Um estudo comparativo de cargas de trabalho e políticas de escalonamento para aplicações paralelas em *clusters* e *grids* computacionais

Juliano Amorim de Oliveira

Um estudo comparativo de cargas de trabalho e políticas de escalonamento para aplicações paralelas em *clusters* e *grids* computacionais

Juliano Amorim de Oliveira

Orientadora: ***Profa. Dra. Regina Helena Carlucci Santana***

Dissertação apresentada ao Instituto de Ciências Matemáticas e de Computação - ICMC-USP, como parte dos requisitos para obtenção do título de Mestre em Ciências - Ciências de Computação e Matemática Computacional.

"VERSÃO REVISADA APÓS A DEFESA"

Data da Defesa:	01/09/2006
Visto do Orientador:	

USP – São Carlos
Setembro/2006

DEDICO,

aos meus pais, Jamir e Ivanilde, meus maiores incentivadores, exemplos de fé e dedicação;

à minha irmã, Liliane, exemplo de força e amizade sincera

Agradecimentos

Ao bom Deus, agradeço o dom da vida. *"O Senhor é meu pastor, nada me faltará."*

A todos os meus familiares, em especial aos tios João, Luzia e Lia, pela confiança, incentivo e orações.

A Professora Dra. Regina Helena Carlucci Santana, pela orientação.

Ao Professor Dr. Rodrigo Fernandes de Mello, por acreditar no meu trabalho e, principalmente, pelo respeito e incentivo, peculiar somente aos grandes Mestres. Seus ensinamentos serão sempre os meus melhores guias.

Aos demais professores e funcionários do programa de mestrado do ICMC/USP, pelo apoio sempre presente.

Aos amigos do LaSDPC, Márcio e Nadja, Luciano e Lilian, Mário, Douglas, Valéria, Caio, Luís, Lucas, pela colaboração neste trabalho e por tornarem esse período mais agradável.

Aos amigos, Michel e Renato, pelo apoio constante e pelo companheirismo nos bons e nos maus momentos.

Ao amigo Bruno, pela amizade sincera, pelas correções criteriosas e opiniões sempre seguras.

A todos os amigos que fiz durante o período deste trabalho, cujos nomes omitirei para não cometer a injustiça de esquecer algum.

Aos professores da UEMG/Passos, em especial ao Professor MSc. Gualberto Rabay Filho e ao Professor MSc. Alessandro de Castro Borges, por me indicarem o caminho dessa conquista.

Ao CNPq, pelo apoio financeiro.

“O sucesso não deve ser medido pelas culminâncias que uma pessoa atinge, mas pelos obstáculos que teve de transpor para alcançá-las.”

Booker T. Washington

Resumo

OLIVEIRA, J. A. *Um estudo comparativo de cargas de trabalho e políticas de escalonamento para aplicações paralelas em clusters e grids computacionais*. Dissertação (Mestrado) — ICMC–USP, São Carlos, 2006.

Diversas políticas de escalonamento para aplicações paralelas voltadas a ambientes computacionais distribuídos têm sido propostas. Embora tais políticas apresentem bons resultados, elas são, geralmente, avaliadas em cenários específicos. Quando o cenário muda, com diferentes ambientes distribuídos e condições de carga, essas políticas podem ter seu desempenho deteriorado. Nesse contexto, este trabalho apresenta um estudo comparativo envolvendo dez políticas de escalonamento avaliadas em diferentes cenários. Cada uma das políticas foi submetida a uma combinação de quatro cargas de trabalho de ocupação da UCP e três variações da taxa de comunicação média entre os processos, utilizando a rede. Foram considerados ainda três sistemas distribuídos distintos: dois *clusters*, com diferentes quantidades de nós, e um *grid* computacional. Foi utilizada a simulação com ambientes próximos ao real e cargas de trabalho obtidas de modelos realísticos. Os resultados demonstraram que, embora as políticas sejam voltadas a ambientes computacionais paralelos e distribuídos, quando o cenário muda, o desempenho cai e a ordem de classificação entre as políticas se altera. Os resultados permitiram ainda demonstrar a necessidade de se considerar a comunicação entre os processos durante o escalonamento em grids computacionais.

Palavras-chave: políticas de escalonamento, computação paralela e distribuída, avaliação de desempenho, simulação

Abstract

OLIVEIRA, J. A. *A comparative study of workloads and policies for parallel job scheduling on clusters and grid computing*. 2006. Master Thesis – ICMC, USP, São Carlos, 2006

Several scheduling policies for parallel applications directed to the distributed computational environments have been proposed. Although such policies present good results, they, generally, are evaluated in specific scenarios. When scenario change, by using different distributed environments and workload conditions, these policies can have its performance spoiled. In this context, this work presents a comparative study involving ten scheduling policies evaluated on different scenarios. Each policy was submitted to a combination of four CPU occupation workloads and three variations of interprocess average communication rates, using the network. Three different distributed systems had been yet considered: two clusters, with different amounts of nodes, and one grid computing. Simulation was used with environments near to the real and workloads obtained of realistic models. Although the policies are directed to parallel and distributed environments, the results have demonstrated that when scenario change, the performance falls and the ranking between the policies changes too. The results have still allowed to demonstrate the necessity of considering interprocess communication during the scheduling in a grid computing.

Keywords: scheduling policies, parallel and distributed computing, performance evaluation, simulation

Lista de Ilustrações

2.1 Taxonomia de arquiteturas paralelas segundo Tanenbaum (1999) 35
2.2 Modelos de comunicação de computadores paralelos (STALLINGS, 2003) 35
(a) Multiprocessador 35
(b) Multicomputador 35

3.1 Taxonomia Hierárquica de Casavant e Kuhl (1988) 52
3.2 Modelos de escalonadores (BUYYA et al., 2002) 65
(a) Centralizado 65
(b) Descentralizado 65
(c) Hierárquico 65

4.1 Soluções de um modelo (SANTANA et al., 1994) 71
4.2 Propostas de Avaliação e Técnicas de Aferição (SANTANA et al., 1994) 73

5.1 Os três tipos genéricos de sistemas de fila (FEITELSON; RUDOLPH, 1998) 79

7.1 Diagrama do sistema utilizado na parametrização 116

Lista de Gráficos

6.1 Histograma de Tamanho de *Job* – Feit96 (1000 *jobs*) 100
6.2 Histograma de Tamanho de *Job* – Jann97-ASCI (1000 *jobs*) 101
6.3 Histograma de Tamanho de *Job* – Jann97-CTC (1000 *jobs*) 101
6.4 Histograma de Tamanho de *Job* – Lub99 (1000 *jobs*) 102
6.5 Histograma de Tempo de Execução – Feit96 (1000 *jobs*) 102
6.6 Histograma de Tempo de Execução – Jann97-ASCI (1000 *jobs*) 103
6.7 Histograma de Tempo de Execução – Jann97-CTC (1000 *jobs*) 103
6.8 Histograma de Tempo de Execução – Lub99 (1000 *jobs*) 103
6.9 Histograma de Tempo de Execução – Feit96 (*Jobs* com até 20 minutos) 104
6.10 Histograma de Tempo de Execução – Jann97-ASCI (*Jobs* com até 20 minutos) . . . 104
6.11 Histograma de Tempo de Execução – Jann97-CTC (*Jobs* com até 20 minutos) . . . 104
6.12 Histograma de Tempo de Execução – Lub99 (*Jobs* com até 20 minutos) 105
6.13 Correlação Tamanho de Job e Tempo de Execução – Feit96 105
6.14 Correlação Tamanho de Job e Tempo de Execução – Jann97-ASCI 106
6.15 Correlação Tamanho de Job e Tempo de Execução – Jann97-CTC 106
6.16 Correlação Tamanho de Job e Tempo de Execução – Lub99 107
6.17 Histogramas de Intervalos entre Chegadas (1000 jobs) 107
6.18 Correlação entre intervalo de chegada e quantidade de jobs submetidos 108
7.1 Cluster 32 nós com a carga Feit96, sem rede 120
7.2 Cluster 32 nós com a carga Feit96, 100 Kbps 120
7.3 Cluster 32 nós com a carga Feit96, 1000 Kbps 120
7.4 Cluster 32 nós com a carga Jann97-ASCI, sem rede 121
7.5 Cluster 32 nós com a carga Jann97-ASCI, 100 Kbps 122
7.6 Cluster 32 nós com a carga Jann97-ASCI, 1000 Kbps 122
7.7 Cluster 32 nós com a carga Jann97-CTC, sem rede 123
7.8 Cluster 32 nós com a carga Jann97-CTC, 100 Kbps 123
7.9 Cluster 32 nós com a carga Jann97-CTC, 1000 Kbps 123
7.10 Cluster 32 nós com a carga Lub99, sem rede 125
7.11 Cluster 32 nós com a carga Lub99, 100 Kbps 125
7.12 Cluster 32 nós com a carga Lub99, 1000 Kbps 125
7.13 Cluster 128 nós com a carga Feit96, sem rede 127
7.14 Cluster 128 nós com a carga Feit96, 100 Kbps 127
7.15 Cluster 128 nós com a carga Feit96, 1000 Kbps 128
7.16 Cluster 128 nós com a carga Jann97-ASCI, sem rede 128
7.17 Cluster 128 nós com a carga Jann97-ASCI, 100 Kbps 129
7.18 Cluster 128 nós com a carga Jann97-ASCI, 1000 Kbps 129
7.19 Cluster 128 nós com a carga Jann97-CTC, sem rede 130
7.20 Cluster 128 nós com a carga Jann97-CTC, 100 Kbps 130
7.21 Cluster 128 nós com a carga Jann97-CTC, 1000 Kbps 130

7.22 Cluster 128 nós com a carga Lub99, sem rede 131
7.23 Cluster 128 nós com a carga Lub99, 100 Kbps 131
7.24 Cluster 128 nós com a carga Lub99, 1000 Kbps 132
7.25 Grid 512 nós (média 8/rede) com a carga Feit96, sem rede 133
7.26 Grid 512 nós (média 8/rede) com a carga Feit96, 100 Kbps 133
7.27 Grid 512 nós (média 8/rede) com a carga Feit96, 1000 Kbps 133
7.28 Grid 512 nós (média 8/rede) com a carga Jann97-ASCI, sem rede 135
7.29 Grid 512 nós (média 8/rede) com a carga Jann97-ASCI, 100 Kbps 135
7.30 Grid 512 nós (média 8/rede) com a carga Jann97-ASCI, 1000 Kbps 135
7.31 Grid 512 nós (média 8/rede) com a carga Jann97-CTC, sem rede 136
7.32 Grid 512 nós (média 8/rede) com a carga Jann97-CTC, 100 Kbps 136
7.33 Grid 512 nós (média 8/rede) com a carga Jann97-CTC, 1000 Kbps 137
7.34 Grid 512 nós (média 8/rede) com a carga Lub99, sem rede 137
7.35 Grid 512 nós (média 8/rede) com a carga Lub99, 100 Kbps 138
7.36 Grid 512 nós (média 8/rede) com a carga Lub99, 1000 Kbps 138

Lista de Tabelas

2.1 Taxonomia de Flynn para computadores paralelos (FLYNN; RUDD, 1996) 33
2.2 Resumo das características típicas de diferentes sistemas paralelos (CIRNE, 2002) . . 47

3.1 Quadro comparativo dos ambientes de escalonamento (SENGER, 2005) 63

5.1 Comparação dos modelos de carga de trabalho disponíveis no *Parallel Workload Archive* (PWA, 2005) 87
5.2 Descrição dos campos do *Standard Workload Format* (PWA, 2005) 89

7.1 Configuração do Sistema Parametrizado 116
7.2 $\theta_{s,k}$ para cada par de computadores do sistema parametrizado (em segundos) 117

A.1 Resumo dos logs disponíveis no *Parallel Workload Archive* (PWA, 2005) 160

B.1 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Feit96, s/ rede 162
B.2 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Feit96, 100 Kbps 162
B.3 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Feit96, 1000 Kpbs 162
B.4 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-ASCI, s/ rede 163
B.5 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-ASCI, 100 Kbps 163
B.6 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-ASCI, 1000 Kpbs 163
B.7 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-CTC, s/ rede 164
B.8 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-CTC, 100 Kbps 164
B.9 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-CTC, 1000 Kpbs 164
B.10 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Lub99, s/ rede 165
B.11 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Lub99, 100 Kbps 165
B.12 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Lub99, 1000 Kpbs 165
B.13 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Feit96, s/ rede 166

B.14 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Feit96, 100 Kbps . . . 166
B.15 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Feit96, 1000 Kpbs . . . 166
B.16 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-ASCI, s/ rede . . . 167
B.17 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-ASCI, 100 Kbps . . . 167
B.18 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-ASCI, 1000 Kpbs . . . 167
B.19 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-CTC, s/ rede . . . 168
B.20 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-CTC, 100 Kbps . . . 168
B.21 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-CTC, 1000 Kpbs . . . 168
B.22 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Lub99, s/ rede . . . 169
B.23 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Lub99, 100 Kbps . . . 169
B.24 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Lub99, 1000 Kpbs . . . 169
B.25 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Feit96, s/ rede . . . 170
B.26 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Feit96, 100 Kbps . . . 170
B.27 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Feit96, 1000 Kpbs . . . 170
B.28 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-ASCI, s/ rede . . . 171
B.29 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-ASCI, 100 Kbps . . . 171
B.30 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-ASCI, 1000 Kpbs . . . 171
B.31 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-CTC, s/ rede . . . 172
B.32 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-CTC, 100 Kbps . . . 172
B.33 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-CTC, 1000 Kpbs . . . 172
B.34 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Lub99, s/ rede . . . 173
B.35 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Lub99, 100 Kbps . . . 173
B.36 Tempo Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Lub99, 1000 Kpbs . . . 173

Lista de Siglas e Abreviaturas

AG	Algoritmos Genéticos
AMIGO	*dynAMical flexIble schedulinG envirOnment*
API	*Application Program Interface*
ASP	*Application Service Provider*
ATM	*Asynchronous Transfer Mode*
CC-NUMA	*Cache Coherent NUMA*
CISC	*Complex Instruction Set Computers*
CLUMP	*Cluster of SMPs*
COMA	*Cache Only Memory Access*
CORBA	*Common Object Request Broker Architecture*
COW	*Cluster of Workstations*
DPWP	*Dynamic Policy Without Preemption*
DSM	*Distributed Shared Memory*
E/S	Entrada/Saída
EP	Elemento de Processamento
FLOPS	*Floating Point Operations Per Second*
GAS	*Genetic Algorithm Scheduling*
Gbps	Gigabits por segundo
I/O	*Input/Output*
IBL	*Instance-Based Learning*
IPC	*InterProcess Communication*
IP	*Internet Protocol*
JMS	*Job Management System*
LAN	*Local Area Network*
LaSDPC	Laboratório de Sistemas Distribuídos e Programação Concorrente
LBM	*Load Balance Manager*
LIM	*Load Information Manager*
Mbps	Megabits por segundo
MFLOPS	*Millions of Floating Point Operations Per Second*
MIMD	*Multiple Instruction, Multiple Data*

MIPS	Milhões de Instruções Por Segundo
MISD	*Multiple Instruction, Single Data*
MPI	*Message Passing Interface*
MPMD	*Multiple Program Multiple Data*
MPP	*Massively Parallel Processing*
NBSP	*Network Bound Scheduling Policy*
NC-NUMA	*Non-Coherent NUMA*
NFS	*Network File System*
NoC	*Network of Computers*
NOM	*Network of Machines*
NOW	*Network of Workstations*
NUMA	*Non-Uniform Memory Access*
P2P	*Peer-to-Peer*
PC	*Personal Computer*
PVM	*Parallel Virtual Machine*
RISC	*Reduced Instruction Set Computers*
RMI	*Remote Method Invocation*
RPC	*Remote Procedure Call*
SD	Sistema Distribuído
SIMD	*Single Instruction, Multiple Data*
SISD	*Single Instruction, Single Data*
SMP	*Symmetric MultiProcessor*
SPMD	*Single Program Multiple Data*
SWF	*Standard Workload Format*
TCP	*Transmission Control Protocol*
TLBA	*Tree Load Balancing Algorithm*
UCP	Unidade Central de Processamento
UDP	*User Datagram Protocol*
ULA	Unidade Lógica e Aritmética
UMA	*Uniform Memory Access*
UniMPP	*Unified Modeling for Predicting Performance*
VLSI	*Very Large Scale Integration*
WAN	*Wide Area Network*
XML	*eXtended Markup Language*

Sumário

1 Introdução **23**
1.1 Contextualização . . . 23
1.2 Motivação e Objetivos . . . 24
1.3 Organização desta dissertação . . . 27

2 Computação Paralela e Distribuída **29**
2.1 Computação Paralela . . . 30
2.1.1 Conceitos Gerais . . . 31
2.1.2 Arquiteturas Paralelas . . . 33
2.2 Sistemas Distribuídos . . . 37
2.2.1 Principais Características . . . 38
2.2.2 Redes de Computadores . . . 39
2.2.3 Comunicação entre Processos . . . 40
2.3 *Clusters* e NOWs . . . 42
2.4 Ferramentas para Computação Paralela Distribuída . . . 43
2.4.1 Parallel Virtual Machine . . . 44
2.4.2 Message Passing Interface . . . 45
2.5 Grids Computacionais . . . 45
2.6 Considerações Finais . . . 48

3 Escalonamento de Aplicações Paralelas em Sistemas Distribuídos **49**
3.1 Terminologia . . . 50
3.2 Escalonamento . . . 51
3.2.1 Taxonomia para o escalonamento . . . 52
3.2.2 Escalonamento local . . . 54
3.2.3 Escalonamento global . . . 54
3.2.4 Políticas de Escalonamento . . . 57
3.2.5 Objetivos e Desempenho do Escalonamento . . . 58
3.3 Ambientes de escalonamento . . . 59
3.4 Escalonamento em *grids* . . . 63
3.5 Considerações Finais . . . 66

4 Avaliação de Desempenho **67**
4.1 Métricas de Desempenho . . . 68
4.2 Técnicas para Avaliação de Desempenho . . . 69
4.2.1 Técnicas de Modelagem . . . 69
4.2.2 Soluções para os Modelos . . . 71
4.2.3 Técnicas de Aferição . . . 72
4.3 *Benchmarking* . . . 74

4.4 Considerações finais 76

5 Cargas de trabalho e Métricas de Desempenho 77
5.1 Avaliação de Desempenho do Escalonamento 78
5.2 Cargas de trabalho 80
5.2.1 Modelos de cargas de trabalho 82
5.2.2 Formato Padrão para Cargas de Trabalho 87
5.3 Métricas de Desempenho 88
5.4 Considerações Finais 91

6 Estudo Comparativo de Cargas de Trabalho e Políticas de Escalonamento 93
6.1 Políticas de Escalonamento 94
6.1.1 Round Robin 94
6.1.2 Random, Disted, Lowest, Central e Global 94
6.1.3 DPWP 97
6.1.4 NBSP 97
6.1.5 GAS 98
6.1.6 TLBA 99
6.2 Análise das cargas de trabalho 99
6.3 Simulador 109
6.4 Considerações Finais 113

7 Resultados 115
7.1 Parametrização do Modelo de Simulação 115
7.2 Simulações 118
7.3 Resultados 119
7.4 Considerações Finais 138

8 Conclusões 141
8.1 Conclusões e Contribuições 142
8.2 Trabalhos Futuros 145

Referências 147

A Logs do Parallel Workload Archive 159

B Intervalos de Confiança 161

Capítulo

1

Introdução

1.1 Contextualização

Os avanços tecnológicos obtidos nas últimas décadas, principalmente no que se refere ao desenvolvimento dos microprocessadores e das redes de comunicação, possibilitaram rápidas e profundas mudanças. A computação, que era baseada unicamente no uso de computadores seqüenciais e centralizados, é caracterizada, atualmente, pela grande conectividade dos recursos, tornando cada vez mais comum o uso de Sistemas Distribuídos (COULOURIS et al., 2001).

Sistemas Distribuídos são compostos por computadores autônomos, que interligados por uma rede de comunicação e utilizando software apropriado, coordenam suas tarefas (COULOURIS et al., 2001). Eles surgiram da necessidade de compartilhar recursos de modo eficiente, mas seu uso tem se estendido a diversos domínios.

Uma das áreas que se beneficiou dos Sistemas Distribuídos é a Computação Paralela (QUINN, 1994). A Computação Paralela explora a existência de tarefas concorrentes, que podem ser executadas em paralelo, a fim de obter um tempo de resposta menor. Para dar suporte à Computação Paralela, utiliza-se um computador com múltiplos elementos de processamento (EPs), capazes de comunicar e cooperar para resolver grandes problemas mais rapidamente (ALMASI; GOTTLIEB, 1994).

Uma forma de obter uma arquitetura com múltiplos EPs é utilizar um sistema distribuído. Os avanços em capacidade de processamento, em tecnologias de rede e em ferramentas de software tornaram os Sistemas Distribuídos uma infra-estrutura conveniente para o processamento paralelo (HARIRI; PARASHAR, 2004). Essa convergência, entre Computação Paralela e Sistemas Distribuídos, é denominada Computação Paralela e Distribuída (ZOMAYA, 1996) e

vêm se tornando bastante atrativa, especialmente por prover uma boa relação custo/benefício.

É preciso ressaltar, no entanto, que existem problemas com essa abordagem. Nos Sistemas Distribuídos os elementos de processamento são, potencialmente, heterogêneos e são interligados por uma rede de comunicação compartilhada e de baixa velocidade. Além disso, é preciso considerar que existem cargas de trabalho externas às aplicações paralelas. Esses fatores tornam o software paralelo e distribuído ainda mais complexo e se refletem diretamente no desempenho do sistema, no que se refere à adequada utilização dos recursos disponíveis. O escalonamento das tarefas para os recursos disponíveis nesses ambientes se torna uma tarefa de extrema importância (HARIRI; PARASHAR, 2004).

O escalonamento refere-se à atividade de alocar os recursos disponíveis entre as tarefas que compõem cada aplicação. Em qualquer sistema, um bom escalonamento, baseado nos objetivos propostos, é determinante para obter um bom desempenho (CASAVANT; KUHL, 1988). O escalonamento em Sistemas Distribuídos tem sido largamente estudado (HAMSCHER et al., 2000; KARATZA; HILZER, 2003; SHIM, 2004; SENGER, 2005; ISHII et al., 2005). Há uma intensa atividade de pesquisa, buscando formas eficientes de alocar aplicações paralelas nos principais sistemas distribuídos utilizados para processamento paralelo: *Network of Workstations* (NOW), *clusters* e *grids* computacionais. Muitos conceitos e técnicas utilizados para o escalonamento de aplicações em computadores paralelos não são adequados quando se trata de um sistema distribuído, havendo a necessidade de adaptar essas técnicas ou criar novas técnicas que garantam um bom desempenho.

As decisões comuns, que devem ser tomadas pelo software que realiza o escalonamento, são determinadas por políticas de escalonamento. As políticas de escalonamento determinam, por exemplo, quais aplicações obterão acesso aos recursos disponíveis, a quantidade de recursos que serão destinados para as aplicações e a localização desses recursos. Para obter um bom desempenho é necessário empregar boas políticas de escalonamento, adequadas ao ambiente computacional alvo e a carga de trabalho esperada.

1.2 Motivação e Objetivos

Clusters e *grids* são uma alternativa interessante e viável para a execução de aplicações paralelas e têm sido largamente empregados. Enquanto máquinas verdadeiramente paralelas possuem um alto custo de aquisição e manutenção, além de grande dependência de um fabricante, os sistemas paralelos distribuídos apresentam uma excelente relação custo/benefício e uma grande flexibilidade. Além disso, a disponibilidade de diversas ferramentas de software facilitam o desenvolvimento de uma aplicação paralela. Essas ferramentas são particularmente interessantes porque permitem que aplicações paralelas possam ser portáveis entre diferentes sistemas computacionais paralelos. Exemplos dessas ferramentas são *Parallel Virtual Machine* (PVM) (GEIST et al., 1994), *Message Passing Interface* (MPI) (SNIR et al., 1996) e

o projeto Globus (FOSTER, 2005) para *grids* computacionais.

Devido à grande evolução dos sistemas paralelos e distribuídos, diversas políticas de escalonamento para aplicações paralelas têm sido propostas para esses ambientes (ISHII, 2004; MELLO, 2003; DANDAMUDI; ZHOU, 2004; MELLO et al., 2006). Essas políticas têm atraído a atenção de pesquisadores, uma vez que os resultados obtidos têm indicado bons desempenhos. Entretanto, tais políticas são avaliadas com o objetivo apenas de demonstrar sua validade e/ou superioridade sob um determinado aspecto. Sob diferentes condições de carga, a política pode ter seu desempenho deteriorado, o que, na maioria das vezes, não é verificado. Analisar o comportamento de uma política de escalonamento sob diferentes cenários é uma importante tarefa para se compreender como a mesma se comportaria em um ambiente real e, assim, garantir um bom desempenho.

Diversos trabalhos comparam diferentes estratégias para o escalonamento em *clusters*, NOWs (ARNDT et al., 2000; KARATZA, 2000; KARATZA, 2002a; KARATZA; HILZER, 2003; SHIM, 2004) e em *grids* (HAMSCHER et al., 2000). Entretanto, nesses trabalhos não há a preocupação em verificar o comportamento da política de escalonamento sob diferentes condições de carga que podem ocorrer e nem sua aplicabilidade a outros ambientes distribuídos. Uma política de escalonamento que apresenta um bom desempenho sob determinada carga de trabalho e ambiente distribuído pode ter seu desempenho reduzido quando o cenário muda. O contrário também pode ocorrer. Uma política que obtém mau desempenho sob determinada carga pode se comportar bem sob outra.

Este trabalho é motivado pela necessidade de verificar o comportamento de políticas de escalonamento para aplicações paralelas sob diferentes cenários. Complementar o trabalho do grupo de pesquisa em que este trabalho se insere é também uma importante motivação para a definição e execução deste trabalho.

O grupo de pesquisa do Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC) – Instituto de Ciências Matemáticas e de Computação da Universidade de São Paulo (ICMC–USP) tem direcionado esforços nas linhas de pesquisa em que este trabalho se insere. O grupo vêm acumulando conhecimento nas áreas de avaliação de desempenho de sistemas computacionais e no escalonamento de aplicações paralelas em Sistemas Distribuídos. Um dos primeiros resultados desse esforço é o *dynAMical flexIble schedulinG envirOnment* (AMIGO), descrito na seção 3.3. Seguindo a linha desse trabalho diversos trabalhos relacionados ao escalonamento foram desenvolvidos, dentre os quais pode-se destacar:

- Especificação e implementação do AMIGO (SOUZA, 2000)
- Implementação de uma interface gráfica para o ambiente AMIGO (CAMPOS, 2001)
- Integração AMIGO-PVM e implementação da política *Dynamic Policy Without Preemption* (DPWP) (ARAÚJO, 1999)
- Integração AMIGO-MPI (FIGUEIREDO, 2000)

- Integração AMIGO-CORBA (SANTOS, 2001)
- Monitoração aplicada ao escalonamento em sistemas distribuídos e Análise de métricas de desempenho (SOUZA, 2004)
- Definição de índices de carga para definir o grau de heterogeneidade de um sistema distribuído (BRANCO, 2004)
- Definição de uma política de escalonamento para aplicações com alta comunicação na rede, *Network Bound Scheduling Policy* (NBSP) (ISHII, 2004)
- Aquisição do conhecimento de aplicações paralelas aplicado ao escalonamento de processos e definição da política GAS (SENGER, 2005)
- Estudo de caso para o escalonamento em *grids* computacionais (REIS, 2005)
- Definição de índices de carga para políticas de escalonamento *memory-bound* (VOORSLUYS, 2006)
- Um simulador para o escalonamento em ambientes paralelos e distribuídos heterogêneos (MELLO; SENGER, 2006)

Nesse contexto, o problema central estudado neste trabalho é a avaliação de desempenho de políticas de escalonamento para aplicações paralelas sob diferentes cenários. O objetivo principal deste trabalho é analisar o comportamento de políticas de escalonamento voltadas a ambientes paralelos e distribuídos quando submetidas a diferentes condições de carga e quando aplicadas a diferentes ambientes computacionais distribuídos.

Dentre as contribuições esperadas com a execução deste trabalho, pode-se destacar as seguintes:

- Complementar trabalhos que propõem e avaliam políticas de escalonamento, apontando restrições e características inerentes às políticas que não foram identificadas nos trabalhos originais. Como discutido, grande parte da avaliação de desempenho conduzida nos trabalhos que propõem novas políticas tem o objetivo apenas de validar a proposta. Geralmente, as restrições e a aplicabilidade da política a diferentes cenários não é identificada.
- Verificar a aplicabilidade de políticas de escalonamento voltadas a *clusters* em *grids* computacionais. O escalonamento em *grids* é uma intensa atividade de pesquisa, motivada pelas restrições que esse ambiente impõe ao escalonador. Políticas de escalonamento utilizadas em *clusters* podem ter sua aplicabilidade reduzida em *grids*, devido a tais restrições.
- Verificar que política melhor se aplica a determinado cenário. Um estudo, realizado aplicando uma avaliação de desempenho adequada, deve permitir analisar, comparativamente, diversas políticas e apontar aquela que obteve o melhor desempenho sob um determinado cenário estudado. Um aspecto importante desse estudo é que o mesmo se aproxime ao máximo de um ambiente real.

- Definição de uma sistemática para avaliar, comparativamente, políticas de escalonamento. Poucos trabalhos tratam da metodologia de avaliação de desempenho de políticas e discutem técnicas para comparar políticas. Pretende-se com este trabalho, revisar e apresentar técnicas aplicáveis para comparar políticas de escalonamento.
- Analisar o impacto da comunicação entre os processos das aplicações em *grids* computacionais. Grids são compostos por computadores pertencentes a diferentes organizações. Entre essas organizações existe uma rede com alto atraso e baixa largura de banda (geralmente, a Internet). Aplicações que se comunicam pela rede podem ter seu desempenho reduzido devido aos atrasos gerados pela rede de comunicação.

1.3 Organização desta dissertação

Esta dissertação está organizada em oito capítulos e dois apêndices, sendo o primeiro capítulo esta Introdução.

Os três capítulos seguintes destinam-se à revisão bibliográfica das áreas em que este trabalho se insere. No Capítulo 2 é tratada a Computação Paralela e Distribuída, destacando-se as arquiteturas utilizadas para a execução de aplicações paralelas. Ferramentas disponíveis para a programação paralela em ambientes distribuídos também são discutidas. No Capítulo 3, o escalonamento em *clusters*, NOW e *grids* é tratado. Detalhes inerentes a estes ambientes que interferem no escalonamento e, conseqüentemente, no desempenho, são tratados ao longo do capítulo. O Capítulo 4 destina-se a revisar as técnicas empregadas na avaliação de desempenho.

A revisão apresentada nestes três capítulos visa fornecer ao leitor o embasamento necessário à compreensão deste trabalho. Esses capítulos foram organizados de forma sucinta, mas bastante completa. O leitor que se sentir a vontade com os assuntos tratados nesses capítulos pode prosseguir diretamente para o Capítulo 5.

No Capítulo 5, é tratada a avaliação de desempenho no contexto específico do escalonamento, buscando enfatizar as escolhas realizadas neste trabalho com relação às cargas de trabalho, à métrica de desempenho e à utilização de simulação, como técnica de avaliação de desempenho.

O Capítulo 6 descreve o trabalho desenvolvido neste mestrado. As políticas de escalonamento estudadas são detalhadas, apontando seu funcionamento. Além disso, as cargas de trabalho são analisadas, provendo detalhes das distribuições de grau de paralelismo, tempo de execução e utilização da rede. O modelo de simulação empregado neste trabalho, assim como o simulador que o implementa são descritos com detalhes. As extensões aplicadas ao simulador durante esse trabalho também são detalhadas.

Os resultados obtidos neste trabalho são descritos no Capítulo 7. A parametrização do modelo de simulação é discutida e, em seguida, os resultados são analisados. Observações a

respeito dos resultados obtidos também estão disponíveis.

Finalmente, no Capítulo 8, são apresentadas as conclusões finais e as principais contribuições obtidas com o desenvolvimento deste trabalho de mestrado. Além disso, são apontados caminhos para trabalhos futuros.

O Apêndice A apresenta detalhes dos *logs* disponíveis no repositório *Parallel Workload Archive* (PWA, 2005), visando complementar a discussão apresentada na seção 5.2, a respeito dos modelos de carga de trabalho. Tais modelos foram gerados a partir de *logs*, muito dos quais estão disponíveis nesse repositório.

O Apêndice B complementa os resultados expostos no Capítulo 7. São apresentados os intervalos de confiança para os experimentos conduzidos neste trabalho e é apresentado o modo como o cálculo foi efetuado.

Capítulo

2

Computação Paralela e Distribuída

A Computação Paralela e Distribuída é a convergência de duas áreas surgidas com motivações e características distintas, a Computação Paralela e os Sistemas Distribuídos, trazendo benefícios para ambas (ZOMAYA, 1996). Essa convergência surgiu dos avanços tecnológicos que tornaram os Sistemas Distribuídos uma infra-estrutura de suporte à Computação Paralela.

Para dar suporte à Computação Paralela diversas arquiteturas podem ser empregadas. Arquiteturas MIMD[1] com memória distribuída têm sido usadas para aplicações que demandam grande potência computacional. Exemplo são as arquiteturas de processamento maciçamente paralelo — *Massively Parallel Processing* (MPP), constituídas por um grande número de elementos de processamento (EPs) fortemente acoplados por uma rede de interconexão proprietária de alta velocidade.

Outra forma de obter uma arquitetura paralela com memória distribuída é utilizando um Sistema Distribuído (SD). Um SD é logicamente similar a um MPP, custando uma fração do preço e apresentando-se bastante flexível (devido à independência de fabricante). Os Sistemas Distribuídos têm evoluído bastante e possuem várias características desejáveis na computação paralela: transparência de acesso aos recursos, confiabilidade, tolerância à falhas (HARIRI; PARASHAR, 2004).

A Computação Paralela sobre Sistemas Distribuídos, ou Computação Paralela Distribuída, surgiu como alternativa ao uso de supercomputadores paralelos caros, especializados e proprietários. Os avanços obtidos em capacidade de processamento, tecnologia de rede e em ferramentas de software tornaram os Sistemas Distribuídos uma infra-estrutura conveniente para o processamento paralelo (HARIRI; PARASHAR, 2004).

[1]MIMD é uma das categorias da Taxonomia de Flynn para computadores paralelos, discutida na Seção 2.1.2

A Computação Paralela Distribuída, é, portanto, bastante atrativa. A idéia básica é utilizar os computadores que compõem um sistema distribuído como elementos de processamento de uma máquina paralela virtual (ou ambiente paralelo virtual). Entretanto é preciso ressaltar que existem desvantagens e problemas. Técnicas e mecanismos antes utilizados na Computação Paralela nem sempre são convenientes na Computação Paralela e Distribuída. Fatores característicos dos Sistemas Distribuídos, tais como heterogeneidade, atraso na rede e carga de trabalho externa às aplicações paralelas, tornam a utilização e o gerenciamento desses sistemas tarefas complexas.

Neste capítulo são abordados conceitos inerentes à Computação Paralela Distribuída. Para facilitar o estudo, Computação Paralela e Sistemas Distribuídos são tratados separadamente. Em seguida, a convergência das duas áreas é apresentada em termos dos sistemas computacionais distribuídos empregados para executar aplicações paralelas e as ferramentas de software empregadas para esse fim. O objetivo desse capítulo é identificar características do hardware e do software básico que influenciam no desempenho.

2.1 Computação Paralela

Embora os computadores uniprocessados tradicionais tornem-se mais rápidos, existem aplicações que não podem ser executadas com eficiência em arquiteturas seqüenciais. Algumas aplicações, em virtude do uso de algoritmos complexos e/ou conjuntos de dados extensos, demandam grande potência computacional; outras são naturalmente paralelas. Com o objetivo principal de prover melhores desempenho e adequação para essas aplicações, surgiu a Computação Paralela (QUINN, 1994; DONGARRA et al., 2003; ALMASI; GOTTLIEB, 1994).

A Computação Paralela explora a existência de tarefas concorrentes, que podem ser executadas em paralelo a fim de obter um tempo de resposta menor. Para dar suporte à Computação Paralela utiliza-se um computador com múltiplos EPs, capazes de comunicar e cooperar para resolver grandes problemas mais rapidamente (ALMASI; GOTTLIEB, 1994).

Paralelismo não é um conceito novo em computação. Por exemplo, no nível de instruções são comuns organizações *pipeline*, possibilitando o processamento paralelo de instruções e, conseqüentemente, um desempenho maior, se comparado ao processamento puramente seqüencial. Embora o paralelismo de baixo nível apresente melhorias, só é possível obter grande aumento de desempenho, a um fator teoricamente ilimitado, através da replicação de toda a UCP (ou de suas partes principais) (TANENBAUM, 1999).

Um único processador possui limitações físicas. A velocidade do circuito não pode ultrapassar a velocidade da luz e o desenvolvimento de processadores já esbarra em problemas de refrigeração e nos efeitos da mecânica quântica (TANENBAUM, 1999). Esses fatores, aliados às inovações tecnológicas (como a VLSI) e o conseqüente barateamento dos componentes de hardware, têm tornado a construção de computadores paralelos bastante atrativa. Existem tam-

bém motivações relacionadas ao software. A tolerância a falhas, obtida através da replicação de EPs, é uma característica altamente desejada para aplicações que necessitam de confiabilidade. Entretanto, é importante ressaltar que o software é o grande desafio dessa área.

Um computador paralelo tem pouca ou nenhuma utilidade sem um bom programa paralelo. Embora várias técnicas de programação paralela tenham sido desenvolvidas (FOSTER, 1995; DONGARRA et al., 2003), a construção de aplicações paralelas é ainda uma onerosa tarefa para o programador. A divisão de uma aplicação em processos, que possam ser atribuídos aos EPs de forma eficiente, e a sincronização são tarefas extremamente complexas.

2.1.1 Conceitos Gerais

Concorrência e Paralelismo

Quando dois ou mais processos inicializaram e não terminaram sua execução diz-se que existe concorrência. Geralmente, existe disputa entre esses processos pelo uso de um processador. Para que haja paralelismo é preciso que dois ou mais processos estejam em execução (alocando processador) num mesmo intervalo de tempo (ALMASI; GOTTLIEB, 1994). Assim, processos paralelos são também concorrentes, mas o contrário nem sempre é verdade.

O Paralelismo implica no uso de um sistema paralelo, composto por múltiplos elementos de processamento. Já a concorrência pode ocorrer tanto em sistemas paralelos quando em sistemas com um único processador. Em sistemas uniprocessados tem-se um pseudoparalelismo: a execução de cada processo em pequenos intervalos de tempo dá a impressão de que os processos executam simultaneamente.

Granulosidade ou Níveis de Paralelismo

Granulosidade ou nível de paralelismo refere-se ao tamanho da unidade de trabalho executada pelos processadores, sendo classificada em fina, média e grossa (QUINN, 1994).

A granulosidade fina indica um paralelismo a nível de operações ou de instruções, sendo, geralmente, implementada em hardware. Na granulosidade média, as unidades de trabalho são constituídas de blocos de instruções; geralmente, esse nível de paralelismo exige suporte de linguagens de programação paralela. No último nível da classificação, a granulosidade grossa, o paralelismo é dado a nível dos processos ou tarefas, executando simultaneamente.

Abordagens de Programação Paralela

Os processos e/ou *threads*[2] que compõem uma aplicação paralela podem ser estruturados sob duas abordagens principais: o paralelismo de dados e o paralelismo funcional (HWANG; XU, 1998).

[2]Processos e *threads* são definidos na seção 3.1.

No paralelismo de dados, também denominado *Single Program Multiple Data* (SPMD), os dados a serem processados são divididos em porções, geralmente do mesmo tamanho. Cada processador executa o mesmo código, operando sobre uma porção diferente dos dados. Daí a denominação paralelismo de dados: o paralelismo obtido se deve à decomposição dos dados entre diversos EPs. No paralelismo funcional ou *Multiple Program Multiple Data* (MPMD), cada tarefa possui um conjunto de instruções distinto que podem ou não operar sobre o mesmo conjunto de dados. O nome de deve ao fato de que cada tarefa possui uma função diferente.

Speedup e Eficiência

Speedup e Eficiência são importantes métricas para medir a qualidade de um programa paralelo. O *Speedup* permite verificar o ganho de desempenho obtido com o uso de um programa paralelo em relação ao programa seqüencial mais rápido que executa a mesma tarefa. A Eficiência determina a taxa de utilização dos processadores (QUINN, 1994).

Em sistemas homogêneos (com processadores de mesma potência computacional e mesma quantidade de memória), o *Speedup* é dado pela equação 2.1. O *Speedup* obtido com p processadores, $S(p)$, consiste na razão entre o tempo T_1 gasto para executar o programa seqüencial (em um processador) e o tempo T_p gasto para executar o programa paralelo em p processadores.

$$S(p) = \frac{T_1}{T_p} \tag{2.1}$$

Em sistemas heterogêneos, dados k diferentes EPs e sendo $\Sigma EP = EP_1 + \ldots + EP_k$ o número total de EPs, $T_1(i)$ o tempo para executar o algoritmo seqüencial em um elemento de processamento do tipo i, T_p o tempo para executar o algoritmo paralelo no conjunto de EPs e $T_1(EP)$ o tempo médio ponderado para executar o algoritmo em um EP (equação 2.2), o *Speedup* é dado pela equação 2.3 (NEARY; CAPPELLO, 2000).

$$T_1(EP) = \frac{EP_1 T_1(1) + ... + EP_k T_1(k)}{\Sigma EP} \tag{2.2}$$

$$S = \frac{T_1(EP)}{T_p(EP)} \tag{2.3}$$

A Eficiência $E(p)$ é dada pela equação 2.4, que é a razão entre o *speedup* $S(p)$ e o número de processadores p utilizado.

$$E(p) = \frac{S(p)}{p} \tag{2.4}$$

Pelas equações observa-se que o caso ideal seria $S(p) = p$ e $E(p) = 1$, um ganho igual ao número de processadores utilizado e uma utilização de 100% dos processadores. Entretanto, o caso ideal é dificilmente alcançado, devido a diversos fatores: atraso na comunicação, balanceamento de carga inadequado, partes seqüenciais do programa paralelo, etc.

2.1.2 Arquiteturas Paralelas

Para dar suporte à Computação Paralela diferentes arquiteturas podem ser usadas, existindo, assim, uma grande diversidade de arquiteturas paralelas. Elas variam no que se refere aos componentes utilizados, ao modo de interligá-los e até mesmo ao modelo de programação.

Os elementos de processamento podem variar desde simples ULAs até UCPs e computadores completos. A quantidade de EPs varia, dependendo do propósito da arquitetura paralela e da interconexão utilizada (quanto menor a latência e maior a largura de banda maior o número de EPs). Com relação aos elementos de memória existe também uma grande variedade de características. Geralmente, a memória é utilizada em módulos de forma a permitir acessos paralelos por diferentes EPs. A capacidade total e de cada módulo de memória também varia.

Apesar da diversidade de características apresentada pelos elementos de processamento e de memória, a maior divergência das arquiteturas de computadores paralelos é a forma de interligar esses elementos (TANENBAUM, 1999). Esse é o fator que apresenta grande influência nas características e no desempenho de um computador paralelo e que dá origem a diversas classificações de computadores paralelos.

Taxonomias de computadores paralelos

Diante da diversidade de computadores paralelos, faz-se necessária a introdução de uma classificação. Diversos esquemas foram propostos para classificar arquiteturas de computadores paralelos, mas nenhum teve ampla aceitação (TANENBAUM, 1999). Dentre os esquemas propostos, o mais citado é a taxonomia proposta por Flynn, em 1972.

A Taxonomia de Flynn (FLYNN; RUDD, 1996) é baseada em dois conceitos: fluxos de instruções e fluxos de dados. Um fluxo de instruções é uma seqüência de instruções executadas; cada fluxo corresponde a um contador de programa[3] (TANENBAUM, 1999). Um fluxo de dados consiste em um conjunto de operandos manipulado por um fluxo de instruções (TANENBAUM, 1999; QUINN, 1994). Considerando-se a unicidade ou multiplicidade dos fluxos, obtém-se as quatro categorias que compõem a classificação de Flynn, conforme a tabela 2.1.

Tabela 2.1: Taxonomia de Flynn para computadores paralelos (FLYNN; RUDD, 1996)

Fluxo de Instrução	**Fluxo de Dados**	**Categoria**
Único	Único	SISD
Único	Múltiplo	SIMD
Múltiplo	Único	MISD
Múltiplo	Múltiplo	MIMD

Single Instruction, Single Data **(SISD)** – possui um único fluxo de instrução e um

[3]O contador de programa é o registrador da UCP que armazena a próxima instrução a ser executada

único fluxo de dados. Computadores seriais, baseados no modelo de *von Neumann*, compõem essa categoria. Ainda que exista paralelismo de baixo nível (*pipeline* de instruções), apenas uma instrução é decodificada por vez e, mesmo utilizando múltiplas unidades funcionais, ainda existe apenas uma unidade de controle (QUINN, 1994).

***Single Instruction, Multiple Data* (SIMD)** – um único fluxo de instrução é executado por múltiplos elementos de processamento sobre conjuntos de dados distintos. Há apenas uma unidade de controle que busca a próxima instrução a ser executada paralelamente pelas várias unidades aritméticas sobre seus próprios dados. Processadores vetoriais e matriciais são exemplos de arquiteturas SIMD.

***Multiple Instruction, Single Data* (MISD)** – múltiplos fluxos de instruções operam sobre um mesmo fluxo de dados. Esta categoria é um tanto confusa, existindo uma divergência entre os autores sobre a existência de arquiteturas que possam ser classificadas como MISD. Stallings (2003) afirma que essa estrutura nunca foi implementada. Flynn e Rudd (1996) e Zomaya (1996) classificam *arrays* sistólicos como representantes dessa categoria.

***Multiple Instruction, Multiple Data* (MIMD)** – um conjunto de processadores executam simultaneamente diferentes seqüências de instruções sobre diferentes conjuntos de dados. Essa categoria engloba a maior parte dos computadores paralelos.

Embora útil e bastante difundida, a taxonomia de Flynn se limita às quatro categorias citadas, que não são suficientes para acomodar de forma adequada vários computadores modernos. Assim, os elementos dessa classificação foram mantidos em taxonomias mais abrangentes.

Em 1990, Duncan apresentou uma taxonomia, tendo como um dos objetivos incluir processadores vetoriais com *pipeline* e outras arquiteturas que são difíceis de acomodar na Taxonomia de Flynn. Excluir arquiteturas que empregam apenas paralelismo de baixo nível e manter elementos da Taxonomia de Flynn também foram objetivos da Taxonomia de Duncan (DUNCAN, 1990). Essa classificação é dividida em arquiteturas síncronas e assíncronas.

As arquiteturas Síncronas coordenam operações concorrentes utilizando-se de relógio global, unidade de controle central ou controladoras de unidade vetorial. Esse grupo é composto pelas arquiteturas SIMD da taxonomia de Flynn, pelas arquiteturas vetoriais e sistólicas. Nas arquiteturas Assíncronas não há controle centralizado mantido por hardware; os processadores podem operar de maneira autônoma. Esse grupo é composto, basicamente, pelas arquiteturas MIMD da taxonomia de Flynn, sejam elas convencionais (MIMD com memória distribuída ou compartilhada) ou não-convencionais (MIMD/SIMD, Fluxo de Dados, Redução ou Dirigidas a demanda e Frente de Onda) (DUNCAN, 1990).

Apesar da taxonomia de Duncan ser bastante abrangente, grande parte dos autores (TANENBAUM, 1999; STALLINGS, 2003; GRAMA et al., 2003) preferem utilizar uma taxonomia mais simples que engloba as arquiteturas de computadores paralelos mais utilizadas atualmente. Este esquema se caracteriza por manter as quatro categorias da Taxonomia de Flynn e pela sub-

divisão das categorias SIMD e MIMD, conforme pode ser visto na figura 2.1.

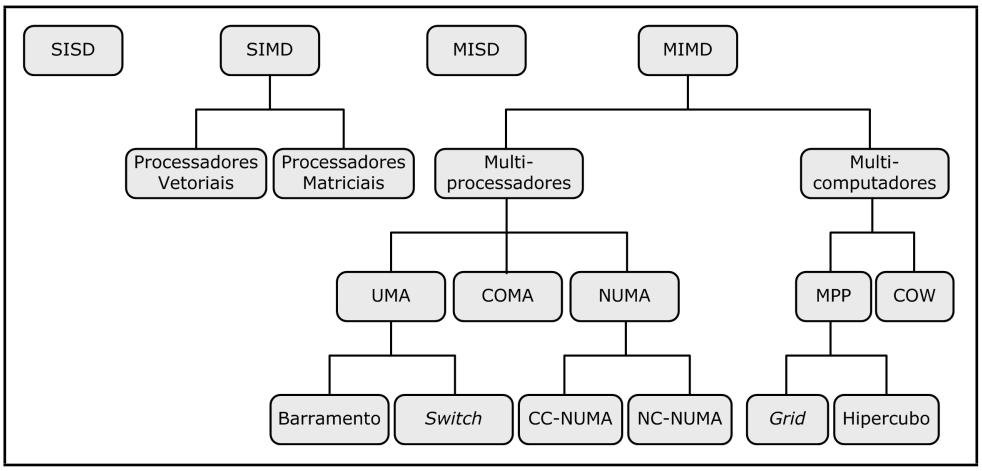
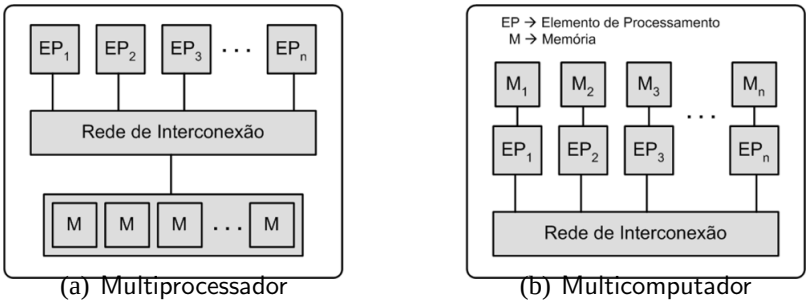


Figura 2.1: Taxonomia de arquiteturas paralelas segundo Tanenbaum (1999)

A categoria SIMD foi dividida em dois grupos: Processadores Vetoriais e Matriciais. O primeiro é composto por supercomputadores numéricos e máquinas que operam sobre vetores, realizando a mesma operação em cada elemento do vetor. O segundo grupo é composto por máquinas com uma unidade de controle principal que repassa a próxima instrução a ULAs independentes. A categoria MIMD é dividida, de acordo com o modelo de comunicação, em multiprocessadores e multicomputadores. Essas arquiteturas são discutidas na seção seguinte.

Modelos de Comunicação

Em qualquer computador paralelo, tarefas paralelas de um mesmo *job*[4] precisam trocar informações. Há duas formas dessa interação ocorrer: acessando um espaço de memória compartilhado ou trocando mensagens. Esses dois modos de interação são suportados, respectivamente, por dois modelos: Multiprocessadores e Multicomputadores. A diferença básica está no modo que a memória é acessível aos EPs, conforme pode ser visto na figura 2.2.



(a) Multiprocessador (b) Multicomputador

Figura 2.2: Modelos de comunicação de computadores paralelos (STALLINGS, 2003)

Nos **Multiprocessadores** os EPs compartilham uma mesma memória física, e por isso são também denominados **Sistemas de Memória Compartilhada**. Embora os módulos de

[4]Os conceitos de *job* e tarefa são discutidos no próximo capítulo

memória possam estar fisicamente separados, existe um único espaço de endereçamento virtual para todos os EPs. Esse modelo de comunicação é esquematizado na figura 2.2(a).

No segundo modelo de comunicação, **Multicomputadores**, a memória encontra-se física e logicamente distribuída, sendo por isso conhecido também como **Sistemas de Memória Distribuída**. Cada EP tem sua própria memória e seu próprio espaço de endereçamento, inacessível a outros EPs por meio de instruções *LOAD* e *STORE*. A figura 2.2(b) representa um multicomputador.

Em sistemas com memória distribuída, a interação entre os EPs ocorre por meio de passagem de mensagens, utilizando a redes de interconexão[5]. Isso tem grandes implicações sobre o software, tornando-o mais complexo. Nos sistemas de memória compartilhada, processos em diferentes EPs podem se comunicar apenas lendo e escrevendo na memória. Isso facilita a sincronização, que pode ser feita utilizando-se semáforos e monitores, mecanismos já utilizados em computadores seriais. Apesar do desenvolvimento de aplicações ser mais trabalhoso, os multicomputadores ainda são mais fáceis de construir e são escaláveis[6].

Para solucionar a dificuldade no desenvolvimento de aplicações, o que se tem feito é construir sistemas híbridos, em que uma das abordagens utilizadas é conhecida como *Distributed Shared Memory* (DSM) (TANENBAUM, 1999). Nessa abordagem o sistema operacional simula um sistema de memória compartilhada que permite à aplicação acessar a memória de todo o computador paralelo como um único espaço de endereçamento.

Existem três tipos de multiprocessadores: ***Uniform Memory Access* (UMA)**, ***Non-Uniform Memory Access* (NUMA)** e ***Cache Only Memory Access* (COMA)**. Os multiprocessadores simétricos – *Symmetric MultiProcessor* (SMP) são exemplos de arquiteturas UMA e são bastante utilizados atualmente. Em arquiteturas UMA qualquer porção de memória têm o mesmo tempo de acesso para qualquer EP, enquanto em arquiteturas NUMA os tempos de acesso a cada região de memória difere de EP para EP (STALLINGS, 2003). Em NUMA cada EP possui uma memória local e o espaço de endereçamento compartilhado é formado pela combinação destas (QUINN, 1994). As arquiteturas NUMA podem possuir coerência de cache em hardware — *Cache Coherent NUMA* (CC-NUMA) — ou não possuir coerência de cache — *Non-Coherent NUMA* (NC-NUMA) (STALLINGS, 2003). Em arquiteturas COMA, a memória fisicamente remota é acessível apenas através de cache.

Multicomputadores podem ser divididos em duas categorias. A primeira é representada pelos **MPPs** (***Massively Parallel Processing***), consistindo de sistemas com **acoplamento forte**, formados por um grande número de (UCPs) interconectadas por uma rede proprietária de alta velocidade. A segunda categoria inclui **NOW** (***Network of Workstations***) e **COW** (***Cluster***

[5]Redes de interconexão conectam UCPs e módulos de memória a outras UCPs e a outros módulos de memória, permitindo a troca de dados. Detalhes podem ser obtidos em Tanenbaum (1999) e em Almasi e Gottlieb (1994).

[6]Segundo Hwang e Xu (1998), um sistema computacional (*hardware + software*) é escalável se ele pode ser aumentado para acomodar a demanda por funcionalidade e desempenho e/ou diminuído para reduzir custos.

of Workstations), sistemas com **acoplamento fraco**, constituídos de computadores autônomos interligados por uma rede de interconexão.

2.2 Sistemas Distribuídos

O desenvolvimento de microprocessadores mais potentes, menores e mais baratos, assim como o aumento da confiabilidade e da velocidade das redes de comunicação de dados mudaram o panorama da computação na década de 80. As organizações possuíam, geralmente, um único *mainframe*, ao qual vários usuários eram conectados por meio de terminais. Esse modelo foi substituído pelo uso de pequenos computadores interconectados por uma rede de comunicação de dados (HARIRI; PARASHAR, 2004). Desse novo modelo surgiram os chamados **Sistemas Distribuídos** (COULOURIS et al., 2001; TANENBAUM; STEEN, 2002). Surgidos da necessidade de compartilhar recursos de modo eficiente, os Sistemas Distribuídos tornaram-se uma alternativa, eficiente e de baixo custo, aos sistemas centralizados.

Os Sistemas Distribuídos apresentam algumas vantagens sobre os sistemas centralizados: melhor relação custo/desempenho; possibilidade de obter capacidade de processamento superior ao de um *mainframe*; confiabilidade (o sistema continua a funcionar quando um computador falha) e crescimento escalar (mais computadores podem ser adicionados ao sistema, conforme a necessidade) (TANENBAUM, 1995a). No entanto, vale ressaltar que, apesar dessas vantagens, a construção de um SD não é uma tarefa trivial. O software é bastante complexo e o projetista deve estar atento a este aspecto.

Segundo Coulouris et al. (2001), um SD é aquele no qual os componentes de hardware ou software localizados em computadores interligados em rede se comunicam e coordenam suas ações utilizando somente passagem de mensagens. Dessa definição destaca-se alguns aspectos importantes de um SD: **a)** pode estar separado por qualquer distância, por exemplo, uma mesma sala ou continentes; **b)** os eventos ocorrem concorrentemente; **c)** não há relógio global.

A Internet e as Intranets são exemplos de Sistemas Distribuídos. Entretanto, há uma grande dificuldade em classificar um sistema computacional como distribuído ou não. Tanenbaum (1997) comenta uma dessas dificuldades: "*existe na literatura uma terrível confusão entre redes de computadores e sistemas distribuídos*". Segundo Tanenbaum (1997), a principal diferença entre as redes e os sistemas distribuídos é a *transparência*. Ao contrário do que ocorre em uma rede de computadores, em que o usuário *explicitamente* invoca uma tarefa, num sistema distribuído os detalhes de um comando são responsabilidade do sistema, dando ao usuário a sensação de um sistema unificado.

A transparência e a autonomia dos computadores são as características chave da definição de Tanenbaum e Steen (2002). Para Mullender (1993) um SD é composto por vários computadores realizando tarefas em conjunto, destacando duas características básicas: interconexão entre os computadores e a existência de cooperação e coordenação para manter o estado

do sistema. Neste trabalho, a seguinte definição de sistema distribuído será considerada:

> *Um sistema distribuído é uma coleção de computadores autônomos interligados por uma rede de comunicação e que utilizam passagem de mensagens para cooperar e coordenar suas ações.*

Embora não exista na literatura especializada uma definição única e abrangente para Sistemas Distribuídos, a maioria dos autores concordam que para ser considerado um sistema distribuído um sistema computacional deve apresentar algumas características. Duas dentre elas são consenso na maioria dos trabalhos, a transparência e o compartilhamento de recursos. Essas e outras características são apresentadas a seguir.

2.2.1 Principais Características

A seguir são descritas as principais características dos Sistemas Distribuídos (TANENBAUM, 1995a; MULLENDER, 1993; COULOURIS et al., 2001).

Heterogeneidade

Embora não seja uma regra, Sistemas Distribuídos são, em geral, compostos por elementos heterogêneos (rede, hardware, sistema operacional, linguagens de programação). Elementos distintos não devem impossibilitar a comunicação ou gerar inconsistências. Na alocação de tarefas a heterogeneidade deve ser considerada: uma vez que existem nós com diferentes potências computacionais, é preciso escolher o que melhor se aplica às restrições de determinada tarefa.

Compartilhamento de recursos

Recurso é todo elemento que pode ser compartilhado através de uma rede, englobando desde componentes de hardware, como processadores e impressoras, até entidades de software, como arquivos e bases de dados. O compartilhamento de recursos merece atenção especial em sistemas distribuídos, visto que os recursos estão separados fisicamente. O propósito do compartilhamento pode ser reduzir custos, melhorar o desempenho, permitir trabalho cooperativo. Há recursos escassos que devem ser gerenciados pelo sistema de modo a garantir a melhor utilização, além de existirem recursos cujo acesso deve ser controlado.

Transparência

Coulouris et al. (2001, p. 23-24) Relaciona-se ao modo como o usuário e o programador de aplicações vêem o sistema. Se um sistema distribuído é visto como um sistema único, ocultando a existência de componentes independentes e separados fisicamente, então ele é transparente. Em ANSA (1989) e ISO (1992), oito formas de transparência são identificadas: acesso, localização, concorrência, replicação, falhas, mobilidade (ou migração), desempenho, propriedade escalar (*scaling*). Embora as oito formas de transparência sejam apenas diretrizes para a construção de sistemas distribuídos, duas delas são muito importantes: transparência de acesso

e de localização. Juntas recebem um nome especial: *transparência de rede* (COULOURIS et al., 2001).

Tratamento de falhas

Falhas podem ocorrer em qualquer sistema. O tratamento de falhas em sistemas distribuídos é particularmente complexo, pois, quando um componente do sistema falha, os demais continuam ativos, podendo gerar inconsistências. Quando um componente do sistema falha, somente o trabalho que estava usando este componente deve ser afetado, devendo ainda ser possível completar o trabalho em outro componente. Para lidar com as falhas algumas técnicas são empregadas: detecção de falhas, mascaramento de falhas (minimizar os efeitos de uma falha), recuperação de falhas (retornar o sistema para um estado correto antes da falha ocorrer) e redundância (replicar componentes, garantindo que, se um componente do sistema falhar, haverá outro, devidamente atualizado, para cumprir sua função).

Capacidade escalar (Scalability)

É a propriedade de manter-se estável quando a quantidade de recursos e de usuários aumenta significativamente (COULOURIS et al., 2001). Um exemplo é a Internet. Teoricamente, não há limite para o número de computadores e de usuários (e nem mesmo para a distância) de um sistema distribuído. Na prática, a obtenção da propriedade escalar é limitada por fatores como a rede. Várias técnicas têm sido empregadas com o objetivo de possibilitar maior expansão do sistema: utilização de algoritmos descentralizados, evitando gargalos; replicação e estruturação hierárquica de servidores e dados, utilização de *caches*.

Concorrência

Concorrência é uma característica natural dos Sistemas Distribuídos, que decorre da existência de múltiplos usuários e do compartilhamento de recursos. Em um mesmo instante, pode haver mais de um usuário tentando acessar o mesmo recurso. O processo (ou objeto) que representa um recurso compartilhado deve assegurar que o mesmo opere corretamente e mantenha seu estado consistente. Isto é obtido com o uso de operações sincronizadas.

Abertura

Um sistema aberto é aquele que pode se comunicar com outro sistema aberto usando regras padronizadas quanto a formato, conteúdo e significado das mensagens enviadas e recebidas (TANENBAUM, 1995a). Em um sistema aberto novos componentes de hardware e software podem ser adicionados, independentemente de fabricantes individuais.

2.2.2 Redes de Computadores

O aumento da velocidade e da confiabilidade das Redes de Computadores tornaram-nas o meio de comunicação mais utilizado para interligar os componentes de um sistema distri-

buído (COULOURIS et al., 1994). Em termos práticos, um SD é construído adicionando-se um conjunto de softwares a uma rede de computadores(TANENBAUM, 1997).

O desempenho de um algoritmo distribuído depende em grande parte da largura de banda e da latência da comunicação entre os nós[7]. Nos últimos anos a busca por alta largura de banda e baixa latência têm sido intensificada com avanços não somente em hardware, mas também com protocolos altamente eficientes que minimizam o *overhead* (DONGARRA et al., 2003; HARIRI; PARASHAR, 2004). Redes de alta velocidade, alcançando taxas de 100 Mbps e 1 Gbps e com uma boa relação custo/desempenho, já são comuns. Entretanto, quando se fala em Sistemas Distribuídos, velocidade não é suficiente sem confiabilidade. O software das Redes de Computadores é organizado como uma pilha de protocolos, o que ajuda a lidar com a complexidade da rede e com aspectos importantes, como a detecção de falhas e a garantia de entrega de um pacote.

Sistemas distribuídos podem ser construídos com uma grande variedade de Redes de Computadores, tanto em escala (redes locais – LANs, redes geograficamente distribuídas – WANs, inter-redes) quanto em tecnologia de transmissão (Ethernet, Token Ring, ATM). As tecnologias de interligação em redes são particularmente interessantes, pois possibilitam redes heterogêneas serem integradas. A tecnologia de interligação em redes TCP/IP é utilizada na Internet (COMER, 1995; KUROSE; ROSS, 2003) e é empregada na grande maioria dos sistemas distribuídos, operando de maneira uniforme, tanto em redes locais quanto em redes que atravessam países (COULOURIS et al., 2001). As Redes de Computadores são amplamente discutidas em Tanenbaum (1997), Comer (1995) e Kurose e Ross (2003).

2.2.3 Comunicação entre Processos

Os processos em um sistema distribuído, assim como nos sistemas centralizados, precisam se comunicar e sincronizar suas ações. Devido à ausência de memória compartilhada, mecanismos como semáforos e monitores não podem ser usados. Nos Sistemas Distribuídos a comunicação entre processos – *InterProcess Communication* (IPC) – é realizada através de passagem de mensagens, utilizando o suporte provido pela rede (TANENBAUM; STEEN, 2002).

Nas redes TCP/IP, dois protocolos da camada de transporte provêem tipos diferentes de serviço para passagem de mensagens. O *Transmission Control Protocol* (TCP), baseado num *stream* de bytes e orientado à conexão, garante a entrega da mensagem, mas gera maior overhead que o *User Datagram Protocol* (UDP), que se baseia em pacotes e não possui conexão. O UDP apresenta maior desempenho porém não há garantia de entrega. A arquitetura TCP/IP provê uma API que permite programar facilmente utilizando os protocolos TCP e UDP. Ela

[7] A Largura de Banda é uma métrica de desempenho de uma rede e se refere à quantidade de dados que pode ser enviada por unidade de tempo. Outra métrica importante é a Latência, definida como o tempo total necessário para transferir uma mensagem da origem até o destino. Diversos atrasos podem ocorrer na transmissão de uma mensagem: sobrecarga (*overhead*), propagação, roteamento e contenção (tráfego na rede).

fornece um conjunto de operações que possibilitam a comunicação e abstrações úteis como o *Socket*, uma referência a uma porta usada por um processo em um *host*, que pode ser usado para entregar uma mensagem a um processo.

Passagem de Mensagens

A passagem de mensagens é a forma mais simples de IPC (COULOURIS et al., 2001). A passagem de mensagens é suportada pelas operações *send* e *receive*, que envolvem um par de processos. Um processo envia uma mensagem para outro processo através da rede e o processo destinatário recebe a mensagem.

O mecanismo de comunicação pode ser síncrono ou assíncrono (COULOURIS et al., 2001). No mecanismo síncrono, os processos emissor e receptor sincronizam em cada mensagem e, neste caso, *send* e *receive* são bloqueantes. Assim, toda vez que um processo emissor envia uma mensagem, ele fica parado até que a mensagem seja recebida pelo processo receptor e toda vez que um processo receptor chama *receive*, ele bloqueia à espera da chegada da mensagem. Quando o mecanismo é assíncrono o processo emissor faz uma chamada *send* e continua sua execução normal, ou seja, ele não bloqueia à espera do envio da mensagem. O processo receptor no mecanismo assíncrono continua sua execução normal e quando a mensagem chega, ele pode sofrer uma interrupção ou pode, a qualquer instante de sua execução, verificar num *buffer* se a mensagem chegou. O mecanismo síncrono é mais simples no que se refere à programação, mas provoca perda de desempenho, pois impede que outras operações ocorram paralelamente ao envio e ao recebimento de uma mensagem.

Modelos de Comunicação

Nos Sistemas Distribuídos, os processos se comunicam utilizando, basicamente, dois modelos:

Comunicação cliente-servidor: baseia-se num protocolo Requisição–Resposta. Um processo cliente envia uma mensagem para um processo servidor solicitando um serviço. O processo servidor executa o serviço e envia uma mensagem, contendo o resultado, para o processo cliente. A resposta é a confirmação do recebimento e, possivelmente, da execução da solicitação. O protocolo pode ser ainda mais confiável com a utilização de uma confirmação de recebimento da resposta, enviada do cliente para o servidor.

Comunicação de grupo: é utilizada quando há necessidade de um processo comunicar-se com mais de um processo, em que o modelo cliente–servidor é inadequado. Nesse caso, uma operação *multicast* é mais adequada. Em operações *multicast*, uma mesma mensagem pode ser enviada para cada processo membro de um grupo. A comunicação em grupo pode possuir vários objetivos, entre eles: tolerância a falhas, melhoria de desempenho por meio de serviços replicados, atualização múltipla.

RPC, RMI e CORBA

A *Remote Procedure Call* (RPC) introduz uma camada de abstração que permite realizar uma chamada remota do mesmo modo que é feita uma chamada a um procedimento local, facilitando o trabalho de programação. Ele baseia-se no modelo cliente–servidor e utiliza passagem de mensagens para implementar a comunicação de fato. O uso de uma interface de serviço, contendo os métodos que podem ser chamados remotamente, é a chave do funcionamento da RPC. A partir dessa interface é possível gerar código *stub* (que encapsula as chamadas do cliente) e código *skeleton* (que encapsula a comunicação do servidor).

Remote Method Invocation (RMI) (DOWNING, 1998), uma tecnologia da plataforma Java, e *Common Object Request Broker Architecture* (CORBA) (SANTOS, 2001) são similares ao RPC, porém destinam-se ao uso com o paradigma de programação orientado a objetos. Elas realizam invocações a métodos em objetos contidos no espaço de endereçamento de um processo remoto.

2.3 Clusters e NOWs

É possível encontrar na literatura da área diferentes denominações para os sistemas distribuídos utilizados para a execução de aplicações paralelas, de acordo com características particulares: *Network of Workstations* (NOW), *Cluster of Workstations* (COW), *Network of Computers* (NoC), *Network of Machines* (NOM), *Cluster of SMPs* (CLUMP) (ZAKI et al., 1995; BUYYA, 1999; LASTOVETSKY, 2003). Entretanto, a grande maioria dos trabalhos utiliza apenas os termos NOW (YANG et al., 2003; KARATZA; HILZER, 2003; SHIM, 2004) e COW (CHIOLA; CIACCIO, 2000; ZHANG et al., 2000; ZHANG et al., 2001; KARATZA, 2002a).

Os termos NOWs e COWs são usados para designar uma rede convencional de computadores independentes. Esses computadores possuem seu próprio sistema operacional e podem variar desde estações de trabalho ou PCs até SMPs ou MPPs (BUYYA, 1999). O termo COW (ou, simplesmente, *Cluster*) sugere o uso de uma rede de alta velocidade e computadores homogêneos, formando uma arquitetura projetada especialmente para processamento paralelo. NOWs são usadas para designar um conjunto de computadores, geralmente heterogêneos, interligados por uma rede de propósito geral. Os computadores de uma NOW são usados tanto por aplicações paralelas quanto por usuários interativos.

A Computação Paralela e Distribuída explora a possibilidade de utilizar os computadores de NOWs e COWs como elementos de processamento de uma máquina paralela virtual. NOWs e clusters são as arquiteturas paralelas mais comuns em pequenas e também em grandes organizações (LASTOVETSKY, 2003). Seu uso é motivado por sua semelhança com os MPPs, incluindo o uso de passagem de mensagens como modelo de programação básico, para o qual existem ferramentas maduras e consolidadas (*vide* seção 2.4). Além disso, NOWs e clusters apresentam vantagens sobre os computadores paralelos especializados, tais como: maior

accessibilidade no que se refere ao custo e à alta disponibilidade; aumento da capacidade de processamento dos computadores pessoais; aumento da largura de banda e diminuição da latência das redes, sendo comum taxas de transferências na ordem de gigabits/segundo; facilidade de integrar novos computadores e de aumentar a capacidade de processamento do sistema.

Apesar das vantagens das NOWs e dos *clusters* sobre os computadores paralelos especializados, existem fatores que trazem novos problemas para o software paralelo. Um desses fatores é a interatividade dos usuários. É preciso lidar com cargas de trabalho externas às aplicações paralelas. Os usuários das estações de trabalho não podem ter suas atividades deterioradas em função da carga de trabalho paralela. Ainda não está claro como o escalonador deve lidar nesse caso. O escalonamento em NOWs será discutido no próximo capítulo.

A heterogeneidade, que por um lado pode ser vista como uma vantagem, por outro é o maior problema para as aplicações paralelas. É possível tirar proveito da heterogeneidade alocando o conjunto de computadores mais adequado às necessidades computacionais de uma aplicação paralela. Além disso, os processos possuem demandas computacionais diferentes, sendo interessante alocar os melhores computadores para os processos que demandam maior potência computacional.

Entretanto, é preciso lidar com as diversas heterogeneidades possíveis. Essa heterogeneidade surge em níveis configuracionais e arquiteturais. Uma NOW pode ser construída apenas com estações de trabalho ou pode conter, também, outras arquiteturas, como SMPs e MPPs. Ainda que seja construída com uma única arquitetura os nós podem ter potência computacional diferente e/ou diferentes quantidades de memória. Além disso, os nós podem estar rodando sistemas operacionais distintos (Unix, Windows), possuir processadores com diferentes arquiteturas (RISC, CISC) e, conseqüentemente, diferentes formatos de dados.

A rede também pode ser heterogênea, contendo diversas tecnologias de transmissão com diferentes largura de banda e latência. Alguns nós podem ser interligados, por exemplo, à taxa de transferência de 1 Gbps e outros por taxas de apenas 100 ou 10 Mbps.

A heterogeneidade torna o desenvolvimento de software paralelo ainda mais complexo. Existem ferramentas que agregam em suas funcionalidades o tratamento da heterogeneidade. Elas permitem a distribuição dos processos e o gerenciamento de recursos de modo transparente, liberando o programador dos detalhes da comunicação. Algumas dessas ferramentas são discutidas na seção seguinte.

2.4 Ferramentas para Computação Paralela Distribuída

A programação paralela em sistemas paralelos e distribuídos exige que toda a cooperação entre os computadores ocorra por meio de troca de mensagens. Isso pode ser feito manualmente, utilizando *sockets* ou através de ambientes de passagem de mensagens.

Os ambientes de passagem de mensagens implementam diretamente o modelo de pro-

gramação paralela por troca de mensagens, permitindo aos programadores implementar programas paralelos sem se preocupar diretamente com a comunicação entre os processos. Os ambientes de passagem de mensagens são extensões de linguagens seqüenciais, implementadas através de bibliotecas de passagem de mensagens. Esses ambientes provêem os recursos necessários à programação paralela, como criação, comunicação, sincronização de processos.

Existem diversos ambientes de passagens de mensagens, que diferem em detalhes de implementação do mesmo paradigma de passagem de mensagens. Os ambientes de passagem de mensagens mais populares são MPI e PVM. A absoluta maioria dos programas que utilizam passagem de mensagens é escrita utilizando um desses ambientes (LASTOVETSKY, 2003).

2.4.1 Parallel Virtual Machine

Parallel Virtual Machine (PVM) faz com que, na visão do usuário e do programador, um conjunto de computadores heterogêneos sejam uma única máquina paralela virtual. O PVM é composto por um conjunto de bibliotecas e ferramentas que permitem o desenvolvimento eficiente de programas paralelos e a manipulação transparente de mensagens, conversão de dados e escalonamento de tarefas através de uma rede de computadores com arquiteturas imcompatíveis (GEIST et al., 1994).

O PVM foi desenvolvido inicialmente para estações de trabalho Unix, mas atualmente existem implementações para uma variedade de plataformas. A primeira versão do PVM foi desenvolvida no *Oak Ridge National Laboratory*. A versão 2 foi escrita na Universidade do Tennessee e distribuída em março de 1991. Depois de um grande número de alterações, em 1993 foi completada a versão 3. As principais características do PVM são (GEIST et al., 1994):

- coleção de máquinas (host pool) configurada pelo usuário;
- transparência de acesso ao hardware;
- computação baseada em processos
- paradigma de passagem de mensagens explícito;
- utilização de ambientes heterogêneos;
- suporte a multiprocessadores;

O modelo de computação do PVM é baseado na existência de várias tarefas em uma mesma aplicação. O PVM suporta tanto o paralelismo de dados quanto o paralelismo funcional. Para que o programador possa introduzir paralelismo no código, possibilitando a cooperação entre as tarefas da aplicação, o PVM provê uma biblioteca de rotinas. Essa biblioteca provê primitivas que o programador pode usar para passar mensagens, criar processos, coordenar tarefas e modificar a máquina virtual.

O PVM possui também um processo *daemon* que executa em cada computador que compõe a máquina virtual, denominado *pvmd3* ou *pvmd*. Os *daemons* utilizam algoritmos

distribuídos para inicializar e coordenar a operação da máquina virtual, incluindo o roteamento de mensagens (SUNDERAM; GEIST, 1999).

2.4.2 Message Passing Interface

Message Passing Interface (MPI) é um padrão "*de facto*" para a implementação de ambientes de passagem de mensagem. Uma variedade de bibliotecas de passagem de mensagens foram desenvolvidas desde a década de 80. Elas diferiam umas das outras, tornando difícil construir programas paralelos portáveis. Em 1992, foi formado o MPI Forum com o objetivo de estabelecer uma interface padrão para a implementação de ambientes de passagem de mensagens (SNIR et al., 1996).

Devido ao limite de tempo imposto à formulação do padrão 1.0 do MPI, em 1994 definiu-se um conjunto básico de rotinas relacionadas à comunicação ponto-a-ponto e em grupo. Rotinas e funcionalidades extras seriam adicionadas partir da versão 2.0 (MPI Forum, 2005a; MPI Forum, 2005b). O MPI 1.0 (SNIR et al., 1996) define um conjunto de rotinas, que oferecem os seguintes serviços: comunicação ponto-a-ponto, comunicação em grupo, suporte a grupos de processos, suporte a contextos de comunicação, suporte a topologia de processos.

O grande número de implementações MPI demonstraram uma significativa aceitação do padrão MPI 1.1 (SOUZA, 1997). O MPI-2 foi concluído em 1997. Ele apresenta diversas correções e esclarecimentos à versão 1.1, além de definir novos conjuntos de rotinas, dentre os quais incluem-se (MPI Forum, 2005a; MPI Forum, 2005b): gerenciamento de processos, entrada e saída paralelas.

Várias implementações do MPI, tanto comerciais como de domínio público, estão disponíveis. As funções básicas da biblioteca de troca de mensagens têm diversas variações dentre as diferentes implementações existentes. As implementações de domínio público mais comuns são LAM/MPI (BURNS et al., 1994; LAM Team, 2004) e MPICH (NEVIN, 1996).

2.5 Grids Computacionais

Grids Computacionais, ou simplesmente *grids*, são a mais nova e promissora infraestrutura para a Computação Paralela Distribuída (ABBAS; ABBAS, 2003; BERMAN et al., 2003). O termo "*grid*" surgiu da analogia com a rede elétrica: os consumidores simplesmente usam a energia sem conhecimento de como e onde ela é gerada. *Grids* serão capazes de prover um serviço semelhante: o usuário conecta-se ao grid e se torna capaz de resolver seus problemas sem qualquer conhecimento dos recursos envolvidos (HARIRI; PARASHAR, 2004).

Não há uma definição precisa de grids, globalmente aceita e que não se sobreponha com tecnologias relacionadas (MINOLI, 2004). Segundo Foster (2002) três pontos definem um grid:

1. Coordena recursos que não estão sujeitos a controle centralizado
2. Usa interfaces e protocolos de propósito geral, abertos e padronizados

3. Provê qualidades de serviço não triviais

Muitos pesquisadores veêm *Grids* Computacionais como uma evolução de tecnologias familiares como computação distribuída, web[8], *Peer-to-Peer* (P2P), *Application Service Provider* (ASP).

Os *grids* consistem no compartilhamento de recursos computacionais (dados, capacidade de processamento, armazenamento), pertencentes a diferentes organizações e geograficamente separados. O compartilhamento se dá através de redes de interconexão e de softwares de monitoramento e controle de informações. Os recursos podem ser bastante heterogêneos e por estarem geograficamente distribuídos, geralmente, a Internet é usada. A maior dificuldade no uso de *grids* surge pelo fato dos recursos pertencerem a diferentes organizações. Assim, é preciso haver políticas bem definidas para se administrar o uso desses recursos. Os *grids* devem ser seguros, consistentes, pervasivos e possuir um baixo custo de acesso.

A construção de *grids* possui diversas motivações. Uma delas é a confiabilidade, obtida devido à existência de múltiplos recursos. Assim, se um componente falha outro pode substituí-lo sem maiores prejuízos para a aplicação. Também relacionada à existência de múltiplos recursos, outra motivação (talvez a principal) é a busca por maior potência computacional. Utilizando *grids* é possível compartilhar um imenso número de computadores para a execução de aplicações paralelas, tirando proveito da existência de muitos recursos computacionais sub-utilizados em diferentes organizações. Esses recursos, quando integrados, oferecem alta capacidade de processamento. Outros atrativos são: o compartilhamento de caros supercomputadores, como MPPs, entre diversas instituições (REIS, 2005); o acesso transparente a recursos remotos, como bases de dados e softwares; a computação em tempo real e sob-demanda.

Apesar dos grandes benefícios que um *grid* traz, é preciso estar atento às suas limitações. A tabela 2.2 (CIRNE, 2002) apresenta as principais características dos sistemas utilizados para a execução de aplicações paralelas. Os *grids* possuem características fortemente distintas dos demais sistemas paralelos. Assim, um algoritmo proposto para outras arquiteturas tem sua eficiência reduzida em *grids*.

Nem toda aplicação pode ser paralelizada para execução em *grids*. Algumas aplicações requerem o desenvolvimento de novos algoritmos, o que, geralmente, representa uma tarefa onerosa a fim de obter um tempo de resposta satisfatório (BERTIS, 2002). Quando se trata de *grids*, alguns princípios básicos devem ser observados (BAKER et al., 2002):

- *Grids* envolvem recursos **heterogêneos**, abrangendo um grande número de tecnologias.
- Os recursos estão geograficamente distribuídos entre **múltiplos domínios administrativos** e são propriedade de organizações distintas, que têm **autonomia** sobre tais recursos.

[8]World Wide Web – um sistema de informação global baseado em hipertexto que usa a Internet como mecanismo de transporte para intercambiar informação textual, imagens, vídeo, áudio.

Tabela 2.2: Resumo das características típicas de diferentes sistemas paralelos (CIRNE, 2002)

	SMPs	MPPs	*Clusters*[a]	*Grids*
Conectividade	excelente	muito boa	boa	média/ruim
Heterogeneidade	nula	baixa	média	alta
Compartilhado	não	não	sim	sim
Imagem	única	comum	comum	múltipla
Escala	10	1.000	1.000	100.000

[a]Na tabela original, em Cirne (2002), esta coluna é identificada por NOWs, mas as características descritas se aplicam melhor a *Clusters*.

- *Grids* devem ser **escaláveis**. Um grid pode crescer e potenciais problemas de desempenho podem surgir. Aplicações devem ser descentralizadas e projetadas para alta latência e baixa largura de banda.
- *Grids* devem ser **dinâmicos** e **adaptáveis**. Esses sistemas são bastante suscetíveis a falhas e devem ter meios de se recuperar delas.

Para prover as funcionalidades de um *grid* e lidar com as características citadas, os grids necessitam de alguns serviços básicos, como gerenciamento de recursos, autenticação, comunicação e transferência de dados eficiente e confiável, monitoração do estado dos recursos do grid, escalonamento (BERTIS, 2002). Assim, os componentes de um grid devem ser bem estruturados. Segundo Baker et al. (2002), quatro componentes são necessários para formar um grid:

- **Elementos físicos** – recursos globalmente acessíveis através da Internet. Exemplos são computadores (PCs, MPPs, SMPs); redes; sistemas operacionais de cada computador (Unix, Linux, Windows), incluindo suas bibliotecas e protocolos de rede; dispositivos de armazenamento; bases de dados; instrumentos científicos, como telescópio, sensores.
- ***Middleware* núcleo** – oferece os principais serviços do *grid*: gerenciamento de processos remotos; co-alocação de recursos; acesso aos dispositivos de armazenamento; descoberta e registro de informações; segurança; qualidade de serviço, como reserva de recursos.
- ***Middleware* de nível de usuário** – inclui ambientes de desenvolvimento, ferramentas de programação, *brokers* para a gerência dos recursos, escalonamento.
- **Aplicações e portais** – Aplicações de grid são desenvolvidas usando, tipicamente, os recursos providos pelo ambiente (linguagens e utilitários, como MPI, potência computacional, instrumentos científicos. Os portais oferecem serviços através de uma interface Web, em que os usuários podem submeter e coletar resultados das suas aplicações em recursos remotos.

A infra-estrutura de um *grid* é uma combinação de capacidades e recursos identificados para um problema e ambiente específicos (JOSEPH; FELLENSTEIN, 2003). Essa infra-estrutura pode mudar conforme as funcionalidades que se deseja prover num *grid*. Entretanto, essas diferenças não devem ser sinônimo de imcompatibilidades.

Devido à expectativa gerada em torno dos *grids*, muitos dos chamados Sistemas para Computação em Grid, pacotes de software que implementam os serviços necessários ao funcionamento de um *grid* têm sido propostos. Entretanto, eles têm sido criados sem padronização, retardando o crescimento e a consolidação dos *grids*. Para estabelecer padrões foi criado o Global Grid Forum (GGF). O GGF especificou a OGSI (*Open Grid Services Infrastructure*) e a OGSA (*Open Grid Service Architecture*), que estabelecem a natureza dos serviços que respondem às mensagens dos protocolos presentes em um grid (REIS, 2005).

O projeto Globus (FOSTER, 2005), é, provavelmente, o Sistema de Computação em Grid mais conhecido e com maior aceitação por parte dos administradores. Ele provê um conjunto de softwares que possibilita às aplicações manipular recursos computacionais heterogêneos distribuídos como uma única máquina virtual. Ele provê os serviços básicos e as capacidades requeridas para construir um Grid Computacional: segurança, localização e gerenciamento de recursos e comunicação (BAKER et al., 2002). O Globus permite a adoção de diferentes escalonadores, possuindo interfaces com o Condor, LoadLeveler, PBS (*vide* seção 3.3) (REIS, 2005).

2.6 Considerações Finais

Apesar do constante aumento de desempenho dos computadores seqüenciais baseados no modelo de *von Neuman*, a Computação Paralela já é uma necessidade real nos mais variados setores da sociedade. No entanto, os altos custos dificultam a aquisição e a utilização de computadores paralelos. Essa situação tem levado à busca de modos alternativos para prover alto desempenho a um custo menor. Os Sistemas Distribuídos surgem então como uma alternativa viável, de baixo custo e alta flexibilidade.

A arquitetura sobre a qual um sistema distribuído é construído se assemelha em muitos aspectos a alguns dos computadores paralelos mais utilizados atualmente, os MPPs. É possível visualizar um Sistema Distribuído, representado pelas NOWs e COWs, como uma máquina paralela virtual. A utilização de ambientes de passagem de mensagens e o aumento da confiabilidade e da eficiência dos Sistemas Distribuídos têm tornado a Computação Paralela e Distribuída bastante comum.

Entretanto, a Computação Paralela Distribuída traz novas complexidades, sendo ainda alvo de intensa pesquisa. Muitos aspectos antes utilizados em Computação Paralela se mostram inadequados para um ambiente com as características dos Sistemas Distribuídos. Um ponto crítico é o escalonamento. O desempenho de uma aplicação paralela depende de um bom escalonamento, o que não é fácil de se obter quando é preciso lidar com a heterogeneidade das NOWs e COWs, com cargas de trabalho externas às aplicações paralelas e com uma rede de comunicação relativamente lenta.

Capítulo

3

Escalonamento de Aplicações Paralelas em Sistemas Distribuídos

O escalonamento refere-se à atividade de alocar os recursos disponíveis, tal como elementos de processamento, disco e rede, entre as aplicações e entre as tarefas que compõem cada aplicação. Em qualquer sistema, um bom escalonamento, baseado nos objetivos propostos, é determinante para obter um bom desempenho (CASAVANT; KUHL, 1988).

O escalonamento de aplicações paralelas (*parallel job scheduling*) tem sido amplamente estudado na literatura de Computação Paralela e Distribuída (FEITELSON, 1999; SOUZA, 2000; ZHANG et al., 2001; ARNDT et al., 2000; CHOE; PARK, 2002; KARATZA, 2002a; KARATZA, 2002b; KARATZA; HILZER, 2003; KARATZA, 2003; FRACHTENBERG et al., 2003; SHIM, 2004; ISHII et al., 2005). Enquanto em computadores verdadeiramente paralelos as pesquisas relacionadas ao escalonamento já estão consideravelmente consolidadas (ESHAGHIAN; WU, 1997), em sistemas computacionais distribuídos há uma intensa atividade de pesquisa, buscando formas eficientes de alocar aplicações paralelas em NOWs e COWs. Muitos conceitos e técnicas utilizados para o escalonamento de aplicações em computadores paralelos não são adequados quando se trata de um sistema distribuído, havendo a necessidade de adaptar essas técnicas ou criar novas técnicas que garantam um bom desempenho.

Em Computação Paralela Distribuída, o escalonamento é um fator crítico para alcançar eficiência (KARATZA, 2000). Sistemas distribuídos oferecem considerável potência computacional. Entretanto, fatores como a característica de arquitetura de propósito geral das NOWs, a heterogeneidade e os atrasos na rede de comunicação dificultam as decisões do escalonamento e interferem no desempenho. Além disso, o escalonador, software responsável pelo escalona-

mento, deve lidar com objetivos conflitantes propostos para o escalonamento.

Este capítulo destina-se à abordagem dos principais conceitos e características relacionados ao escalonamento de aplicações paralelas em sistemas distribuídos, como NOWs e COWs. Embora o enfoque seja dado à Computação Paralela Distribuída, o assunto aqui tratado se aplica também às arquiteturas verdadeiramente paralelas.

3.1 Terminologia

A terminologia da área de escalonamento é bastante inconsistente. Existe na literatura um desacordo em relação aos termos usados e, conseqüentemente, uma grande dificuldade em relacionar os vários trabalhos desenvolvidos. O trabalho de Souza (2000) trata em detalhes os vários problemas existentes com a terminologia da área.

A fim de esclarecer os principais termos utilizados no escalonamento e ao longo deste trabalho, esta seção é destinada à terminologia da área.

Processos, programas, tarefas, threads, aplicações e jobs

Um **processo** é um **programa** em execução, contendo, além de um conjunto de instruções, uma entrada, uma saída e um estado (pilha, contador de programa e demais informações necessárias à execução do programa) (TANENBAUM, 2001). O termo **tarefa** é empregado como sinônimo de processo. ***Threads*** ou *lightweight processes* são, segundo Tanenbaum (1995b), linhas de execução (concorrentes) que podem existir em um mesmo processo. Cada *thread* executa de forma seqüencial, similarmente a um processo, mas compartilha com outras *threads* o mesmo espaço de endereçamento de memória do processo a que pertence. Isso facilita as trocas de contexto. Os termos **aplicação** e ***job*** designam o mesmo elemento: a maior "entidade" de execução. Uma aplicação pode conter apenas um processo (quando é uma aplicação seqüencial) ou vários processos que cooperam para a resolução de um problema (no caso de uma aplicação concorrente) (SOUZA, 2000).

Políticas, estratégias, mecanismos e algoritmos de escalonamento

Política de escalonamento e **estratégia de escalonamento** tem o mesmo sentido. Ambos os termos são usados para designar as decisões que devem ser tomadas em um escalonamento. Um exemplo de política ou estratégia de escalonamento é a decisão de quando e como serão coletadas informações a respeito da carga do sistema. Os **mecanismos de escalonamento** são os comandos e/ou instruções que, atuando como "ferramentas", desempenham as atividades básicas para a realização do escalonamento. As instruções que coletam informações sobre a carga de um sistema são exemplos de mecanismos de escalonamento (SOUZA, 2000). O termo **algoritmo de escalonamento** será utilizado neste trabalho num sentido mais amplo. O algoritmo de escalonamento inclui políticas e mecanismos de escalonamento.

Escalonamento de processos, escalonamento de aplicações paralelas, balanceamento de carga, alocação de recursos

O maior problema de se falar sobre escalonamento é que este termo tem diferentes significados para diferentes pessoas. Um erro bastante comum é o uso do termo **balanceamento de cargas** como sinônimo de escalonamento realizado em tempo de execução da aplicação. Balanceamento de cargas é um dos possíveis objetivos do escalonamento (SOUZA, 2000). Um termo comumente empregado como sinônimo de escalonamento é **alocação de recursos**. Segundo Casavant e Kuhl (1988) os termos escalonamento e alocação de recursos designam a mesma atividade sob pontos de vista diferentes. Enquanto o termo **escalonamento** aborda a questão em termos de processos ou aplicações (os consumidores dos recursos) a **alocação de recursos** leva em consideração os elementos de processamento (os recursos). Os termos **escalonamento de processos** e **escalonamento de aplicações paralelas** são sinônimos da mesma atividade, dando ênfase ao que está sendo escalonado. (FEITELSON; RUDOLPH, 1995).

3.2 Escalonamento

O escalonamento é definido por Casavant e Kuhl (1988) como um "*recurso para o gerenciamento de recursos*", dividido em três componentes básicos:

- **Consumidores** (aplicações e processos, que compõem cada aplicação)
- **Recursos** (elementos de processamento, memória, rede de comunicação, etc.)
- **Políticas de escalonamento**

O escalonador é responsável pelo gerenciamento dos recursos. Os consumidores obtêm acesso aos recursos através do escalonador, que, utilizando as regras de uma política de escalonamento, atribui os recursos disponíveis entre os consumidores. Essas regras são definidas com base nos objetivos propostos para o escalonamento. Assim, as decisões do escalonador são afetadas pelos objetivos propostos. Possíveis objetivos são: diminuir o tempo médio de resposta, diminuir atrasos na comunicação, maximizar a utilização dos recursos disponíveis, balancear a carga entre os EPs. Objetivos do escalonamento são discutidos na seção 3.2.5.

O escalonador pode ser implementado na própria aplicação, no nível do sistema operacional ou em um aplicativo desenvolvido para este fim. Em Sistemas Distribuídos o escalonamento pode ser realizado em duas etapas. Na primeira etapa, o escalonador decide, entre as aplicações submetidas ao sistema, qual o conjunto de aplicações terá acesso aos recursos computacionais, qual a localização desses recursos e a quantidade destes que será disponibilizada para cada aplicação. Na segunda etapa, as tarefas que compõem a aplicação são mapeadas localmente nos EPs.

Diferentes técnicas e metodologias foram desenvolvidas para o escalonamento. Com o objetivo de organizar o conhecimento adquirido na área de escalonamento e facilitar a comparação entre os trabalhos desenvolvidos, várias classificações foram propostas (SOUZA, 2000).

3.2.1 Taxonomia para o escalonamento

Casavant e Kuhl (1988) propuseram uma taxonomia para o escalonamento em sistemas distribuídos de propósito geral, cujo foco é o escalonamento realizado com vários processadores, denominado escalonamento global (*vide* seção 3.2.3). Essa taxonomia merece destaque por ser uma das mais completas, além der ser bastante utilizada e referenciada. Em virtude dessas características e por ser bastante apropriada, essa taxonomia será empregada para classificar as estratégias de escalonamento discutidas neste trabalho.

A taxonomia de Casavant e Kuhl (1988) divide-se em **classificação hierárquica**, conforme apresentado na figura 3.1, e **classificação plana**, na qual os termos relacionam-se de modo independente. A primeira divisão da classificação hierárquica é o escalonamento **local** ou **global**. No escalonamento local os processos são atribuídos, segundo uma política de compartilhamento de tempo, a um único processador. O escalonamento global refere-se à atividade de decidir *onde* os processos que compõem um *job* serão alocados em um sistema com vários elementos de processamento.

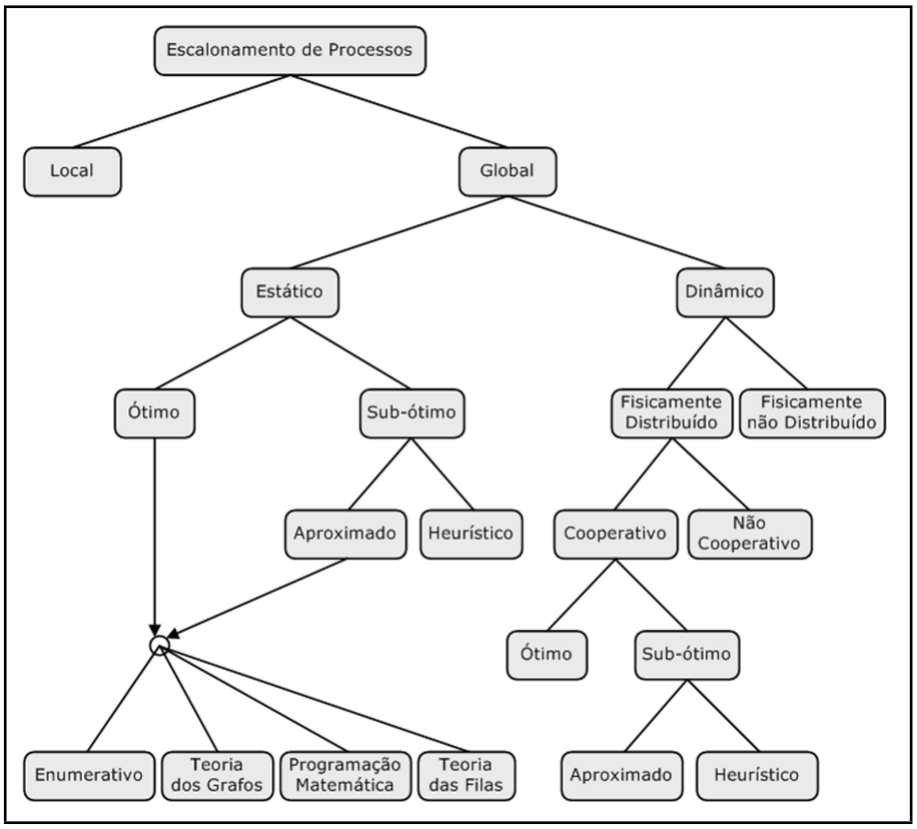


Figura 3.1: Taxonomia Hierárquica de Casavant e Kuhl (1988)

Essa taxonomia divide o escalonamento global em **estático** e **dinâmico**, de acordo com o momento em que as decisões do escalonador são tomadas. No escalonamento estático, as decisões são tomadas no momento da compilação da aplicação, sendo necessário que as informações sobre o sistema estejam disponíveis. É comum a modelagem da aplicação através de **grafos**,

onde os processos são representados por nós e as dependências (relacionadas à comunicação) entre os processos são representados pelos arestas. É comum também utilizar **Programação Matemática** e **Teoria das Filas** para modelar as aplicações. Para resolver um modelo são empregados algoritmos que buscam uma solução **ótima** (o que é um problema NP-completo) ou, quando algumas informações não estão disponíveis, **sub-ótima**. As soluções sub-ótimas empregam o uso de **Heurísticas** ou resultados **aproximados** na busca de uma solução aceitável para o problema.

No escalonamento dinâmico, assume-se que muito pouca informação a respeito dos recursos necessários às aplicações é conhecida *a priori*. Assume-se também que o ambiente em que a aplicação será executada é desconhecido ou as características do ambiente podem mudar dinamicamente. As informações sobre o estado do sistema são coletadas, em tempo de execução, para auxiliar nas decisões do escalonamento, dando maior flexibilidade e aumentando o desempenho global.

As decisões do escalonamento dinâmico podem ser **fisicamente distribuídas** entre diversos EPs ou **fisicamente não distribuídas** (quando as decisões são realizadas em um único EPs). Quando as decisões são **fisicamente distribuídas**, o escalonamento pode ser **cooperativo** ou não **cooperativo**. No escalonamento cooperativo os escalonadores trabalham cooperativamente, interagindo durante a alocação de recursos, na busca da melhor alocação dos recursos do sistema. No escalonamento não cooperativo os escalonadores trabalham isoladamente, tendo autonomia para decidir como alocar seus próprios recursos. No caso do escalonamento cooperativo, soluções **ótimas** e **sub-ótimas** podem ser ser empregadas. Como no escalonamento estático, as soluções sub-ótimas podem ser obtidas através de resultados **aproximados** ou utilizando **heurísticas**.

A classificação plana da taxonomia de Casavant e Kuhl (1988) engloba características do escalonamento que não têm relação de hierarquia e são, portanto, independentes, como **balanceamento de carga**, **atribuição inicial**, **reatribuição dinâmica** e **escalonamento adaptativo**.

O **balanceamento de carga (*load balancing*)** e o **compartilhamento de carga (*load sharing*)** são mecanismos utilizados para a distribuição de carga. O compartilhamento de carga busca evitar que EPs fiquem sobrecarregados enquanto outros EPs possam estar disponíveis. O balanceamento de carga estende o mecanismo de compartilhamento de carga, buscando manter a mesma carga em cada elemento de processamento (SHIVARATRI et al., 1992).

O escalonamento com **reatribuição dinâmica** (CASAVANT; KUHL, 1988) é comumente utilizado para se obter balanceamento de cargas. Na reatribuição dinâmica, um processo, que já foi atribuído a um EP e, possivelmente, já iniciou sua execução, pode ser migrado para outro EP. O caso contrário à reatribuição dinâmica é denominado **atribuição inicial**, em que um processo inicia e termina sua execução no mesmo EP para o qual foi atribuído (mesmo sob altas variações de carga). A **migração** de processos na reatribuição dinâmica é uma ati-

vidade que gera muita sobrecarga, principalmente em sistemas distribuídos, pois, além de ser necessário salvar todo o contexto do processo, é necessário transferir o processo pela rede de comunicação.

Os mecanismos citados buscam aumentar o desempenho, assim como no **escalonamento adaptativo**, em que o comportamento das políticas de escalonamento são alteradas em função do estado atual do sistema.

3.2.2 Escalonamento local

No escalonamento local os processos são atribuídos a um único elemento de processamento. O escalonamento local é comum em computadores pessoais com sistemas operacionais multitarefa. Em geral, esse tipo de escalonamento é realizado pelo sistema operacional e baseia-se no compartilhamento do elemento de processamento por meio de técnicas de compartilhamento de tempo (*time-sharing*).

Nas técnicas de *time-sharing* o tempo de uso de um EP é dividido em pequenas fatias de tempo (*time-slices*). Cada processo, em uma fila de processos aguardando pela utilização do EP, utiliza o EP durante um *time-slice*, liberando-o, em seguida, para uso por outro processo na fila. Caso o processo não tenha terminado sua execução neste *time-slice*, ele volta à fila (geralmente, ao final) para utilizar mais um *time-slice* e assim sucessivamente. O sistema operacional é responsável pela suspensão e ativação dos processos por meio de técnicas de preempção.

Através dessas técnicas, conhecidas também como multiprogramação, é possível otimizar o uso do processador, que não fica parado enquanto um processo está bloqueado[1]. Além disso, a utilização de *time-slices* pequenos favorece processos interativos, dando ao usuário a impressão de que vários processos rodam em paralelo, quando na verdade, devido à existência de apenas um processador, existe um pseudoparalelismo entre processos concorrentes.

Detalhes sobre o escalonamento local podem ser encontrados em (TANENBAUM, 2001).

3.2.3 Escalonamento global

No escalonamento global, caracterizado pela existência de mais do que um elemento de processamento, os processos de cada aplicação são distribuídos entre os EPs disponíveis. Várias características relacionadas ao escalonamento global foram discutidas na seção 3.2.1.

Os elementos de processamento de um sistema paralelo precisam ser compartilhados

[1]Utilizando técnicas de compartilhamento de tempo, um processo pode, de uma maneira geral, assumir três estados: pronto, bloqueado, executando. Um processo é dito executando quando está alocando o processador. Um processo executando pode ir para o estado estado pronto (seu *time-slice*, também denominado *quantum*, expirou e ainda não acabou sua execução) ou bloqueado. Um processo é bloqueado quando faz um pedido ao sistema operacional, E/S, por exemplo, e permanece nesse estado até que o pedido seja atendido. Quando o pedido é atendido o processo vai para o estado pronto. Um processo no estado pronto é aquele capaz de continuar sua execução, mas não pode alocar o processador porque outro processo está executando.

entre as aplicações que são submetidas ao sistema. Nesse sentido, as políticas de escalonamento podem ser organizadas através de duas dimensões: compartilhamento de tempo (*time sharing*) e compartilhamento de espaço (*space sharing*).

As políticas de compartilhamento de espaço dividem os EPs disponíveis em conjuntos disjuntos, denominados partições. Cada partição é atribuída a uma aplicação paralela, que, geralmente, ocupa a partição até que termine sua execução. Com relação à maneira como as partições são criadas e mantidas as políticas de compartilhamento de espaço são classificadas em fixas (são pré-estabelecidas durante a configuração do sistema e não são alteradas), adaptativas (o estado do sistema é considerado a fim de realizar o particionamento) e dinâmicas (considerando o estado do sistema, definem, sob demanda, o tamanho da partição, que pode ser alterado em tempo de execução) (NAIK et al., 1997).

A utilização de políticas de compartilhamento de espaço reduz o custo com troca de contextos, mas seu uso isolado não é adequado. Como o particionamento é realizado considerando o número de elementos de processamento e não a potência computacional de cada EP, as políticas de compartilhamento de espaço são pouco eficazes em sistemas heterogêneos. Além disso, é muito difícil para uma única aplicação utilizar eficientemente todos os EPs. A multiprogramação é necessária para melhorar os tempos de resposta do conjunto de aplicações submetidas ao sistema e melhorar a utilização dos recursos (MAJUMDAR; PARSONS, 2000).

A multiprogramação, no entanto, é um problema complexo para o escalonamento em sistemas distribuídos. A existência de múltiplas aplicações, com diferentes requisitos computacionais, exige o uso de boas técnicas de compartilhamento de tempo.

O compartilhamento de tempo em sistemas paralelos e distribuídos é uma extensão do compartilhamento de tempo usado no escalonamento local. O tempo de processamento dos diversos EPs pode ser compartilhado sob duas abordagens. Na primeira, cada processo em uma fila global obtêm acesso a um dos EPs por um intervalo de tempo. Quando esse tempo expira, o processo volta a fila para que seja escalonado novamente, possivelmente para outro EP. Essa abordagem apresenta dois problemas: o acesso a fila global pode causar disputa, tornando-se um gargalo e não é possível tirar proveito da utilização da memória principal e da *cache* de um EP (FEITELSON; RUDOLPH, 1998). Além disso, a utilização dessa abordagem requer modificações no sistema operacional.

A segunda abordagem tem sido utilizada nos sistemas distribuídos de propósito geral. Em NOWs e COWs o escalonamento de uma aplicação paralela é dividido em um etapa global e local. Na etapa global, em um dado momento, o escalonador decide, dentre as aplicações submetidas ao sistema, qual aplicação terá acesso aos elementos de processamento. As tarefas que compõem a aplicação são, então, mapeadas para os EPs disponíveis, através de processos e *threads*. Os processos (da mesma aplicação ou de aplicações diferentes) escalonados para um mesmo EP competem pela utilização do recurso. O sistema operacional local é responsável

pelo compartilhamento do tempo de processamento entre esses processos, tomando decisões isoladamente dos demais EPs.

Um problema dessa segunda abordagem, distribuída, é a diferença de carga de trabalho entre os EPs. É possível atribuir muita carga a determinados EPs e subutilizar outros. A diferença de carga surge também pela existência de usuários que geram cargas de trabalho externas às aplicações paralelas. Dessa forma, torna-se necessário utilizar políticas de escalonamento que tenham por objetivo o compartilhamento ou o balanceamento de carga. Uma forma de obter balanceamento de cargas é utilizar a migração de processos

Uma característica de suma importância em uma NOW é a interação entre as cargas de trabalho seqüencial e paralela. Em uma NOW, o que se tem buscado é não degradar o desempenho das aplicações seqüenciais e/ou interativas dos usuários, utilizando apenas EPs ociosos para executar aplicações paralelas. Quando o proprietário de uma estação de trabalho volta a utilizá-la duas estratégias podem ser empregadas. Uma mais agressiva, migra os processos das aplicações paralelas para outros EPs. A outra, mais conservadora, atribui aos processos paralelos uma prioridade menor do que os processos seqüenciais (SHIM, 2004).

Outro problema da segunda abordagem, as políticas de compartilhamento de tempo distribuídas, é a falta de sincronização entre os EPs, gerando um atraso considerável na execução das aplicações. O atraso ocorre porque um processo rodando em um EP pode querer se comunicar com um processo em outro EP que não está rodando. Assim, é preciso esperar até que o processo ocupe o processador. Estratégias de escalonamento cooperativo (*coscheduling*) tem sido estudadas para permitir que as tarefas de uma mesma aplicação sejam escalonadas em seus respectivos EPs ao mesmo tempo. Entretanto, essas estratégias requerem alguma forma de sincronização entre os escalonadores locais. Devido às características dos sistemas distribuídos, a sincronização explícita é pouco atrativa, levando à busca de estratégias que utilizem a sincronização implícita (ARPACI-DUSSEAU, 2001). Um exemplo de política de escalonamento cooperativo é o *gang scheduling* (KARATZA, 2002a).

Para que o escalonamento cumpra seus objetivos, apresentando um desempenho adequado, é necessário, além do conhecimento a respeito dos recursos computacionais do sistema, conhecimento a respeito da carga de trabalho e das aplicações paralelas. Em vários trabalhos o conhecimento sobre as aplicações paralelas tem sido utilizado para obter melhor desempenho. O trabalho de Senger (2005), desenvolvido no Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC), investiga a obtenção e utilização do conhecimento sobre as aplicações paralelas no escalonamento em sistemas computacionais distribuídos, através da definição de mecanismos eficientes para prover esse conhecimento ao escalonador.

Atualmente, grande parte das políticas de escalonamento têm levado em consideração as características da aplicação que está sendo executada, procurando tratar adequadamente aplicações com diferentes requisitos. Três classes principais de aplicações têm sido consideradas

no escalonamento de aplicações paralelas:

CPU–Bound – aplicações dessa classe possuem alta demanda por processamento e baixa comunicação entre os processos (ARAÚJO, 1999).

Memory–Bound – escalonar adequadamente aplicações dessa classe é importante uma vez que falhas de páginas[2] podem deteriorar consideravelmente o desempenho. Em geral, o que se busca ao escalonar aplicações *memory-bound* é evitar falhas de páginas, o que pode ser alcançado escalonando os processos para os EPs com maiores quantidades de memória (MAJUMDAR; PARSONS, 2000).

I/O–Bound – Essa classe engloba aplicações que realizam muita operação em dispositivos de entrada/saída. Aplicações *I/O–Bound* são também classificadas como *Disk–Bound* (quando fazem muito acesso a disco) e *Network–Bound* (quando efetuam muita comunicação, gerando alto tráfego na rede) (ISHII et al., 2005).

3.2.4 Políticas de Escalonamento

As políticas de escalonamento determinam quais, quando e como os mecanismos de escalonamento serão empregados para que o escalonamento seja efetuado. Em geral, são as políticas que determinam o desempenho do escalonamento.

Um algoritmo de escalonamento pode ser organizado como um conjunto de políticas de escalonamento, permitindo uma visão modular das diferentes atividades desempenhadas por ele. A quantidade e a função dessas políticas variam de acordo com a visão e o enfoque que se dá à área de escalonamento de processos (SOUZA, 2000).

Um enfoque bastante interessante é apresentado por (SHIVARATRI et al., 1992) e é aplicado a algoritmos de escalonamento dinâmico, que são baseados na distribuição da carga entre os diversos elementos de processamento utilizado migração de processos. Segundo Shivaratri et al. (1992), um algoritmo de escalonamento é composto por quatro políticas:

Política de Informação – é responsável por definir em que momento informações sobre o estado do sistema são necessárias, de que elementos de processamento essas informações devem ser coletadas, e que tipo de informação deve ser utilizada

Política de Seleção – relaciona-se à decisão de qual processo será transferido após a definição de qual elemento de processamento será um transmissor.

Política de Localização – responsável por definir um elemento de processamento parceiro para participar da transferência.

[2]Situação que ocorre em sistemas operacionais que utilizam memória virtual. Quando a memória principal é insuficiente, uma área do disco (memória secundária) é destinada a armazenar temporariamente dados de um programa que não estão sendo utilizados no momento. Quando o programa precisa desses dados ocorre uma falha de página. Então, o sistema operacional busca os dados no disco e coloca na memória principal.

Política de Transferência – determina a disponibilidade de um elemento de processamento para participar de uma transferência de processos, seja como transmissor ou como receptor.

A carga de um elemento de processamento é uma informação de grande importância, pois permite avaliar o quanto um EP está carregado. Essa informação permite dizer se o EP está apto a receber mais tarefas para executar ou se está sobrecarregado (podendo, até mesmo, ser necessário transferir tarefas para outro EP). De modo geral, a carga é medida por meio de índices de carga, que são valores inteiros não-negativos. O valor zero de um índice de carga indica que não há carga e, à medida que a carga aumenta, o valor de um índice de carga também aumenta. Índices baseados nas filas de acesso a algum recurso, como EP e memória; índices baseados na taxa de utilização de um recurso e índices baseados no tempo de resposta são alguns exemplos de índices de carga.

A obtenção e a utilização de índices de carga são tarefas complexas. A medição de um índice gera sobrecarga sobre o sistema, sendo necessário garantir que a medição interfira o mínimo possível no comportamento do sistema. A escolha de índices de carga ruins pode fornecer resultados imprecisos para aplicações que possuam requisitos bem definidos em termos de recursos (por exemplo, o mais importante para uma aplicação CPU-Bound é saber o quanto a CPU está ocupada). É possível combinar índices para obter uma informação mais precisa do estado do sistema, chegando a conclusões mais próximas da realidade (SOUZA, 2000).

Devido à heterogeneidade dos sistemas distribuídos, índices de carga absolutos podem levar a erros na utilização dos índices. Por exemplo um processador mais rápido que esteja mais carregado que outro mais lento pode ser uma melhor opção para receber uma nova tarefa. Assim, os índices de carga precisam ser normalizados de forma que a potência computacional dos elementos de processamento seja considerada. Índices de carga são discutidos com maiores detalhes em Santana e Zaluska (1988), Branco (2004), Voorsluys (2006).

Algumas políticas de escalonamento serão discutidas ao longo deste trabalho. Duas políticas utilizadas neste trabalho foram desenvolvidas no Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC). A DPWP (*Dynamical Policy Without Preemption*) é uma política voltada a aplicações *CPU–Bound*, enquanto a NBSP (*Network–Bound Scheduling Policy*), como o próprio nome diz, é voltada para aplicações *Network–Bound*. Ambas têm como objetivo melhorar o tempo de resposta das aplicações.

3.2.5 Objetivos e Desempenho do Escalonamento

Os objetivos propostos para um escalonamento são os principais elementos que devem ser traduzidos para uma política de escalonamento. Como é a política que toma as decisões do escalonamento de uma aplicação paralela, cabe a esta cumprir os objetivos propostos.

O objetivo primário do escalonamento de aplicações paralelas é garantir que as apli-

cações possam ser executadas e consigam terminar sua execução. Apesar desse objetivo ser bastante óbvio, ele deve ser sempre considerado, pois nenhum outro objetivo deve interferir na execução de uma aplicação (ou de seus processos). Os objetivos propostos para o escalonamento podem ser bastante variados. Os mais comumente empregados são objetivos ligados à utilização dos recursos e à satisfação do usuário, como:

Minimizar o tempo de resposta – o tempo de resposta é o tempo gasto desde a chegada de uma aplicação no sistema até o término de sua execução. Menor tempo de resposta significa que a aplicação foi escalonada e/ou foi executada mais rapidamente.

Maximizar o throughput – o *throughput* é o número de tarefas processadas por unidade de tempo. Esse objetivo está relacionado à exploração da capacidade do sistema. Quando o throughput é igual ao número de tarefas que chegam significa que o sistema não está saturado.

Maximizar a utilização dos recursos – esse objetivo representa a porcentagem da capacidade de um recurso, como um EP, que está sendo utilizada. O cumprimento desse objetivo caracteriza-se, principalmente, pelo esforço em não permitir que os recursos fiquem ociosos. Isso pode ser obtido com multiprogramação.

É importante notar que algumas vezes os objetivos podem ser contrastantes. Por exemplo, quando se procura maximizar a utilização do processador, o que geralmente se faz é escalonar processos grandes primeiro. Isso acaba por aumentar o tempo de resposta. Portanto, maximizar a utilização dos recursos e minimizar o tempo de resposta pode não ser possível.

Uma medida de desempenho útil para verificar a qualidade do escalonamento é verificar se uma política de desempenho cumpre os objetivos para a qual foi proposta. Os objetivos de uma política de escalonamento podem ser formalizadas através de métricas de desempenho, permitindo avaliar quantitativamente se os objetivos estão sendo atingidos (KRALLMANN et al., 1999; SOUZA, 2000). Medidas de desempenho são discutidas na seção 4.1.

3.3 Ambientes de escalonamento

O escalonamento em NOWs e COWs é, geralmente, realizado por um software que roda sobre o sistema operacional, como um *middleware* para as aplicações paralelas. É comum encontrar na literatura o termo *Job Management System* (JMS) (EL-GHAZAWI et al., 2004) como sinônimo desses ambientes de escalonamento.

Existem vários ambientes de escalonamento disponíveis. Grande parte desses ambientes são (ou possuem) versões de domínio público. Eles são otimizados com base no sistema computacional para o qual foram construídos, podendo, inclusive, ser parte integrante do sistema operacional (SOUZA, 2004). Os ambientes de escalonamento, implementados externamente ao sistema operacional, apresentam vantagens, como: controle e planejamento da utilização dos recursos por parte do administrador, adaptabilidade e portabilidade a sistemas heterogêneos.

Alguns dos ambientes de escalonamento mais conhecidos são discutidos a seguir. No final desta seção é apresentado um quadro comparativo entre os ambientes discutidos.

Condor

Condor (Condor Team, 2006) é um sistema de escalonamento *batch* para LANs. Os *jobs* executam em segundo plano, utilizando estações de trabalho ociosas. Quando o proprietário de uma estação de trabalho retorna, os processos paralelos são suspensos. Provido com suporte à migração e à *checkpoint*, os processos paralelos são migrados para outro EP se o usuário continuar a utilizar a estação de trabalho.

O Condor possui uma interface de gerenciamento denominada CARMI (*Condor Application Resource Management Interface*). Através dessa interface o usuário pode especificar a arquitetura do EP, a quantidade de memória, a quantidade de *swap* e o sistema operacional. No Condor um gerente centralizado é responsável pelo escalonamento e atribuição de tarefas aos EPs. Prioridades são implementadas através de um algoritmo *up-down*, que incrementa a prioridade de aplicações que estão a mais tempo na fila e reduz a prioridade de aplicações que utilizaram vários recursos em instantes de tempos passados.

Load Leveler

O LoadLeveler (Maui HPC Center, 2005) é um produto usado em estação de trabalho IBM baseado no Condor. No Load Leveler as políticas de escalonamento podem ser alteradas através de uma coleção de utilitários e através de uma interface gráfica.

LSF (Load Sharing Facility)

O LSF (ZHOU et al., 1993; XU, 2001) é baseado no compartilhamento de carga entre os EPs. Permite a execução remota de forma transparente e o escalonamento de aplicações paralelas em *clusters* heterogêneos de larga escala (FEITELSON, 1999). O usuário pode especificar características que são usadas pelo software escalonador durante a distribuição de tarefas. Alguns exemplos são: quantidade de memória principal, quantidade de espaço em disco, arquitetura dos EPs. O administrador do sistema pode inserir medidas de desempenho adicionais, a fim de tornar as decisões do escalonador mais flexíveis.

Sun Grid Engine

O Sun Grid Engine visa a utilização ótima dos recursos computacionais em ambientes distribuídos. Ele suporta várias plataformas, principalmente, sistemas UNIX. Esse ambiente possui um conjunto de filas e um escalonador mestre, que recebe as aplicações dos usuários. Os processos *daemons*, que executam em cada EP, buscam os EPs apropriados para as aplicações. Se os recursos que satisfaçam os requisitos da aplicação não estiverem disponíveis, o sistema aguarda a liberação desses recursos.

O Sun Grid Engine implementa migração de processos e *checkpointing*, o que possibilita a recuperação do trabalho em caso de falhas. Outro mecanismo de tolerância a falhas é a replicação do escalonador mestre. O ambiente possui suporte à customização, permitindo ao usuário submeter aplicações e selecionar arquitetura específica, quantidade de memória, sistema operacional. Além disso, prioridades podem ser solicitadas e o administrador pode configurar restrições de acesso aos usuários do ambiente. As políticas, assim como as filas e os períodos de verificação de carga dos EPs, podem ser configurados em tempo de execução, sem necessidade de interrupção na execução do ambiente.

DQS (Distributed Queueing System)

O ambiente DQS possui várias filas, nas quais os usuários submetem suas aplicações. Esse ambiente possui implementação para diversos sistemas operacionais. O DQS implementa tolerância a falhas, com a existência de um escalonador auxiliar, que assume as atividades do escalonador mestre em caso de falhas. A fim de realizar o escalonamento e a atribuição de tarefas o DQS se baseia na carga média do sistema, alocando aplicações e tarefas para os EPs ociosos. O usuário pode especificar os requisitos computacionais de sua aplicação, como quantidade de memória, arquitetura dos EPs e podem também especificar características da aplicação. O DQS pode ser configurado para aplicar restrições aos usuários.

NQE (Network Queuing Environment)

É formado por um processo mestre, que é responsável pelo escalonamento das aplicações, e de processos *daemon*, que coletam informações sobre a carga nos EPs para serem submetidas ao processo mestre. O usuário fornece as características das aplicações, que são inseridas em filas específicas de acordo com essas características. Recursos importantes para o escalonador podem ser configurados pelo administrador, assim como as políticas de escalonamento e o número de filas do ambiente podem ser alterados em tempo de execução.

PBS (Portable Batch System):

O PBS visa fornecer controle sobre a execução e o escalonamento de aplicações, tanto em uma rede de estações de trabalho, quanto em máquinas com processamento maciçamente paralelo (MPPs). Ele segue o modelo de filas múltiplas. O PBS utiliza políticas configuradas pelo usuário, que podem conter prioridades e informações sobre a aplicação para realizar o escalonamento.

AMIGO (dynAMical flexIble schedulinG envirOnmet):

O AMIGO (*dynAMical flexIble schedulinG envirOnment*) foi desenvolvido no Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC). Ele foi especificado e implementado através do trabalho de doutorado de Souza (2000) e dos trabalhos de mestrado

de Araújo (1999), Figueiredo (2000), Campos (2001) e Santos (2001).

Esse software foi desenvolvido com a finalidade de gerenciar o escalonamento de aplicações em sistemas computacionais heterogêneos e distribuídos, executando sistemas UNIX, de maneira flexível e dinâmica (SOUZA, 2000). O AMIGO é flexível porque possui a possibilidade de se alterar configurações (por exemplo, inserir ou remover políticas de escalonamento) de acordo com as necessidades do usuário. A parte dinâmica refere-se à adequação do ambiente em relação a novas situações/configurações em tempo de execução (ARAÚJO et al., 1999; SOUZA et al., 1999).

O AMIGO foi projetado com uma estrutura modular de duas camadas, independentes em tempo de execução. A camada superior é composta por uma interface gráfica, na qual o usuário tem acesso a detalhes do ambiente, como: seleção de políticas de escalonamento, inserção de informações sobre o *hardware*, definição de métricas para monitoramento da política de escalonamento (CAMPOS, 2001). Todas essas configurações são armazenadas em arquivos, que são utilizados pela camada inferior. Os arquivos de configuração são o único elo de ligação entre as camadas.

A camada inferior é composta por três módulos: o ambiente de passagem de mensagens, o AMIGO Daemom (AMIGOD) e as políticas de escalonamento. O ambiente de passagem de mensagens atua requisitando elementos de processamento, nos quais as aplicações paralelas devem ser executadas. O AMIGO pode ser acoplado a qualquer ambiente, desde que uma interface de comunicação tenha sido previamente construída. Atualmente, existem interfaces para PVM, MPI e CORBA.

O AMIGOD é um processo servidor distribuído, que executa em cada EP. Funciona como um elo entre a política e o ambiente de passagem de mensagens. Sua responsabilidade se resume a rotear e gerenciar as mensagens trocadas entre os módulos da camada inferior, iniciar e terminar a execução de políticas de escalonamento e manter informações sobre execução de outros AMIGODs. Não há uma hierarquia. Mesmo com a possibilidade maior complexidade e sobrecarga de comunicação, os AMIGODs são completamente distribuídos, possibilitando melhor tolerância a falhas e propriedade escalar. As políticas de escalonamento são responsáveis por verificar e repassar ao AMIGOD, quais são os elementos de processamento adequados para a execução das aplicações.

Utilizando o AMIGO, as aplicações paralelas não precisam ser modificadas ou adaptadas e o escalonamento é transparente, ocultando detalhes de implementação e das políticas usadas. O AMIGO não possui mecanismos de *checkpointing* e migração de processos.

Um comparativo entre os ambientes de escalonamento acima descritos é apresentado na tabela 3.1. São relacionadas algumas características consideradas importantes para os ambientes de escalonamento para NOWs e COWs. As características são classificadas em suportadas totalmente, parcialmente suportadas e não suportadas pelo ambiente de escalonamento. As

características citadas são: **heterogeneidade**, **passag. mensagens** (se o ambiente suporta ambientes de passagem de mensagens como PVM e MPI), **políticas dinâmicas** (possibilidade de reconfiguração em tempo de execução), **sem recompilação** (escalona sem a necessidade de recompilação do código da aplicação), ***checkpointing***, **migração**, **balanceamento de carga**, **gratuito** (se é ou possui uma versão de domínio público).

Tabela 3.1: Quadro comparativo dos ambientes de escalonamento (SENGER, 2005)

	Condor	Load Leveler	LSF	Sun Grid Engine	DQS	NQE	PBS	AMIGO
heterogeneidade	◆	◆	◆	◆	◆	◆	◆	◆
passag. mensagens	◇	◇	◆	◆	◇	⬙	◇	◆
políticas dinâmicas	◇	◆	◆	◆	◆	◆	◆	◆
sem recompilação	◇	⬙	◆	⬙	⬙	◆	◆	◆
checkpointing	◆	◆	◆	◆	◆	◇	◇	◇
migração	◆	◆	◆	◆	◇	◇	◇	◇
balanceamento de carga	◇	◆	◆	◆	◆	◆	◆	◆
gratuito	◆	◇	◇	◆	◆	◇	◆	◆

Legenda: ◆ totalmente suportado ⬙ parcialmente suportado ◇ não suportado

3.4 Escalonamento em grids

Em grids computacionais o escalonamento é, particularmente, desafiador. Embora as políticas de escalonamento propostas para uso em Clusters e NOWs possam ser usadas, elas tem sua aplicabilidade reduzida em grids, oferecendo desempenhos ruins nesses sistemas (BERMAN et al., 2003). A tabela 2.2, apresentada na seção 2.5 resume as características típicas dos principais sistemas utilizados para computação paralela de alto desempenho. Até mesmo entre grids e NOWs as diferenças são consideráveis.

Dessa forma, o que se tem buscado é o desenvolvimento de políticas de escalonamento dedicadas a esses sistemas. Políticas que contemplam o tratamento das características inerentes aos grids obtêm ganhos de desempenho significativos quando comparadas às políticas de escalonamento que não comtemplam (REIS, 2005). Algumas características dos grids dificultam as decisões do escalonador.

Nos grids, os recursos de hardware e software são bastante heterogêneos, apresentando grandes diferenças de desempenho, múltiplas arquiteturas e múltiplas imagens de sistema operacional. A grande quantidade de recursos é outro dificultador para a atividade de escalonamento. O gerenciamento de recursos envolve descobrir, monitorar e alocar recursos. Assim, a existência de muitos recursos pode tornar o escalonador um gargalo e dificultar manter consis-

tente as informações de estado dos recursos disponíveis.

Os recursos são compartilhados por vários usuários; redes e computadores pertencem a diferentes domínios administrativos, cada qual com sua própria política de utilização. Tudo isso, pode levar a uma grande variação de carga e a uma grande variação nos recursos disponíveis para as aplicações paralelas. Os usuários de um determinado domínio administrativo têm prioridade sobre seus recursos; o escalonador deve priorizar as aplicações locais, só escalonando aplicações paralelas em momentos de baixa utilização.

A comunicação e a consistência das informações necessárias ao escalonamento é um dos mais graves problemas. A transferência de informações em grids deve ser limitada para que não haja degradação no desempenho causada pela contenção na rede. Deve-se evitar muita comunicação e réplicas, que devem ser atualizadas com freqüência.

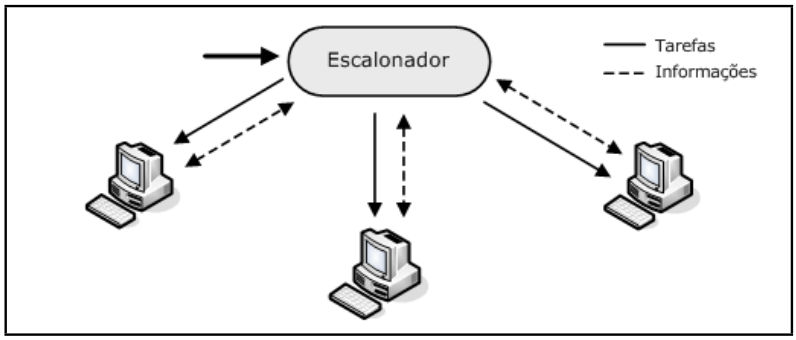
O tipo de aplicação dos grids também está relacionado à comunicação. Geralmente, a intensidade de comunicação entre as tarefas de uma aplicação determina à que tipo de plataforma ela se aplica. Em plataformas com fraco acoplamento as tarefas que compõem uma aplicação devem trocar pouca ou nenhuma informação. Aplicações desse tipo, em que suas tarefas são independentes, recebem o nome de *Bag-of-Tasks* e são comuns em grids (REIS, 2005). Vale ressaltar que é possível submeter aplicações paralelas com comunicação intensiva ao escalonador de um grid, mas cabe ao escalonador procurar, dentre os recursos do grid, aquele que provenha um bom desempenho (um MPP, por exemplo).

As características acima citadas influenciam em outro aspecto do escalonador: sua organização. Tipicamente, um escalonador centralizado não é aplicável a grids computacionais, pois, num grid, nenhum sistema pode controlar todos os recursos (BERMAN, 1999). Devido à grande escala, ampla distribuição e existência de múltiplos domínios administrativos (em que os administradores de cada site não desejam dispor do controle sobre seus recursos), é pouco viável utilizar uma abordagem centralizada. Os Grids Computacionais utilizam escalonadores capazes de dividir a carga de trabalho entre si, cooperativamente.

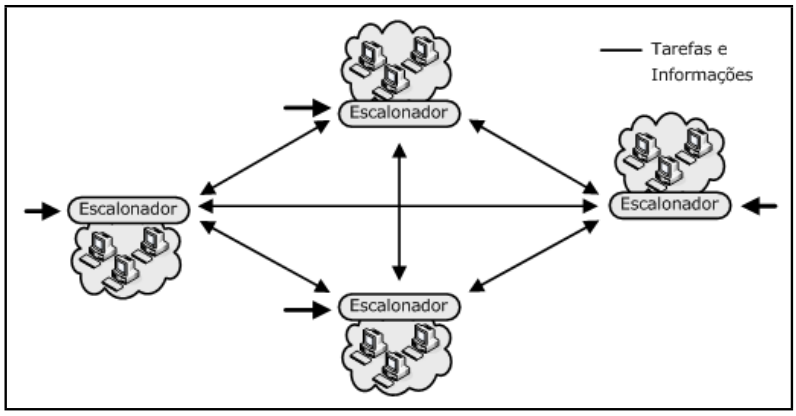
Existem, basicamente, três tipos de escalonadores: centralizado, descentralizado e hierárquico (HAMSCHER et al., 2000; BUYYA et al., 2002).

Um esquema de escalonador centralizado é apresentado na figura 3.2(a). Esse tipo de escalonador, empregado, tradicionalmente, em clusters e NOWs, controla todos os recursos do sistema, podendo se tornar um gargalo. Ele toma todas as decisões de quando, como e onde cada aplicação e tarefa deve executar. Além disso, ele é responsável por receber as requisições de todos os usuários e arbitrar, entre estes, qual terá acesso aos recursos compartilhados.

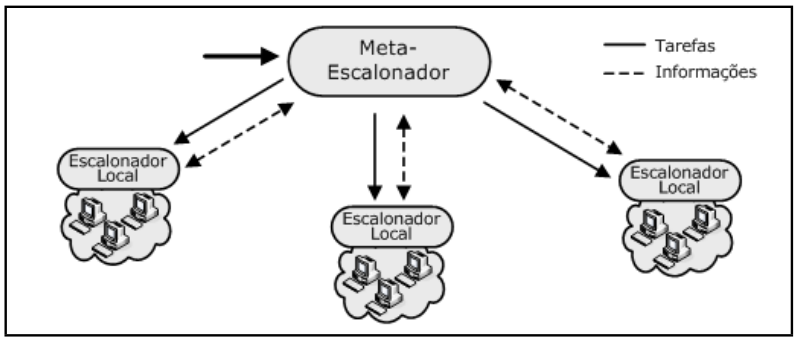
Uma alternativa aos escalonadores centralizados é o escalonador dito, descentralizado. Ele é esquematizado na figura 3.2(b). Nesse modelo a carga de trabalho é dividida entre escalonadores do mesmo nível. Cada escalonar pode decidir em que recurso do sistema determinada tarefa será escalonada. Uma característica importante desse modelo é que cada escalonador tem



(a) Centralizado



(b) Descentralizado



(c) Hierárquico

Figura 3.2: Modelos de escalonadores (BUYYA et al., 2002)

conhecimento do estado dos demais, sendo capaz de promover um balanceamento da carga. Quando uma requisição chega a um escalonador sobrecarregado, ele verifica qual dos seus vizinhos possui menor sobrecarga e envia a solicitação para ele. Um problema desse modelo é a sobrecarga gerada pela comunicação para manter o estado dos escalonadores vizinhos atualizado.

O terceiro tipo de escalonador é o hierárquico, esquematizado na figura 3.2(c). Nesse modelo existem dois ou mais níveis de escalonadores. Cada escalonador do nível mais baixo é responsável por um conjunto de recursos computacionais. Nos níveis superiores os escalonadores, conhecidos como meta-escalonadores, não são mais responsáveis por recursos e sim por um grupo de escalonadores de nível inferior. A decisão de escalonamento ocorre no escalonador de mais alto nível. Para tanto, os demais escalonadores (de nível inferior) devem enviar,

periodicamente, seus estados ao meta-escalonador.

3.5 Considerações Finais

O escalonamento é uma atividade natural na computação que decorre da existência de vários usuários requisitando o uso de um ou mais recursos. O escalonamento é uma atividade de extrema importância. Apenas com a adequada utilização dos recursos, provendo às aplicações os requisitos computacionais necessários, é possível obter um bom desempenho. Em clusters, NOWs e, principalmente, em *grids* o escalonamento ainda é alvo de intensa pesquisa.

A heterogeneidade, a variação de carga, a interatividade do usuário, o fraco acoplamento entre os EPs são características desses sistemas que devem ser consideradas nas políticas de escalonamento. Além disso, diferentes objetivos são propostos para a atividade de escalonamento e aplicações com diferentes requisitos são submetidas ao sistema. Tudo isso torna extremamente complexo o desenvolvimento de uma única política de escalonamento que trate adequadamente todos os sistemas e todas as aplicações.

Existe uma forte tendência em desenvolver políticas que tratem situações e problemas específicos. Essa tendência é evidenciada no LSF e no AMIGO (*vide* seção 3.3), que suportam a troca dinâmica das políticas de escalonamento baseado no estado do sistema e no tipo de aplicação a ser escalonada. Dessa forma, muitas abordagens têm sido propostas para o escalonamento em NOWs, tornando difícil comparar diferentes alternativas. Nesse contexto, a avaliação de desempenho tem um importante papel. Somente com critérios bem definidos, utilizando técnicas de avaliação de desempenho que permitam classificar e comparar as políticas de escalonamento disponíveis, é possível escolher a melhor abordagem.

Capítulo

4

Avaliação de Desempenho

A avaliação de desempenho está se tornando uma atividade cada vez mais importante em todo o processo computacional. Isto se deve ao fato de que uma boa avaliação pode garantir maior desempenho a um custo menor. Além disso, a crescente complexidade dos sistemas computacionais (hardware + software) exige o uso de técnicas elaboradas para verificar o desempenho (SANTANA et al., 1994; FORTIER; MICHEL, 2003).

Antes de qualquer coisa, é preciso definir o que é desempenho. Uma figura de desempenho para um sistema computacional corresponde ao resultado de uma análise da quantidade de serviços prestados em relação ao tempo total decorrido desde o início desses serviços (FORTIER; MICHEL, 2003). A satisfação do usuário em relação à qualidade de um serviço é também um parâmetro relacionado ao desempenho, embora não seja um parâmetro confiável. O desempenho só pode ser definido com maior clareza ao se conhecer os objetivos do serviço que será avaliado. As **métricas de desempenho** permitem formalizar e quantificar os objetivos de um serviço a fim de obter resultados confiáveis da avaliação de desempenho (SANTANA et al., 1994).

A importância da avaliação de desempenho é refletida pela existência de uma grande quantidade de pesquisas na área e de um grande número de conferências e jornais especializados. Além disso, grande parte dos eventos possuem seções voltadas especialmente para a avaliação de desempenho em domínios específicos, como o escalonamento de aplicações paralelas. No entanto, na maioria das vezes, a avaliação de desempenho é simplesmente ignorada e freqüentemente usada de forma incorreta. Jain (1991), logo no início do seu livro, enumera vinte e dois erros comuns na avaliação de desempenho e como podem ser evitados. Em geral, o desuso e a incorreta aplicação da avaliação de desempenho se deve ao fato de que essa

é uma tarefa difícil, cujas técnicas nem sempre são dominadas. A experiência do analista de desempenho é um fator primordial.

A avaliação de desempenho é empregada em muitas situações diferentes e com vários propósitos. Em sistemas computacionais distribuídos a avaliação de desempenho pode ser aplicada na análise de sistemas existentes, análise de sistemas em desenvolvimento ou seleção de um determinado sistema (SANTANA et al., 1994). Em sistemas existentes a avaliação de desempenho é empregada com o objetivo de maximizar a eficiência e a utilização de um dado recurso ou processamento, minimizando os custos e o tempo de resposta, entre outros objetivos. Os sistemas em desenvolvimento são avaliados para predizer seu comportamento, sendo que a avaliação estende-se desde a concepção do sistema até sua instalação. Quando se trata de selecionar um sistema, uma comparação do desempenho é realizada segundo algum parâmetro (FORTIER; MICHEL, 2003).

No restante deste capítulo são abordadas as métricas e técnicas utilizadas para verificar a qualidade de um serviço, seja com o objetivo de seleção ou de melhoria. A próxima seção destina-se a conceituar métricas de desempenho e a apresentar algumas métricas largamente utilizadas. Em seguida, as técnicas são abordadas, procurando enfatizar as características e os domínios em que cada técnica é recomendada. *Benchmarking*, uma das técnicas de avaliação de desempenho, é abordada em detalhes na seção 4.3.

4.1 Métricas de Desempenho

Uma avaliação de desempenho baseia-se na análise de métricas de desempenho. As métricas de desempenho permitem formalizar os objetivos de um serviço e quantificar os resultados de uma avaliação de desempenho. Para afirmar se o desempenho está bom ou ruim ou o quanto o desempenho é bom ou ruim, é preciso verificar o valor de uma métrica de desempenho adequada para avaliar o objetivo considerado (KRALLMANN et al., 1999).

Para avaliar o desempenho, de forma objetiva, uma métrica deve ser definida. As métricas tradicionalmente empregadas para avaliar o desempenho são (FORTIER; MICHEL, 2003):

Tempo de resposta médio – é a média do tempo total que os *jobs* esperam no sistema, desde a chegada até a partida. O tempo de resposta é considerado ser uma métrica orientada ao usuário. Minimizando essa métrica os resultados são retornados mais rapidamente.

Throughput – é o número de tarefas que o sistema processa por unidade de tempo. Em sistemas não saturados, o *throughput* é igual à taxa de chegada, assim, tipicamente o que é considerado é a taxa máxima de chegada que o sistema pode sustentar (*throughput* sustentável). O *throughput* sustentável é considerado ser uma métrica orientada ao sistema. Maximizar essa métrica significa que mais trabalho é completado por unidade de tempo.

4.2 Técnicas para Avaliação de Desempenho

A avaliação de desempenho consiste em coletar informações associadas aos parâmetros significativos para a análise. As técnicas para a avaliação de desempenho são os métodos utilizados para a obtenção dessas informações (SANTANA et al., 1994).

Selecionar uma técnica é um passo importante da avaliação de desempenho. Muitas considerações estão envolvidas nesta escolha, por exemplo, o estágio de desenvolvimento do sistema que se quer avaliar, o tempo disponível para avaliação, a disponibilidade de ferramentas e o nível de precisão desejado. Em geral, apenas uma técnica é utilizada na avaliação, no entanto, nada impede que mais de uma técnica seja usada seqüencialmente, de forma complementar (JAIN, 1991).

As técnicas para avaliação de desempenho podem ser divididas em dois grandes grupos: técnicas de modelagem e técnicas de aferição.

4.2.1 Técnicas de Modelagem

Em alguns casos, existe a necessidade de avaliar o desempenho sem utilizar o sistema computacional em si. Pode ser que não se deseje interferir no comportamento do sistema ou este ainda não foi desenvolvido. Uma boa alternativa é a utilização de técnicas de modelagem.

As técnicas de modelagem baseiam-se na representação de um determinado sistema computacional na forma de modelos. Um modelo é uma abstração de um sistema computacional que preserva as características mais relevantes desse sistema. O modelo de um sistema deve ser o mais simples possível, contendo somente características essenciais. A inclusão de muitos detalhes em um modelo pode torná-lo complexo sem necessidade (SANTANA et al., 1994) e, conseqüentemente, dificultar a avaliação.

Diversas técnicas de modelagem podem ser utilizadas para desenvolver modelos. A literatura apresenta várias delas, porém, as mais utilizadas são Redes de Filas, Redes de Petri, *Statecharts* e Cadeias de Markov (ORLANDI, 1995).

Redes de Filas

Existem diversas situações nos sistemas computacionais em que há concorrência por algum recurso compartilhado, como UCP ou disco rígido. Como, geralmente, a quantidade se serviços é maior do que o número de recursos disponíveis, formam-se filas de serviços a serem atendidos. Para permitir modelar esse tipo de sistema existe uma técnica denominada Redes de Filas (SANTANA et al., 1994; BOLCH et al., 2005).

Essa técnica é baseada na Teoria das Filas (um ramo das probabilidades). As Redes de Filas possuem uma sólida base matemática, mas sua representação gráfica é limitada a apenas duas entidades: centro de serviços e usuários, em que o primeiro é um provedor de serviços

que são usufruídos pelo segundo. Um centro de serviço pode possuir mais de um servidor de acordo com o sistema sendo modelado. Existe também uma fila de espera para os usuários que requisitaram um serviço mas ainda não obtiveram acesso. A técnica de Redes de Filas é bastante limitada para modelar sistemas, pois não permite que se modele várias características que podem ser desejáveis conter em um modelo, como a situação dos serviços em estados pronto, bloqueado e processando.

Redes de Petri

Informalmente, Redes de Petri (MARSAN et al., 1998; ZHOU; VENKATESH, 1999), podem ser definidas como uma técnica que permite a especificação de sistemas utilizando representação matemática. Essa técnica possui mecanismos poderosos de análise, sendo bastante adequada aos sistemas computacionais da atualidade, visto que permite modelar sistemas paralelos, concorrentes, assíncronos e não-determinísticos (NONATO, 1999). A representação gráfica de uma rede de petri é bastante poderosa. Redes de Petri permitem modelar, facilmente, várias características de um sistema que não são possíveis representar em Redes de Filas. Uma desvantagem de Redes de Petri é a grande complexidade do modelo quando sistemas maiores são considerados.

Statecharts

Statecharts (FRANCÊS et al., 1999; FRANCÊS et al., 2000) são uma extensão das máquinas de estado finito, possibilitando representar hierarquia, concorrência e comunicação entre os diversos estados de um determinado sistema. São usados principalmente para sistemas que reagem a estímulos externos e internos, normalmente sob condições críticas em relação ao tempo (FRANCÊS, 2001). *Statecharts* são usados para modelar sistemas complexos, nos quais os diagramas de estado não são eficientes.

Cadeias de Markov

As cadeias de Markov (SONG et al., 2004; BOLCH et al., 2005) modelam sistemas de acordo com suas probabilidades de transações. Uma cadeia de Markov é um processo estocástico que tem uma propriedade especial denominada Memoryless. Essa propriedade se refere ao fato de que uma transição para um determinado estado depende somente do estado atual.

As maiores vantagens da utilização de técnicas de modelagem são a flexibilidade dessas técnicas (podem ser aplicadas tanto em sistemas existentes quanto em sistemas ainda não desenvolvidos) e a relação custo–benefício (o custo é baixo para o grau de precisão obtido). No entanto, vale ressaltar que as abstrações dos modelos implicam em simplificações que diminuem a precisão dessas técnicas. As técnicas de modelagem tem uma precisão menor que as técnicas de aferição. Uma grande desvantagem das técnicas de modelagem é a dificuldade em validar o modelo.

4.2.2 Soluções para os Modelos

Uma vez definido um modelo de um sistema computacional, é preciso resolvê-lo para se obter os resultados que demonstram o desempenho do sistema modelado. Um modelo pode ser resolvido utilizando solução analítica ou solução por simulação. A solução analítica exige domínio matemático, mas é mais rápida e barata, enquanto a simulação apresenta resultados com melhor qualidade, permitindo especificar mais detalhes no modelo. Porém, a simulação envolve a construção de um programa computacional que represente o modelo, sendo mais demorada e mais cara. A escolha de uma das abordagens depende de fatores diretamente ligados ao sistema e à avaliação. A figura 4.1 apresenta algumas diretrizes que auxiliam na escolha de uma ou outra abordagem.

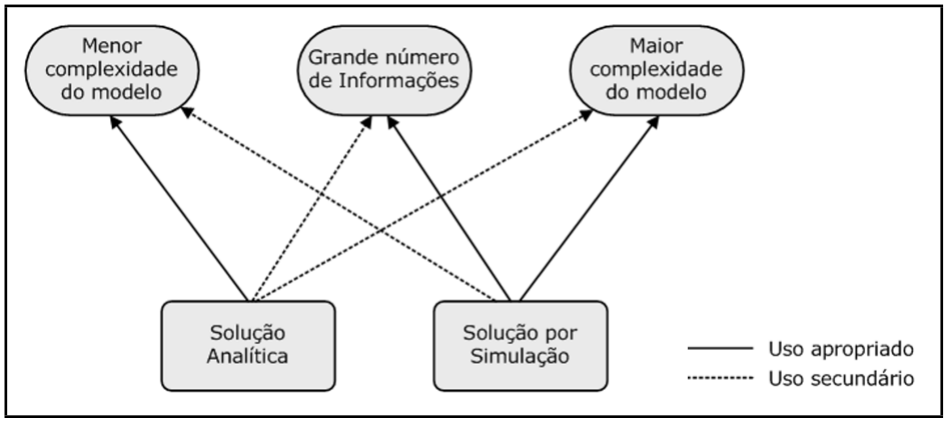


Figura 4.1: Soluções de um modelo (SANTANA et al., 1994)

Solução Analítica

Utilizando a técnica de solução analítica é possível escrever uma relação funcional entre os parâmetros do sistema e os critérios de desempenho escolhidos, em termos de equações. Essas equações podem ser resolvidas por meio de análise matemática (SANTANA et al., 1994).

Em geral, solução analítica é mais rápida e, por isso, é o método preferido para resolver modelos. Essa técnica também fornece resultados mais precisos. No entanto, a solução analítica nem sempre é aplicável, pois, à medida que a complexidade do sistema modelado aumenta, também aumenta a dificuldade da utilização desta técnica. Para que o modelo seja resolvido, algumas simplificações podem ser necessárias. Essas simplificações representam restrições, que, em geral, não correspondem ao que é verificado nos sistemas reais. (SANTANA et al., 1994). Embora seja um método que fornece resultados expressivamente exatos, as restrições podem levar a um modelo que não representa adequadamente o sistema real, tornando sem sentido a utilização desse modelo. Geralmente, os modelos são representados utilizando-se redes de filas. A solução analítica possui um custo relativamente baixo e o tempo para a solução desejada depende da complexidade do problema.

Solução por Simulação

Simulação (BANKS, 1998) é uma das abordagens mais comuns para a avaliação de desempenho de sistemas computacionais. Simular o comportamento de um objeto (ou um sistema de objetos) é uma necessidade em várias áreas do conhecimento. Em computação, simulação refere-se ao emprego de um programa de computador (um simulador) para a implementação do modelo de algum fenômeno ou sistema dinâmico (sistemas cujos estados se alteram com o tempo).

Quando o modelo envolve um grande número de informações e não é possível fazer as simplificações necessárias para a utilização de solução analítica, pode-se optar por simulação, para resolver um modelo. Fatores, como flexibilidade, facilidade na utilização e custos relativamente baixos têm motivado o emprego da simulação. Os maiores problemas na utilização de simulação são a dificuldade de verificar se o simulador está correto e a grande quantidade de computação necessária para obter medidas válidas do ponto de vista estatístico (MAJUMDAR; PARSONS, 2000). A simulação distribuída busca solucionar esse problema inserindo paralelismo em uma simulação executada em uma arquitetura distribuída (FUJIMOTO, 2003).

Souza (2000) utiliza simulação para avaliar políticas de escalonamento. No trabalho de Ishii et al. (2005) e de Senger (2005), a simulação foi utilizada para demonstrar as melhorias obtidas de suas propostas para o escalonamento de aplicações paralelas em Computação Paralela e Distribuída.

4.2.3 Técnicas de Aferição

As técnicas de aferição são aplicadas a sistemas computacionais já implementados ou em fase final de seu desenvolvimento, podendo também ser aplicadas a protótipos. Assim, a partir da experimentação sobre o próprio sistema (ou um protótipo) é possível verificar o desempenho (SANTANA et al., 1994).

As técnicas de aferição são as técnicas que apresentam os resultados mais próximos da realidade. Elas proporcionam maior precisão em relação às técnicas de modelagem, mas, requerem que o sistema computacional (ou ao menos um protótipo) esteja disponível. O maior problema no uso de técnicas de aferição é que estas atuam diretamente sobre o sistema, podendo influenciar no comportamento do mesmo; as técnicas de aferição disputam por recursos com o sistema sendo avaliado.

Algumas das principais técnicas de aferição são construção de protótipos, coleta de dados e *benchmarking*. As duas primeiras técnicas são discutidas a seguir e *benchmarking* é vista na seção 4.3. Assim como nas técnicas de modelagem, a escolha entre uma das técnicas de aferição depende de alguns fatores. A figura 4.2 apresenta um esquema que ajuda a direcionar essa decisão.

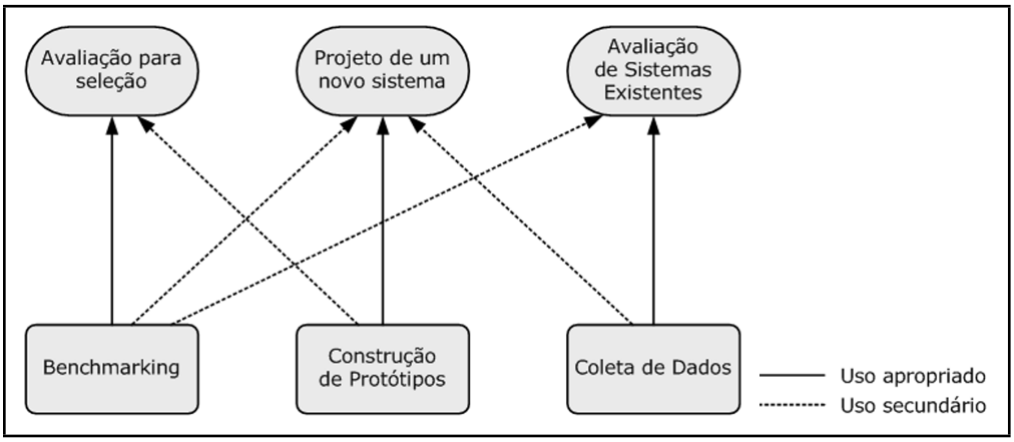


Figura 4.2: Propostas de Avaliação e Técnicas de Aferição (SANTANA et al., 1994)

Construção de Protótipos

Quando se deseja avaliar um sistema computacional que ainda não foi desenvolvido (ou que esteja em desenvolvimento) e a precisão é um fator importante, uma alternativa é a construção de um protótipo. Um protótipo é uma simplificação do sistema computacional que se deseja avaliar, mantendo-se as funcionalidades e as características essenciais de tal sistema. Os protótipos possuem menor custo e maior facilidade de alteração em relação à construção e alteração do sistema real (ORLANDI, 1995). Entretanto, no que se refere às outras técnicas de aferição, a construção de protótipos têm maior custo.

Para se construir um protótipo é preciso, primeiramente, verificar a adequabilidade de um sistema à construção de protótipos. Em seguida, delimitando e conhecendo os domínios funcionais e comportamentais do sistema real é possível desenvolver o protótipo. A construção de um protótipo é constituída de diversas fases de desenvolvimento, teste e melhorias e verificação, até que se tenha um protótipo que reproduza as características fundamentais e as funcionalidades do sistema real (SANTANA et al., 1994).

Coleta de Dados

Quando, numa avaliação de desempenho, os dados podem ser obtidos diretamente do sistema computacional e a precisão é importante, a coleta de dados é uma técnica adequada. Dentre as técnicas de aferição, coleta de dados é a mais precisa (FORTIER; MICHEL, 2003). No entanto, a coleta de dados deve ser realizada de forma criteriosa. A coleta de dados e o sistema sendo avaliado concorrem pela utilização de recursos, como o processador, o que pode interferir no comportamento do sistema real. Sem tratamento adequado o tempo gasto na coleta são acrescentados aos cálculos das métricas de desempenho, gerando dados incorretos.

Uma aplicação útil da coleta de dados é a validação de modelos. A coleta de dados pode ser aplicada a um sistema existente e os dados colhidos podem ser comparados aos dados fornecidos por um modelo desse sistema. O modelo validado pode ser usado para verificar

alternativas.

A coleta de dados é realizada com a utilização de monitores que podem ser de dois tipos (SOUZA, 2004):

Monitores de hardware: são equipamentos de hardware específicos utilizados para coletar dados relacionados ao objeto em estudo e analisá-los. Os monitores de hardware devem se limitar à coleta dos dados, mantendo os dados inalterados e fiéis aos dados originais.

Monitores de software: são usados em casos nos quais se deseja observar características específicas do software. Essas características são bastante peculiares ao software e não podem ser detectadas por monitores de hardware. Exemplos de uso são a verificação da existência ou não de uma fila de espera associada a algum recurso.

O trabalho de doutorado de Souza (2004) aplica monitoração ao escalonamento de processos em sistemas distribuídos. Ele objetiva prover uma arquitetura de monitoração que permita avaliar e obter métricas de desempenho representativas de maneira eficiente e com baixa sobrecarga. Essa arquitetura pode ser usada por qualquer escalonador.

4.3 Benchmarking

Benchmarking é uma técnica de aferição bastante empregada na avaliação de desempenho, principalmente, quando se deseja comparar alternativas. Segundo Jain (1991), *benchmarking* é o processo de comparação de desempenho entre dois ou mais sistemas por meio de medições e, para se efetuar tais medições, são utilizadas cargas de trabalho[1] (*workloads*) denominadas *benchmarks*. Essa denominação deve-se ao fato de que o termo *benchmark* tem sido comumente empregado na literatura como sinônimo de carga de trabalho em geral. Embora alguns autores restrinjam o termo benchmark a cargas de trabalho reais, nesse trabalho benchmarks serão usados como cargas de trabalho em geral.

Os *benchmarks* são programas utilizados para avaliar hardware, software ou sistemas computacionais completos. Esses programas definem cargas de trabalho que representam as características essenciais de um domínio de aplicação. Segundo Weicker (2002), o que torna um programa um *benchmark* é o fato de que esse programa é padronizado (contendo especificações detalhadas), tendo sido projetado ou selecionado para executar em diferentes sistemas computacionais com o objetivo de realizar comparações justas do desempenho desses sistemas. Assim, algumas características são importantes para um *benchmark* (WEICKER, 2002; MADAN, 1997):

Portável – um benchmark deve ter a possibilidade de executar em diferentes sistemas computacionais, não devendo haver qualquer dependência de hardware e software. Um

[1] Conjunto de todas as requisições de serviço (processamento) submetidas a um sistema durante um período de tempo.

benchmark deve poder ser executado tanto em um mainframe quanto em computadores portáteis. Por exemplo um benchmark para avaliar computadores paralelos deve rodar em qualquer computador paralelo que se deseje comparar o desempenho.

Justo – não deve haver qualquer influência em um benchmark que privilegie determinado tipo de hardware ou software. Embora até mesmo programas portáveis possuam certas estruturas que privilegiam o sistema para o qual foram originalmente escritos, os *benchmarks* devem diminuir esse problema e devem ser o mais justo possível.

Relevante – um *benchmark* deve representar adequadamente a área de aplicação para a qual se destina. Ele deve emular (comportar-se como outro aceitando as mesmas entradas e produzindo as mesmas saídas) apenas certos domínios de aplicações específicos (que tem características de comportamento comum para os fatores sendo medidos). Um *benchmark*, que se destina a representar diversos domínios de aplicações, torna-se complexo demais ou seu resultado perde a importância.

Simples – Em suma, um benchmark deve ser fácil de medir e de explicar. A especificação de um benchmark deve ser simples de entender e implementar. Uma especificação simples ajuda a minimizar influências de implementação de uma plataforma para outra. Um benchmark que não é simples dificilmente será adotado pelos desenvolvedores de um produto e pelos usuários.

Ao executar os benchmarks sobre diversos sistemas é possível obter as métricas de desempenho e assim realizar uma análise comparativa, garantindo qual possui melhor desempenho para um dado domínio de aplicação. Freqüentemente, benchmarks utilizam, para medir o desempenho de sistemas computacionais, tanto tarefas mais gerais, como realização de E/S, quanto tarefas específicas, como *Crivo de Eratóstenes*[2] (JAIN, 1991, p. 52), operações sobre matrizes, etc. Qualquer aspecto do desempenho, que seja interessante avaliar, pode ser medido através de benchmarks.

A utilização de *benchmarks* também exige alguns cuidados para se efetuar a avaliação de desempenho. Sendo *Benchmarking* uma técnica de aferição, é necessário verificar se a utilização dos *benchmarks* não influenciará o comportamento do próprio sistema. Outro aspecto que merece atenção é a escolha das métricas de desempenho. As métricas não podem ser genéricas, visto que, *benchmarks* são usados para avaliar características particulares; e devem ser adequadas para caracterizar os aspectos de desempenho sendo avaliados. É importante considerar também as influências de outros componentes do sistema sobre os componentes sendo avaliados. Por exemplo, as unidades MIPS (Milhões de Instruções Por Segundo) e FLOPS (*Floating Point Operations Per Second*) são controvérsias, pois fornecem valores perigosamente absolutos, mesmo diante de fatores distintos (como arquiteturas *Reduced Instruction Set Compu-*

[2] É baseado em um algoritmo utilizado para encontrar todos os números primos abaixo de um dado número n e é utilizado para comparar microprocessadores.

ters (RISC) e *Complex Instruction Set Computers* (CISC)), que podem influenciar os resultados (SANTANA et al., 1994).

A utilização de benchmarks para avaliar o desempenho de políticas de escalonamento de processos para aplicações paralelas é possível. Como dito anteriormente, *benchmarks*, definem cargas de trabalho. Assim, os benchmarks impõem uma carga de trabalho a uma política de escalonamento de processos, que deve portanto gerenciar essa carga de trabalho e responder adequadamente a ela. Assim, um escalonador é apenas um gerenciador de carga de trabalho que deve garantir o desempenho do sistema submetido à avaliação (SOUZA, 2004). Através das medições é possível dizer qual política se comportou melhor sobre determinada carga de trabalho, e que, conseqüentemente, é melhor para determinada aplicação.

4.4 Considerações finais

A avaliação de desempenho, no atual estágio de desenvolvimento das técnicas de produção de software, é essencial para se obter um bom sistema. Segundo Jain (1991), a avaliação de desempenho é uma arte, que não pode ser realizada mecanicamente. Portanto, cabe ao analista de desempenho conhecer e fazer as escolhas certas para obter sucesso em uma avaliação de desempenho.

Existem diversas técnicas de avaliação de desempenho que podem ser utilizadas para avaliar sistemas e políticas de escalonamento de processos. A escolha da técnica a ser utilizada está diretamente relacionada ao estágio de desenvolvimento do sistema (ou política de escalonamento de processos) a ser avaliado e aos propósitos da avaliação.

As técnicas de modelagem são muito úteis quando o sistema ainda não foi desenvolvido. Atualmente, é uma necessidade verificar o comportamento de um sistema antes mesmo de iniciar seu desenvolvimento. Embora as técnicas de modelagem não possuam a mesma precisão das técnicas de aferição, elas são boas alternativas porque fornecem bons resultados a um custo razoável.

Quando o sistema a ser avaliado já está implementado, as técnicas de aferição são bastante adequadas, principalmente pela precisão que fornecem. Cada vez mais as técnicas de aferição são melhoradas no sentido de fornecer resultados mais precisos e influenciar menos na execução do sistema sendo avaliado.

A técnica mais adequada quando se deseja fazer uma escolha entre dois ou mais sistemas, com base numa comparação de desempenho, é *benchmarking*. Os benchmarks fornecem meios adequados para comparar diversas alternativas e garantir qual é melhor para determinado domínio de aplicação.

Capítulo

5

Cargas de trabalho e Métricas de Desempenho

O escalonamento pode afetar profundamente a eficiência de um computador paralelo. Segundo Wong et al. (2000), aumentar a eficiência de uma máquina Cray T3E, contendo 644 processadores de 80% para 90% (taxa de utilização) significa uma economia de 2 milhões de dólares, que seriam gastos na compra de 81 novos processadores para suportar a mesma quantidade de *jobs*. Um bom escalonador reduz a perda de recursos, melhora a utilização do sistema e aumenta o *throughput*. O escalonador é também capaz de prover melhorias do ponto de vista do usuário, diminuindo o tempo de resposta das aplicações. Apesar disso, muitos administradores de ambientes de produção simplesmente escolhem uma política da literatura e a adaptam ao seu ambiente. Embora se obtenha um escalonamento válido, o desempenho pode não ser adequado (KRALLMANN et al., 1999).

Obter desempenho no escalonamento é uma tarefa complexa, especialmente em Computação Paralela Distribuída. No capítulo 3, foram discutidos diversos fatores que influenciam nas decisões do escalonador e, conseqüentemente, na qualidade do serviço provido. A obtenção de um bom escalonamento depende de uma série de características do sistema computacional utilizado e da carga de trabalho imposta ao escalonador. Dessa forma, torna-se difícil encontrar um único algoritmo que seja eficiente para todos os sistemas (KRALLMANN et al., 1999), levando ao desenvolvimento de diversas propostas para cenários específicos. Selecionar uma boa política de escalonamento depende de uma série de restrições.

Nesse contexto, a avaliação de desempenho tem um importante papel, pois permite verificar a validade de uma proposta, a qualidade do escalonamento gerado e possíveis problemas

de desempenho. A avaliação de desempenho permite também comparar e selecionar políticas de escalonamento. Entretanto, é preciso estar atento ao fato de que diferenças de desempenho podem ser causadas também pela metodologia de avaliação (FEITELSON, 2003).

Neste capítulo, é tratada a avaliação de desempenho no contexto específico do escalonamento de aplicações paralelas, relacionando os principais trabalhos da área. O foco do assunto aqui tratado são as cargas de trabalho e as métricas de desempenho que têm sido utilizadas. A seção seguinte dá uma visão geral da metodologia empregada para avaliar o escalonamento, ficando os detalhes inerentes à seleção da carga e das métricas, respectivamente, para as seções 5.2 e 5.3.

5.1 Avaliação de Desempenho do Escalonamento

A avaliação de desempenho do escalonamento requer o uso de uma metodologia adequada. Existem diversos detalhes e opções que devem ser observados a fim de obter uma avaliação confiável, cujos resultados possam ser reproduzidos e facilmente explicados. Não existe na literatura muitos trabalhos que tratam das metodologias empregadas na avaliação de políticas de escalonamento para aplicações paralelas. Grande parte do conhecimento resulta de trabalhos que apresentam novas propostas para o escalonamento e utilizam a avaliação de desempenho para demonstrar a validade e/ou a superioridade da sua proposta (BEAUMONT et al., 2003; STREIT, 2003; ISHII et al., 2005).

Como visto no capítulo anterior, diversas técnicas se aplicam à avaliação de políticas de escalonamento. Quando uma política está implementada num sistema real ou num protótipo, as técnicas de aferição apresentam resultados mais precisos. A solução analítica tem sua aplicabilidade reduzida devido à complexidade. A simulação pode apresentar resultados bastante satisfatórios. Os modelos empregados nos simuladores representam com um alto grau de fidelidade as características de um sistema real. No entanto, o que, geralmente, se deseja numa avaliação de desempenho é comparar alternativas e selecionar aquela que obteve o melhor desempenho (FEITELSON, 2003). Nesse caso, uma técnica bastante adequada é *Benchmarking* (*vide* seção 4.3).

Em um dos mais importantes trabalhos voltados à avaliação de desempenho do escalonamento, Feitelson e Rudolph (1998) discutem métricas de desempenho e propõem o uso de *benchmarks* para avaliar o escalonamento de aplicações paralelas. Segundo os autores, ao contrário de *benchmarks* padrão, que consistem de um conjunto de *jobs* representativos, executados isoladamente, um *benchmark* para avaliar o escalonamento especifica o momento da submissão ao sistema e a estrutura interna de cada *job* (consumo de UCP e de memória, paralelismo, acesso a dispositivos de I/O). Um aspecto importante dessa carga de trabalho, usada como *benchmark*, é que ela seja padronizada, pois, assim, é possível realizar comparações justas e confiáveis entre diferentes escalonadores. Além disso, é possível comparar diferentes propostas

independentemente se a carga de trabalho é aplicada a um sistema real, em simulação ou em solução analítica.

Parte da análise de desempenho é verificar sob que condições de carga uma política de escalonamento não é mais capaz de processar *jobs*. Isso é particularmente útil para o planejamento de capacidade de um sistema. Além disso, as diferenças de desempenho aparecem quando os sistemas sob estudo são submetidos a uma carga que estresse sua capacidade. Tal análise só é possível em sistemas abertos on-line (FEITELSON; RUDOLPH, 1998).

Escalonadores são sistemas de filas: os *jobs* chegam, podem aguardar um tempo na fila, recebem o serviço requerido e, finalmente, deixam o sistema. Os sistemas de filas podem ser on-line ou off-line. No caso de sistemas on-line um sistema pode ainda ser aberto ou fechado. A figura 5.1 ilustra os três tipos de sistemas de fila, explicados a seguir.

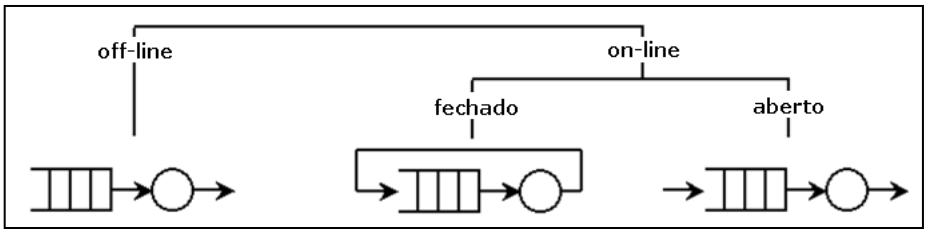


Figura 5.1: Os três tipos genéricos de sistemas de fila (FEITELSON; RUDOLPH, 1998)

Em um sistema off-line todos os *jobs* e os recursos requeridos por esses *jobs* são conhecidos no início. Nenhum *job* chega depois que o sistema está em execução. Assim, um escalonamento ótimo pode ser empregado. Sistemas off-line são frequentemente usados para escalonadores em lotes (*batch*) ou que utilizam compartilhamento de espaço. Nesse caso, a avaliação utilizando técnicas analíticas fornece uma boa predição do desempenho.

Em um sistema on-line os *jobs* podem chegar a qualquer instante. O escalonador deve processar novos *jobs* no momento em que eles chegam, sem qualquer conhecimento de chegadas futuras. Assim, planejamento é possível somente no estado atual. Possivelmente, o próximo *job* que chega requer um novo planejamento, tornando necessário alterar o escalonamento atual. Assim, computar uma nova solução deve ser rápido e, freqüentemente, essa decisão é crítica, pois o estado do sistema pode ser profundamente afetado no futuro. Em sistemas fechados on-line assume-se um conjunto fixo de *jobs*, havendo um limite máximo de *jobs* no sistema. Novas chegadas só ocorrerão quando *jobs* deixam o sistema. Nesse caso, o sistema nunca excede sua capacidade.

Um modelo mais realístico e representativo dos escalonadores atuais é o sistema aberto on-line. Os *jobs* chegam num fluxo infinito, independentemente de outros *jobs* terminarem e das condições de carga do sistema. Esse modelo é mais complexo, necessitando que o processo de chegada também seja modelado. O escalonador em um sistema aberto on-line deve ser capaz de lidar com situações extremas, com intervalos de alta carga. Assim, é interessante avaliar quando

o escalonador não é mais capaz de manipular a carga que chega. O sistema fica saturado, uma vez que um aumento na carga de trabalho não resulta em um alta utilização, mas a qualidade do serviço provido (tempo de resposta médio, por exemplo) cai significativamente.

Krallmann et al. (1999) apresentam recomendações para a seleção e avaliação de sistemas de escalonamento de *jobs* em sistemas paralelos. Segundo os autores, é possível comparar diferentes escalonadores se a mesma "função objetivo" e a mesma carga de trabalho forem usadas. O termo "função objetivo" equivale ao termo métrica de desempenho, enfatizando que a métrica deve ser escolhida de acordo com o objetivo a que determinada política de escalonamento se propõe (SOUZA, 2004). A função objetiva atribui um valor escalar, denominado custo de escalonamento, a cada escalonamento realizado. Assim, é possível verificar se o escalonamento cumpre seu objetivo, distinguindo bons e maus escalonamentos e classificando escalonadores.

Tipicamente, o que se faz para comparar políticas de escalonamento é aplicar a mesma carga de trabalho às diferentes alternativas e verificar (por meio das métricas coletadas) como elas se comportam. Espera-se que as diferenças nos resultados obtidos durante a avaliação reflitam as diferenças dos sistemas em estudo. Entretanto, os resultados dependem não somente do sistema, mas também da métrica usada e da carga de trabalho imposta a ele. Existem várias interações entre sistema, métrica e carga de trabalho que podem levar a resultados incoerentes (FEITELSON, 2002b). A seleção de cargas de trabalho e das métricas de desempenho são discutidas nas seções seguintes.

5.2 Cargas de trabalho

O desempenho de um sistema computacional depende da carga de trabalho que lhe é imposta (CALZAROSSA; SERAZZI, 1993; FEITELSON, 2002c). Assim, bons modelos de carga são necessários para avaliar o escalonamento. Existem diversos trabalhos que discutem sobre a aplicação, caracterização e uso de cargas de trabalho na avaliação de políticas de escalonamento para aplicações paralelas (FEITELSON; RUDOLPH, 1998; DOWNEY; FEITELSON, 1999; CALZAROSSA et al., 2000; CIRNE; BERMAN, 2001; FEITELSON, 2002c).

As cargas de trabalho utilizadas para avaliar sistemas abertos on-line são compostas por duas componentes principais: um intervalo de chegada de *jobs* e uma descrição da estrutura interna de cada *job* (FEITELSON; RUDOLPH, 1998). A primeira componente especifica em que momento, de um dado período de tempo, um *job* será submetido ao escalonador. Essa componente está relacionada às características do usuário (humano), podendo haver ciclos diários, devido aos padrões de trabalho. Além disso, usuários tipicamente reagem ao desempenho do escalonador. Se o tempo de resposta é pequeno, mais *jobs* são submetidos. Em um certo nível de carga (muito alta) o escalonador não é mais capaz de prover um bom serviço e o tempo de resposta aumenta. Nesse ponto, o usuário pára de submeter novos *jobs* e, possivelmente, o ciclo

se repete. A segunda componente especifica os requerimentos de recurso de cada *job*. Consumo de UCP e elementos da estrutura interna de um *job* podem ser modelados, tal como paralelismo, barreira, comunicação, acesso a dispositivos de I/O.

Existe uma grande discussão a respeito de que elementos devem ser modelados, tanto no processo de chegada, quanto na descrição da estrutura de um *job*. Segundo Feitelson e Rudolph (1998) quanto mais características forem modeladas na carga de trabalho, mais características do escalonador serão exercitadas. Isso seria útil, por exemplo, para avaliar o impacto de determinada característica provida por um escalonador sobre outro que não provê a mesma característica. Entretanto, quanto mais detalhes uma carga de trabalho provê mais complexa ela se torna e mais difícil é compreender a interação entre seus componentes e os possíveis resultados sobre a avaliação de desempenho. Assim, segundo Feitelson (2002b), as cargas de trabalho devem ser mantidas tão simples quanto possível.

A escolha da carga de trabalho está diretamente relacionada às características do sistema em estudo e ao objetivo da avaliação. As informações da estrutura interna dos *jobs* podem variar de carga para carga. Por exemplo, se o objetivo é avaliar o desempenho de uma política de escalonamento voltada a aplicações CPU-Bound não faz sentido manter informações de acesso à rede. Entretanto, se o objetivo é verificar a melhoria de desempenho provida por uma política que trata a comunicação entre os processos sob outra que não trata, então a informação de acesso à rede é essencial.

Existem, basicamente, duas opções para se obter uma carga de trabalho para avaliar o escalonamento. Uma é utilizar diretamente logs coletados da execução de aplicações paralelas, a outra é utilizar modelos de cargas de trabalho. Existe uma grande discussão a respeito das vantagens e desvantagens do uso de cada abordagem (LO et al., 1998; DOWNEY; FEITELSON, 1999; FEITELSON, 2002c).

Os modelos, também denominados modelos de cargas de trabalho sintéticas, usam distribuições de probabilidade para gerar os dados da carga de trabalho. Essas distribuições podem ser definidas com base em pouco ou nenhum conhecimento (modelos puros ou "*naive*") ou através da análise de logs de cargas de trabalhos reais (modelos realísticos).

O processo de gerar um modelo de carga de trabalho a partir de logs coletados em ambientes reais é denominado caracterização de carga. Basicamente, o que se faz é analisar características importantes do *log*, definindo distribuições de probabilidades (juntamente com seus parâmetros) que reflitam um comportamento similar no modelo. Um exemplo é o intervalo entre chegadas que pode ser modelado por meio de uma Poisson com média 1500, como utilizado em (ISHII et al., 2005). A caracterização de carga é uma importante área de pesquisa que têm sido intensificada nos últimos anos (JANN et al., 1997; CALZAROSSA et al., 2000; FEITELSON, 2002c; LUBLIN; FEITELSON, 2003; ZILBER et al., 2005; FEITELSON, 2006).

Segundo Lo et al. (1998), a escolha, simplesmente, entre logs e modelos de carga de trabalhos realísticos não afeta o desempenho relativo das políticas de escalonamento avaliadas, ou seja, se uma política A obtêm o melhor desempenho para um log ela também será a melhor para a carga de trabalho sintética. Isso reforça a idéia de que para se comparar políticas o importante é que a mesma carga de trabalho seja utilizada.

Algumas observações, no entanto, podem nortear o uso de modelos de carga de trabalho realísticos:

- Os logs podem conter anomalias que tornam a carga não representativa. Por exemplo, é possível encontrar submissões repetidas de um mesmo *job* por um mesmo usuário. Esse comportamento pode dominar a carga de trabalho por um período relativamente pequeno e afetar o resultado do desempenho. Segundo Tsafrir e Feitelson (2003), essas anomalias devem ser retiradas da carga de trabalho.
- As modelagem de carga de trabalho é realizada basicamente com base em cargas de trabalhos reais, obtidas dos logs. Assim, o comportamento do modelo e do log são similares.
- Os modelos permitem alterar alguns parâmetros para ajustar a carga de trabalho à avaliação. Por exemplo, pode não fazer sentido utilizar a carga de trabalho coletada em um sistema de 400 nós em outro sistema contendo apenas 64 nós. Utilizando um modelo é possível ajustar a carga a esse parâmetro, mantendo as mesmas características da carga de trabalho real.

Diante do exposto, neste trabalho optou-se pelo uso de modelos realísticos de carga de trabalho. A fim de obter tais modelos foram pesquisados dois repositórios de cargas de trabalho: *HPC Workload/Resource Trace Respository* (HPC, 2005) e o *Parallel Workload Archive* (PWA, 2005). O primeiro contém apenas logs e, além disso, dados sobre o sistema paralelo e as configurações do escalonador não estão disponíveis. O *Parallel Workload Archive* fornece logs e modelos realísticos. Esses modelos são discutidos na seção seguinte.

5.2.1 Modelos de cargas de trabalho

A maior parte dos modelos disponíveis no *Parallel Workload Archive* foram criados com base na análise estatística de logs coletados em diferentes sistemas paralelos de larga escala. Grande parte desses logs também estão disponíveis no repositório (*vide* Apêndice A). No Parallel Workload Archive é fornecida uma descrição detalhada de cada modelo e referências, tanto do autor do modelo quanto de trabalhos que utilizam-no. Está também disponível código para gerar cargas de trabalho utilizando o modelo e há um grande esforço em utilizar um formato padrão para definir a carga. Esse padrão é discutido na seção seguinte.

Os modelos disponíveis se dividem em duas categorias: modelos de *jobs* rígidos e modelos de *jobs* flexíveis Chapin et al. (1999). Essa classificação se deve aos tipos de *jobs* existentes.

A flexibilidade dos *jobs* é classificada em rígida, moldável, *evolving* e maleável (FEITELSON et al., 1997).

Quando o escalonador deve fornecer ao *job* exatamente a quantidade de recursos requerida e esta não se altera durante a execução, o *job* é rígido. Só assim um *job* rígido é capaz de executar. Os *jobs* moldáveis, ao contrário, são capazes de executar com uma quantidade mínima de recursos, melhorando seu desempenho até que uma quantidade máxima de recursos lhe seja atribuída. O escalonador decide, dentro do intervalo, quantos recursos atribuir. Nos *jobs evolving* a quantidade de recursos requerida varia durante a execução. O *job* passa por diferentes fases de paralelismo, decicindo se mais ou menos recursos são necessários. Cabe ao escalonador apenas prover a quantidade requisitada. Nos *jobs* maleáveis a quantidade de recursos atribuída aos *jobs* pode ser alterada pelo sistema. O escalonador decide se o *job* deve liberar recursos ou se recursos adicionais lhe serão atribuídos.

Os modelos para *jobs* rígidos fornecem o tempo de submissão, número de recursos requeridos e tempo de execução de cada *job*. Ao contrário, um modelo de *jobs* flexíveis descreve como a aplicação executa em diferentes níveis de paralelismo. Isso pode ser expresso, por exemplo pelo total de computação e uma função de *speedup*.

Os modelos Feitelson96, Jann97 e Lublin99 foram utilizados neste trabalho e serão analisados no capítulo seguinte. Entretanto, para fundamentar a escolha adotada neste trabalho, todos os modelos são detalhados a seguir. A escolha realizada neste trabalho baseia-se no fato de que modelos de jobs moldáveis não se aplicam às políticas de escalonamento avaliadas neste trabalho. Isso será discutido em detalhes no próximo capítulo.

Calzarossa85 (CALZAROSSA; SERAZZI, 1985)

Embora não seja um modelo para aplicações paralelas, este é um modelo bastante útil. Ele especifica o processo de chegada para *jobs* interativos em um ambiente multiusuário. O modelo fornece a taxa de chegada como uma função da hora do dia, incluindo padrões de chegada cíclicos. A função para dias normais fornece a taxa de chegada com base no tempo t, relativo à hora do dia num intervalo de 8h30min e 18h. Para ser usada em simulação baseada em eventos, onde é necessário o intervalo entre chegadas, a função deve ser derivada (FEITELSON; RUDOLPH, 1998).

Leland86 (LELAND; OTT, 1986)

Este modelo também não é voltado a aplicações paralelas. Ele fornece o tempo de execução de processos em um ambiente interativo. A distribuição proposta é válida para processos maiores que três segundos. Devido a essa restrição a probabilidade de que um processo executará mais do que t segundos é proporcional a t^k, em que $-1.25 < k < -1.05$. Num estudo posterior, Harchol-Balter e Downey (1995) reafirmaram o modelo e forneceram outros valores para k, além de outras fórmulas para obter o tempo de execução.

Sevcik94 (SEVCIK, 1994)

Este modelo é útil para *jobs* flexíveis. Ele fornece características de speedup de aplicações paralelas: desbalanceamento, trabalho serial e *overhead* paralelo (STREIT, 2003). Ele apresenta uma função que quantificando o desbalanceamento, a quantidade de trabalho serial e por processador e os atrasos de comunicação e congestionamento fornece o tempo de execução de um *job* com n processadores alocados. Nesse trabalho, são fornecidos dois conjuntos de valores. O primeiro obtém um baixo *speedup* para *jobs* pequenos e o segundo conjunto de valores obtém um *speedup* razoável para *jobs* grandes.

Feitelson96 (FEITELSON, 1996)

Este é, provavelmente, o primeiro modelo detalhado para aplicações paralelas. Ele baseia-se na observação de seis logs de máquinas paralelas em ambientes de produção: NASA Ames iPSC/860, Argonne[1] IBM SP1 com 128 nós, SDSC[2] Paragon com 416 nós, LLNL[3] "Butterfly"com 126 nós, CTC[4] IBM SP2 com 512 nós e ETH Paragon com 96 nós. O modelo foi criado com o objetivo de refletir cargas de trabalho reais.

Da observação dos logs foram constatadas algumas características:

- A predominância de *jobs* pequenos e com tamanho igual a potência de dois, mesmo sem razões arquiteturais. Todos os logs possuem um número significativo de *jobs* seqüenciais e de *jobs* que usam menos de dez processadores.
- Correlação entre o grau de paralelismo e tempo de execução. Há pequenos *jobs* que rodam por um longo tempo e grandes *jobs* que rodam por um curto tempo, mas a tendência é grandes *jobs* rodarem por um longo período.
- Repetidas execuções de um mesmo *job* por um mesmo usuário.

O grau de paralelismo (ou tamanho do *job*) é dado por uma distribuição *Harmônica* de ordem 1.5. Essa distribuição foi ajustada para enfatizar *jobs* de tamanhos pequenos e potência de dois. Para gerar o tempo de execução de um *job* é usada uma distribuição *Hiper-Exponencial*, tal que o coeficiente de variação seja maior que um. Existe uma relação linear entre o tamanho do *job* e a probabilidade de usar a exponencial com média maior. Assim, há uma distribuição diferente para cada tamanho de *job*, com tempo de execução médio maior para *jobs* maiores. Uma distribuição *Zipf* é usada para definir o número de execuções repetidas de um *job*. A probabilidade de um *job* ser executado n vezes é proporcional a $n^{-2.5}$.

O processo de chegada usado no modelo segue uma distribuição *Poisson*, ajustada para a repetição de *jobs*, uma vez que um *job* repetido só chega ao sistema depois que o anterior

[1] Argonne National Lab
[2] San-Diego Supercomputer Center
[3] Lawrence Livermore National Lab
[4] Cornell Theory Center

termina. Entretanto, existem duas versões do modelo: a original de 1996 e uma melhorada em 1997, cuja diferença está nos detalhes das distribuições. Observando o código fornecido em PWA (2005) a distribuição de intervalo entre chegadas é dada por uma *Exponencial* de média 1500 segundos.

Downey97 (DOWNEY, 1998)

Este é um modelo para *jobs* moldáveis e maleáveis, que inclui, portanto, uma função de *speedup*. O modelo baseia-se nas observações dos logs do SDSC e no log do CTC SP2. A dominância de *jobs* com tamanho potência de dois é ignorada nesse modelo. Segundo o autor, isso é resultado do particionamento do sistema. O grau de paralelismo é dado por uma distribuição *Log-Uniforme*.

O modelo utiliza um tempo de execução cumulativo, que equivale à soma do tempo de execução em todos os processadores. O tempo de execução médio é obtido por meio da função de *speedup*, fornecendo como parâmetro o número de processadores alocados pelo escalonador. Uma distribuição *Log-Uniforme* é usada. A função de *speedup* é modelada usando três parâmetros: o grau de paralelismo, a variância no paralelismo e o número de processadores. As equações para o cálculo do *speedup* são detalhadas em (DOWNEY, 1998).

O processo de chegada segue uma distribuição *Poisson*. Entretanto, as chegadas só acontecem durante a primeira parte (correspondente ao dia num ambiente real). Na segunda parte (noite) não há chegadas; o sistema apenas executa os *jobs* colocados na fila.

Jann97 (JANN et al., 1997)

Este é um modelo detalhado da primeira parte do log CTC SP2. Ele foi construído para *jobs* rígidos. O modelo usa a distribuição *Hiper-Erlang* de Ordem Comum tanto para o tempo de execução quanto para o intervalo entre chegadas. Os parâmetros usados baseiam em combinar os três primeiros momentos do log. A distribuição Hiper-ErLang é um generalização das distribuições Exponencial, Hiperexponencial e Erlang. Ela é caracterizada por 4 parâmetros.

Como as características dos *jobs* com diferentes tamanhos diferem, existem diferentes parâmetros de acordo com o grau de paralelismo. Foram definidos intervalos com base em potência de dois ou múltiplos de cinco. Cada intervalo possui seus próprios parâmetros e, além disso, existem parâmetros diferentes para intervalo entre chegadas e tempo de execução. Tabelas com todos os valores dos parâmetros estão disponíveis no artigo que descreve o modelo.

Esse modelo é reutilizado para modelar a carga de trabalho do LLNL ASCI Blue-Pacific. Esse modelo difere do CTC SP2 apenas no que se refere aos valores dos parâmetros empregados. Esses valores estão disponíveis em (FRANKE et al., 1999).

Feitelson98 (FEITELSON; RUDOLPH, 1998)

Este não é propriamente um modelo com distribuições de probabilidade. Trata-se de um *framework* para criar modelos da estrutura interna de aplicações paralelas com o objetivo de investigar as conexões entre o comportamento da aplicação e o escalonamento. Esse framework deve ser completado com valores de parâmetros específicos para criar um modelo. Entretanto, nenhum conjunto de parâmetros foi ainda sugerido devido à falta de logs de dados reais a respeito da estrutura interna das aplicações (PWA, 2005).

Lublin99 (LUBLIN; FEITELSON, 2003)

Trata-se de um modelo detalhado para *jobs* rígidos. Inclui padrão de chegada baseado em um ciclo diário com intervalos de pico e correlação entre tempo de execução e o número de nós requisitados. Ele é baseado em três logs: SDSC Paragon, LANL[5] CM-5 e KTH[6] IBM SP2.

Segundo o autor, *jobs* com tamanho potência de dois devem ser enfatizados, assim como *jobs* seqüenciais, conforme encontrado nos logs. Dessa forma, os *jobs* são divididos em três classes: seqüenciais, potência de dois e o restante. A probabilidade de que um *job* seja seqüencial é 0.24. Se ele for paralelo, a probabilidade de que seja potência de dois é 0.75. A distribuição para o tamanho de *jobs* paralelos é uma *Uniforme de dois estágios*, com os parâmetros $l = 0.2 -$ tamanho mínimo de job, $m =$ tamanho máximo $- 2.5$, $h =$ tamanho máximo de job, $p = 0.86$.

O tempo de execução é modelado com base na correlação entre tamanho do *job* e tempo de execução. É usada uma distribuição *Hiper-Gamma* composta de duas *Gammas* e um parâmetro p. Esse parâmetro é linearmente dependente do tamanho do *job*.

Ao contrário da grande maioria dos modelos, o processo de chegada é detalhado. As chegadas são baseadas em duas distribuições *Gamma*: uma para modelar picos no intervalo entre chegadas e outra para modelar o ciclo diário.

O modelo faz ainda distinção entre *jobs batch* e interativos, existindo parâmetros distintos para cada um deles.

Cirne01 (CIRNE; BERMAN, 2001)

Esse é um modelo para gerar *jobs* moldáveis. Ele é composto por um modelo para gerar *jobs* rígidos e um modelo para converter *jobs* rígidos em moldáveis. Para isso ele gera um conjunto de alternativas contendo tamanho de partição e tempo de execução. Cirne01 baseia-se no modelo de *speedup* de Downey (1998), propondo novos parâmetros para as distribuições.

[5] Los Alamos National Lab
[6] Swedish Royal Institute of Technology

Tsafrir05 (TSAFRIR et al., 2005)

Trata-se de um modelo detalhado para gerar tempo de execução estimado pelo usuário de forma bastante realística. O modelo captura o comportamento do usuário. Pelas observações quando um usuário submete uma aplicação paralela ele fornece valores muito pequenos e, freqüentemente, fornece o maior valor permitido para a estimativa do tempo de execução.

O modelo é composto por duas partes: **1)** um gerador realístico de distribuições de estimativa de tempo de execução do usuário **2)** um aplicativo que adiciona a estimativa a logs e modelos existentes (utilizando formato padrão de cargas de trabalho).

A tabela 5.1 apresenta um resumo das características dos modelos disponíveis no *Parallel Workload Archive* (PWA, 2005). Cada modelo recebe um nome composto pelo seu principal autor e o ano de sua criação. Alguns modelos são para *jobs* paralelos (rígidos ou flexíveis), outros para *jobs* seqüencias em ambiente Unix. As informações providas por cada modelo variam. Algumas características são parcialmente informadas.

Tabela 5.1: Comparação dos modelos de carga de trabalho disponíveis no *Parallel Workload Archive* (PWA, 2005)

Modelo	***Jobs***	**Grau de paralelismo**	**Tempo de execução**	***Speedup***	**Intervalo entre chegadas**	**Tempo estimado de execução**
Calzarossa85	Unix	não	não	não	sim	não
Leland86	Unix	não	sim	não	não	não
Sevcik94	moldável	não	sim	sim	não	não
Feitelson96	rígido	sim	sim	não	parcial	não
Downey97	moldável	sim	sim	sim	parcial	não
Jann97	rígido	parcial	sim	não	sim	não
Feitelson98	variado	parcial	parcial	implícito	não	não
Lublin99	rígido	sim	sim	não	sim	não
Cirne01	moldável	sim	sim	sim	sim	não
Tsafrir05	n/a[a]	n/a	n/a	n/a	n/a	sim

[a] não se aplica

Para facilitar a utilização desses modelos e dos logs das cargas de trabalho disponíveis no *Parallel Workload Archive* foi criado um formato padrão. Ele é discutido na seção seguinte.

5.2.2 Formato Padrão para Cargas de Trabalho

O Formato Padrão para Cargas de Trabalho, *Standard Workload Format* (SWF) (CHAPIN et al., 1999), visa facilitar o uso de logs e modelos em programas e simuladores. Com o SWF é possível utilizar diferentes cargas de trabalho (seja um log ou um modelo) sem necessi-

dade de alterar o código que interpreta os arquivos da carga.

O SWF foi especificado para ser portável e fácil de interpretar. Cada carga de trabalho é armazenada em um arquivo ASCII, em que cada linha armazena os dados de um *job*. As linhas contém uma quantidade pré-definida de campos; quando um campo não se aplica ao tipo de carga seu valor é "-1". Geralmente, o arquivo começa com comentários sobre o modelo ou sobre o ambiente em que os logs foram coletados. Os comentários são linhas iniciadas por ";".

Os campos que compõem cada linha no arquivo que especifica uma carga de trabalho são descritos na tabela 5.2. Uma descrição detalhada sobre o SWF pode ser obtida em Chapin et al. (1999) ou no *Parallel Workload Archive* (PWA, 2005).

5.3 Métricas de Desempenho

A avaliação de desempenho não depende apenas do sistema em estudo e da carga de trabalho que lhe é imposta, mas também da métrica de desempenho (FEITELSON, 2002b). Assim, para obter resultados conclusivos de uma avaliação de políticas de escalonamento para aplicações paralelas, a escolha de uma métrica adequada tem grande relevância. Diversos trabalhos discutem o uso de métricas nesse contexto (FEITELSON; RUDOLPH, 1998; KRALLMANN et al., 1999; FEITELSON, 2001; FEITELSON, 2003; SOUZA, 2004; SABIN; SADAYAPPAN, 2005; FRACHTENBERG; FEITELSON, 2005).

A métrica de desempenho deve ser escolhida com base em diversos aspectos: objetivo da avaliação, características do sistema em estudo e características da carga de trabalho. Geralmente, para um dado problema existem diversas métricas que se aplicam. É importante que a métrica de desempenho reflita com fidelidade o comportamento do sistema sob a carga de trabalho imposta.

Feitelson e Rudolph (1998) apresentam uma discussão da escolha da métrica com relação ao tipo de sistema de fila. Segundo os autores, existe uma métrica de desempenho apropriada e comumente usada para cada um dos três sistemas de fila. Para sistemas *off-line* a métrica escolhida é *Makespan*, enquanto que para sistemas fechados *on-line* e abertos *online* as métricas mais apropriadas são, respectivamente, *Throughput* e Tempo de resposta.

Makespan é dado pelo tempo total gasto para executar toda a carga de trabalho. Como em sistemas *off-line* todos os *jobs* são conhecidos no início cabe ao escalonador otimizar a alocação de recursos fornecendo o melhor tempo de resposta para cada um dos *jobs* e, conseqüentemente, o melhor tempo de resposta total. Em um sistema fechado on-line o processo de chegada depende do desempenho do sistema. Cada vez que um *job* termina ele é imediatamente re-submetido. Nesse caso, a análise é verificar o *Throughput*: o número de *jobs* executados por unidade de tempo.

Tempo de Resposta é o tempo decorrido desde a submissão até o término de um *job*. Segundo Feitelson e Rudolph (1998), esta é a principal métrica para sistemas abertos on-line

Tabela 5.2: Descrição dos campos do *Standard Workload Format* (PWA, 2005)

Campo	**Descrição**
Número do *job*	Contador, iniciando em 1.
Tempo de submissão	Instante, em segundos, em que o *job* foi submetido ao sistema. Inicia em 0. As linhas do arquivo são ordenadas em ordem ascendente de acordo com esse campo.
Tempo de espera	Diferença, expressa em segundos, entre o instante da submissão e o início da execução de um *job*. É relevante apenas em logs.
Tempo de execução	Dado em segundos. Equivale à diferença entre o momento do término de um *job* e o momento de início da execução.
Número de processadores alocados	Número de EPs alocados pela aplicação.
Tempo médio de uso de UCP	A média do tempo de UCP usada nos EPs alocados pela aplicação.
Memória usada	Quantidade média por processador de memória usada pela aplicação, em kilobytes.
Número de processadores requeridos	Número de EPs requeridos pela aplicação.
Tempo requerido	Estimativa fornecida pelo usuário do tempo de execução da aplicação.
Memória requisitada	Estimativa fornecida pelo usuário da quantidade média de memória por processador requerida pela aplicação.
Completado	Possui valor igual a 1 se a aplicação foi finalizada naturalmente e igual a 0 se ocorreu algum erro ou houve alguma restrição do escalonador.
Identificador de usuário	Um inteiro entre 1 e o número total de usuários existente na carga de trabalho. Identifica um usuário particular.
Identificador de grupo	Um inteiro entre 1 e o número total de grupos existente na carga de trabalho. Identifica um grupo particular.
Número do executável	Um inteiro entre 1 e o número total de aplicações existente na carga de trabalho.
Número da fila	Quando existe mais de uma fila de espera por recursos, identifica uma fila específica do escalonador.
Número da partição	Se o ambiente é composto por mais de uma partição, esse campo indica em que partição a aplicação foi executada.
Número do *job* antecessor	Expecifica a aplicação previamente executada.
Tempo de pensar do *job* antecessor	Diferença, em segundos, entre o término da aplicação anterior e o início da aplicação atual.

porque Utilização e *Throughput* são determinados pela taxa de chegada e pelos requisitos do *job*, não pelo escalonador.

Feitelson e Rudolph (1998) afirmam que a métrica Tempo de Resposta Médio enfatiza *jobs* longos ao invés de *jobs* pequenos, que são mais comuns nas cargas de trabalho. Por exemplo, o tempo de resposta médio de 100 *jobs* de 1 hora e um *job* de 3 semanas é 6 horas. Nesse trabalho e em outro trabalho posterior, Feitelson (2002a), os autores afirmam que isso é um problema e apresentam a métrica *Slowdown* como uma possível solução.

A métrica *Slowdown* apresenta um problema: *jobs* muito pequenos com atrasos razoáveis podem levar a valores de *Slowdown* excessivamente altos (FEITELSON; RUDOLPH, 1998). Por exemplo, um *job* que executa por apenas um segundo, mas aguarda 20 minutos na fila para ser escalonado terá um *Slowdown* igual a 1200.[7] A solução para tal problema é uma outra métrica que impõe um limite ao *Slowdown*, denominado *Slowdown* Limitado (*Bounded Slowdown*). O Slowdown limitado é dado pela razão entre o $maior$(tempo de resposta, limite) e o $maior$(tempo de execução real, limite). Assim, o mesmo *job* do exemplo anterior teria, utilizando um limite de 10 segundos, um *Slowdown* Limitado igual a 120.

Outra abordagem para a escolha da métrica de desempenho é apresentada por Krallmann et al. (1999) e Souza (2004). Segundo os autores, a métrica de desempenho deve ser escolhida com base no objetivo do escalonamento. Relacionar a métrica ao objetivo torna mais fácil a escolha da métrica, pois diminui-se o número possível de métricas que podem ser utilizadas. Um exemplo são políticas de escalonamento que tem como objetivo otimizar o acesso à memória. Com base nesse objetivo métricas relacionadas ao uso da memória serão escolhidas, como: percentual de utilização da memória, quantidade de paginação, percentual de utilização de swap. Entretanto, ainda existem muitas métricas que se aplicam a esse objetivo. Souza (2004) analisa diversas métricas de desempenho quanto à sua adequação aos objetivos do escalonamento. O autor também analisa diversas métricas quanto à sua relação com os tipos de sistemas de filas e quanto à sua aplicação em sistemas distribuídos heterogêneos.

Segundo Krallmann et al. (1999), cada escalonador tem seu próprio objetivo e, assim, não seria possível comparar dois escalonadores. Por outro lado a comparação de diferentes políticas de escalonamento faz sentido se a mesma métrica de desempenho for utilizada. Quando duas políticas de escalonamento são submetidas à mesma carga de trabalho e analisadas sob a mesma métrica de desempenho é possível concluir qual obteve melhor desempenho sob um determinado aspecto, mesmo que elas possuam objetivos distintos. Quando se deseja, por exemplo, avaliar políticas com o objetivo de verificar qual otimiza o uso da rede, pode-se utilizar a taxa de utilização da rede para avaliar uma política que otimize o acesso à memória. Provavelmente, a política se comportará mal, mas o fato é que a política está sendo analisada sob o aspecto da rede. Nesse sentido, a comparação é plausível.

Quando se compara duas ou mais políticas de escalonamento espera-se que a métrica de desempenho seja capaz de fornecer resultados conclusivos à respeito da qualidade de um escalonamento. Espera-se que a métrica ofereça uma comparação justa entre diferentes políticas, permitindo que se estabeleça uma relação do tipo $escalonador_A > escalonador_B$. É preciso estar atento ao fato de que a métrica reduz o escalonamento de milhares de *jobs* em um único número, de forma que escalonadores sejam facilmente comparáveis. Assim, explicar os efeitos capturados pela métrica de desempenho e responder à grande questão de qual escalonador é

[7] Esse valor foi obtido usando a primeira definição de *Slowdown*, tempo de resposta de um *job* normalizado pelo tempo em que o *job* está verdadeiramente executando.

melhor depende, também, da carga de trabalho.

Diante de todo o exposto, a métrica Tempo de Resposta Médio é bastante atrativa. Alguns pontos adicionais descrevem a escolha dessa métrica para a avaliação de desempenho conduzida neste trabalho:

- Tempo de resposta é uma das métricas mais conhecidas e utilizadas, sendo mais fácil para um pesquisador comparar seu trabalho com os demais utilizando uma métrica bastante conhecida.
- Grande parte dos objetivos de um escalonador tem relação com diminuir o tempo de resposta. Por exemplo, reduzir o percentual de swap é uma forma de otimizar o uso da memória que resulta em um melhor tempo de resposta.
- O problema apresentado por Feitelson e Rudolph (1998) e discutido acima não é visto neste trabalho como um problema. Isso é apenas um reflexo do uso de uma média aritmética. Além disso, não é incorreto o fato de *jobs* longos terem grande influência na métrica, uma vez que *jobs* grandes consomem recursos por um longo período e provocam atrasos nos demais *jobs*.

5.4 Considerações Finais

Neste capítulo foram apresentados alguns aspectos relacionados à avaliação de desempenho no contexto do escalonamento de aplicações paralelas. A obtenção de resultados conclusivos em uma avaliação de desempenho requer o uso de uma metodologia adequada, principalmente quando o objetivo é comparar alternativas. A escolha da carga de trabalho e da métrica de desempenho têm grande influência na avaliação.

Quando se deseja comparar diferentes alternativas a mesma carga de trabalho e a mesma métrica de desempenho devem ser usadas. Entretanto, alguns aspectos relacionados à escolha de cargas de trabalho e métricas devem ser observados. Assim, diretrizes para a escolha da carga de trabalho foram apresentadas, apontando vantagens de modelos sobre logs.

A escolha pela métrica Tempo de Resposta Médio para a avaliação conduzida neste trabalho foi justificada, uma vez que acredita-se que a utilização dessa métrica oferece resultados conclusivos. Diversos trabalhos que comparam políticas de escalonamento de aplicações paralelas, referentes aos sistemas computacionais distribuídos, fazem uso da métrica Tempo de Resposta (ARNDT et al., 2000; KARATZA, 2000; KARATZA, 2002a; KARATZA; HILZER, 2003; SHIM, 2004).

Capítulo

6

Estudo Comparativo de Cargas de Trabalho e Políticas de Escalonamento

Diversas políticas de escalonamento para aplicações paralelas em ambientes computacionais distribuídos têm atraído a atenção de pesquisadores, uma vez que os resultados obtidos com tais políticas têm indicado bons desempenhos. Entretanto, tais políticas são avaliadas com o objetivo apenas de demonstrar sua superioridade sob um determinado aspecto. Sob diferentes condições de carga, a política pode ter seu desempenho deteriorado, o que, na maioria das vezes, não é verificado. Analisar o comportamento de uma política de escalonamento sob diferentes cenários é uma importante tarefa para se compreender como a mesma se comportaria em um ambiente real. Além disso, o conhecimento de como a política se comporta sobre diferentes condições de carga pode garantir que se obtenha bons desempenhos.

Vários trabalhos comparam diferentes estratégias para o escalonamento em *clusters* e NOW (ARNDT et al., 2000; KARATZA, 2000; KARATZA, 2002a; KARATZA; HILZER, 2003; SHIM, 2004) e até mesmo em *grids* (HAMSCHER et al., 2000). Entretanto, nesses trabalhos não há a preocupação em verificar o comportamento da política de escalonamento sob diferentes condições de carga que podem ocorrer.

Nesse contexto, o presente trabalho, estuda o comportamento de diferentes políticas de escalonamento, verificando sua aplicabilidade a *clusters* e *grids* computacionais e a diferentes situações de carga. Foram selecionadas 10 políticas voltadas ao escalonamento em ambientes paralelos e distribuídos. Utilizando simulação, cada uma das políticas foi submetida a uma combinação de 4 cargas de UCP e 3 cargas de rede, num total de 12 cargas de trabalho distintas. Cada uma das cargas de trabalho foi aplicada à política de escalonamento quando em execução

em 3 diferentes sistemas computacionais distribuídos.

Os sistemas computacionais considerados neste estudo foram: um *cluster* de 32 nós interligados por uma rede Gigabit Ethernet; um *cluster* de 128 nós interligados por uma rede Gigabit Ethernet e um *grid* de 512 nós, composto por *sites* que têm em média 8 computadores. Cada um dos *sites* do *grid* se comunica com os demais por meio da Internet. A obtenção dos dados dos sistemas utilizados na parametrização do simulador é discutido no capítulo seguinte.

Este capítulo destina-se a descrever as políticas avaliadas neste trabalho, as cargas de trabalho utilizadas e o simulador. Quanto às políticas de escalonamento, suas principais características são discutidas, focando nos aspectos que influenciam seu comportamento e, conseqüentemente, no desempenho obtido. As cargas de trabalho selecionadas são analisadas, sob diversos aspectos, com o objetivo de compreender os resultados obtidos nas simulações. No que se refere ao simulador, o modelo UniMPP, utilizado para predizer o tempo de resposta; os detalhes da implementação do simulador, assim como as extensões aplicadas ao mesmo durante este trabalho, são discutidas.

6.1 Políticas de Escalonamento

Nesta seção, são apresentadas as políticas de escalonamento estudadas neste trabalho.

6.1.1 Round Robin

A política de escalonamento Round Robin é talvez a mais simples e mais conhecida das políticas de escalonamento. Utilizando essa política os processos são atribuídos aos elementos de processamento de maneira cíclica, até que não haja mais nenhum processo a ser escalonado (TANENBAUM, 2001).

Esta política é interessante e bastante utilizada devido à coleta de informações do sistema, como tamanho das tarefas e taxa de utilização da UCP, não ser necessária. Entretanto, em Sistemas Distribuídos, a utilização da política de escalonamento Round Robin não é adequada. Nesses sistemas, ela pode gerar um grande desbalanceamento de carga entre os processadores, uma vez que máquinas com baixa potência computacional recebem a mesma quantidade de carga de máquinas com grande potência computacional. Assim, devido à heterogeneidade configuracional dos Sistemas Distribuídos, essa política apresenta, freqüentemente, tempos de resposta muito mais altos do que políticas que consideram a heterogeneidade.

6.1.2 Random, Disted, Lowest, Central e Global

Zhou e Ferrari (1987) analisam algoritmos de balanceamento de carga, estudando seus efeitos no desempenho de sistemas distribuídos fracamente acoplados. Todos os algoritmos são compostos, basicamente, por dois elementos:

- *Load Information Manager* (LIM), responsável pelas políticas de informação e de locali-

zação (*vide* seção 3.2.4). A coleta e gerenciamento da informação de carga e a tomada de decisão da localização de um *job* são realizados pelos LIMs que executam em cada host que compõe o sistema. Os LIMs monitoram a carga constantemente e cooperam entre eles do modo definido pelo algoritmo de balanceamento de carga.

- *Load Balance Manager* (LBM), responsável pelo mecanismo de balanceamento de carga. Existe um LBM em cada host. O LBM estabelece uma conexão entre o host destino e o host remoto e, interagindo com os componentes do sistema, permite que o processo execute no host remoto.

A separação entre política e mecanismo de escalonamento torna possível a experimentação de diferentes algoritmos alterando apenas o LIM. Assim, cinco algoritmos foram estudados: Random, Disted, Lowest, Central e Global. O índice de carga utilizado é o mesmo para todos os algoritmos. O índice de carga do ambiente é calculado com base no comprimento da fila de processos em espera pela UCP. Assim, computadores com muitos processos na fila estariam, segundo este índice de carga, mais carregados.

O funcionamento de cada um dos 5 algoritmos é descrito a seguir.

Disted

Neste algoritmo, a política de informação é inteiramente distribuída. Periodicamente, os LIMs de cada host coletam a informação de carga do sistema local e calculam o índice de carga. Caso o novo índice seja significativamente diferente do último índice calculado, o índice atual é enviado para todos os demais LIMs em cada host do sistema.

A política de localização baseia suas decisões nos índices de carga armazenados localmente. Quando um processo é iniciado, o LIM verifica os índices de todos os hosts para decidir qual receberá o novo processo. Na versão do algoritmo descrita em Zhou e Ferrari (1987), o processo será executado no host com menor carga somente se a carga local estiver acima de um dado limiar. No trabalho de Mello (2003), o processo sempre será executado no host com a menor carga. Esta é também a versão utilizada neste trabalho. Se todos os hosts estiverem carregados, ou seja, o índice de carga estiver acima de um dado limite, o processo é executado no host que o iniciou.

Global

Tendo em vista que o algoritmo Disted gera muitas mensagens para a troca dos índices de carga, o algoritmo Global propõe uma abordagem que visa reduzir o número de mensagens trocadas entre os hosts. Neste algoritmo existe um LIM mestre que recebe todos os índices de carga do sistema, enquanto existe um LIM escravo em cada um dos demais hosts. Os LIMs escravos calculam o índice de carga local e enviam para o LIM mestre quando há mudanças significativas. Em intervalos de tempo, o LIM mestre envia para cada um dos escravos um vetor contendo os índices de carga de todos os hosts.

A política de localização deste algoritmo é igual ao do algoritmo Disted. A decisão de alocação dos processos continua sendo distribuída. Quando um processo é iniciado, o LIM verifica os índices de carga locais para decidir em qual host o processo será executado.

Central

Neste algoritmo, existe um LIM mestre que centraliza todas as decisões. Tanto a informação de carga coletada, quanto as requisições de execução de processos são direcionadas para o LIM mestre. Os LIMs escravos periodicamente coletam e calculam o índice de carga e enviam para o LIM mestre, que armazena as informações de carga de todo o sistema. Quando um processo chega num host que executa um LIM escravo, a requisição de execução do processo é repassada ao LIM mestre. Exceto pelo fato de que a decisão de que em que host um processo será executado é tomada apenas pelo LIM mestre, a política de localização deste algoritmo é igual à política de localização do algoritmo Disted.

É preciso atentar para o fato de que neste algoritmo o LIM mestre pode se tornar um gargalo e de que tal algoritmo tem sua aplicabilidade reduzida em sistemas escaláveis de larga escala.

Lowest

Este algoritmo troca informações de carga sob demanda. Quando um novo processo chega ao sistema, o LIM do host que iniciou o processo define um conjunto de hosts, cujo limite é definido pela política de informação. O LIM local solicita, então, para cada LIM desse conjunto de hosts, a informação de carga. De posse desses dados, a política de localização define qual o host com a menor carga e seleciona este host para executar o novo processo. Este algoritmo difere dos anteriores, pois o número de mensagens trocadas no sistema independe do número de computadores (ZHOU; FERRARI, 1987).

Random

Diferentemente dos algoritmos Disted, Global, Central e Lowest, o algoritmo Random não utiliza qualquer informação a respeito da carga dos computadores que compõem o sistema. Quando uma requisição para execução de um processo chega ao sistema, o computador que irá receber o processo é escolhido de forma aleatória.

No trabalho de Zhou e Ferrari (1987) os cinco algoritmos foram avaliados e comparados em termos de seu tempo de resposta médio. Com exceção do algoritmo Random, os demais algoritmos têm desempenho muito próximo. As políticas de informação dos algoritmos Disted, Global e Central oferecem uma melhor alocação dos processos, mas consomem muitos recursos do sistema, sobrecarregando o meio de comunicação.

A partir de todas as análises efetuadas, foi possível concluir também que algoritmos centralizados, tais como o Global, respondem melhor para ambientes onde a carga é constan-

temente alterada, enquanto algoritmos distribuídos, tais como o Lowest, impõem menor sobrecarga no sistema e atingem maior escala. Finalmente, conclui-se que o algoritmo Lowest é o mais indicado para ambientes distribuídos (MELLO, 2003).

6.1.3 DPWP

A *Dynamic Policy Without Preemption* (DPWP) é apresentada no trabalho de mestrado de Araújo (1999). Trata-se de uma política voltada a aplicações *CPU–Bound* que busca balancear a carga entre os elementos de processamento em ambientes distribuídos heterogêneos. As principais características da DPWP são:

- Dinâmica. Essa política tem a capacidade de se adequar às mudanças do ambiente computacional, durante a execução das aplicações paralelas.
- Na DPWP não está incluído nenhum mecanismo de *checkpoint* e migração.
- Tratamento da heterogeneidade no que tange às diferenças de potência computacional dos EPs do sistema.

No trabalho de Araújo (1999), a DPWP utiliza como índice de carga o tamanho da fila de processos normalizado pela capacidade de processamento da UCP. A versão utilizada neste trabalho é apresentada em Senger (2005). O índice de carga utilizado é a ocupação dos processos normalizado pela capacidade de processamento do EP. A ocupação é dada pela soma dos tempos de execução dos processos na fila do EP. Isto se deve ao fato de que o tamanho da fila de processos não é um bom parâmetro para o índice de carga. Um EP pode conter 1 processo de 100 segundos enquanto outro tem 10 processos de 1 segundo. No segundo caso, o tamanho da fila é maior, entretanto, o primeiro EP estará mais carregado.

A DPWP mantém um vetor que armazena o índice de carga de cada EP do sistema. Esse vetor é mantido em ordem ascendente. Quando um *job* chega ao sistema os índices de carga são recalculados e o vetor é reordenado. Em seguida, a DPWP distribui cada um dos processos desse *job* seqüencialmente sobre os EPs, na ordem definida no vetor. Se o número de processos é maior que a quantidade de EPs, a DPWP reinicia a distribuição dos processos a partir do EP na primeira posição do vetor. Assim, privilegia-se a alocação de processos, primeiramente, nos EPs com menor carga.

6.1.4 NBSP

A *Network Bound Scheduling Policy* (NBSP) foi proposta no trabalho de mestrado de Ishii (2004). Trata-se de uma extensão da política de escalonamento DPWP para contemplar o tratamento de aplicações *Network–Bound*. O escalonamento realizado pela NBSP baseia-se em quantificar o impacto no processamento causado pela comunicação.

A NBSP incorpora duas modificações à DPWP: a adoção de equações que quantificam o impacto da comunicação e a distribuição dos processos com base em uma constante k, que

define a utilização máxima da rede de comunicação.

São empregadas duas técnicas de atribuição dos processos aos EPs do sistema. A primeira escalona da mesma forma empregada na DPWP e a segunda realiza uma alocação denominada escalonamento em grupo, na qual todos os processos de um mesmo *job* são alocados em um único EP. A adoção de uma das técnicas depende da constante k.

Enquanto a largura de banda utilizada por um EP é menor do que a constante k, o escalonamento realizado pela NBSP é idêntico ao escalonamento realizado pela DPWP. Para cada processo alocado em um determinado EP, a largura de banda utilizada por esse elemento de processamento é calculada. Se esta largura de banda exceder o valor da constante k é aplicado o escalonamento em grupo. Todos os processos de um *job* que ainda não foram escalonados são alocados no EP que ocupa a primeira posição do vetor (segundo a DPWP).

A taxa de utilização de 25% como valor da constante k mostrou-se adequada, segundo os experimentos conduzidos em Ishii et al. (2005).

6.1.5 GAS

Genetic Algorithm Scheduling (GAS) é um algoritmo de escalonamento definido no trabalho de doutorado de Senger (2005). Ele emprega o conhecimento obtido pelas aplicações paralelas na utilização de recursos, com o objetivo de melhorar o escalonamento em ambientes distribuídos heterogêneos.

O algoritmo utiliza como entradas a aplicação paralela (número de EPs requisitados), o padrão de comportamento na utilização de recursos da aplicação, empregando o modelo de aquisição de conhecimento descrito em Senger (2005) e a predição de sua ocupação, em milhões de instruções, obtida através do algoritmo *Instance-Based Learning* (IBL), descrito no mesmo trabalho. O algoritmo também usa informações do ambiente distribuído, como capacidade e carga dos EPs, custo de comunicação.

O GAS utiliza Algoritmos Genéticos (AG) para obter uma solução aceitável de quais os EPs mais adequados para atender aos requisitos das aplicações paralelas. AGs são baseados em técnicas de busca probabilísticas usadas para a exploração de grandes espaços de busca (CORTES, 2004). Para tanto, utiliza-se uma função de *fitness* para calcular a aptidão de cada indivíduo de uma população.

A função de *fitness* utilizada no GAS é composta por 2 termos. O primeiro denota a estimativa do custo computacional de atribuir um processo a um determinado EP. O segundo termo considera o custo que será adicionado ao realizar a comunicação entre os processos. Os custos são ponderados utilizando 2 variáveis que quantificam a porcentagem de processamento e a porcentagem de comunicação entre os processos de um *job*. O custo computacional do algoritmo GAS é proporcional à quantidade de EPs solicitados pela aplicação (SENGER, 2005).

6.1.6 TLBA

O algoritmo *Tree Load Balancing Algorithm* (TLBA) visa o balanceamento de cargas em sistemas distribuídos heterogêneos e com possibilidade de crescimento em escala. Ele é o resultado do trabalho de doutorado de Mello (2003).

Esse algoritmo cria uma topologia de interconexão lógica entre os EPs em formato de árvore e realiza migração de processos para melhorar o balanceamento de cargas no sistema. Assim, o algoritmo gera um baixo número de mensagens durante as operações de balanceamento de carga, oferece estabilidade em situações de carga pesada e obtém valores baixos para tempos de resposta médio das aplicações paralelas.

O TLBA utiliza uma nova técnica para calcular o índice de carga, que se baseia na ocupação de UCP e de memória. No TLBA, os índices de carga são calculados periodicamente. Quando o valor do índice de carga de um EP varia acima ou abaixo de um dado limiar (*threshold*), o índice é propagado para demais EPs na árvore, no sentido da raiz.

Quando é preciso localizar o melhor EP para receber um processo, a TLBA busca, a partir do nó raiz da árvore (mantida em memória principal), o EP com menor carga. Quando um EP encontra-se sobrecarregado, ele solicita ao EP raiz a média de ociosidade dos EPs do sistema. Em seguida, a viabilidade de transferir processos de um EP carregado para outro ocioso é analisada. Se for viável, o receptor é localizado e o processo migrado, desde que este ainda não tenha sido migrado k vezes no ambiente. O valor k impede que um processo seja migrado indefinidamente em determinadas situações, como no caso em que todos os EPs estão 100% ocupados. A TLBA também tem mecanismos para tratamento de falhas (MELLO, 2003)

6.2 Análise das cargas de trabalho

Para avaliar as políticas de escalonamento estudadas nesta dissertação, foram definidas algumas cargas de trabalho. Embora seja interessante avaliar um *grid* computacional com cargas caracterizadas especialmente para esse ambiente, após uma vasta pesquisa, não foram encontradas tais cargas. Existe um esforço com o objetivo de prover modelos de cargas de trabalho para *grids*. Em um dos mais significativos trabalhos, Li et al. (2004) discutem as características da execução de aplicações paralelas em um *grid* formado por *clusters* em cinco universidades. Entretanto, os logs coletados não têm informação de co-alocação e o sistema tem uma baixa utilização, que varia de 7,3% a 22%.

As cargas de trabalho utilizadas foram obtidas a partir de 3 modelos disponíveis no *Parallel Workload Archive* (*vide* seção 5.2.1). Dentre os 10 modelos disponíveis, 6 (*Feitelson96*, *Downey97*, *Jann97*, *Feitelson98*, *Lublin99*, *Cirne01*) fornecem as informações necessárias para obter uma carga de trabalho de aplicações paralelas (intervalo entre chegadas, tempo de execução e grau de paralelismo ou função de *speedup*). Entretanto, os modelos para *jobs* moldáveis

não são aplicáveis às políticas de escalonamento avaliadas neste trabalho, uma vez que elas não utilizam a informação de *speedup* fornecida pelo modelo. Diante disso, foram desconsiderados também os modelos *Downey97* e *Cirne01*. Como *Feitelson98* se trata de um *framework* para criar modelos que ainda precisa ser completado, foram selecionados, para este trabalho, os modelos *Feitelson96*, *Jann97* e *Lublin99*.

A partir desses 3 modelos foram geradas quatro cargas de trabalho. Elas foram denominadas *Feit96*, *Jann97-CTC*, *Jann97-ASCI* e *Lub99* e foram geradas, respectivamente, a partir dos modelos *Feitelson96*, *Jann97* com os parâmetros obtidos do log CTC, *Jann97* com parâmetros obtidos do log LLNL ASCI Blue-Pacific e *Lublin99*. Para gerar as cargas, utilizou-se o código disponível no *Parallel Workload Archive*[1], ajustando-se seus parâmetros. Os parâmetros de todos os modelos foram alterados para gerar *jobs* com um grau de paralelismo máximo de 64 nós e mínimo de 1 nó. Para o modelo de Lublin99, foram utilizados os parâmetros para *jobs* interativos.

De todos os códigos, exceto *Feitelson96*, as informações extraídas foram o tempo de submissão, o grau de paralelismo e o tempo de execução de cada *job*. O código do modelo *Feitelson96* forneceu valores negativos para o tempo de submissão. Assim, tal informação foi descartada e o tempo de submissão foi obtido por meio do próprio simulador (descrito abaixo), utilizando uma função de distribuição de probabilidades *Exponencial* com média 1500 segundos. A informação de que é essa a distribuição para o tempo de chegada no modelo *Feitelson96*, foi obtida a partir do código[2].

As cargas de trabalho utilizadas neste trabalho são analisadas a seguir. Nos gráficos 6.1, 6.2, 6.3, 6.4 são apresentados os histogramas de grau de paralelismo, respectivamente, para as cargas Feit96, Jann97-ASCI, Jann97-CTC e Lub99. Foi utilizada uma amostragem de 1000 *jobs*. É apresentada a freqüência de *jobs* com cada grau de paralelismo.

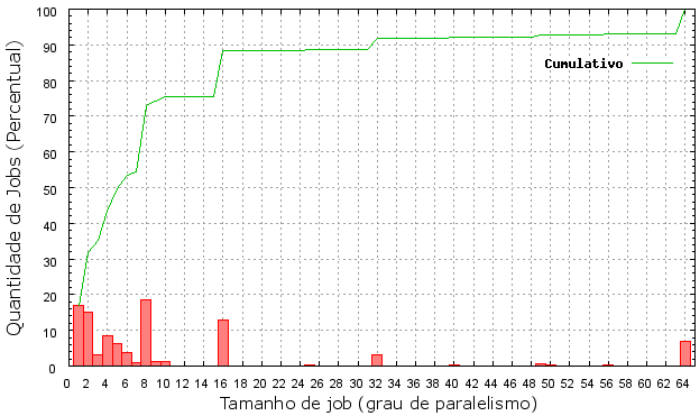


Gráfico 6.1: Histograma de Tamanho de *Job* – Feit96 (1000 *jobs*)

[1]Os códigos são bem documentados e sua compreensão é complementada pelo artigo que propõe o modelo.

[2]O código disponível data de 1997, ano em que o modelo foi estendido. Originalmente, o modelo não tinha distribuição para intervalo de chegada e alguns trabalhos assumiram o uso da *Poisson* (PWA, 2005).

Feit96 apresenta uma grande quantidade de *jobs* seqüenciais e de *jobs* com um grau de paralelismo baixo. Observando o gráfico 6.1, nota-se que mais de 18% dos *jobs* são seqüenciais e quase 90% dos *jobs* têm, no máximo, 16 processos. Outra característica observada é que aproximadamente 55% dos *jobs* paralelos têm tamanho potência de dois. Esse comportamento difere da carga Jann97-ASCI e volta a se repetir na carga Lub99. Jann97-CTC se assemelha apenas por ter uma grande quantidade de *jobs* seqüenciais. A carga Feit96 tem uma quantidade significativa de *jobs* com grau de paralelismo 64, que se aproxima de 8%, enquanto nas demais cargas tal valor não ultrapassa 2%.

No gráfico 6.2, observa-se que a quantidade de *jobs* com um mesmo grau de paralelismo é melhor distribuída. Os graus de paralelismo que têm maior freqüência variam de 1 a 8 e todos estão próximos de 6%. Além disso, a distribuição cumulativa de tamanho de *jobs* chega a 90% apenas quando o grau de paralelismo é igual a 46, enquanto que nas cargas Feit96, Jann97-CTC e Lub99 90% corresponde, respectivamente, aos graus de paralelismo 32, 21 e 22.

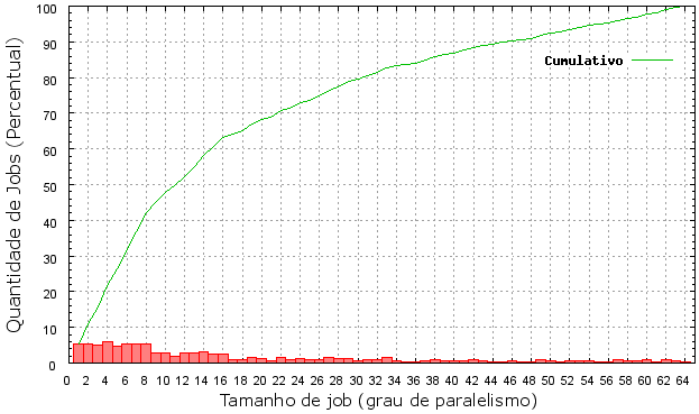


Gráfico 6.2: Histograma de Tamanho de *Job* – Jann97-ASCI (1000 *jobs*)

Em Jann97-CTC, aproximadamente 40% dos *jobs* são seqüenciais e existem muitos *jobs* pequenos: quase 70% dos *jobs* têm no máximo 8 processos. Um aspecto interessante que pode ser claramente observado no gráfico 6.3 é que a freqüência diminui à medida que o grau de paralelismo aumenta. A freqüência de *jobs* com tamanho potência de dois é bem pequena, sendo insignificante a preferência por tais *jobs*.

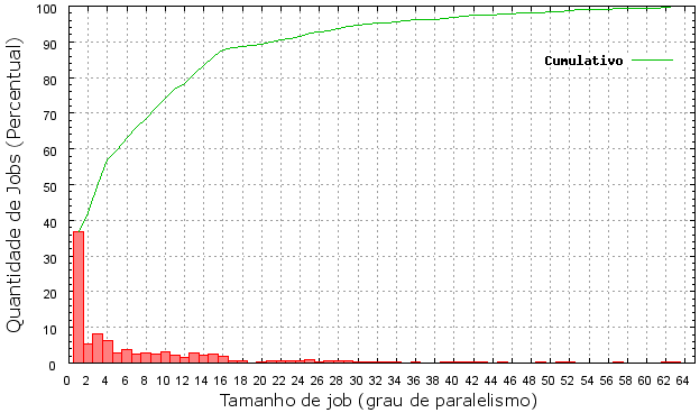


Gráfico 6.3: Histograma de Tamanho de *Job* – Jann97-CTC (1000 *jobs*)

Pode-se observar pelo gráfico 6.4 que em Lub99, ao contrário, a maioria dos *jobs* tem tamanho potência de dois: cerca de 65%. Essa carga tem apenas 12% de *jobs* seqüenciais, valor pequeno em relação às demais cargas. Novamente a quantidade de *jobs* pequenos é alta, com aproximadamente 88% dos *jobs* com grau de paralelismo 16.

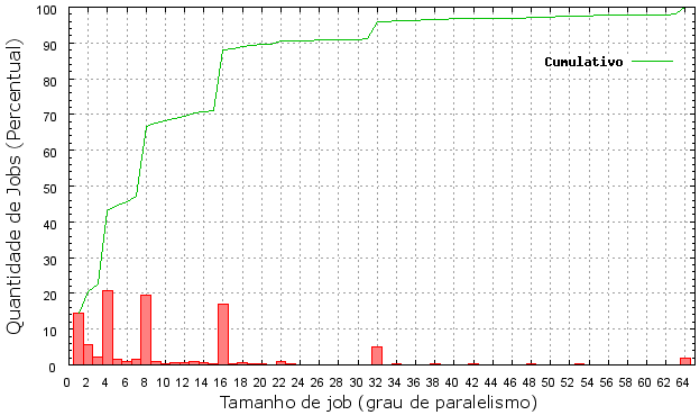


Gráfico 6.4: Histograma de Tamanho de *Job* – Lub99 (1000 *jobs*)

Outro aspecto importante a se observar nas cargas de trabalho é o tempo de execução dos *jobs*. Para tanto, foram gerados os histogramas a seguir. Nos gráficos 6.5, 6.6, 6.7 e 6.8 os tempos de execução foram divididos em intervalos de 10 minutos. Com uma amostragem de 1000 *jobs*, é apresentada a freqüência de *jobs* em cada intervalo. Por exemplo, no gráfico 6.5, observando o intervalo 0, nota-se que, aproximadamente 70% dos *jobs* tem um tempo de execução maior que 0 e menor que 10 minutos.

A respeito da carga Feit96, observa-se no gráfico 6.5 que o maior *job* da amostra tem entre 1050 e 1060 minutos de tempo de execução. Para a carga Jann97-ASCI esse valor é menor, com o maior *job* variando entre 300 e 310 minutos, enquanto para Jann97-ASCI, o tempo de execução do maior *job* se aproxima dos 1800 minutos. Lub99 apresenta um valor moderado, sendo que o maior *job* tem um tempo de execução de aproximadamente 750 minutos.

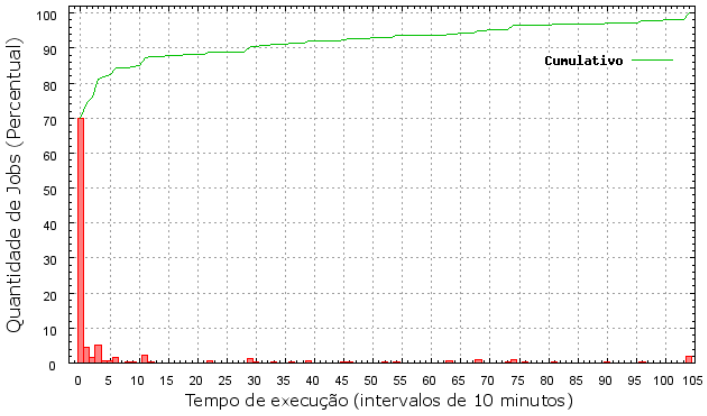


Gráfico 6.5: Histograma de Tempo de Execução – Feit96 (1000 *jobs*)

Em todas as cargas, exceto Jann97-ASCI, observa-se que há uma quantidade significativa de *jobs* grandes, sendo que estes estão mais distribuídos entre os intervalos. No gráfico 6.8,

aproximadamente 7% dos *jobs* estão entre 1 e 10 horas de execução. Essa porcentagem é ainda maior nos gráficos 6.5 e 6.7. No primeiro nota-se alguns intervalos em que a quantidade de *jobs* se aproxima de 2%, como é o caso do último intervalo.

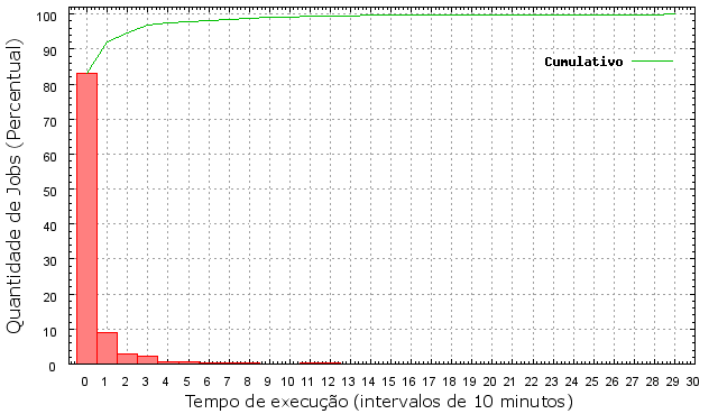


Gráfico 6.6: Histograma de Tempo de Execução – Jann97-ASCI (1000 *jobs*)

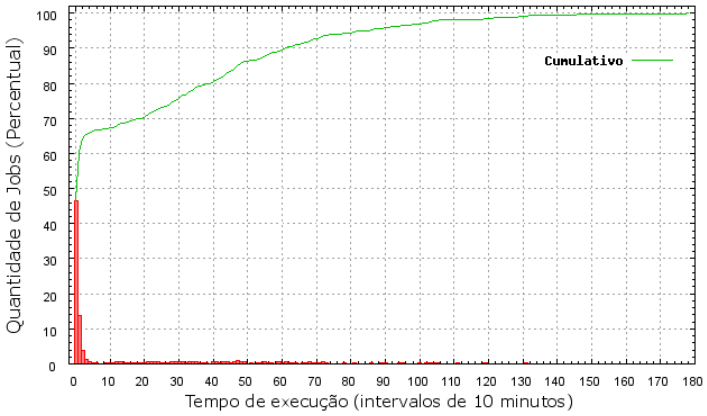


Gráfico 6.7: Histograma de Tempo de Execução – Jann97-CTC (1000 *jobs*)

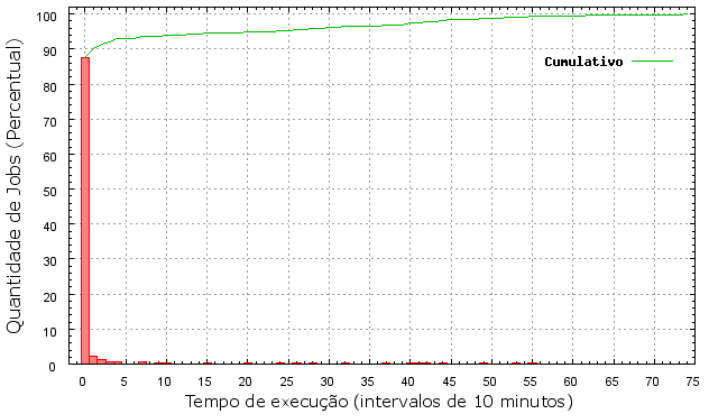


Gráfico 6.8: Histograma de Tempo de Execução – Lub99 (1000 *jobs*)

Embora existam *jobs* grandes, nota-se que *jobs* com um tempo de execução pequeno são bem mais freqüentes. Nas cargas Feit96, Jann97-ASCI, Jann97-CTC e Lub99 a quantidade de

jobs com até 10 minutos de tempo de execução se aproxima, respectivamente, de 70%, 84%, 48% e 88%. Para se ter uma visão melhor dessa tendência, a freqüência dos *jobs* com até 20 minutos é apresentada, em intervalos de 10 segundos, nos gráficos 6.9, 6.10, 6.11 e 6.12. Em todos os casos a tendência se confirma, havendo uma quantidade maior de *jobs* com tempo de execução pequeno. No gráfico 6.11, nota-se que os *jobs* são melhores distribuídos com relação ao tempo de execução para a carga Jann97-CTC.

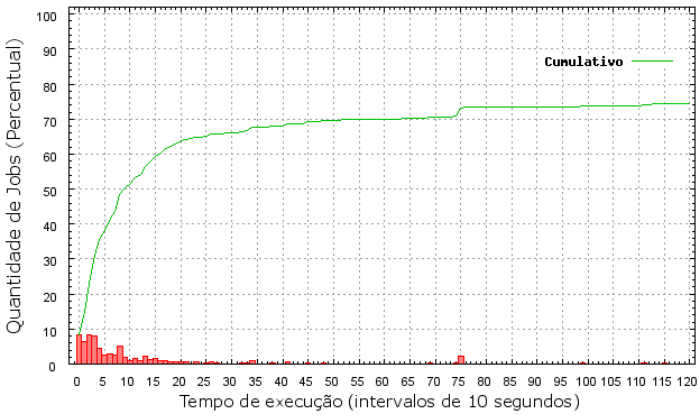


Gráfico 6.9: Histograma de Tempo de Execução – Feit96 (*Jobs* com até 20 minutos)

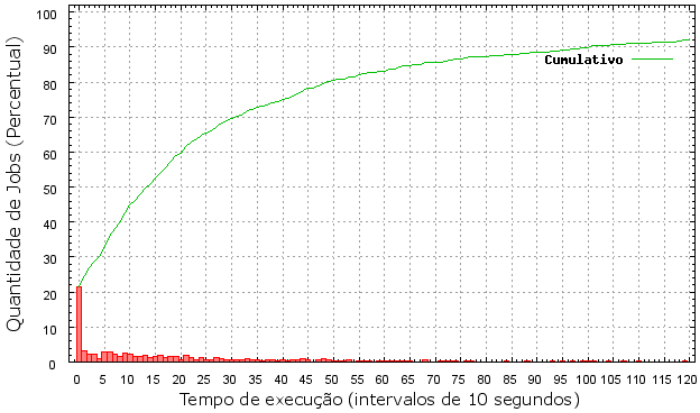


Gráfico 6.10: Histograma de Tempo de Execução – Jann97-ASCI (*Jobs* com até 20 minutos)

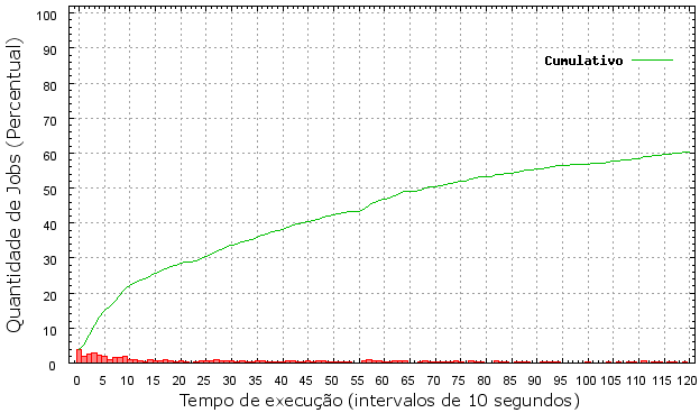


Gráfico 6.11: Histograma de Tempo de Execução – Jann97-CTC (*Jobs* com até 20 minutos)

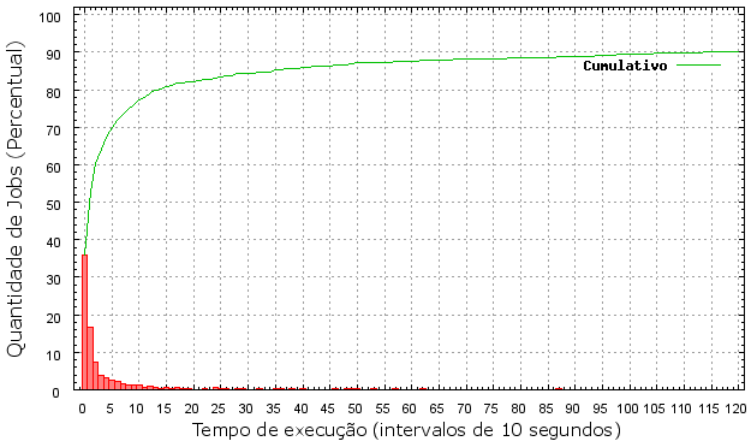
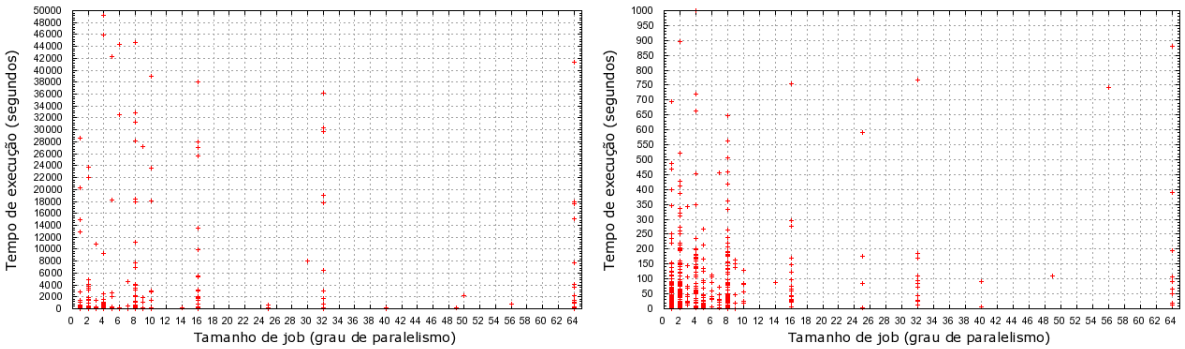


Gráfico 6.12: Histograma de Tempo de Execução – Lub99 (*Jobs* com até 20 minutos)

Observando-se os histogramas de tamanho de *job* e tempo de execução, nota-se que a freqüência de jobs pequenos e com baixo tempo de execução é bem maior. A combinação de um grau de paralelismo baixo com um tempo de execução também baixo nos jobs pode levar a uma baixa utilização do sistema. Para verificar esse aspecto nas cargas utilizadas neste trabalho foram gerados os gráficos 6.13(a), 6.13(b), 6.14(a), 6.14(b), 6.15(a), 6.15(b), 6.16(a) e 6.16(b). Eles correlacionam grau de paralelismo (eixo das abscissas) e tempo de execução dos jobs (eixo das ordenadas). Nos gráficos da esquerda foram representados todos os jobs de uma amostragem de 1000 jobs. Utilizando a mesma amostragem, nos gráficos da direita são representados apenas os jobs com até 1000 segundos de tempo de execução. Os gráficos da direita permitem visualizar melhor uma tendência percebida nas cargas de trabalho: uma freqüência alta de jobs com baixo grau de paralelismo e baixo tempo de execução.



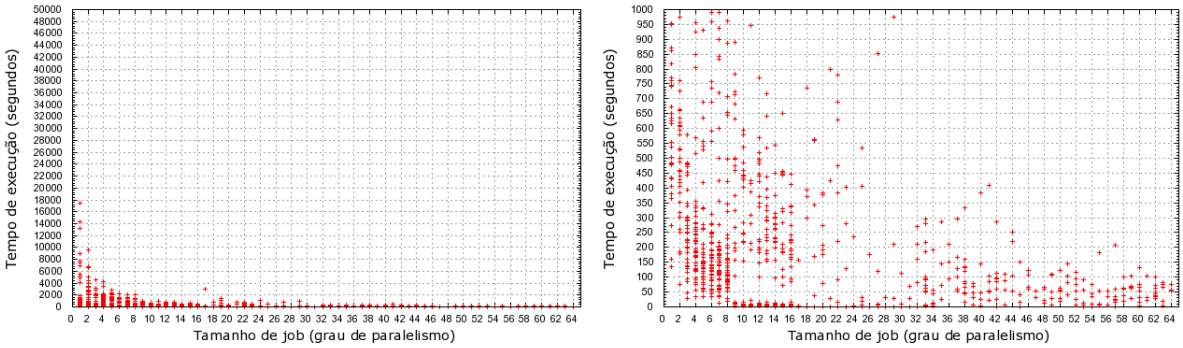
(a) 1000 jobs (b) jobs até 1000 seg.

Gráfico 6.13: Correlação Tamanho de Job e Tempo de Execução – Feit96

Nos gráficos 6.13(a) e 6.13(b) nota-se um agrupamento maior na parte inferior esquerda dos gráficos. Na carga Feit96 grande parte dos jobs possuem baixo grau de paralelismo e baixo tempo de execução. Entretanto, nota-se que existem jobs que impõem uma alta carga, havendo jobs que combinam alto grau de paralelismo ou alto tempo de execução. Existe ainda jobs com grau de paralelismo e tempo de execução altos. Por exemplo, existe um job com grau de paralelismo 64 que possui um tempo de execução de cerca de 42000 segundos. Assim, apesar

da existência de um grande número de jobs que impõem uma baixa carga, existem jobs com elevados tempo de execução e/ou grau de paralelismo.

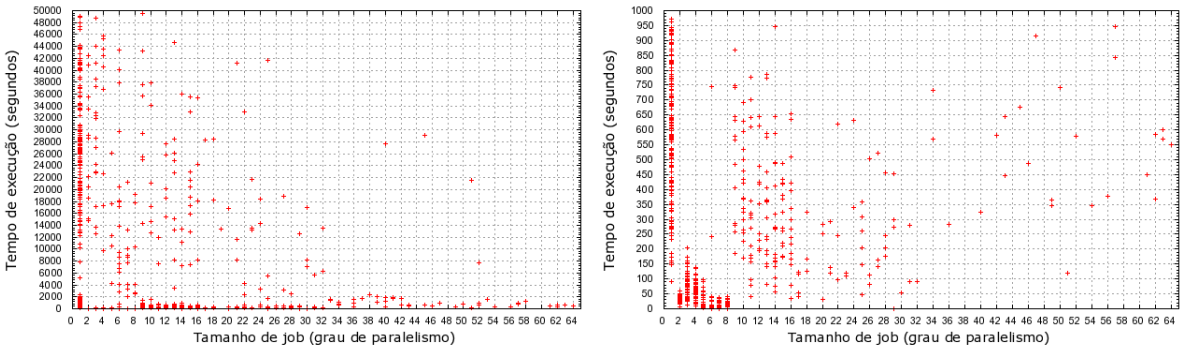
O contrário a essa tendência pode ser percebido nos gráficos 6.14(a) e 6.14(b) para a carga Jann97-ASCI. Nota-se um grande agrupamento de jobs na parte inferior do gráfico 6.14(a) e ainda mais elevado à esquerda. A tendência apresentada pela curva é que quanto maior o grau de paralelismo, menor o tempo de execução do job. Enquanto existe job seqüencial com tempo de execução de aproximadamente 18000 segundos (gráfico 6.14(a)), para o grau de paralelismo 64 um job possui um tempo de execução de aproximadamente 100 segundos, conforme pode ser observado no gráfico 6.14(b). Isso pode levar a uma baixa utilização do sistema.



(a) 1000 jobs (b) jobs até 1000 seg.

Gráfico 6.14: Correlação Tamanho de Job e Tempo de Execução – Jann97-ASCI

A correlação entre grau de paralelismo e tempo de execução na carga Jann97-CTC se assemelha ao comportamento da carga Feit96. Mas existem algumas diferenças. No gráfico 6.15(a) nota-se uma freqüência pequena de jobs com um alto tempo de execução para os jobs com grau de paralelismo maior que 32. Nota-se ainda um grande número de jobs seqüenciais com elevado tempo de execução. Apesar de existir um agrupamento de jobs pequenos e com baixo tempo de execução, os jobs estão bastante dispersos em toda a área esquerda do gráfico, com uma quantidade significativa de jobs com tempo de execução alto.

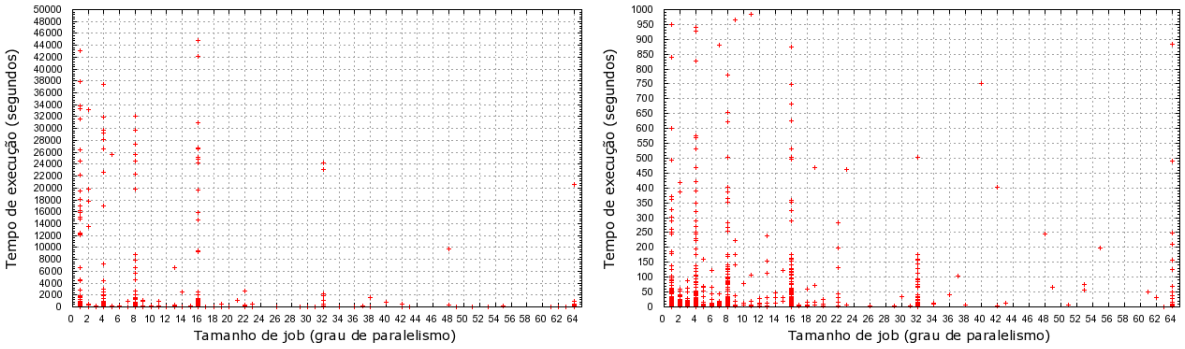


(a) 1000 jobs (b) jobs até 1000 seg.

Gráfico 6.15: Correlação Tamanho de Job e Tempo de Execução – Jann97-CTC

Na carga Lub99 a grande maioria dos jobs com elevado tempo de execução são os jobs

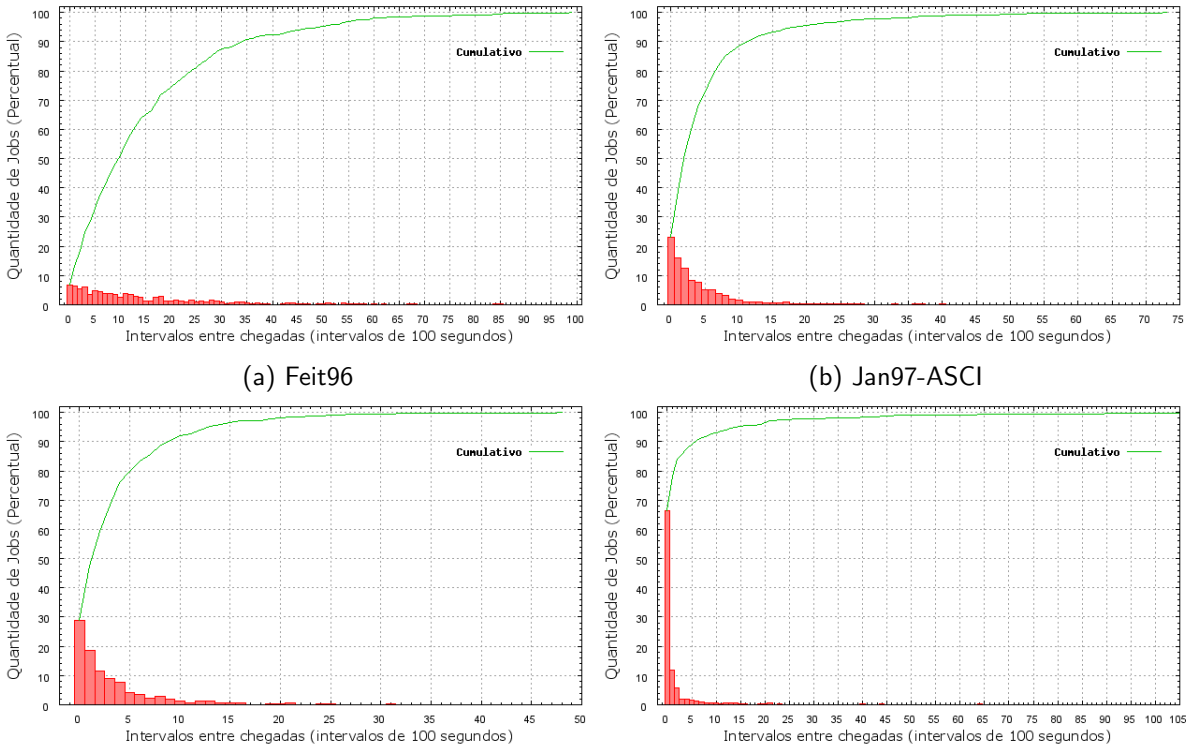
com grau de paralelismo potência de dois, conforme pode ser observado nos gráficos 6.16(a) e 6.16(b). Essa carga, assim como na carga Feit96, apresenta um número significativo de jobs que impõem alta carga ao sistema. Essa característica só não é percebida na carga Jann97-ASCI.



(a) 1000 jobs (b) jobs até 1000 seg.

Gráfico 6.16: Correlação Tamanho de Job e Tempo de Execução – Lub99

Complementando o estudo das cargas de trabalho utilizadas, a submissão de jobs ao sistema é analisada nos gráficos a seguir. A submissão de jobs ao sistema, modelada por meio do intervalo entre chegadas, é um importante parâmetro para especificar uma carga de trabalho para sistemas abertos on-line (*vide* seções 5.1 e 5.2 para informações sobre sistemas abertos on-line e sobre o intervalo de chegada como um dos componentes de cargas de trabalho para avaliar tais sistemas).



(a) Feit96 (b) Jan97-ASCI

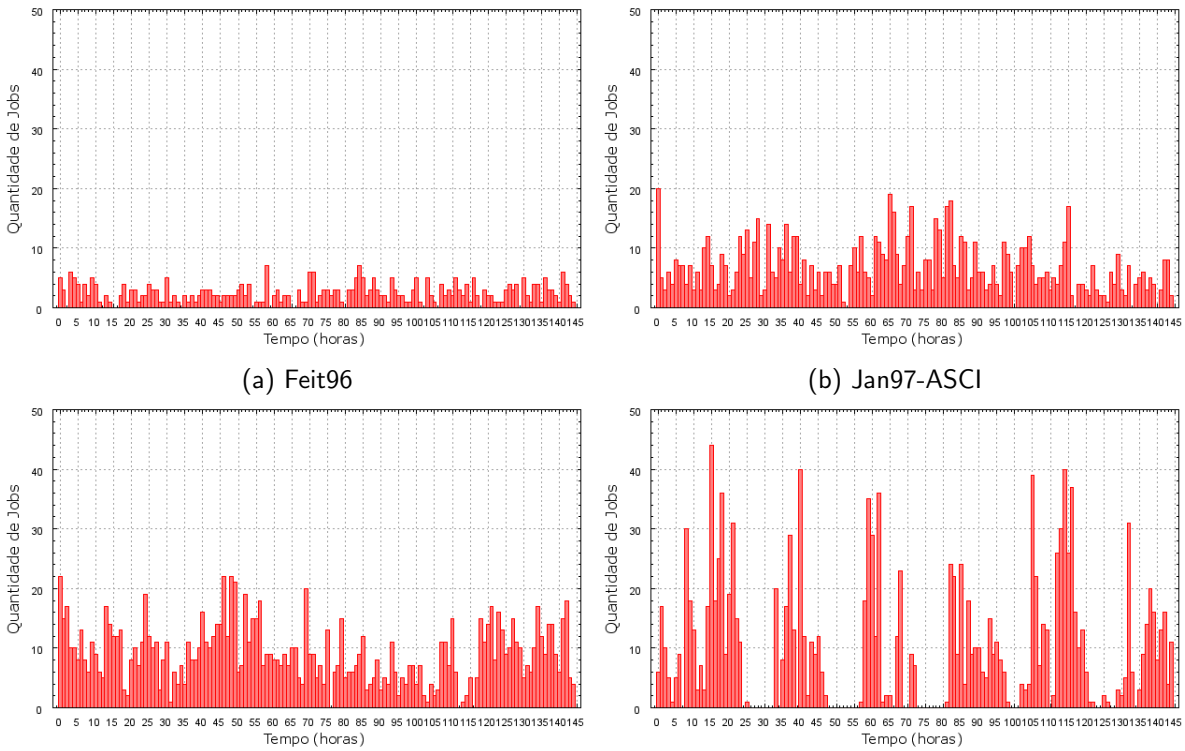
(c) Jan97-CTC (d) Lub99

Gráfico 6.17: Histogramas de Intervalos entre Chegadas (1000 jobs)

Os gráficos 6.17(a), 6.17(b), 6.17(c) e 6.17(d) representam os intervalos entre chegadas de 1000 jobs, respectivamente para as cargas Feit96, Jan97-ASCI, Jan97-CTC e Lub99. Os intervalos entre chegadas foram agrupados no gráfico em intervalos de 100 segundos, ou seja, se um job possui um intervalo entre chegadas igual a 350 segundos, esse job é acrescentado ao percentual do 3º intervalo.

Em todos os gráficos nota-se uma concentração maior de intervalos entre chegadas menores. No gráfico 6.17(a), para a carga Feit96, 80% dos jobs tem um intervalo entre chegadas próximo ou inferior a 2500 segundos. Para a carga Jan97-ASCI, representada no gráfico 6.17(b), este índice é alcançado próximo dos 700 segundos. Para as cargas Jan97-CTC e Lub99 80% dos jobs tem um intervalo entre chegadas inferior a, respectivamente, 500 segundos e 300 segundos, conforme pode ser observado nos gráficos 6.17(c) e 6.17(d).

Os gráficos 6.18(a), 6.18(b), 6.18(c) e 6.18(d) fornecem uma representação do padrão de submissões dos jobs em relação ao tempo, respectivamente para as cargas Feit96, Jan97-ASCI, Jan97-CTC e Lub99. Os gráficos representam 144 horas de execução de um sistema paralelo. O tempo é dividido em intervalos de 1 hora e é dada a quantidade de jobs submetidos em cada um desse intervalos.



(a) Feit96 (b) Jan97-ASCI

(c) Jan97-CTC (d) Lub99

Gráfico 6.18: Correlação entre intervalo de chegada e quantidade de jobs submetidos

Os gráficos acima ressaltam o comportamento do usuário que submete os jobs. Nota-se nos gráficos que após intervalos com alta submissão existe intervalos em que a quantidade de jobs submetidos ao sistema cai bastante, havendo intervalos sem submissão. Esse comporta-

mento é destacado em (FEITELSON; RUDOLPH, 1998). O usuário reage ao tempo de resposta obtido no sistema, ou seja, se o tempo de resposta é pequeno os usuários submetem mais jobs. Quando esse tempo de resposta aumenta, devido à existência de muitos jobs no sistema, o usuário para de submeter jobs.

Uma informação essencial para esse trabalho e que não é fornecida nas cargas de trabalho descritas acima é a comunicação entre os processos utilizando a rede. Assim, para avaliar o impacto da comunicação sobre o tempo de resposta obtido com o uso das políticas de escalonamento, cada umas das 4 cargas de trabalho descritas acima foram combinadas com 3 variações do uso da rede de comunicação, gerando um total de 12 cargas de trabalho.

A primeira variação é sem uso da rede, ou seja, os processos não se comunicam por meio da rede, sendo, nesse caso, aplicações "*bag-of-tasks*". Na segunda variação é utilizada uma distribuição de probabilidades para o envio e o recebimento de mensagens utilizando a rede de comunicação que segue uma *Poisson* com média 100 Kbps. Na terceira variação também é utilizada uma distribuição de probabilidades para modelar o uso da rede de comunicação e, nesse caso, utiliza-se uma Poisson com média 1000 Kbps. Essa função de distribuição de probabilidades foi adotada com base nos experimentos conduzidos em Ishii (2004), que demonstraram que a *Poisson* segue o mesmo comportamento da função *Hiper-Exponencial*, definida no trabalho de Chodnekar et al. (1997).

6.3 Simulador

A avaliação de desempenho conduzida neste trabalho utiliza simulação. Para realizar tal simulação, foi adotado o modelo *Unified Modeling for Predicting Performance* (UniMPP), definido por Mello e Senger (2006). Por meio desse modelo, é possível predizer o tempo de resposta de aplicações paralelas em ambientes distribuídos heterogêneos, como *clusters* e *grids* computacionais.

O UniMPP unifica os modelos de Amir (2000) e de Mello et al. (2002) para consumo de UCP, os modelos de Culler et al. (1993) e de Sivasubramaniam (1997) para tempo gasto na transmissão de mensagens e os modelos de Chodnekar et al. (1997) e Vetter e Mueller (2003) para volume de mensagens e distribuições de probabilidades de geração de mensagens. O UniMPP agrega ainda características de atrasos voluntários das aplicações e acesso a disco. Mello e Senger (2006) conduziram experimentos a fim de validar o modelo e demonstraram que o o mesmo é capaz de predizer, com precisão, o comportamento da execução de aplicações paralelas em ambientes distribuídos heterogêneos. O tempo de resposta gerado por meio do modelo foi similar ao observado em execuções reais, apresentando erros com valores próximos a 0.

Nesse modelo, um processo p_j chega ao sistema no instante a_j, seguindo uma função de distribuição de probabilidades e é iniciado no computador c_i, que mantém uma fila $q_{i,t}$ de seus

processos no instante t. Cada computador c_i é composto pela sêxtupla $\{pc_i, mm_i, vm_i, dr_i, dw_i, lo_i\}$, onde: pc_i é a potência computacional total do computador i, medida em instruções por unidade de tempo (MIPS, por exemplo); mm_i é o total de memória principal; vm_i é o total da memória virtual; dr_i é o *throughput* de leitura do disco rígido; dw_i é o *throughput* de escrita no disco rígido e lo_i é o tempo consumido no envio e recebimento de mensagens. Os processos das aplicações paralelas são representadas pela sêxtupla $\{mp_j, sm_j, pdfdm_j, pdfdr_j, pdfdw_j, pdfnet_j\}$, onde: mp_j representa o consumo de processamento, sm_j é a quantidade de memória estática alocada pelo processo, $pdfdm_j$ é a função de distribuição de probabilidades para a ocupação dinâmica de memória, $pdfdr_j$ é a função de distribuição de probabilidades para leitura no disco rígido, $pdfdw_j$ é a distribuição de probabilidades para escrita no disco rígido, $pdfnet_j$ é a distribuição de probabilidades para envio e recebimento de mensagens utilizando a rede de comunicação.

O UniMPP fornece as equações para quantificar os atrasos impostos a um processo e para obter o tempo de resposta do mesmo. Quando um processo p_j é executado em um computador c_i, seu tempo de execução TE_{p_j,c_i} é dado pela equação 6.1. Essa equação se aplica a condições ideais, ou seja, não há nenhum atraso causado por competição, por acesso à memória ou a dispositivos de I/O. Assim, TE_{p_j,c_i} é dado pela razão entre mp_j e pc_i, que devem ser expressos na mesma métrica, como MIPS ou MFLOPS.

$$TE_{p_j,c_i} = \frac{mp_j}{pc_i} \tag{6.1}$$

A equação 6.2 fornece o tempo de execução de um processo considerando o atraso gerado pelo uso das memórias principal e virtual. TE_{p_j,c_i} é dado pela equação 6.1 e α representa uma porcentagem obtida por meio de uma função de atraso. A função de atraso é gerada por uma ferramenta de *benchmark* (explicada a seguir) em que no eixo x é dada a ocupação da memória até que toda a memória virtual seja usada e no eixo y é dado o valor de α (o atraso imposto à execução do processo devido à ocupação da memória). Por meio de experimentos, Mello e Senger (2006) observaram que a função de atraso por ocupação da memória tem um comportamento linear durante o uso da memória principal e exponencial a partir do momento em que a memória virtual começa a ser usada. Assim, são consideradas 2 funções para obter o valor α. As funções usadas neste trabalho são detalhadas na seção 7.1.

$$TEM_{p_j,c_i} = TE_{p_j,c_i} * (1 + \alpha) \tag{6.2}$$

Considerar o atraso na execução de um processo causado pelo uso da memória não é suficiente para predizer com precisão seu tempo de resposta. Assim, o modelo UniMPP foi complementado com equações para quantificar atrasos gerados pelo acesso ao disco, tanto leitura quanto escrita (equações 6.3 e 6.4, respectivamente) e atrasos gerados pelo envio e recebimento

de mensagens na rede de comunicação (equação 6.5).

$$SLDR_{p_j,c_i} = \sum_{k=1}^{nr} \frac{bsize_k}{dr_i} + wtdr_k \tag{6.3}$$

$$SLDW_{p_j,c_i} = \sum_{k=1}^{nw} \frac{bsize_k}{dw_i} + wtdw_k \tag{6.4}$$

A equação 6.3 modela o atraso imposto à execução de um processo gerado por leitura ao disco rígido, em que nr é o número de acesso de leitura; $bsize$ representa o tamanho do *buffer* de dados, dr_i representa o *throughput* para leitura no disco e $wtdr_k$ representa o tempo de espera para usar o recurso. Para operações de escrita no disco rígido é fornecida a equação 6.4, em que nw é o número de acessos de escrita, $bsize$ representa o tamanho do buffer de dados para escrita, dw_i representa o *throughput* para escrita no disco rígido e $wtdw_k$ representa o tempo de espera para usar o recurso.

Os atrasos causados pelo uso da rede de comunicação variam de acordo com diversos fatores: largura de banda, latência, *overhead* do protocolo de comunicação, tamanho da mensagem. O atraso causado pelo envio e recebimento de mensagens é definido na equação 6.5, na qual nm é o número de mensagens recebidas e enviadas; $\theta_{s,k}$, dado pela equação 6.6, é o tempo gasto para enviar e receber mensagens na rede de comunicação, sem considerar a espera pelo uso do recurso e wtn_k representa o tempo de fila para enviar ou receber uma mensagem quando o recurso está ocupado. Na equação 6.6, $o_{s,k}$ representa o *overhead* da mensagem que é multiplicada por 2 para quantificar o empacotamento no emissor e o desempacotamento no receptor e $l_{s,k}$ representa a latência para transmitir uma mensagem.

$$SLN_{p_j,c_i} = \sum_{k=1}^{nm} \theta_{s,k} + wtn_k \tag{6.5}$$

$$\theta_{s,k} = 2 * o_{s,k} + l_{s,k} \tag{6.6}$$

No modelo UniMPP, é proposta a equação 6.7, que unifica os modelos de atrasos definidos nas equações acima. Por meio dessa equação, é possível predizer o tempo de resposta de um processo p_j em um computador c_i, em que lz é o atraso voluntário do processo, gerado por chamadas de sistema *sleep*. Quando há migração do processo, os parâmetros (capacidade da rede, memória, disco) do computador em que o processo está executando podem alterar os atrasos impostos ao processo, assim o tempo de resposta de um processo p_j que migra entre n computadores é obtido pela equação 6.8.

$$SL_{p_j} = SL_{p_j,c_i} = TEM_{p_j,c_i} + SLDR_{p_j,c_i} + SLDW_{p_j,c_i} + SLN_{p_j,c_i} + lz \quad (6.7)$$

$$SL_{p_j} = \sum_{k=1}^{n} SL_{p_j,c_k} \quad (6.8)$$

O modelo UniMPP foi implementado em um simulador, denominado SchedSim[3] (MELLO; SENGER, 2006). Ele foi implementado em Java (DEITEL; DEITEL, 2001) e utiliza conceitos de orientação a objetos, que facilitam sua extensão e adição de funcionalidades. O simulador é parametrizado com a configuração de um ambiente computacional (capacidade de UCP, memória principal e virtual, *throughput* de disco, capacidade da rede de comunicação) e recebe processos para execução, cujos dados podem ser obtidos de funções de distribuição de probabilidade ou de arquivos de dados. Utilizando esse simulador, é possível avaliar diferentes estratégias de escalonamento.

A fim de obter uma parametrização mais próxima de um ambiente de execução real, é definido, juntamente com o simulador, um conjunto de *benchmarks* que permitem extrair informação de um sistema real. Esses benchmarks foram utilizados para parametrizar as simulações conduzidas neste trabalho. Essa parametrização é detalhada na seção 7.1. As ferramentas de *benchmark* são:

- **mips** – mede a capacidade do processador em Milhões de Instruções Por Segundo (MIPS).
- **memo** – mede os atrasos causados pela troca de contexto entre processos, durante a ocupação de toda a memória principal e virtual.
- **discio** – mede o *throughput* médio de escrita (com e sem buffer) e o *throughput* médio de leitura em discos rígidos e em dispositivos de armazenamento remotos (via NFS).
- **net** – permite avaliar o tempo gasto no envio e recebimento de mensagens sobre a rede de comunicação, por meio de duas aplicações, uma cliente e outra servidora.

Visando realizar a avaliação de desempenho aqui conduzida, o simulador SchedSim (MELLO; SENGER, 2006) foi utilizado e estendido para se adequar às necessidades deste trabalho.

A primeira extensão foi a implementação de outras duas políticas. O simulador permite a implementação de diferentes algoritmos de escalonamento, empregando o conceito de Abstração. Dentre as dez políticas utilizadas, duas foram implementadas neste trabalho: a política

[3] A versão original de Mello e Senger (2004) está disponível em http://143.107.183.115/mello/outros/schedSim-v0.0.1.tar.gz

Round Robin e a política NBSP. As políticas Random, Disted, Global, Central, Lowest e TLBA foram implementadas no SchedSim durante o trabalho de Mello e Senger (2004). As políticas DPWP e GAS foram implementadas por Senger (2005). A política DPWP utilizada neste trabalho é uma versão estendida. Na versão original, a DPWP utiliza como índice de carga o tamanho da fila de processos normalizado pela capacidade do processador. Neste trabalho, ao invés do tamanho da fila é utilizada a ocupação dos processos (*vide* seção 6.1.3).

Originalmente, o simulador obtém o intervalo entre chegadas utilizando uma função de distribuição de probabilidades. Para permitir também a obtenção do intervalo entre chegadas a partir de um arquivo, o simulador foi modificado. Essa modificação permite que o intervalo de chegadas seja obtido a partir do campo *Tempo de submissão* do formato padrão para cargas de trabalho (SWF). O intervalo entre chegadas de um job_i e um job_{i-1} é obtido pela diferença entre $Tempo\ de\ submissao_{job_i}$ e $Tempo\ de\ submissao_{job_{i-1}}$.

Outra extensão aplicada ao simulador se refere à geração de mensagens na rede de comunicação. Ela foi necessária para permitir a geração de uma carga de rede pesada e para obter mais controle sobre o número de mensagens geradas. A cada *quantum* do processador fornecido para a execução de um processo, este gera um dado número de mensagens. Com a extensão no simulador, o número de mensagens geradas é dado pela taxa de comunicação nesse intervalo dividida por 32 bytes, o tamanho (fixo) de cada mensagem. A taxa de comunicação de um processo é dada por uma função de distribuição de probabilidades. Neste trabalho, quando há uso da rede, a taxa de comunicação segue uma *Poisson* com média de 100 Kbps ou 1000 Kbps.

Para tornar o simulador mais flexível, foram implementados dois geradores de configurações de clusters e NOW's. O primeiro utiliza dados de computadores parametrizados por meio dos *benchmarks* fornecidos com o simulador e permite a replicação de tais dados. O inconveniente de tal opção é que quando se deseja um sistema de larga escala com base em poucos computadores parametrizados, obtém-se um sistema com pouca heterogeneidade. O segundo gerador resolve esse problema. Ele utiliza funções de distribuição de probabilidades para obter cada um dos parâmetros que compõem um computador segundo o modelo UniMPP. A segunda opção foi utilizada neste trabalho e será detalhada na seção 7.1. Os dados gerados a respeito do sistema é armazenado em formato XML, podendo ser interpretado, posteriormente, pelo simulador.

6.4 Considerações Finais

Este capítulo destinou-se a apresentar o estudo conduzido neste trabalho. Foram discutidas as políticas de escalonamento avaliadas e as cargas de trabalho a que tais políticas foram submetidas. Todas as políticas são voltadas a ambientes computacionais distribuídos e buscam minimizar a comunicação para realizar o escalonamento. Exceto a política NBSP e GAS, todas as políticas são voltadas a aplicações *CPU-Bound* e não tratam da comunicação entre os proces-

sos que compõem uma aplicação. A NBSP é voltada a aplicações *Network-Bound* e a política GAS contempla o tratamento da comunicação entre os processos que compões um *job*.

A análise das cargas demonstrou características distintas em cada uma. A carga *Jann97-ASCI* se distingue bastante das demais, apresentando uma baixa requisição de serviço. A quantidade de serviço requerida por cada carga de trabalho também difere. As cargas *Feit96* e *Lub99* apresentam uma grande freqüência de *jobs* com tamanho potência de 2, enquanto isso não ocorre nas demais (Jann97-ASCI e Jann97-CTC). Em todas as cargas, a freqüência de *jobs* com um baixo grau de paralelismo (menor que 16) e um baixo tempo de execução (menos de 1 hora) ultrapassa 50%. A combinação dessas cargas com as cargas de rede garante que as principais características das políticas e dos sistemas computacionais serão consideradas.

A avaliação de desempenho foi conduzida com a utilização de simulação. Assim, o simulador utilizado, assim como o modelo que ele implementa foram discutidos. O modelo *UniMPP* mostra-se bastante adequado à avaliação realizada neste trabalho, fornecendo resultados precisos. A flexibilidade do simulador desenvolvido em Java e orientado a objetos garantiu que as extensões necessárias a este trabalho pudessem ser implementadas. As extensões foram a implementação de novas políticas, o modo de leitura do intervalo de chegada, a geração de mensagens para a rede de comunicação e os geradores de sistemas computacionais para parametrizar o simulador.

Desse modo, foi possível conduzir o estudo proposto neste trabalho, obtendo bons resultados da comparação das diferentes políticas sob as diferentes cargas de trabalho em *clusters* e *grids*. No capítulo seguinte são apresentados a parametrização do simulador e os resultados obtidos na avaliação de desempenho.

Capítulo

7

Resultados

Neste capítulo, é discutida a parametrização do modelo de simulação utilizado neste trabalho, assim como os resultados obtidos. Conforme discutido anteriormente, a simulação foi utilizada para obter o resultado da execução das políticas de escalonamento. O resultado de cada execução é expresso em Tempo de Resposta Médio. A escolha por essa métrica é discutida na seção 5.3.

7.1 Parametrização do Modelo de Simulação

Simulação oferece bons resultados quando se utiliza um modelo adequado. O modelo UniMPP (*vide* seção 6.3), utilizado neste trabalho, oferece resultados bastante precisos, permitindo cumprir um dos objetivos propostos para o estudo conduzido nesta dissertação: obter resultado o mais próximo possível de um ambiente real. Para tanto, o modelo precisa ser parametrizado com dados da configuração de *clusters* e *grids*, assim como os encontrados em ambientes de produção e acadêmico. Tais sistemas são, geralmente, compostos por um elevado número de computadores interligados por uma rede de alta velocidade. Entretanto, tais *clusters* e *grids* não estão disponíveis para efetuar a parametrização.

A solução encontrada foi gerar tais sistemas de forma estatística, utilizando funções de distribuição de probabilidades para obter cada um dos parâmetros necessários à avaliação de desempenho. As distribuições usadas para cada um dos parâmetros que compõem a sêxtupla que modela um computador c_i no modelo UniMPP foram:

- pc_i – *Normal* com média 1500 MIPS
- mm_i – *Normal* com média 1048576 KB
- vm_i – *Normal* com média 1048576 KB

- dr_i – *Normal* com média 50 MB/s
- dw_i – *Normal* com média 35 MB/s
- lo_i – *Exponencial* com média 1 segundo

Para aproximar os sistemas gerados de sistemas reais, foram utilizados alguns limiares, obtidos de um sistema real parametrizado por meio dos *benchmarks* providos com o simulador. Essa parametrização foi realizada nos laboratórios de computação do LaSDPC e da Universidade Estadual de Ponta Grossa, Paraná neste trabalho e no trabalho de Senger (2005). A figura 7.1 esquematiza esse sistema. Ele possui 4 redes distintas. Os computadores denominados antrax, pegasus e andromeda formam um cluster interligado por uma rede Gigabit Ethernet. Os computadores w3_uepg01 e w3_uepg02 são interligados aos demais através da Internet, assim como em *grids* computacionais.

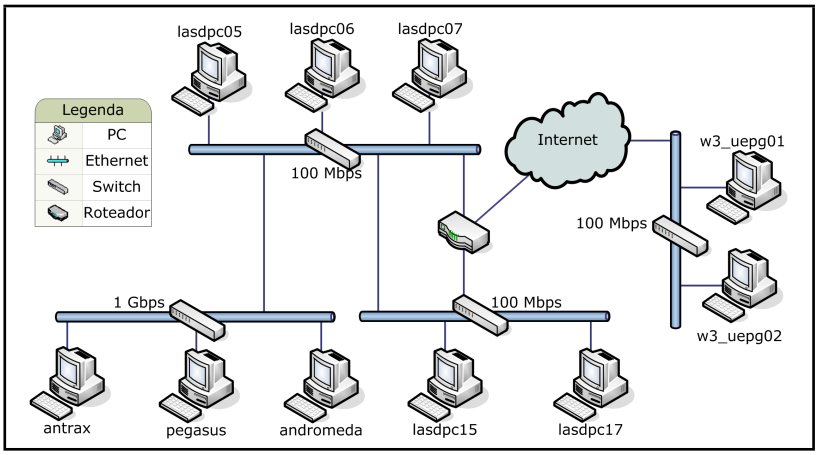


Figura 7.1: Diagrama do sistema utilizado na parametrização

As capacidades de processamento, memória e disco de cada computador são apresentadas na tabela 7.1 e o tempo gasto para enviar e receber mensagens na rede de comunicação, $\theta_{s,k}$, entre cada par de computadores é mostrado na tabela 7.2.

Tabela 7.1: Configuração do Sistema Parametrizado

			MEMÓRIA (MB)		DISCO (MB)	
NÓ	**DESCRIÇÃO**	**UCP (MIPS)**	**PRINCIPAL**	**VIRTUAL**	**LEITURA**	**ESCRITA**
0	antrax	1135.31	1024.0	1024.0	76.33	66.85
1	andromeda	1145.86	1024.0	1024.0	76.28	65.55
2	pegasus	1148.65	1024.0	1024.0	75.21	66.56
3	lasdpc05	187.54	256.0	256.0	17.82	16.60
4	lasdpc06	313.38	256.0	256.0	10.25	6.70
5	lasdpc07	151.50	64.0	128.0	7.30	7.21
6	lasdpc15	1052.87	256.0	256.0	14.42	15.74
7	lasdpc17	1052.87	256.0	256.0	14.42	15.74
8	w3_uepg01	350.00	512.0	512.0	10.25	6.70
9	w3_uepg02	350.00	512.0	512.0	10.25	6.70

Tabela 7.2: Tempo gasto para enviar e receber mensagens na rede de comunicação, $\theta_{s,k}$, para cada par de computadores do sistema parametrizado (em segundos)

EP	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	0.000000	0.000040	0.000040	0.000087	0.000074	0.000136	0.000404	0.000404	0.063291	0.063291
1	0.000040	0.000000	0.000040	0.000081	0.000074	0.000173	0.000440	0.000440	0.060240	0.060240
2	0.000040	0.000040	0.000000	0.000081	0.000074	0.000173	0.000440	0.000440	0.060240	0.060240
3	0.000087	0.000081	0.000081	0.000000	0.000161	0.000161	0.000442	0.000442	0.064020	0.064020
4	0.000074	0.000074	0.000074	0.000161	0.000000	0.000145	0.000420	0.000420	0.064000	0.064000
5	0.000136	0.000173	0.000173	0.000161	0.000145	0.000000	0.000492	0.000492	0.064000	0.064000
6	0.000404	0.000440	0.000440	0.000442	0.000420	0.000492	0.000000	0.000092	0.260563	0.260563
7	0.000404	0.000440	0.000440	0.000442	0.000420	0.000492	0.000092	0.000000	0.260563	0.260563
8	0.063291	0.060240	0.060240	0.064020	0.064000	0.064000	0.260563	0.260563	0.000000	0.000092
9	0.063291	0.060240	0.060240	0.064020	0.064000	0.064000	0.260563	0.260563	0.000092	0.000000

Observando os valores do sistema parametrizado, verifica-se que existem limites entre as capacidades de cada elemento de um computador. Por exemplo, dificilmente um computador com 1200 MIPS terá apenas 16 MB de memória principal e 1024 MB de memória virtual. A fim de gerar dados de computadores, mantendo os limites entre as capacidades de componentes próximos aos limites em um computador real, foram definidos alguns limiares.

Para gerar a capacidade de processamento do computador i, pc_{c_i}, utiliza-se apenas a distribuição *Normal* com média 1500. A fim de gerar a quantidade de memória principal, mm_{c_i}, limiares são considerados da seguinte maneira: se $0.68 \geq \frac{pc_{c_i} * 1024}{mm_{c_i}} \leq 4.1$, então mm_{c_i} é considerado, senão um novo valor de mm_{c_i} é gerado, utilizando a distribuição, até que a relação seja verdadeira. Os valores dos limiares, 0.68 e 4.1, foram obtidos do menor e do maior valor da relação $\frac{MIPS}{memoria\ principal}$ observada nos computadores parametrizados com *benchmarks*, conforme a tabela 7.1.

Para obter os valores da memória virtual, mv_{c_i}, do *throughput* de escrita, dw_{c_i}, e do *throughput* de leitura, dr_{c_i}, do disco rígido, uma idéia semelhante foi aplicada. Se $0.5 \geq \frac{mm_{c_i}}{mv_{c_i}} \leq 1.0$, então o valor de mv_{c_i} é considerado, senão mv_{c_i} é gerado novamente utilizando a distribuição. Para o disco, se a relação $0.916 \geq \frac{dr_{c_i}}{dw_{c_i}} \leq 1.530$ for satisfeita, então dr_{c_i} e dw_{c_i} são considerados, senão dw_{c_i} é gerado novamente, até que a relação seja verdadeira. Os limiares para a memória virtual e o disco foram obtidos da mesma maneira. Os valores 0.5 e 1.0 são, respectivamente, o menor e o maior valor da relação $\frac{memoria\ principal}{memoria\ virtual}$ observada na tabela 7.1. Os valores 0.916 e 1.530 foram obtidos da relação $\frac{throughput\ leitura\ em\ disco}{throughput\ escrita\ em\ disco}$, respectivamente para as nós 6 e 4 da tabela 7.1.

Para gerar a matriz de tempo gasto para enviar e receber mensagens na rede de comunicação, $\theta_{s,k}$, entre cada par de computadores, é considerado, primeiramente, o número de redes,

nn. Utilizando uma função de distribuição de probabilidade *Normal* com média nn, cada computador é colocado em uma das nn redes. Considerando-se um tempo nulo para a comunicação entre processos no mesmo computador, os demais casos são dados pelo seguinte: se os dois computadores fazem parte da mesma rede, então $\theta_{s,k} = 0.00004$ (esse é o valor de $\theta_{s,k}$ para 2 computadores em uma rede Gigabit Ethernet), senão $\theta_{s,k}$ segue uma distribuição *Exponencial* com média 1 segundo. Os valores obtidos são observados na comunicação entre 2 computadores que utilizam a Internet.

Dessa forma, foram gerados os 3 sistemas computacionais distintos para parametrizar o simulador. Foram gerados 2 *clusters*, 1 com 32 e outro com 128 nós. Nesses dois *clusters*, foi considerado uma única rede e, assim, todos os pares de computadores têm $\theta_{s,k}$ igual a 0.00004. O terceiro sistema gerado é um *grid* com 512 nós. Cada site foi gerado com uma média de 8 computadores, ou seja, $nn = 64$. No *grid*, computadores no mesmo site têm $\theta_{s,k}$ igual a 0.00004, enquanto computadores em sites distintos possuem valores de $\theta_{s,k}$ elevados para representar o uso da Internet.

Utilizando o sistema parametrizado, foram obtidas as equações que fornecem o valor de α na equação 6.2 e que permitem quantificar os atrasos impostos pelo uso da memória principal e virtual. Essas equações foram obtidas por meio da execução do *benchmark* memo. Enquanto a quantidade de memória utilizada é menor que a memória principal a equação $\alpha = y = 0.0069x - 0.089$ fornece o valor de α. Nesta equação, x é a quantidade de memória utilizada e y é o valor de α. A partir do momento em que a memória virtual passa a ser utilizada $\alpha = y = 1.1273 * e^{0.0045x}$.

A parametrização dos processos que chegam ao sistema é obtida das cargas de trabalho definidas na seção 6.2. Das cargas Feit96, Jann97-ASCI, Jann97-CTC e Lub99 são extraídas as distribuições de probabilidade para intervalo entre chegadas, consumo de UCP por processo e grau de paralelismo. A distribuição de probabilidades para o envio e o recebimento de mensagens utilizando a rede de comunicação segue uma *Poisson* com média 0 Kbps, 100 Kbps ou 1000 Kbps.

7.2 Simulações

Utilizando a parametrização apresentada na seção anterior, diversas simulações foram conduzidas. Como discutido, simulação oferece bons resultados, próximos de uma execução real. Além disso, uma das grandes vantagens do uso de simulação é o controle sobre os parâmetros de entrada. Assim, a fim de verificar a influência das cargas de trabalho e dos diferentes ambientes computacionais sobre as políticas de escalonamento, foram conduzidos diversos experimentos, em um total de 3240 simulações.

As 10 políticas de escalonamento discutidas na seção 6.1 (Random, Lowest, Disted, Global, Central, TLBA, GAS, Round Robin, DPWPExt, NBSP) foram submetidas às cargas Feit96, Jann97-ASCI, Jann97-CTC e Lub99 (*vide* seção 6.2) quando em execução em um *cluster* de 32 nós, em um *cluster* de 128 nós e em um *grid* de 512 nós. Para cada um desses cenários, foram ainda aplicadas 3 cargas de rede distintas. Por fim, cada uma das combinações anteriores foi simulada com diferentes quantidades de *jobs*: 10, 25, 50, 75, 100, 150, 200, 250 e 300.

Cada experimento foi executado 10 vezes a fim de obter a média aritmética. Para aferir a significância estatística de tais experimentos, optou-se por utilizar intervalos de confiança. As médias obtidas de cada experimento e o intervalo de confiança de 95% são apresentados no Apêndice B. Neste apêndice é também discutido o cálculo do intervalo de confiança para amostras pequenas.

7.3 Resultados

Os resultados dos experimentos descritos acima são apresentados nos gráficos 7.1 a 7.36. Cada gráfico sumariza os resultados obtidos da execução das 10 políticas estudadas neste trabalho quando submetidas à execução em um determinado ambiente computacional, impondo-lhes uma carga de UCP e uma carga de rede. O eixo x representa a quantidade de *jobs* executados e o eixo y representa o tempo de resposta médio. Para facilitar a visualização dos resultados obtidos com as cargas de rede, os gráficos foram agrupados, 3 a 3, pelo ambiente computacional e pela carga de UCP, variando-se apenas a informação da rede.

Os três gráficos, na página seguinte, sumarizam os resultados da execução da carga de UCP Feit96 em um *cluster* de 32 nós interligados por uma rede *Gigabit Ethernet*. Nos gráficos 7.1, 7.2 e 7.3 foram representadas as curvas resultantes da execução das 10 políticas, respectivamente, para uma carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

No gráfico 7.1 nota-se dois conjuntos de políticas com tempos de resposta médios bastante próximos. As políticas Lowest, Disted, Global, Central e TLBA formam o primeiro conjunto, com os menores tempos de resposta médios. O segundo conjunto é formado pelas políticas Round Robin, NBSP e DPWP. A política Random obteve um tempo de resposta médio maior que esses dois conjuntos, apresentando melhor desempenho apenas do que a política GAS. A GAS obteve o pior desempenho e distanciou-se significativamente das demais políticas de escalonamento, que obtiveram o comportamento esperado. Os resultados obtidos pela política GAS devem se a função de *fitness* utilizada, que, provavelmente, apresenta melhores resultados em sistemas com alta carga.

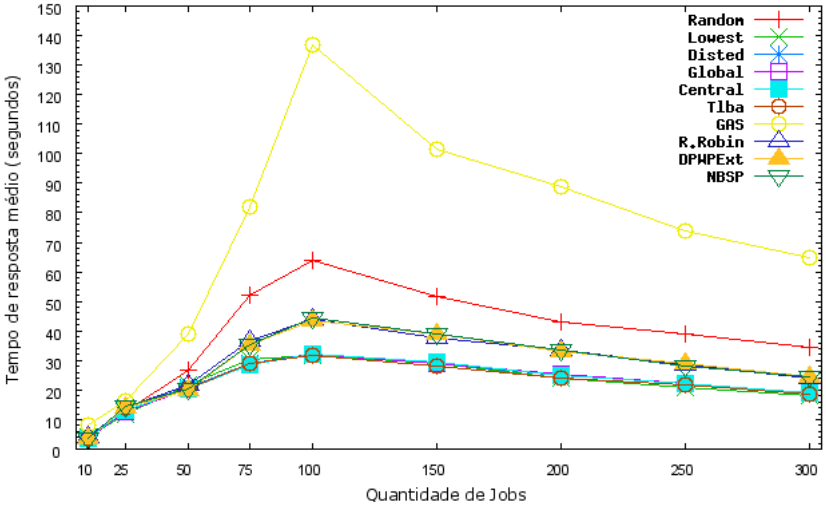


Gráfico 7.1: Cluster 32 nós com a carga Feit96, sem rede

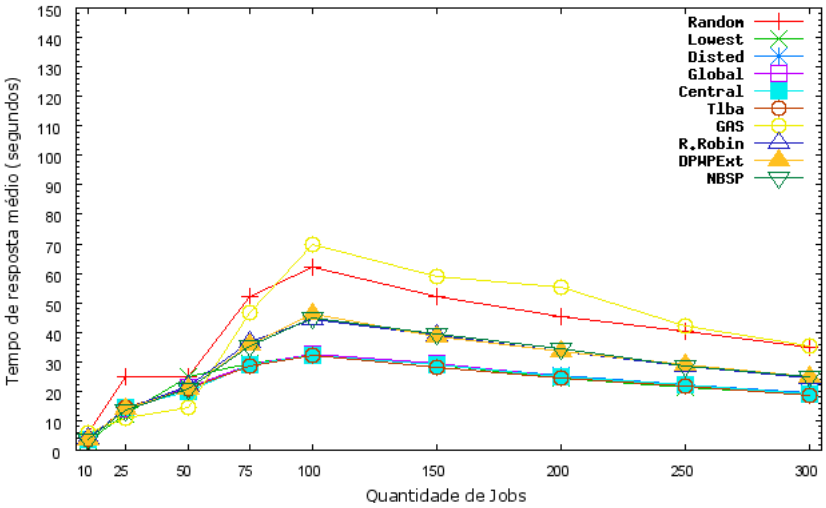


Gráfico 7.2: Cluster 32 nós com a carga Feit96, 100 Kbps

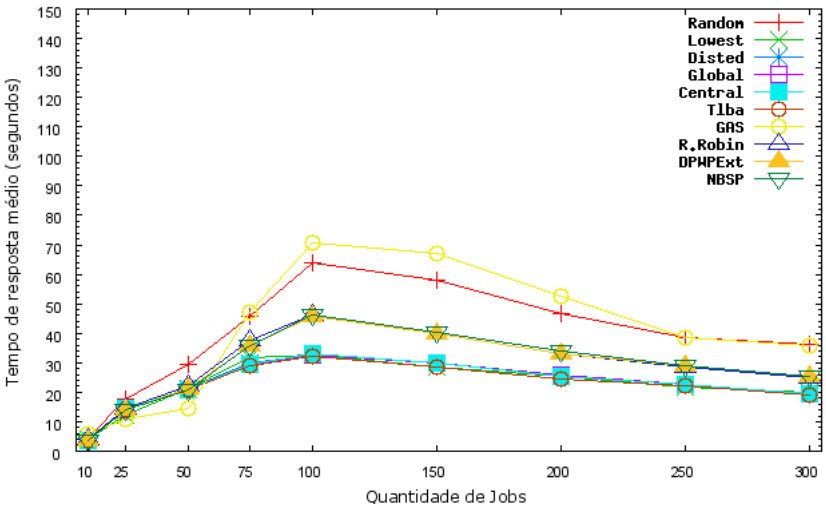


Gráfico 7.3: Cluster 32 nós com a carga Feit96, 1000 Kbps

Um aspecto a se observar é o fato da política Round Robin possuir resultados muito próximos às políticas NBSP e DPWP, o que não deveria ocorrer. Considerando-se que o sistema utilizado é bastante heterogêneo, esse aspecto pode ser explicado por uma baixa carga imposta ao sistema. Verificando-se as características da carga Feit96 nota-se que isso pode ocorrer. Essa carga possui intervalos entre chegadas altos e uma alta freqüência de jobs com baixo tempo de execução e baixo grau de paralelismo (*vide* seção 6.2 para uma discussão das cargas de trabalho).

A ordem de classificação das políticas no gráfico 7.1 se mantém nos gráficos 7.2 e 7.3 para a maior parte das quantidades de jobs. Entretanto, nota-se uma menor variação dos tempos de resposta médios da política GAS em relação às demais políticas. Isso evidencia o fato da função de fitness se comportar melhor em condições de carga mais elevadas. Quanto maior a carga de rede, melhor o comportamento da política GAS.

Os três gráficos a seguir, sumarizam os resultados da execução da carga de UCP Jan97-ASCI em um cluster de 32 nós interligados por uma rede *Gigabit Ethernet*. Os gráficos 7.4, 7.5 e 7.6 representam os resultados da execução das políticas, respectivamente, para uma carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

Em todos os gráficos nota-se um tempo de resposta muito baixo, o que se deve à baixa utilização que carga Jann97-ASCI impõe ao sistema. Neste sistema, com uma rede Gigabit *Ethernet*, a introdução de carga de rede não altera o cenário. No gráfico 7.4, observa-se que a política GAS obtém os piores resultados, seguida pela Random. As demais políticas de escalonamento, apresentam um comportamento similar. Esse comportamento se repete quando a carga de rede aumenta, conforme pode ser visto nos gráficos 7.5 e 7.6.

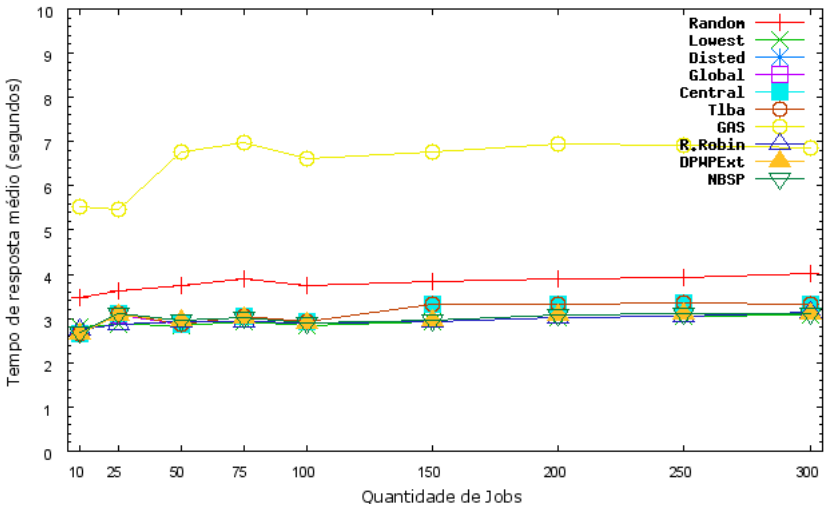


Gráfico 7.4: Cluster 32 nós com a carga Jann97-ASCI, sem rede

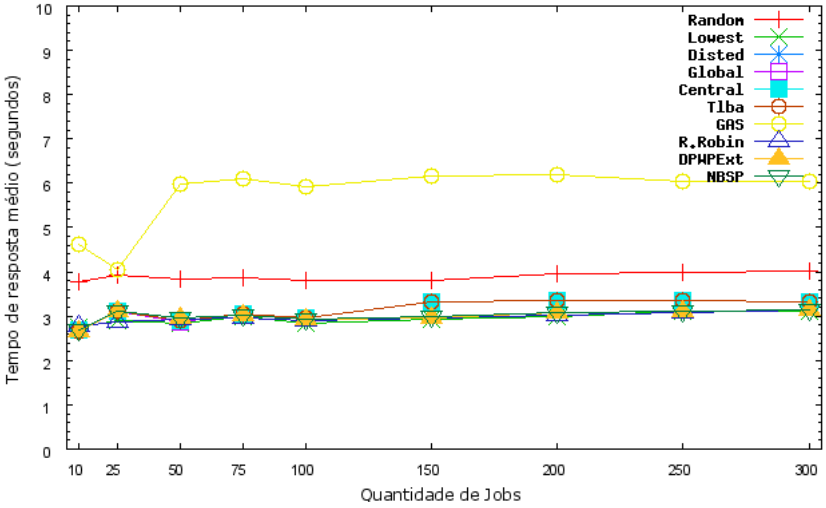


Gráfico 7.5: Cluster 32 nós com a carga Jann97-ASCI, 100 Kbps

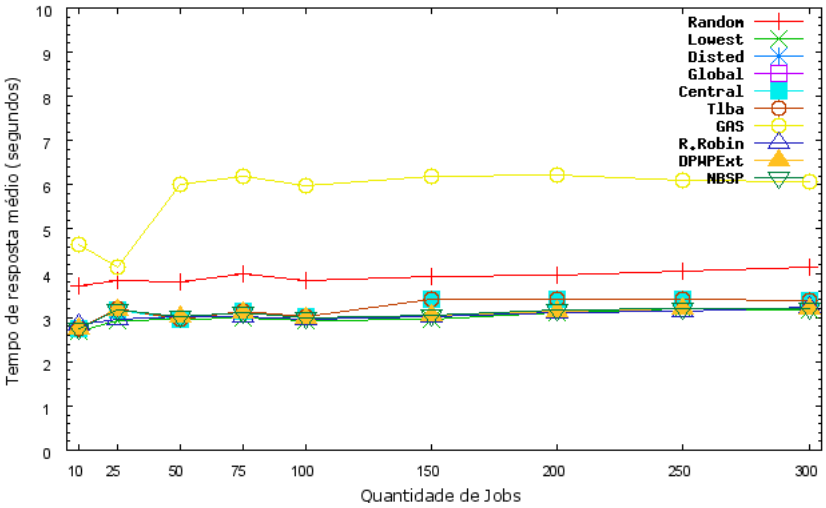


Gráfico 7.6: Cluster 32 nós com a carga Jann97-ASCI, 1000 Kbps

Na próxima página são apresentados os resultados da execução da carga de UCP Jan97-CTC para um cluster de 32 nós interligados por uma rede *Gigabit Ethernet*. Os gráficos 7.7, 7.8 e 7.9 representam os resultado da execução das políticas, respectivamente, para uma carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

Observando os gráficos 7.7, 7.8 e 7.9 nota-se novamente que o aumento da taxa de comunicação entre os processos não aumenta, significativamente, o tempo de resposta dos processos. A taxa de comunicação não impõe uma alta ocupação à rede. Entretanto, o aumento da taxa de comunicação evidencia, mais uma vez, que a função de *fitness* da política GAS se comporta melhor em condições de maior carga de rede. A política GAS se aproxima das demais políticas, chegando a obter melhores desempenhos que a política Random, à medida que a carga de rede aumenta.

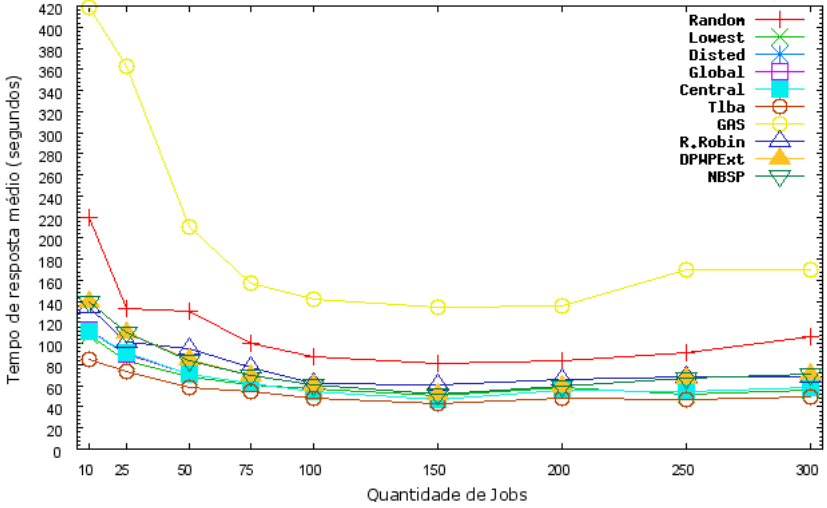


Gráfico 7.7: Cluster 32 nós com a carga Jann97-CTC, sem rede

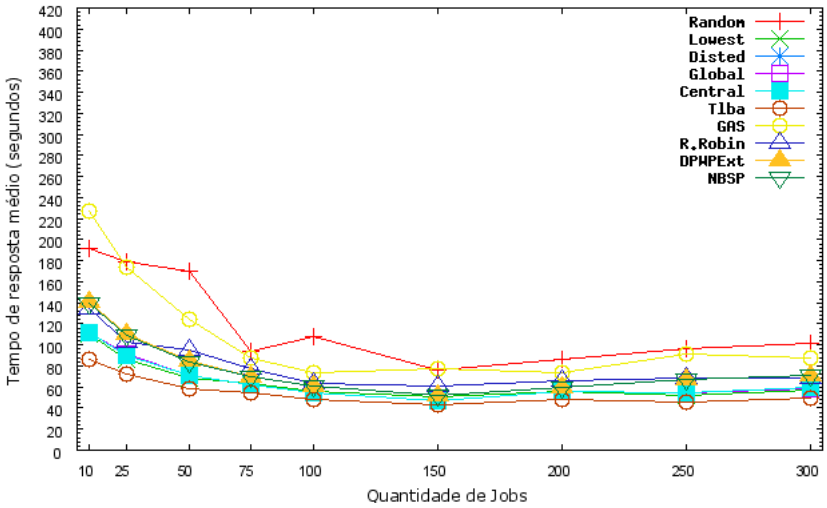


Gráfico 7.8: Cluster 32 nós com a carga Jann97-CTC, 100 Kbps

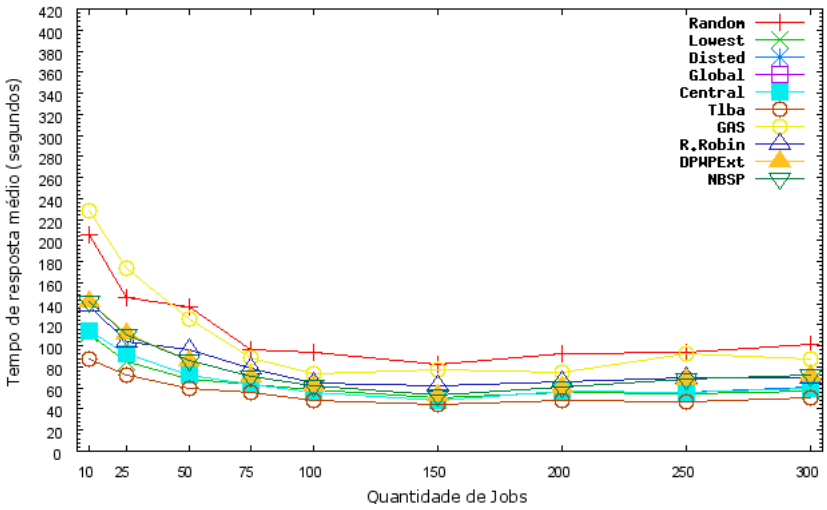


Gráfico 7.9: Cluster 32 nós com a carga Jann97-CTC, 1000 Kbps

Nos gráficos, 7.7, 7.8 e 7.9, independentemente do aumento da carga de rede, as políticas TLBA, Lowest, Disted, Central, Global, NBSP, DPWPExt e Round Robin apresentam a mesma ordem de classificação. A política TLBA obtém o menor tempo de resposta médio, seguida pela Lowest, Central, Disted, Global. Isso é um aspecto esperado em todos os cenários, uma vez que o escalonamento produzido por essas políticas é muito próximo. Com exceção da política Lowest, os índices de carga dessas políticas são coletados e utilizados nas decisões do escalonamento de forma bastante semelhante.

A política Random apresenta tempos de resposta médios bastante próximos para as três cargas de rede, conforme observado nos gráficos. Já a política GAS, quando a rede não é utilizada, apresenta o pior tempo de resposta dentre as 10 políticas. Na presença de comunicação por intermédio da rede a política GAS se aproxima das demais, obtendo melhor desempenho que a Random. Isso novamente evidencia que a função de *fitness* se comporta melhor em cenários com uso da rede.

Em todos os gráficos anteriores as políticas NBSP e DPWPExt apresentam um comportamento bastante similar entre elas. Isso acontece quando não há sobrecarga na rede. Em casos em que há sobrecarga na rede a DPWPExt tende a obter desempenhos piores. Dessa forma, pode se concluir que o aumento da taxa de comunicação não provoca sobrecarga na comunicação entre os processos utilizando a rede e, consequentemente não causa impacto sobre o tempo de resposta obtido na execução das políticas.

Outro aspecto a se observar é o comportamento da política Round Robin, que possui tempos de resposta médios muito próximos aos das demais políticas. Tal aspecto pode ocorrer em sistemas homogêneos ou quando a carga imposta ao sistema não é significativa. Uma vez que o *cluster* utilizado nesses cenários é heterogêneo, e provável que esse aspecto esteja relacionado à carga imposta ao sistema.

Para finalizar a análise dos cenários que utilizam o cluster de 32 nós, com rede *Gigabit Ethernet*, são apresentados nos gráficos 7.10, 7.11 e 7.12 os resultados da execução da carga de UCP Lub99. Novamente, esses gráficos representam os resultado da execução das políticas, respectivamente, para uma carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

Grande parte das características apresentadas pelas políticas de escalonamento para as cargas de trabalho Feit96, Jann97-ASCI e Jann97-CTC se repetem para a carga Lub99. A ordem de classificação entre as políticas é também o mesmo, com a TLBA apresentando os melhores tempos de resposta médios e a formação de dois conjuntos de políticas, com tempos de resposta médios bem próximos: TLBA, Lowest, Central, Disted e Global num conjunto e NBSP, DPWPExt e Round Robin noutro conjunto. Outro aspecto que se repete é o pior tempo

de resposta da política GAS, com um elevado tempo de resposta (em comparação com as demais políticas) na ausência de carga de rede.

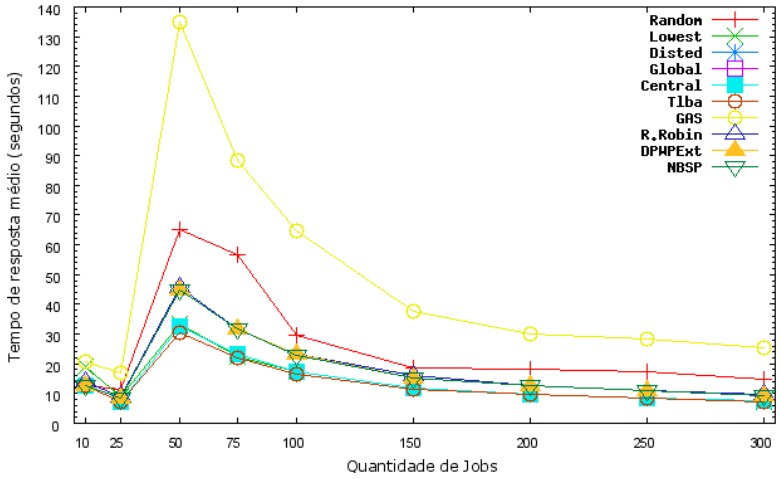


Gráfico 7.10: Cluster 32 nós com a carga Lub99, sem rede

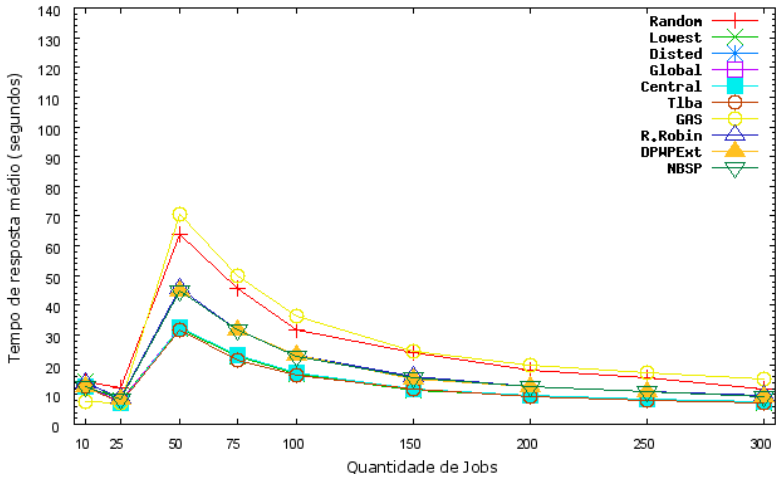


Gráfico 7.11: Cluster 32 nós com a carga Lub99, 100 Kbps

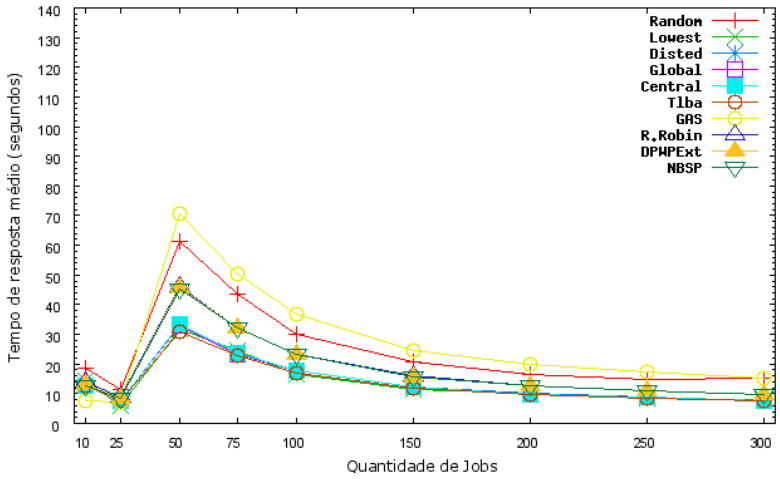


Gráfico 7.12: Cluster 32 nós com a carga Lub99, 1000 Kbps

O comportamento das políticas para a carga Lub99 se assemelha ao comportamento das políticas para a carga Feit96. Nota-se um elevado pico no tempo de resposta médio para 60 jobs submetidos ao sistema. Este aspecto é notório para a carga Lub99 que possui intervalos com um elevado número de jobs submetidos ao sistema seguidos por uma elevada queda desse número (*vide* seção 6.2). Esse aspecto, assim como na carga Feit96 representa a interação dos usuários com o sistema, modelando a reação do mesmo ao tempo de resposta. Quando o tempo de resposta é baixo, mais jobs são submetidos ao sistema e quando o tempo de resposta aumenta o número de jobs submetidos tende a diminuir.

Os cenários apresentados a seguir envolvem o uso de um cluster de 128 nós interligados por uma rede *Gigabit Ethernet*. Os próximos 12 gráficos sumarizam os resultados da execução das cargas de trabalho de UCP Feit96, Jan97-ASCI, Jan97-CTC e Lub99 nesse *cluster*. Novamente os cenários são variados, executando as 10 políticas sem carga de rede, com uma carga de 100 Kbps em média por processo e com uma carga de 1000 Kbps em média por processo.

Os tempos de resposta médios e o comportamento das políticas NBSP, DPWPExt, TLBA, GAS, Central, Global e Disted quando executadas no *cluster* de 128 nós são bastante similares aos resultados obtidos nos cenários utilizando um *cluster* de 32 nós. Já as políticas Random, Round Robin, Lowest quando executadas no cluster de 128 nós obtiveram desempenhos piores aos observados no *cluster* de 32 nós, além de uma maior variação na curva dos tempos de resposta médios. Esses resultados podem ser explicados por dois motivos principais:

- O sistema não foi bem exercitado, ou seja, a carga de trabalho não foi suficiente para causar impacto sobre o tempo de resposta médio.
- O *cluster* de 128 nós é bem mais heterogêneo e provê maior custo para o algoritmo de escalonamento. As políticas Round Robin e Random não coletam/utilizam qualquer índice de carga, podendo realizar escalonamentos em computadores sobrecarregados. Embora a política Lowest utilize índices de carga, apenas uma pequena fração da quantidade de computadores total é considerada no momento de um escalonamento. Assim, computadores com maior potência computacional livre podem ser desconsiderados para receber processos, utilizando tal política. Nas simulações realizadas em ambos os *clusters* a política Lowest coleta e utiliza índice de carga de 8 computadores, selecionados aleatoriamente.

A seguir são apresentados os resultados da execução da carga de UCP Feit96 no cluster de 128 nós interligados por uma rede *Gigabit Ethernet*. Os gráficos 7.13, 7.14 e 7.15 representam os resultado da execução das 10 políticas, respectivamente, para uma carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

Analisando os gráficos 7.13, 7.14 e 7.15 para a carga de trabalho Feit96 em um *cluster* de 128 nós, juntamente com os gráficos 7.1, 7.2 e 7.3 para a mesma carga em um *cluster* de

32 nós, observa-se que, exceto pelas políticas Random, Round Robin e Lowest, as políticas se comportam de forma semelhante nesses dois sistemas. No cluster de 128 nós, as políticas Random, Round Robin e Lowest apresentam tempo de resposta médio superiores ao observados no *cluster* de 32 nós. Conforme discutido, isso se deve à maior heterogeneidade deste cluster e das políticas utilizarem pouca ou nenhuma informação de carga.

Assim como nos gráficos anteriores a política GAS obtém melhores resultados para cargas de rede mais altas. No intervalo entre 10 e 20 jobs a GAS obtém o melhor tempo de resposta dentre todas as políticas, mas perde desempenho quando o número de jobs aumenta. A política TLBA obtém os melhores resultados para quase todos os pontos da curva nos três gráficos. As políticas NBSP, DPWPExt, Disted, Global e Central apresentam resultados muito próximos à TLBA.

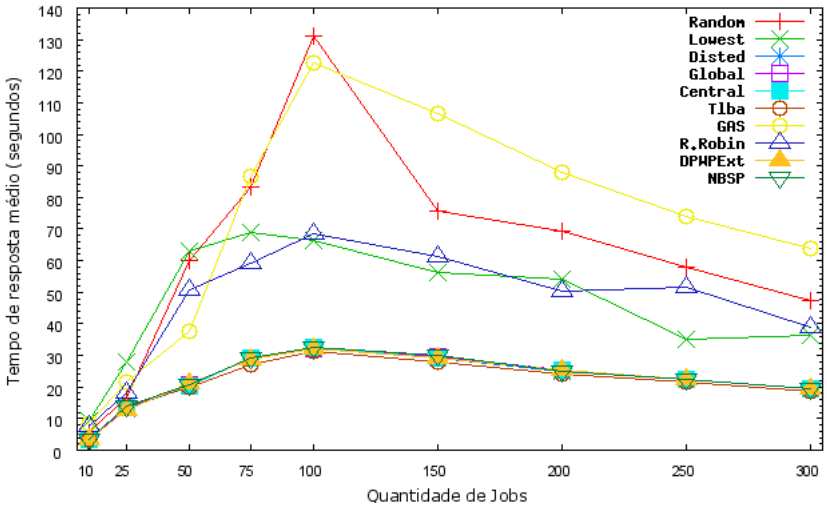


Gráfico 7.13: Cluster 128 nós com a carga Feit96, sem rede

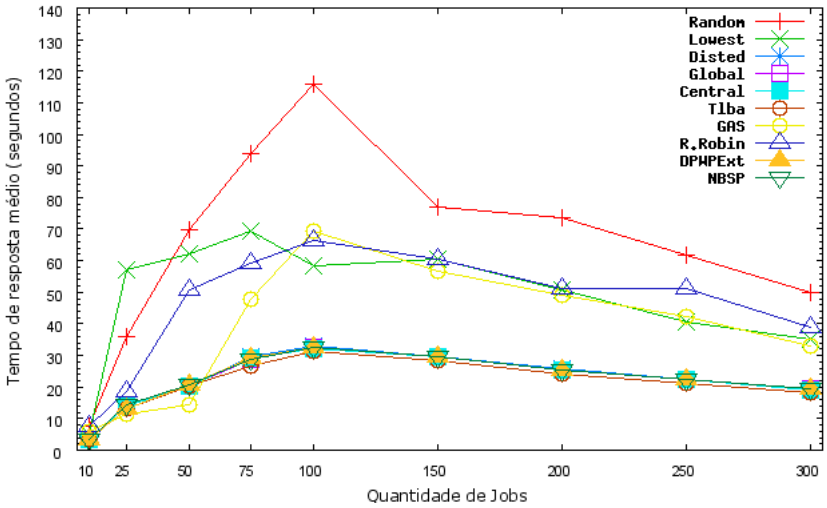


Gráfico 7.14: Cluster 128 nós com a carga Feit96, 100 Kbps

No gráfico 7.3 há uma grande variação na curva apresentada pela política Random. Essa

política realiza o escalonamento selecionando aleatoriamente os computadores que receberão os processos de um job. Devido à heterogeneidade essa política pode obter desempenhos muito ruins.

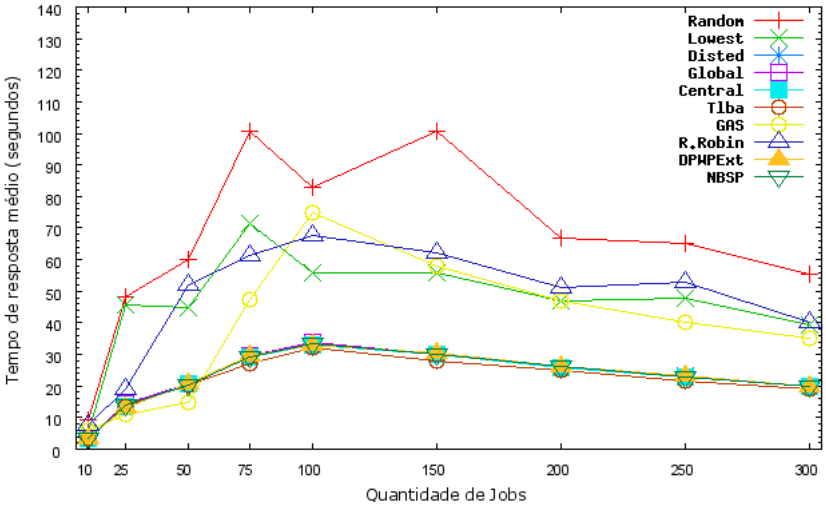


Gráfico 7.15: Cluster 128 nós com a carga Feit96, 1000 Kbps

Os resultados da execução da carga de UCP Jan97-ASCI em um cluster de 128 nós interligados por uma rede *Gigabit Ethernet* são apresentados nos gráficos a seguir. Os gráficos 7.16, 7.17 e 7.18 contêm os resultados da execução das políticas, respectivamente, para uma carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

Como nos resultados do cluster de 32 nós o tempo de resposta obtido pelas políticas é bem pequeno, evidenciando a baixa utilização do sistema, inclusive da rede de comunicação. Independentemente da taxa de comunicação, os tempos de resposta médios das mesmas políticas, nos 3 gráficos, são muito próximos.

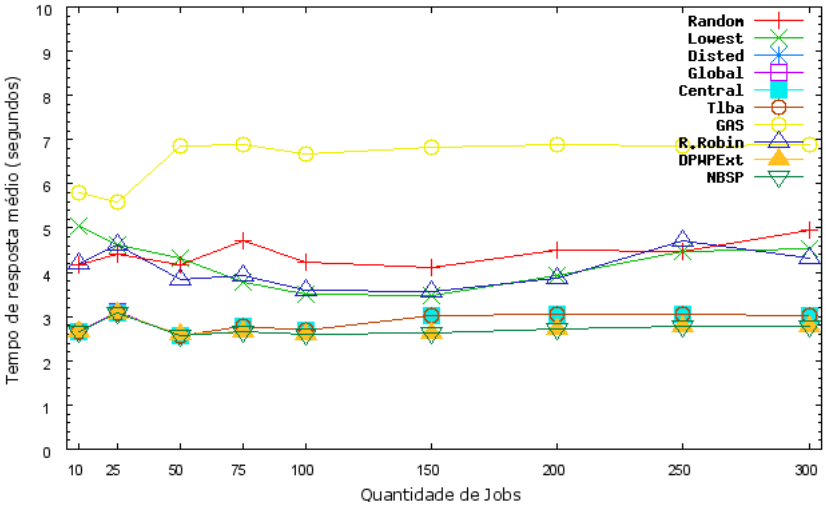


Gráfico 7.16: Cluster 128 nós com a carga Jann97-ASCI, sem rede

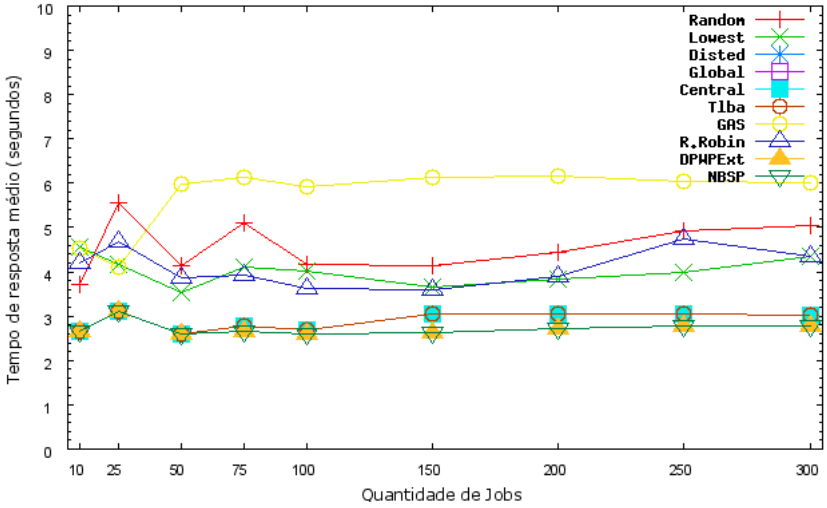


Gráfico 7.17: Cluster 128 nós com a carga Jann97-ASCI, 100 Kbps

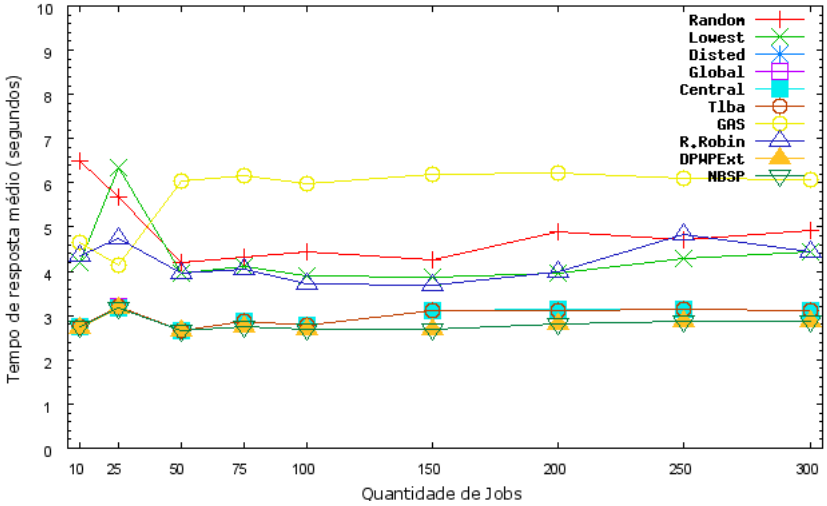


Gráfico 7.18: Cluster 128 nós com a carga Jann97-ASCI, 1000 Kbps

É interessante notar que, para a carga Jann-ASCI, as políticas NBSP e DPWPExt apresentam os melhores tempos de resposta, ao contrário do que acontece para as demais cargas, em que a política TLBA apresenta os melhores resultados.

Na próxima página são apresentados os gráficos 7.19, 7.20 e 7.21 para a carga de UCP Jan97-ASCI em um cluster de 128 nós interligados por uma rede *Gigabit Ethernet*. A carga de rede é nula para o primeiro gráfico, de 100 kbps em média por processo no segundo e de 1000 kbps em média por processo no terceiro gráfico.

O comportamento das políticas de escalonamento para essa carga no *cluster* de 128 nós é similar ao comportamento das políticas para a carga Feit96 nesse mesmo ambiente computacional. A mesma ordem de classificação das políticas pode ser observada. Pode se notar, entretanto, que a política GAS obtém tempos de resposta com menor variação para as diferentes quantidades de jobs nos gráficos 7.20 e 7.21.

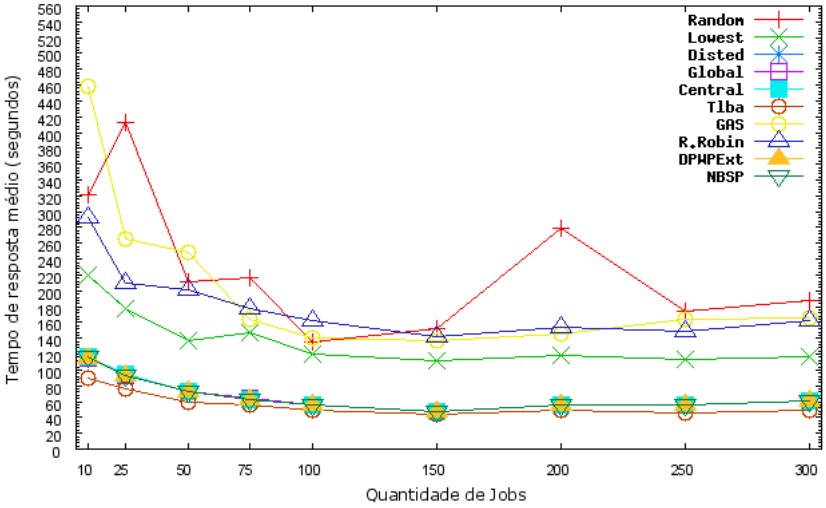


Gráfico 7.19: Cluster 128 nós com a carga Jann97-CTC, sem rede

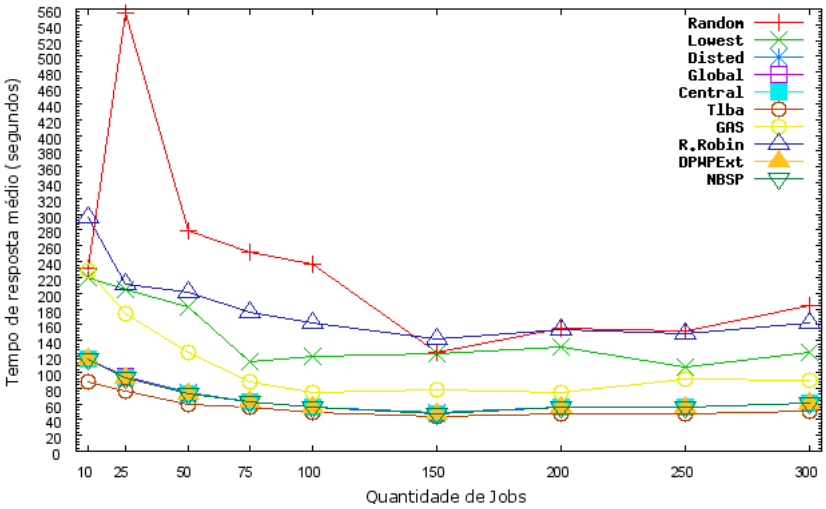


Gráfico 7.20: Cluster 128 nós com a carga Jann97-CTC, 100 Kbps

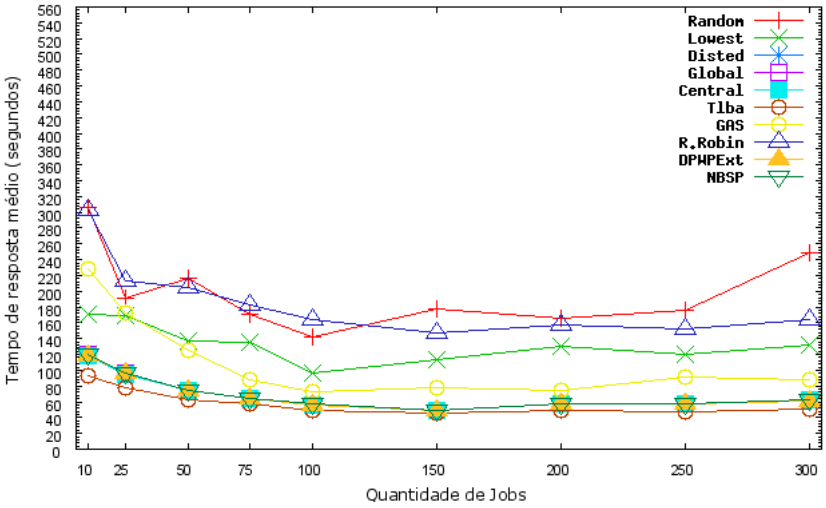


Gráfico 7.21: Cluster 128 nós com a carga Jann97-CTC, 1000 Kbps

Os gráficos 7.22, 7.23 e 7.24 apresentam os resultados obtidos da execução da carga Lub99 no cluster de 128 nós com rede *Gigabit Ethernet*. As mesmas cargas de rede dos casos anteriores foram avaliadas: carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

Os resultados obtidos para essa carga é semelhante aos resultados obtidos para as cargas Feit96 e Jan97-CTC, incluindo a ordem de classificação das políticas. A única exceção é a política GAS que obtém o pior tempo de resposta entre todas as políticas no gráfico 7.22 e o segundo pior resultado nos gráficos 7.23 e 7.24, obtendo desempenho superior apenas a política Random.

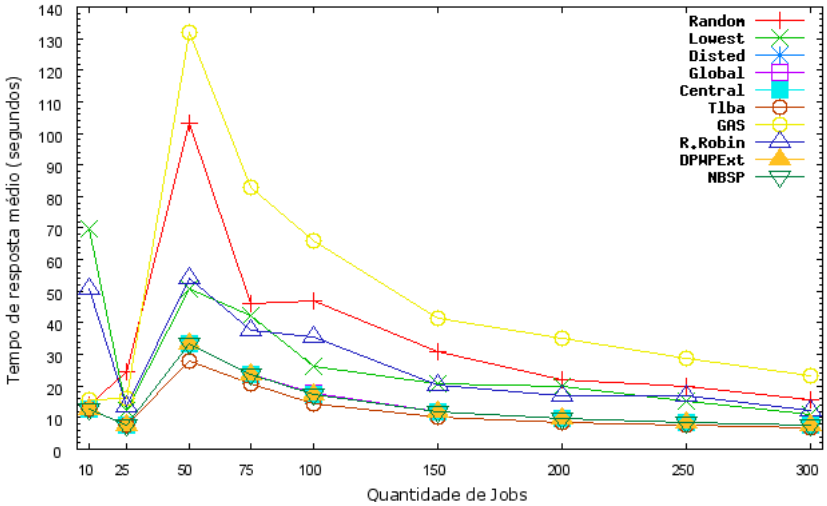


Gráfico 7.22: Cluster 128 nós com a carga Lub99, sem rede

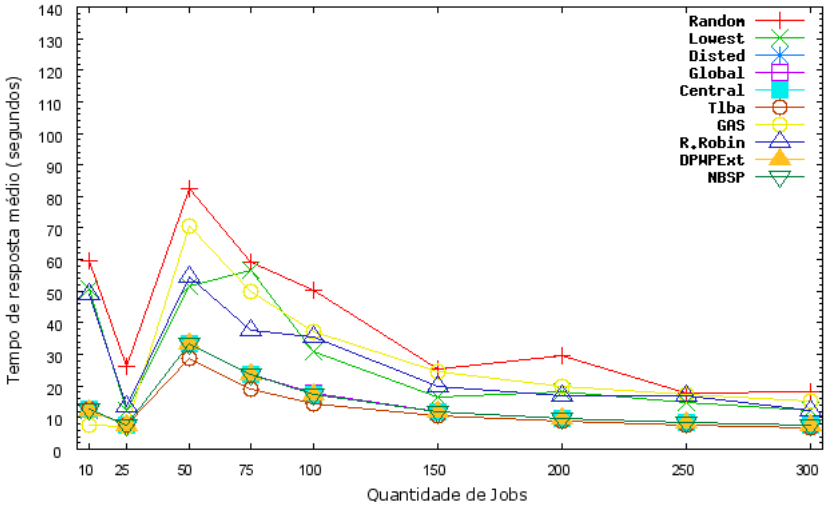


Gráfico 7.23: Cluster 128 nós com a carga Lub99, 100 Kbps

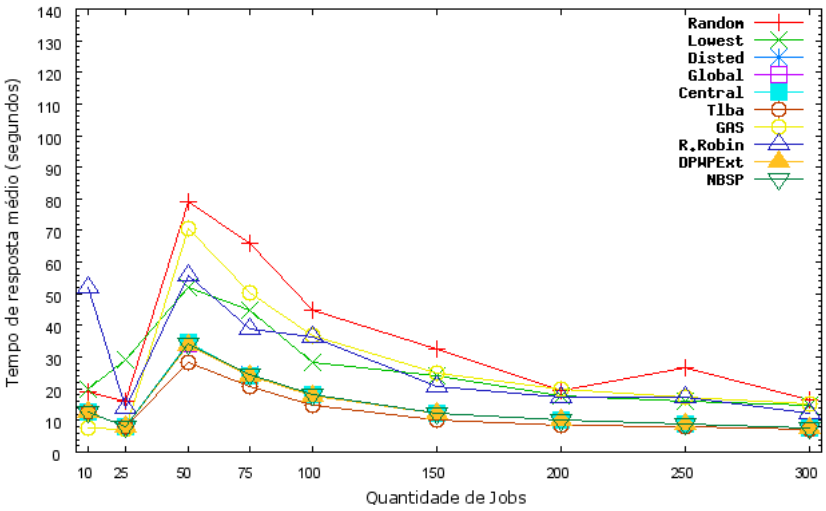


Gráfico 7.24: Cluster 128 nós com a carga Lub99, 1000 Kbps

Em todos os cenários que utilizaram clusters verificou-se que nem mesmo a inserção de carga de rede de 1000 kbps em média por processo provocou impacto nos tempos de resposta obtidos com a utilização das políticas de escalonamento. Essa taxa de comunicação não foi suficiente para provocar sobrecarga de comunicação numa rede com a largura de banda de uma rede *Gigabit Ethernet*. De modo contrário, nos cenários que utilizam grids computacionais o aumento da carga de rede causou grande impacto nos tempos de resposta médios obtidos com a utilização das políticas. A inserção de carga de rede mais pesada alterou, inclusive a ordem de classificação das políticas estudadas. Estes cenários são analisados nos gráficos a seguir.

Os próximos três gráficos sumarizam os resultados da execução da carga de UCP Feit96 em um *grid* de 512 nós. Os nós em um mesmo site são interligados por uma rede *Gigabit Ethernet* e cada site é interligado aos demais utilizando a Internet. A seção 7.1 detalha a parametrização desse sistema no simulador. Nos gráficos 7.1, 7.2 e 7.3 são representadas as curvas resultantes da execução das 10 políticas, respectivamente, para uma carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

Observando os gráficos 7.25 e 7.13 nota-se que na ausência de rede, a execução em um *grid* possui comportamento e tempos de resposta médios próximos ao *cluster*. A diferença principal está nos resultados obtidos pelas políticas Random, Round Robin e Lowest no *grid*, que obtêm tempos de resposta menores que no cluster. Isto se deve, possivelmente, à existência de um número maior de recursos, ampliando as chances de tais políticas escalonarem para nós com baixa utilização.

No grid, à medida que a taxa de comunicação na rede aumenta o comportamento das políticas se altera significativamente e os tempos de resposta médios aumentam em comparação aos *clusters*. No gráfico 7.25 o maior tempo de resposta médio é da política GAS, que atinge valores próximos a 140 segundos, enquanto no gráfico 7.26 o maior tempo de resposta pertence à política Random com valor próximo a 3600 segundos. Já no gráfico 7.27 a política Round

Robin atinge o valor de 32000 segundos.

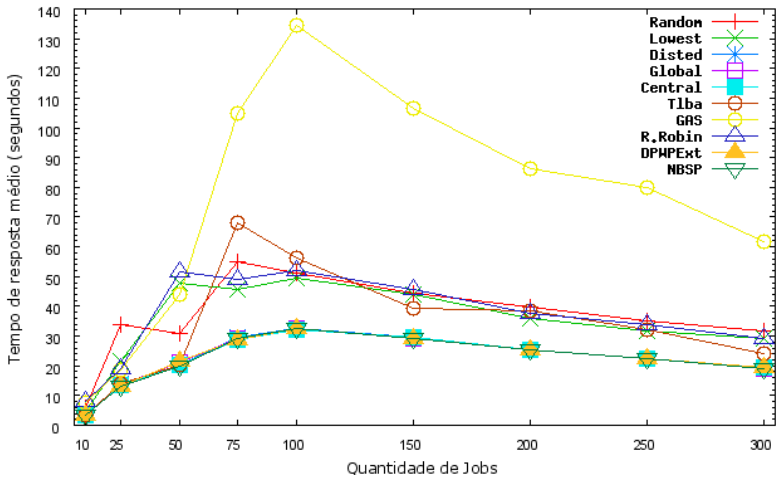


Gráfico 7.25: Grid 512 nós (média 8/rede) com a carga Feit96, sem rede

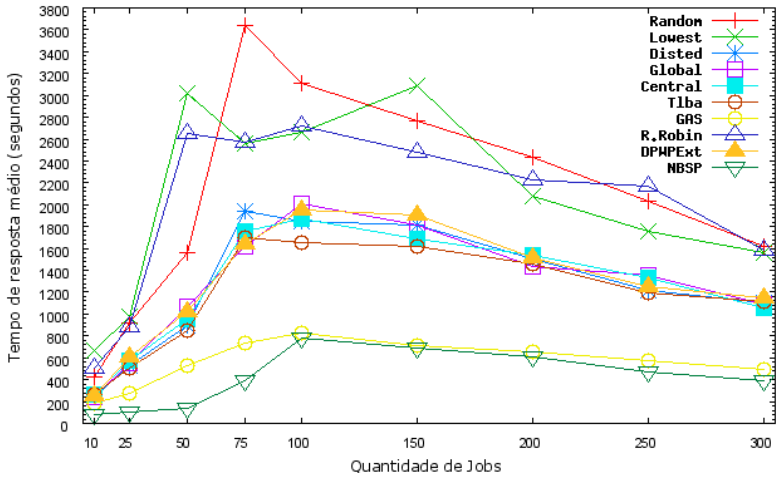


Gráfico 7.26: Grid 512 nós (média 8/rede) com a carga Feit96, 100 Kbps

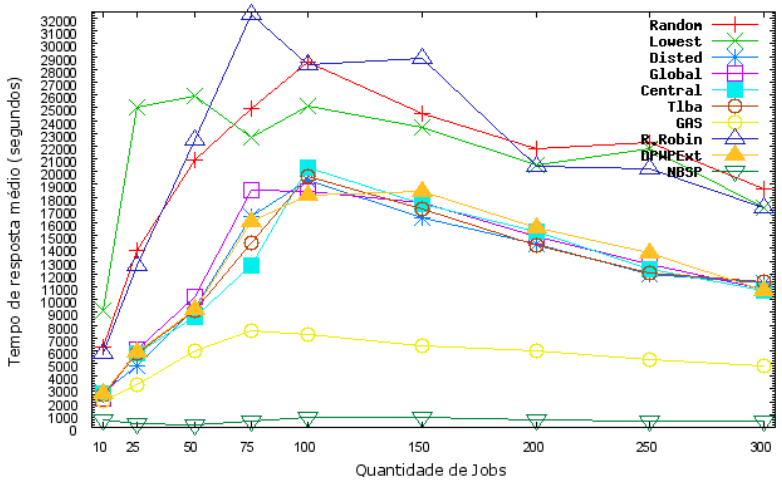


Gráfico 7.27: Grid 512 nós (média 8/rede) com a carga Feit96, 1000 Kbps

Na presença de carga de rede (gráficos 7.26 e 7.27) as políticas Random, Round Robin e Lowest apresentam os piores resultados. Isso é esperado já que tais políticas utilizam pouca ou nenhuma informação da carga do sistema. As políticas TLBA, Disted, Central, Global, DPWPExt formam um grupo de políticas com tempos de respostas médio muito próximos. O tempo de resposta dessas políticas aumenta à medida que a carga na rede aumenta, demonstrando que a comunicação causa grande impacto na execução da carga de trabalho.

Os melhores tempos de resposta médios, com a utilização da rede, pertencem às políticas GAS e NBSP. A NBSP apresenta o melhor resultado também na ausência de carga na rede, conforme pode ser observado no gráfico 7.28. Neste gráfico a NBSP possui um comportamento muito próximo das políticas DPWPExt e Central. O comportamento similar da NBSP e da DPWPExt para cargas de trabalho *CPU-Bound* é esperado, sendo que a NBSP apresenta vantagens quando a ocupação da rede de comunicação aumenta, conforme pode ser observado nos gráficos 7.26 e 7.27.

Os resultados obtidos com as políticas GAS e NBSP nos gráficos 7.25, 7.26 e 7.27 demonstram a necessidade de se considerar a rede no escalonamento em grids computacionais. Essas duas políticas utilizam informação da carga de comunicação na rede. No gráfico 7.28 a política GAS apresenta o pior tempos de resposta médio dentre as 10 políticas, enquanto nos gráficos 7.26 e 7.27 ela apresenta o segundos melhor tempo de resposta, atrás apenas da política NBSP. A NBSP apresenta um tempo de resposta muito menor que as demais políticas, inclusive do que a GAS no gráfico 7.27, demonstrando claramente a necessidade de considerar o a comunicação no escalonamento em grids computacionais. A política NBSP escalona com base no impacto que a comunicação provoca na execução de uma carga de trabalho.

Os resultados apresentados nos gráficos a seguir também demonstram tal necessidade. Os gráficos 7.28, 7.29 e 7.30 apresentam os resultados da execução da carga de UCP Jan97-ASCI no *grid*. As taxas de comunicação da rede em cada gráfico são, respectivamente, nula, 100 kbps em média por processo e 1000 kbps em média por processo.

A observação dos gráficos 7.29 e 7.30 em relação aos gráficos 7.5, 7.6, 7.17 e 7.18 atesta o fato de que uma rede de baixa largura de banda e alta latência, comum nos *grids*, tem grande influência sobre o desempenho. Enquanto a carga Jann97-ASCI impõe uma baixa carga para *clusters*, em *grids* o tempo de resposta médio é elevado, alcançando 3000 segundos no gráfico 7.30. Novamente as políticas que obtém os melhores resultados, distanciando-se significativamente das demais, são NBSP e GAS. Essas políticas consideram a rede de comunicação no momento do escalonamento. A NBSP obtém os melhores resultados porque considera o impacto da utilização da rede sobre os processos.

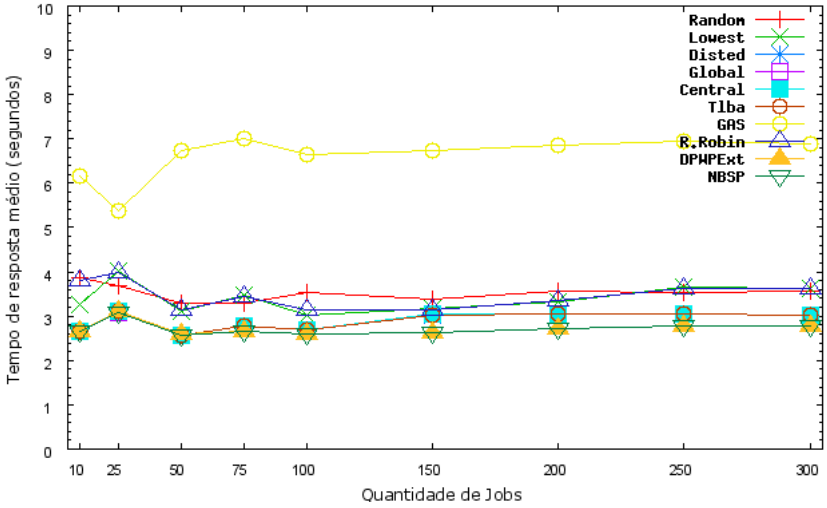


Gráfico 7.28: Grid 512 nós (média 8/rede) com a carga Jann97-ASCI, sem rede

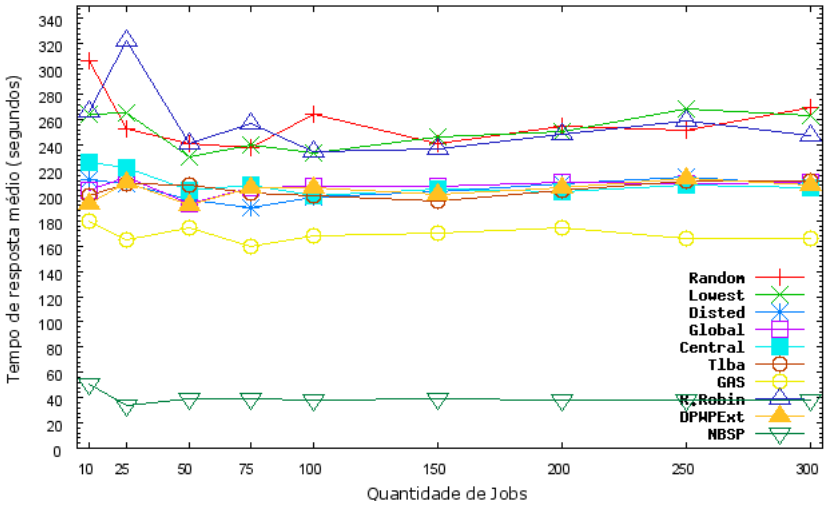


Gráfico 7.29: Grid 512 nós (média 8/rede) com a carga Jann97-ASCI, 100 Kbps

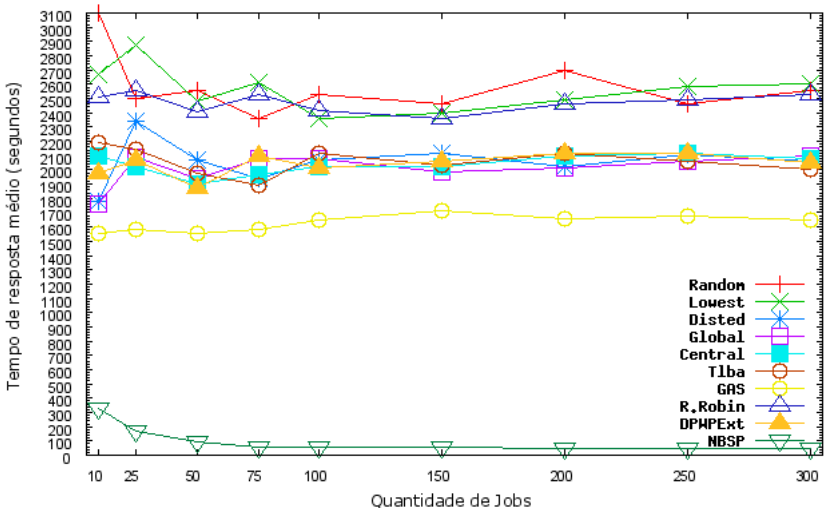


Gráfico 7.30: Grid 512 nós (média 8/rede) com a carga Jann97-ASCI, 1000 Kbps

Os gráficos 7.31, 7.32 e 7.33 apresentam os resultados da execução da carga de UCP Jan97-CTC no grid. As taxas de comunicação da rede em cada gráfico são, respectivamente, nula, 100 kbps em média por processo e 1000 kbps em média por processo.

O comportamento das políticas avaliadas sob a carga Jann97-CTC no *grid* apresenta um comportamento semelhante à carga Feit96 para o mesmo ambiente computacional. Nota-se que a inserção de uma carga de rede mais pesada deteriora sobremaneira o desempenho das políticas de escalonamento. A política GAS, no entanto, apresenta tempos de resposta médios melhores para uma carga de rede maior. Embora o comportamento das políticas para essa carga se assemelhe ao comportamento para a carga Feit96 os tempos de resposta médios obtidos para esta carga é um tanto maior do que para a carga Feit96.

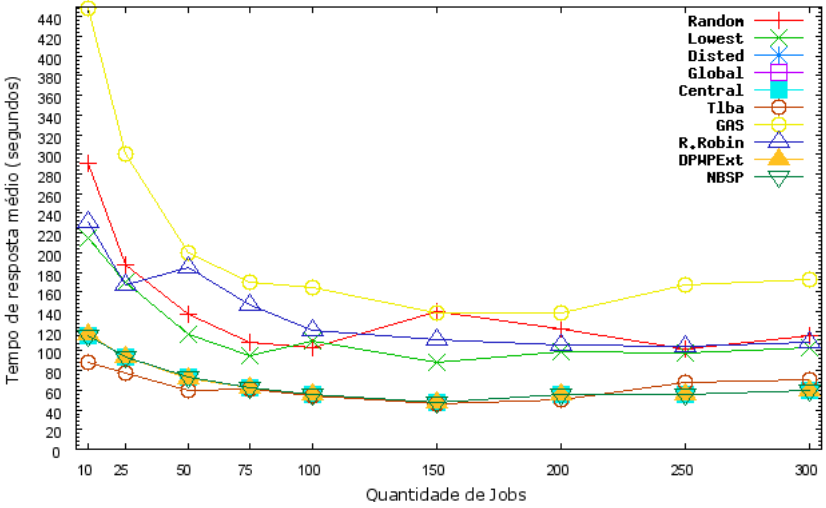


Gráfico 7.31: Grid 512 nós (média 8/rede) com a carga Jann97-CTC, sem rede

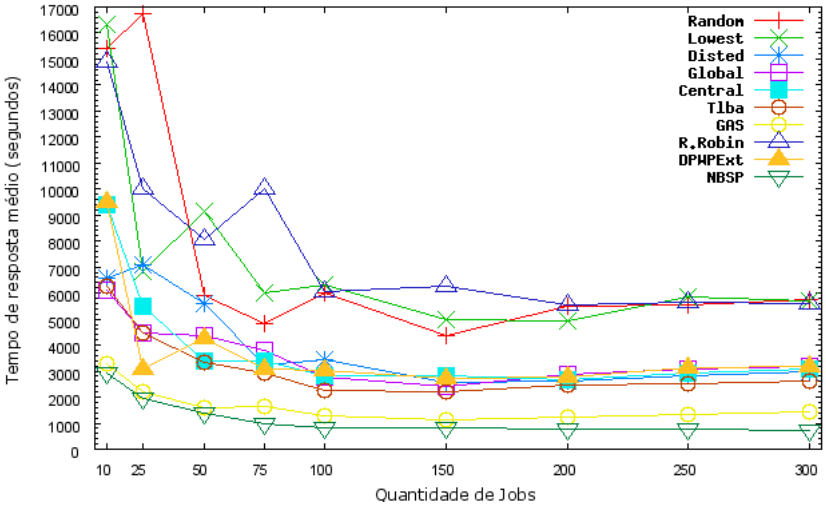


Gráfico 7.32: Grid 512 nós (média 8/rede) com a carga Jann97-CTC, 100 Kbps

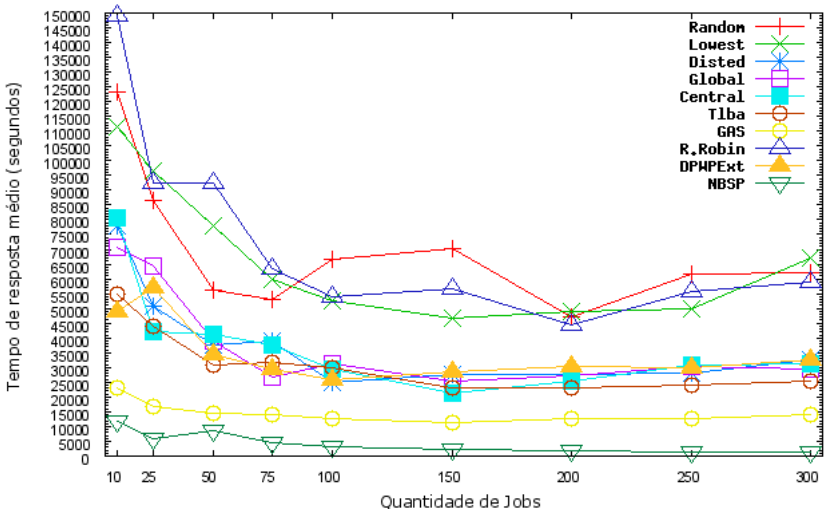


Gráfico 7.33: Grid 512 nós (média 8/rede) com a carga Jann97-CTC, 1000 Kbps

Finalizando os resultados das simulações conduzidas neste trabalho são apresentados a seguir os resultados para a carga de UCP Jan97-CTC no *grid*. Os gráficos 7.34, 7.35 e 7.36 sumarizam os resultados, respectivamente, para uma carga de rede nula, de 100 kbps em média por processo e de 1000 kbps em média por processo.

O comportamento das políticas para essa carga é também semelhante ao comportamento para a carga Feit96. A política NBSP obtém novamente o melhor desempenho na presença de carga de rede, seguida pela política GAS. Um aspecto que pode ser notado aqui, assim como nos demais resultados para o *grid* computacional é que quanto maior a taxa de comunicação maior o desempenho da política NBSP em relação as demais políticas estudadas neste trabalho.

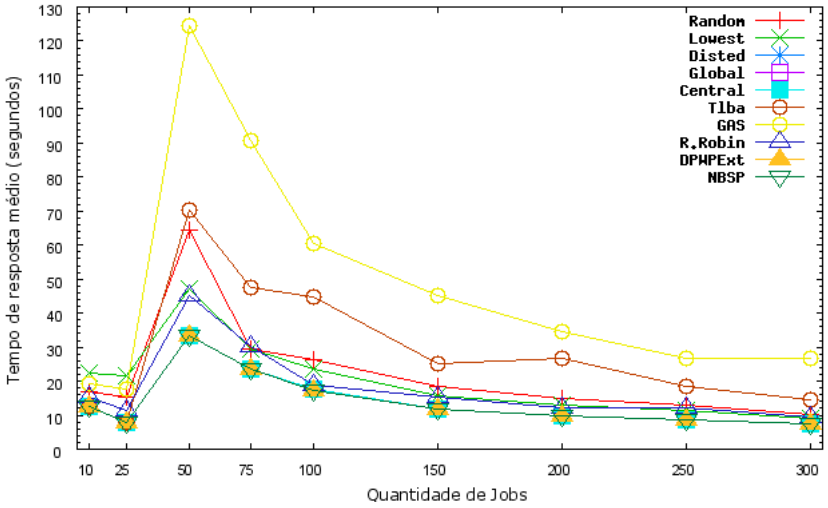


Gráfico 7.34: Grid 512 nós (média 8/rede) com a carga Lub99, sem rede

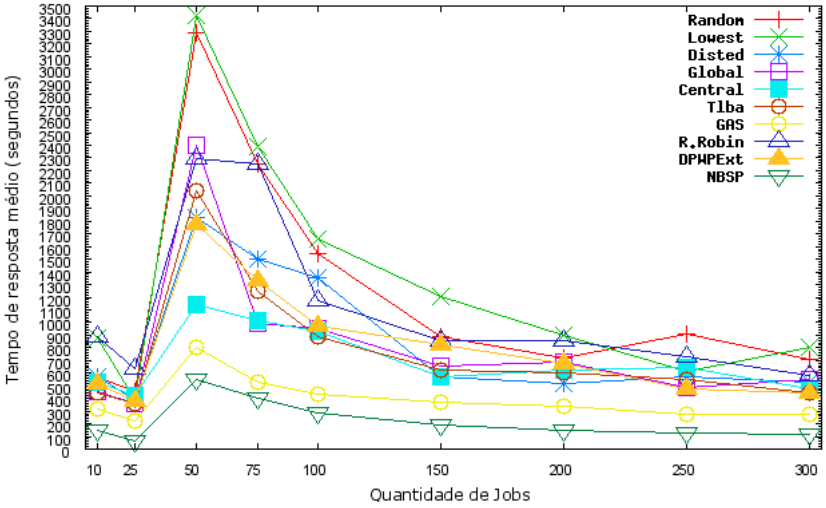


Gráfico 7.35: Grid 512 nós (média 8/rede) com a carga Lub99, 100 Kbps

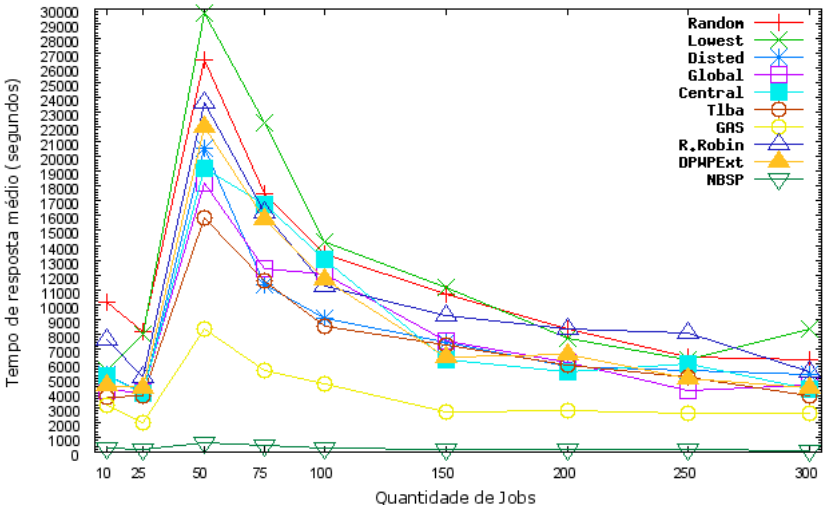


Gráfico 7.36: Grid 512 nós (média 8/rede) com a carga Lub99, 1000 Kbps

7.4 Considerações Finais

Neste capítulo, foi apresentada a parametrização utilizada no simulador. Os ambientes computacionais estudados neste trabalho foram obtidos por meio de funções de distribuição de probabilidades. Alguns limiares obtidos de um sistema parametrizado foram usados a fim de obter ambientes próximos do real.

Os experimentos conduzidos neste trabalho e os resultados obtidos por meio das simulações permitiram cumprir os objetivos propostos para este trabalho. Isso será melhor discutido no próximo capítulo, juntamente com as contribuições deste trabalho. Os resultados permitiram verificar que diferentes cargas de trabalho podem levar a diferentes comportamentos para uma mesma política de escalonamento. Dentre as políticas avaliadas não são adequadas a todos os cenários. Observou-se que a TLBA apresenta bons resultados nos *clusters* avaliados, enquanto

a GAS apresenta escalonamentos ruins nesses sistemas. Por outro lado a GAS apresenta bons escalonamentos em *grids* computacionais e para ambientes cuja carga de comunicação entre os processos é alta. A NBSP apresentou os melhores resultados em *grids*, embora não tenha apresentado os melhores resultados para uma carga *CPU-Bound* e para *clusters*.

Os resultados obtidos com *grids* demonstram que tais sistemas são pouco adequados à execução de aplicações paralelas nas quais há muita comunicação entre seus processos, oferecendo bom desempenho para aplicações *bag-of-tasks*. Os resultados obtidos com a GAS e com a NBSP, principalmente, evidenciam a necessidade de que as políticas de escalonamento voltadas a *grids* computacionais contemplem o tratamento da comunicação entre os processos.

O capítulo a seguir apresenta as conclusões finais e as principais contribuições obtidas com o desenvolvimento deste trabalho de mestrado. Além disso, são apontados caminhos para trabalhos futuros.

Capítulo

8

Conclusões

O escalonamento pode afetar profundamente a eficiência de um sistema. Um bom escalonamento, considerando as características do sistema e da carga de trabalho, pode garantir uma melhor utilização dos recursos computacionais e, em conseqüência, um melhor tempo de resposta para as aplicações paralelas. Entretanto, o escalonamento em sistemas computacionais paralelos e distribuídos é uma atividade complexa, pois uma série de fatores influenciam as decisões do escalonador. Fatores como a heterogeneidade, o baixo desempenho da rede e aplicações com diferentes características exigem o uso de políticas de escalonamento que contemplem o tratamento dessas características nas suas decisões.

Os resultados dos experimentos conduzidos neste trabalho demonstraram a necessidade de considerar características do sistema e da carga de trabalho nas decisões do escalonador. Os resultados demonstraram que políticas que coletam índices de carga e tratam a heterogeneidade como a TLBA, DPWP e NBSP obtêm melhores resultados em sistemas heterogêneos como *clusters* e *grids* computacionais do que políticas que não consideram, como Round Robin e Random. Políticas como a GAS, que consideram características da carga de trabalho, obtêm melhores resultados. Além disso, a NBSP, que considera a utilização da rede nas decisões do escalonamento obteve melhores resultados do que as demais políticas estudadas neste trabalho, sob elevadas condições de carga.

O problema central estudado neste trabalho é a avaliação de desempenho de políticas de escalonamento para aplicações paralelas considerando-se diferentes cenários. A evolução

dos sistemas distribuídos têm tornado tais sistemas uma alternativa viável para a execução de aplicações paralelas. Diversas políticas de escalonamento, geralmente especializadas, têm sido propostas com vistas e melhorar o desempenho nesses sistemas. O problema é que essas políticas são, na grande maioria dos trabalhos que as propõe, avaliadas apenas com o objetivo de demonstrar sua superioridade sob um determinado aspecto. O desempenho dessas políticas sobe diferentes condições de carga e ambientes computacionais não é verificado.

Nesse contexto, uma das principais preocupações deste trabalho foi demonstrar a necessidade de se analisar o comportamento de uma política de escalonamento para aplicações paralelas sob diferentes condições de carga e em diferentes ambientes computacionais distribuídos. Dessa forma, é possível compreender como a mesma se comportaria em um ambiente real. Tal compreensão pode garantir que se obtenha bons desempenhos e que se tome decisões adequadas quando o cenário muda. Souza (2004) propõe a troca da política de escalonamento, de modo *on-line*, quando as condições de carga se alteram em um sistema. Entretanto, é preciso primeiramente conhecer o comportamento das políticas em cada cenário. Daí a relevância deste trabalho.

A avaliação de desempenho das dez políticas de escalonamento descritas neste trabalho nos três ambientes computacionais e sob diferentes cargas de trabalho permitiram alcançar os objetivos propostos. As conclusões deste trabalho são descritas na próxima seção em função das contribuições esperadas. Essas contribuições foram descritas no Capítulo 1. Também são detalhadas algumas contribuições adicionais deste trabalho e trabalhos que ainda podem ser conduzidos com base nesta dissertação.

8.1 Conclusões e Contribuições

Os resultados obtidos da avaliação de desempenho conduzida neste trabalho demonstram, principalmente, a necessidade de se verificar o comportamento de uma política de escalonamento sob diferentes condições de carga. As políticas de escalonamento obtiveram desempenho ruim quando os cenários mudaram. Um aspecto de grande importância que este trabalho permitiu verificar foi que *grids* são pouco adequados à execução de aplicações paralelas com uma alta carga de comunicação entre seus processos sem que haja uma boa política de escalonamento. Políticas voltadas a *grids* computacionais devem contemplar o tratamento da comunicação entre os processos para que sejam obtidos bons escalonamentos.

A seguir são detalhadas as contribuições desta dissertação. Embora a maior parte das conclusões tenham sido discutidas ao longo da seção **??** – Resultados – do capítulo anterior,

algumas conclusões finais, relacionadas às contribuições, também são abordadas.

Revisão Bibliográfica

Uma ampla revisão bibliográfica foi conduzida neste trabalho. O foco deste trabalho é a avaliação de desempenho, mas para se conduzir uma boa avaliação é necessário conhecer os fatores que interferem no desempenho. Dessa forma, a Computação Paralela Distribuída e o Escalonamento foram tratados nesta dissertação. No que se refere à avaliação de desempenho de políticas de escalonamento e a aplicação de técnicas para comparar políticas a revisão bibliográfica contribuiu significativamente para uma área carente de bons trabalhos. Nos capítulos 4 e 5 desta dissertação analisados os principais trabalhos que tratam da avaliação de desempenho no escalonamento.

Esta dissertação fornece material de referência para trabalhos que envolvam escalonamento de aplicações paralelas, *clusters*, *grids* computacionais e, especialmente, técnicas aplicáveis à avaliação de desempenho do escalonamento.

Definição de uma sistemática para avaliar, comparativamente, políticas de escalonamento

A discussão apresentada a respeito de cargas de trabalho e métricas de desempenho fornecem uma base adequada para a condução de diversas outras comparações entre diferentes políticas de escalonamento. Diversas modelos de carga de trabalho foram apresentados e quatro desses foram analisados sob diferentes aspectos que devem ser considerados na avaliação do escalonamento. As discussões a respeito da métrica “tempo de resposta médio” apontaram-na como uma métrica adequada à comparação de políticas.

A avaliação de desempenho conduzida neste trabalho e os critérios utilizados para sua condução podem ser utilizados como uma sistemática para avaliar, comparativamente, políticas de escalonamento. O simulador, a parametrização dos *clusters* e *grids* computacionais e a análise das cargas de trabalho servem como base para futuras comparações entre políticas de escalonamento para aplicações paralelas.

Complemento de trabalhos que propõem e avaliam políticas de escalonamento

Uma das contribuições deste trabalho refere-se ao relacionamento deste com os demais trabalhos desenvolvidos no grupo de pesquisa – Laboratório de Sistemas Distribuídos e Programação Concorrente (LaSDPC). Esses trabalhos foram descritos na seção 1.2. O trabalho apresentado nesta dissertação complementa alguns desses trabalhos. Dentre as políticas ava-

liadas três foram definidas no LaSDPC: DPWP, NBSP e GAS. Neste trabalho, essas políticas foram avaliadas sob aspectos distintos dos providos no trabalho original que propõe a política. Algumas das contribuições que se pode destacar em relação a esses trabalhos são:

- Comprovou-se que a NBSP se aplica a *grids* computacionais e obtém bons resultados, especialmente, sob alta taxa de comunicação na rede.
- Com relação à política GAS pôde se verificar que tal política oferece melhores desempenhos para aplicações *Network-Bound*.
- A DPWP é uma política com desempenho moderado, oferecendo resultados satisfatórios para aplicações *CPU-Bound* tanto em *clusters* quanto em *grids* computacionais

Além dos trabalhos citados e dos trabalhos que propõem as políticas estudadas neste trabalho outros trabalhos foram complementados com o desenvolvimento desta dissertação. Exemplos são os trabalhos que discutem cargas de trabalho e métricas de desempenho e também trabalhos que comparam políticas de escalonamento. Dentre estes pode-se destacar Arndt et al. (2000) e Karatza e Hilzer (2003).

Verificação da aplicabilidade de políticas de escalonamento voltadas a clusters, em grids computacionais

Todas as políticas estudadas neste trabalho foram desenvolvidas originalmente para *clusters*. Embora tais políticas possam ser utilizadas em *grids* não se podia garantir sua aplicabilidade a *grids* computacionais. Este trabalho, demonstrou através de experimentos que essas políticas, com exceção das políticas NBSP e GAS, tem sua aplicabilidade reduzida em *grids* computacionais.

As políticas Round Robin e Random que nem sequer são adequadas a *cluster* heterogêneos obtiveram tempos de resposta médios muito elevados em *grids* computacionais, especialmente em aplicações que utilizam a rede. Essas políticas não utilizam qualquer informação de carga e não tratam a heterogeneidade. A política Lowest também apresentou resultados piores nos *grids*. O fato dessa política coletar informação de apenas um subconjunto de computadores sistema reduz a troca de informações de carga mas leva a escalonamentos piores do que no caso de coletar informação de todo o sistema, como na política Disted.

As políticas Disted, Central, Global, TLBA e DPWPExt também tiveram sua aplicabilidade reduzida nos *grids*. Enquanto a política NBSP obteve valores inferiores a 2500 segundos de tempo de resposta médio para a execução de 300 jobs da carga Jan97-ASCI as políticas citadas obtiveram tempos de resposta entre 25000 e 35000 segundos. Para o mesmo cenário

de carga de trabalho em um *cluster* de 128 nós, essas políticas, inclusive a NBSP, obtiveram tempos de resposta médios entre 50 e 110 segundos.

Verificação das políticas que melhor se aplicam a determinado cenário

Por meio deste trabalho foi possível identificar algumas políticas que melhor ou pior se aplicam a determinados cenários. Os resultados demonstraram que não há uma política que obtenha bons resultados em todos os cenários. Assim, alguns cenários são identificados a seguir juntamente com a(s) política(s) que melhor ou pior se aplica(m).

- A política NBSP foi superior em todos os cenários em que a carga de comunicação na rede causou atrasos na execução dos processos.
- A política GAS foi a que obteve o pior comportamento para aplicações *bag-of-tasks*.
- A política TLBA apresentou o comportamento mais estável dentre as 10 políticas para os cenários que utilizam *clusters*. Os tempos de resposta médios dessa política está entre os 5 melhores para todos os cenários que envolvem *clusters*.

Demonstração do impacto da comunicação entre os processos em grids computacionais

Os resultados obtidos demonstraram que a comunicação entre processos pode gerar grandes atrasos na execução dos processos em um *grid* computacional. Essa característica pode ser observada pela observação dos gráficos 7.34, 7.35 e 7.36. Enquanto no gráfico 7.34, para uma carga de rede nula, o tempo de resposta máximo é de 130 segundos, no gráfico 7.35 o tempo de resposta sobe para um máximo de 3500 segundos para uma carga de 100 kbps em média por processo e para 30000 segundos no gráfico 7.36. Este gráfico sumariza os resultados para uma carga de rede de 1000 kbps em média por processo.

O impacto causado pela comunicação fica ainda mais evidente ao observar a carga Jan97-ASCI num *cluster* e num *grids*. Enquanto nos gráficos 7.16, 7.17 e 7.18 o aumento da carga de rede não provoca qualquer aumento significativo no tempo de resposta médio dos jobs, nos gráficos 7.28, 7.29 e 7.30 os tempos de resposta médios máximos são de respectivamente 7,02 segundos, 322,81 segundos e 3096,06 segundos.

8.2 Trabalhos Futuros

Diversos outros trabalhos podem ser conduzidos com vistas a melhorar e complementar este trabalho. Além disso, os resultados obtidos nesta dissertação motivam a definição de alguns

trabalhos futuros. A seguir estão algumas diretrizes de trabalhos que podem ser conduzidos neste sentido:

- **Avaliar outras políticas de escalonamento**, a fim de verificar seu comportamento sob diferentes condições de carga e em diferentes sistemas computacionais.
- **Definição de cargas de trabalho voltadas a avaliação do escalonamento em *grids* computacionais**. *Logs* podem ser coletados nesses ambientes e utilizados para caracterizar modelos de carga de trabalho. O grande problema é obter um sistema disponível para tal coleta e que possua uma utilização significativa.
- **Propor melhorias nas políticas avaliadas**. Um exemplo é adicionar tratamento da comunicação, como na NBSP à política TLBA, que se comportou bem em todos os cenários envolvendo cluster.
- **Avaliar o custo computacional e de comunicação gasto para coletar e utilizar a informação de carga do sistema nas decisões do escalonamento**. Uma pesquisa interessante seria verificar o impacto de aumentar o tamanho subconjunto de computadores monitorados pela política Lowest e encontrar um valor limite entre esse subconjunto e melhorias no tempo de resposta.

Referências

ABBAS, A.; ABBAS, A. *Grid Computing*: A practical guide to technology and applications. Rockland, MA, USA: Charles River Media, Inc., 2003.

ALMASI, G. S.; GOTTLIEB, A. *Highly Parallel Computing*. 2. ed. Redwood City: The Benjamin/Cummings Publishing Company, Inc, 1994. 689 p.

AMIR, Y. An opportunity cost approach for job assignment in a scalable computing cluster. *IEEE Transactions on Parallel and Distributed Systems*, v. 11, n. 7, p. 760–768, Julho 2000.

ANSA. *The Advanced Network Systems Architecture (ANSA) Reference Manual*. Castle Hill, Cambridge, England, 1989.

ARAÚJO, A. P. F. *DPWP*: Uma nova abordagem para o escalonamento dinâmico em computação paralela virtual. Dissertação (Mestrado) — ICMC-USP, São Carlos, 1999.

ARAÚJO, A. P. F.; SANTANA, M. J.; SANTANA, R. H. C.; SOUZA, P. S. L. DPWP: A new load balancing algorithm. In: *5th International Conference on Information Systems Analysis and Synthesis - ISAS'99*. Orlando, U.S.A.: [s.n.], 1999.

ARNDT, O.; FREISLEBEN, B.; KIELMANN, T.; THILO, F. A comparative study of online scheduling algorithms for networks of workstations. *Cluster Computing*, Kluwer Academic Publishers, Hingham, MA, USA, v. 3, n. 2, p. 95–112, 2000.

ARPACI-DUSSEAU, A. C. Implicit coscheduling: coordinated scheduling with implicit information in distributed systems. *ACM Transactions on Computer Systems*, v. 19, n. 3, p. 283–331, 2001.

BAKER, M.; BUYYA, R.; LAFORENZA, D. Grids and grid technologies for wide-area distributed computing. *Softw. Pract. Exper.*, John Wiley & Sons, Inc., New York, NY, USA, v. 32, n. 15, p. 1437–1466, 2002.

BANKS, J. (Ed.). *Handbook of Simulation*: Principles, methodology, advances, applications, and practice. Hoboken, NJ: Wiley-Interscience, 1998. 864 p.

BEAUMONT, O.; LEGRAND, A.; ROBERT, Y. Scheduling divisible workloads on heterogeneous platforms. *Parallel Comput.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 29, n. 9, p. 1121–1152, 2003.

BERMAN, F. The grid: blueprint for a new computing infrastructure. In: ____. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999. cap. High-performance schedulers, p. 279–309.

BERMAN, F.; FOX, G.; HEY, A. J. G. *Grid Computing*: Making the global infrastructure a reality. New York, NY, USA: John Wiley & Sons, Inc., 2003.

BERTIS, V. *Fundamentals of Grid Computing*. [S.l.], 2002. IBM RedBooks. Disponível em: <http://www.redbooks.ibm.com/redpapers/pdfs/redp3613.pdf>. Acesso em: 20 abr. 2006.

BOLCH, G.; GREINER, S.; MEER, H. de; TRIVEDI, K. S. *Queueing Networks and Markov Chains*. Hoboken, NJ: Wiley-Interscience, 2005. 896 p.

BRANCO, K. R. L. J. C. *Índices de carga e desempenho em ambientes paralelos/ distribuídos*: modelagem e métricas. Tese (Doutorado) — ICMC-USP, São Carlos, dez. 2004.

BURNS, G.; DAOUD, R.; VAIGL, J. LAM: An Open Cluster Environment for MPI. In: *Proceedings of Supercomputing Symposium*. [s.n.], 1994. p. 379–386. Disponível em: <http://www.lam-mpi.org/download/files/lam-papers.tar.gz>. Acesso em: 16 jan. 2006.

BUYYA, R. (Ed.). *High Performance Cluster Computing*. NJ, USA: Prentice Hall PTR, 1999. 881 p.

BUYYA, R.; KRAUTER, K.; MAHESWARAN, M. A taxonomy and survey of grid resource management systems for distributed computing. *Softw. Pract. Exper.*, John Wiley & Sons, Inc., New York, NY, USA, v. 32, n. 2, p. 135–164, 2002.

CALZAROSSA, M.; MASSARI, L.; TESSERA, D. Workload characterization issues and methodologies. In: *Performance Evaluation: Origins and Directions*. Berlim: Springer-Verlag, 2000. (Lecture Notes in Computer Science, v. 1769), p. 459–481.

CALZAROSSA, M.; SERAZZI, G. A characterization of the variation in time of workload arrival patterns. *IEEE Transactions on Computers*, C-34, n. 2, p. 156–162, February 1985.

CALZAROSSA, M.; SERAZZI, G. Workload characterization: A survey. In: *Proceedings of the IEEE*. [S.l.]: IEEE, 1993. v. 81, n. 8, p. 1136–1150.

CAMPOS, A. *Desenvolvimento de uma interface gráfica para um ambiente de escalonamento*. Dissertação (Mestrado) — ICMC-USP, São Carlos, 2001.

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. *IEEE Transactions on Software Engineering*, v. 14, n. 2, p. 141–154, Fev. 1988.

CHAPIN, S. J.; CIRNE, W.; FEITELSON, D. G.; JONES, J. P.; LEUTENEGGER, S. T.; SCHWIEGELSHOHN, U.; SMITH, W.; TALBY, D. Benchmarks and standards for the evaluation of parallel job schedulers. In: FEITELSON, D. G.; RUDOLPH, L. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer-Verlag, 1999, (Lecture Notes in Computer Science, v. 1659). p. 66–89.

CHIOLA, G.; CIACCIO, G. Efficient parallel processing on low-cost clusters with GAMMA active ports. *Parallel Computing*, v. 26, n. 2, p. 333–354, Fevereiro 2000.

CHODNEKAR, S.; SRINIVASAN, V.; VAIDYA, A. S.; SIVASUBRAMANIAM, A.; DAS, C. R. Towards a communication characterization methodology for parallel applications. In: *HPCA '97: Proceedings of the 3rd IEEE Symposium on High-Performance Computer Architecture*. Washington, DC, USA: IEEE Computer Society, 1997. p. 310.

CHOE, T.; PARK, C. A task duplication based scheduling algorithm with optimality condition in heterogeneous systems. In: *International Conference on Parallel Processing Workshops (ICPPW'02)*. [S.l.]: IEEE Computer Society, 2002. p. 531–536.

CIRNE, W. Grids computacionais: Arquiteturas, tecnologias e aplicações. In: *Terceiro Workshop em Sistemas Computacionais de Alto Desempenho (WSCAD)*. Vitória, Espírito Santo: [s.n.], 2002. Disponível em: <http://walfredo.dsc.ufcg.edu.br/papers/Grids Computacionais- WSCAD.pdf>. Acesso em: 20 abr. 2004.

CIRNE, W.; BERMAN, F. A model for moldable supercomputer jobs. In: *IPDPS '01: Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01)*. Washington, DC, USA: IEEE Computer Society, 2001. p. 10059.2.

COMER, D. E. *Internetworking with TCP/IP*: Principles, protocols, and architecture. 3. ed. New Jersey: Prentice Hall, 1995. 640 p.

Condor Team. *Condor Project Homepage*. 2006. Site. Disponível em: <http://www.cs.wisc-.edu/condor/>. Acesso em: 16 jan. 2006.

CORTES, O. A. C. *Um sistema nebuloso-evolutivo para determinar a portabilidade de benchmarks paralelos*. Tese (Doutorado) — ICMC-USP, São Carlos, 2004.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. *Distributed Systems*: Concepts and design. 2. ed. Harlow: Addison Wesley, 1994. 644 p.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. *Distributed Systems*: Concepts and design. 3. ed. Harlow: Addison Wesley, 2001. 771 p.

CULLER, D. E.; KARP, R. M.; PATTERSON, D. A.; SAHAY, A.; SCHAUSER, K. E.; SANTOS, E.; SUBRAMONIAN, R.; EICKEN, T. von. LogP: Towards a realistic model of parallel computation. In: *Principles Practice of Parallel Programming*. [S.l.: s.n.], 1993. p. 1–12.

DANDAMUDI, S. P.; ZHOU, Z. Performance of adaptive space-sharing policies in dedicated heterogeneous cluster systems. *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 20, n. 5, p. 895–906, 2004.

DEITEL, H. M.; DEITEL, P. J. *Java How to Program*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

DONGARRA, J.; FOSTER, I.; FOX, G.; GROPP, W.; KENNEDY, K.; TORCZON, L.; WHITE, A. (Ed.). *Sourcebook of Parallel Computing*. San Francisco: Morgan Kaufmann Publishers, 2003. 842 p.

DOWNEY, A. B. A parallel workload model and its implications for processor allocation. *Cluster Computing*, v. 1, n. 1, p. 133–145, 1998.

DOWNEY, A. B.; FEITELSON, D. G. The elusive goal of workload characterization. *SIGMETRICS Perform. Eval. Rev.*, ACM Press, New York, NY, USA, v. 26, n. 4, p. 14–29, 1999.

DOWNING, T. B. *Java RMI: Remote Method Invocation*. Foster City, CA, USA: IDG Books Worldwide, Inc., 1998.

DUNCAN, R. A survey of parallel computer architectures. In: *Computer*. Los Alamitos, CA, USA: IEEE Computer Society Press, 1990, (Survey & Tutorial Series, 2). p. 5–16.

EL-GHAZAWI, T.; GAJ, K.; ALEXANDRIDIS, N.; VROMAN, F.; NGUYEN, N.; RADZIKOWSKI, J. R.; SAMIPAGDI, P.; SUBOH, S. A. A performance study of job management systems: Research articles. *Concurr. Comput. : Pract. Exper.*, John Wiley and Sons Ltd., Chichester, UK, UK, v. 16, n. 13, p. 1229–1246, 2004.

ESHAGHIAN, M. M.; WU, Y.-C. Resource estimation for heterogeneous computing. *Future Gener. Comput. Syst.*, Elsevier Science Publishers B. V., v. 12, n. 6, p. 505–520, 1997.

FEITELSON, D. G. Packing schemes for gang scheduling. In: *IPPS '96: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*. London, UK: Springer-Verlag, 1996. p. 89–110.

FEITELSON, D. G. Scheduling parallel jobs on clusters. In: ____. *High Performance Cluster Computing*: Architectures and systems. NJ, USA: Prentice Hall, 1999. v. 1, p. 519–533.

FEITELSON, D. G. Metrics for parallel job scheduling and their convergence. In: FEITELSON, D. G.; RUDOLPH, L. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer Verlag, 2001, (Lecture Notes in Computer Science, v. 2221). p. 188–205.

FEITELSON, D. G. *Analyzing the Root Causes of Performance Evaluation Results*. [S.l.], Mar. 2002.

FEITELSON, D. G. The forgotten factor: Facts on performance evaluation and its dependence on workloads. In: *Euro-Par '02: Proceedings of the 8th International Euro-Par Conference on Parallel Processing*. London, UK: Springer-Verlag, 2002. p. 49–60.

FEITELSON, D. G. Workload modeling for performance evaluation. In: CALZAROSSA, M. C.; TUCCI, S. (Ed.). *Performance Evaluation of Complex Systems*: Techniques and tools. Berlim: Springer-Verlag, 2002, (Lecture Notes in Computer Science, v. 2459). p. 114–141.

FEITELSON, D. G. Metric and workload effects on computer systems evaluation. *Computer*, IEEE Computer Society Press, Los Alamitos, CA, USA, v. 36, n. 9, p. 18–25, 2003.

FEITELSON, D. G. *Workload Modeling for Computer Systems Performance Evaluation*. Abr. 2006. Book Draft. Disponível em: <http://www.cs.huji.ac.il/~feit/wlmod/wlmod.pdf>. Acesso em: 25 abr. 2006.

FEITELSON, D. G.; RUDOLPH, L. Parallel job scheduling: Issues and approaches. In: FEITELSON, D. G.; RUDOLPH, L. (Ed.). *Job Scheduling Strategies for Parallel Processing – IPPS'95 Workshop*. Berlim: Springer, 1995. (Lecture Notes in Computer Science, v. 949), p. 1–18.

FEITELSON, D. G.; RUDOLPH, L. Metrics and benchmarking for parallel job scheduling. In: FEITELSON, D. G.; RUDOLPH, L. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim-Heidelberg: Springer-Verlag, 1998, (Lecture Notes in Computer Science, v. 1459). p. 1–24.

FEITELSON, D. G.; RUDOLPH, L.; SCHWIEGELSHOHN, U.; SEVCIK, K. C.; WONG, P. Theory and practice in parallel job scheduling. In: FEITELSON, D. G.; RUDOLPH, L. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer Verlag, 1997, (Lecture Notes in Computer Science, v. 1291). p. 1–34.

FIGUEIREDO, T. C. *Interface AMIGO-MPI*: Uma abordagem flexível e dinâmica para o escalonamento de processos. Dissertação (Mestrado) — ICMC-USP, São Carlos, 2000.

FLYNN, M. J.; RUDD, K. W. Parallel architectures. *ACM Computing Surveys*, v. 28, n. 1, p. 67–70, mar. 1996.

FORTIER, P. J.; MICHEL, H. E. *Computer Systems Performance Evaluation and Prediction*. Boston, USA: Digital Press, 2003. 525 p.

FOSTER, I. *Designing and Bulding Parallel Programs*: Concepts and tools for parallel software engineering. New York: Addison-Wesley Publishing Company, 1995. Disponível em: <http://www-unix.mcs.anl.gov/dbpp/text/book.html>. Acesso em: 16 jan. 2006.

FOSTER, I. *What is the Grid? A Three Point Checklist*. Chicago, 2002. Disponível em: <http://www-fp.mcs.anl.gov/˜foster/Articles/WhatIsTheGrid.pdf>. Acesso em: 09 abr. 2006.

FOSTER, I. Globus toolkit version 4: Software for service-oriented systems. In: *IFIP International Conference on Network and Parallel Computing*. Berlim: Springer-Verlag GmbH, 2005, (Lecture Notes in Computer Science, v. 3779). p. 2–13.

FRACHTENBERG, E.; FEITELSON, D. G. Pitfalls in parallel job scheduling evaluation. In: FEITELSON, D. G.; FRACHTENBERG, E.; RUDOLPH, L.; SCHWIEGELSHOHN, U. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer Verlag, 2005, (Lecture Notes in Computer Science, v. 3834). p. 257–282.

FRACHTENBERG, E.; FEITELSON, D. G.; FERNANDEZ, J.; PETRINI, F. Parallel job scheduling under dynamic workloads. In: FEITELSON, D. G.; RUDOLPH, L.; SCHWIEGELSHOHN, U. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer Verlag, 2003, (Lecture Notes in Computer Science, v. 2682). p. 208–227.

FRANCÊS, C. R. L. *Statecharts Estocásticos e Queuing Statecharts*: Novas abordagens para avaliação de desempenho baseadas em especificação statecharts. Tese (Doutorado) — ICMC-USP, São Carlos, ago. 2001.

FRANCÊS, C. R. L.; SANTANA, M. J.; SANTANA, R. H. C. Statecharts and markov chains for performance evaluation: Its applications and implications. In: *Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*. Monte Carlo Resort, Las Vegas, Nevada, USA: [s.n.], 1999.

FRANCÊS, C. R. L.; SPOLON, R.; SANTANA, M. J.; SANTANA, R. H. C. *Modelagem e Especificação Utilizando Redes de Petri e Statecharts*. São Carlos, 2000.

FRANKE, H.; JANN, J.; MOREIRA, J. E.; PATTNAIK, P.; JETTE, M. A. An evaluation of parallel job scheduling for asci blue-pacific. In: *Supercomputing '99: Proceedings of the 1999 ACM/IEEE conference on Supercomputing (CDROM)*. New York, NY, USA: ACM Press, 1999. p. 45.

FUJIMOTO, R. M. Parallel simulation: distributed simulation systems. In: *WSC '03: Proceedings of the 35th conference on Winter simulation*. New Orleans, Louisiana: Winter Simulation Conference, 2003. p. 124–134.

GEIST, A.; BEGUELIN, A.; DONGARRA, J.; JIANG, W.; MANCHEK, R.; SUNDERAM, V. *Parallel Virtual Machine*: A users' guide and tutorial for networked parallel computing. Cambridge, MA: MIT Press, 1994. 176 p. (Scientific and Engineering Computation). Disponível em: <http://www.netlib.org/pvm3/book/pvm-book.html>. Acesso em: 16 jan. 2006.

GRAMA, A.; KARYPIS, G.; KUMAR, V.; GUPTA, A. *Introduction to Parallel Computing*. 2. ed. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2003. 689 p.

GROVE, D. A.; CODDINGTON, P. D. Communication benchmarking and performance modelling of MPI programs on cluster computers. *J. Supercomput.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 34, n. 2, p. 201–217, 2005.

HAMSCHER, V.; SCHWIEGELSHOHN, U.; STREIT, A.; YAHYAPOUR, R. Evaluation of job-scheduling strategies for grid computing. In: *GRID'00: Proceedings of the First IEEE/ACM International Workshop on Grid Computing*. London, UK: Springer-Verlag, 2000. p. 191–202.

HARCHOL-BALTER, M.; DOWNEY, A. B. *Exploiting Process Lifetime Distributions for Dynamic Load Balancing*. Berkeley, CA, USA, 1995.

HARIRI, S.; PARASHAR, M. *Tools and Environments for Parallel and Distributed Computing*. Hoboken, New Jersey: Wiley-Interscience, 2004. 232 p.

HPC WORKLOAD/RESOURCE TRACE RESPOSITORY. *HPC Workload/Resource Trace Respository*. 2005. Disponível em: <http://www.supercluster.org/research/traces/>. Acesso em: 31 mar. 2006.

HWANG, K.; XU, Z. *Scalable Parallel Computing*: Technology, architecture, programming. 1. ed. New York: McGraw-Hill, 1998. 802 p.

INTERNATIONAL STANDARDS ORGANIZATION. *Basic Reference Model of Open Distributed processing, Part 1: Overview and guide to use*. ISO/IEC JTC1/SC212/WG7 CD 10746-1, 1992.

ISHII, R. P. *NBSP*: Uma política de escalonamento network-bound para aplicações paralelas distribuídas. Dissertação (Mestrado) — ICMC-USP, São Carlos, 2004.

ISHII, R. P.; MELLO, R. F. de; SENGER, L. J.; SANTANA, M. J.; SANTANA, R. H. C. Scheduling based on the impact over process communication of parallel applications. In: *HPCS '05: Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications (HPCS'05)*. Washington, DC, USA: IEEE Computer Society, 2005. p. 149–155.

JAIN, R. *The Art of Computer Systems Performance Analysis*: Techniques for experimental design, measurement, simulation and modeling. New York: John Wiley & Sons, 1991. 685 p.

JANN, J.; PATTNAIK, P.; FRANKE, H.; WANG, F.; SKOVIRA, J.; RIODAN, J. Modeling of workload in MPPs. In: FEITELSON, D. G.; RUDOLPH, L. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer Verlag, 1997, (Lecture Notes in Computer Science, v. 1291). p. 95–116.

JOSEPH, J.; FELLENSTEIN, C. *Grid Computing*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2003. 400 p.

KANT, K. *Introduction to Computer System Performance Evaluation*. New York, NY, USA: McGraw-Hill, 1992.

KARATZA, H. D. Scheduling strategies for multitasking in a distributed system. In: IEEE COMPUTER SOCIETY. *33rd Annual Simulation Symposium*. Washington: IEEE Computer Society, 2000. p. 83–.

KARATZA, H. D. Gang scheduling performance on a cluster of non-dedicated workstations. In: IEEE COMPUTER SOCIETY. *Proceedings of the 35th Annual Simulation Symposium (ANSS'02)*. San Diego, California: IEEE Computer Society, 2002. p. 235–.

KARATZA, H. D. Task scheduling performance in distributed systems with time varying workload. *Neural, Parallel Sci. Comput.*, Dynamic Publishers, Inc., Atlanta, GA, USA, v. 10, n. 3, p. 325–338, 2002.

KARATZA, H. D. Simulation of parallel and distributed systems scheduling. *Applied system simulation: methodologies and applications*, Kluwer Academic Publishers, Norwell, MA, USA, p. 61–80, 2003.

KARATZA, H. D.; HILZER, R. C. Performance analysis of parallel job scheduling in distributed systems. In: *ANSS'03: Proceedings of the 36th annual symposium on Simulation*. Washington, DC, USA: IEEE Computer Society, 2003. p. 109–116.

KRALLMANN, J.; SCHWIEGELSHOHN, U.; YAHYAPOUR, R. On the design and evaluation of job scheduling algorithms. In: FEITELSON, D. G.; RUDOLPH, L. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim-Heidelberg: Springer Verlag, 1999, (Lecture Notes in Computer Science, v. 1659). p. 17–42.

KUROSE, J. F.; ROSS, K. W. *Redes de Computadores e a Internet: uma nova abordagem*. São Paulo: Addison-Wesley, 2003. 548 p. Tradução Arlete Simille Marques.

LAM Team. *LAM/MPI Parallel Computing*. 2004. Disponível em: <http://www.lam-mpi.org-/>. Acesso em: 09 nov. 2004.

LASTOVETSKY, A. L. *Parallel Computing on Heterogeneous Networks*. New York, NY, USA: John Wiley & Sons, Inc., 2003. 440 p.

LELAND, W. E.; OTT, T. J. Load balancing heuristics and process behavior. *ACM Performance Evaluation Review*, v. 14, n. 1, p. 54–69, May 1986.

LI, H.; GROEP, D.; WOLTERS, L. Workload characteristics of a multi-cluster supercomputer. In: FEITELSON, D. G.; RUDOLPH, L.; SCHWIEGELSHOHN, U. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer Verlag, 2004, (Lect. Notes Comput. Sci., v. 3277). p. 176–193.

LO, V.; MACHE, J.; WINDISCH, K. A comparative study of real workload traces and synthetic workload models for parallel job scheduling. In: FEITELSON, D. G.; RUDOLPH, L. (Ed.). *Job Scheduling Strategies for Parallel Processing*. [S.l.]: Springer Verlag, 1998, (Lectures Notes Computer Science, v. 1459). p. 25–46.

LUBLIN, U.; FEITELSON, D. G. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. *J. Parallel Distrib. Comput.*, Academic Press, Inc., Orlando, FL, USA, v. 63, n. 11, p. 1105–1122, 2003.

MADAN, S. A benchmark for artificial intelligence applications on parallel computers - BEAP. In: IEEE. *WESCANEX'97 Proceedings*. Winnipeg: IEEE, 1997. p. 82–87. Conference on Communications, Power and Computing.

MAJUMDAR, S.; PARSONS, E. W. Parallel job scheduling: A performance perspective. In: *Performance Evaluation*: Origins and directions. Berlim: Springer-Verlag, 2000, (Lecture Notes in Computer Science, v. 1769). p. 233–252.

MARSAN, M. A.; BOBBIO, A.; DONATELLI, S. Petri nets in performance analysis: An introduction. In: *Lectures on Petri Nets I: Basic Models, Advances in Petri Nets, the volumes are based on the Advanced Course on Petri Nets*. London, UK: Springer-Verlag, 1998. (Lectures Notes in Computer Science), p. 211–256.

Maui HPC Center. *LoadLeveler*. 2005. Site. Disponível em: <http://www.mhpcc.edu/training-/workshop/loadleveler/MAIN.html>. Acesso em: 16 jan. 2006.

MELLO, R. F. de. *Proposta e avaliação de desempenho de um algoritmo de balanceamento de carga para ambientes distribuídos heterogênos escaláveis*. Tese (Doutorado) — EESC-USP, São Carlos, 2003.

MELLO, R. F. de; PAIVA, M. S. V.; TREVELIN, L. C.; GONZAGA, A. Analysis on the significant information to update the tables on occupation of resources by using a peer-to-peer protocol. In: *HPCS '02: Proceedings of the 16th Annual International Symposium on High Performance Computing Systems and Applications*. Washington, DC, USA: IEEE Computer Society, 2002. p. 229.

MELLO, R. F. de; SENGER, L. J. A new migration model based on the evaluation of processes load and lifetime on heterogeneous computing environments. In: *SBAC-PAD '04: Proceedings of the 16th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD'04)*. Washington, DC, USA: IEEE Computer Society, 2004. p. 222–227.

MELLO, R. F. de; SENGER, L. J. Model for simulation of heterogeneous high-performance computing environments. In: *VECPAR'06 7th International Conference on High Performance Computing in Computational Sciences*. Rio de Janeiro: [s.n.], 2006. (Aceito, mas não publicado ainda).

MELLO, R. F. de; SENGER, L. J.; YANG, L. T. A routing load balancing policy for grid computing environments. In: IEEE COMPUTER SOCIETY. *20th International Conference on Advanced Information Networking and Applications (AINA'06)*. Vienna, Austria: IEEE Computer Society, 2006. v. 1, p. 153–158.

MINOLI, D. *A Networking Approach to Grid Computing*. Hoboken, New Jersey: Wiley-Interscience, 2004. 400 p.

MPI Forum. *MPI-2: Extensions to the Message Passing Interface*. 2005. Disponível em: <http://www.mpi-forum.org/>. Acesso em: 10 dez. 2005.

MPI Forum. *MPI-2 Journal of Development*. 2005. Disponível em: <http://www.mpi-forum-.org/>. Acesso em: 10 dez. 2005.

MULLENDER, S. (Ed.). *Distributed Systems*. 2. ed. New York: Addison-Wesley Publishing Company, 1993. 601 p. (ACM Press Frontier Series).

NAIK, V. K.; SETIA, S. K.; SQUILLANTE, M. S. Processor Allocation in Multiprogrammed Distributed-memory Parallel Computer Systems. *Journal of Parallel and Distributed Computing*, v. 47, n. 1, p. 28–47, 1997.

NEARY, M. O.; CAPPELLO, P. Internet-based tsp computation with javelin++. In: *Proceedings of the 2000 International Workshop on Parallel Processing*. [S.l.]: IEEE Computer Society, 2000. p. 137–144.

NEVIN, N. J. *The Performance of LAM 6.0 and MPICH 1.0.12 on a Workstation Cluster*. Columbus, Ohio, 1996.

NONATO, S. A. *Utilização de Redes de Petri em Simulação de Sistemas Computacionais*. Dissertação (Mestrado) — ICMC-USP, São Carlos, fev. 1999.

ORLANDI, R. C. G. S. *Ferramenta para análise de desempenho de sistemas computacionais distribuídos*. Dissertação (Mestrado) — ICMC-USP, São Carlos, 1995.

PWA: Parallel workloads archive. Mantido por Dror Feitelson. 2005. Disponível em: <http://www.cs.huji.ac.il/labs/parallel/workload/>. Acesso em: 31 mar. 2006.

QUINN, M. J. *Parallel Computing*: Theory and practice. 2. ed. New York: McGraw Hill, 1994. 446 p.

REIS, V. Q. dos. *Escalonamento em grids computacionais*: estudo de caso. Dissertação (Mestrado) — ICMC-USP, São Carlos, 2005.

SABIN, G.; SADAYAPPAN, P. Unfairness metrics for space-sharing parallel job schedulers. In: FEITELSON, D. G.; FRACHTENBERG, E.; RUDOLPH, L.; SCHWIEGELSHOHN, U. (Ed.). *Job Scheduling Strategies for Parallel Processing*. [S.l.]: Springer Verlag, 2005, (Lect. Notes Comput. Sci. vol., v. 3834). p. 238–256.

SANTANA, M. J.; ZALUSKA, E. J. Load balancing in a session-based distributes file-store architecture. *Software Practice and Experience*, v. 18(11), p. 1091–1107, Nov. 1988.

SANTANA, R. H. C.; SANTANA, M. J.; ORLANDI, R. C. G. S.; SPOLON, R.; JÚNIOR, N. C. *Técnicas para Avaliação de Desempenho de Sistemas Computacionais*. São Carlos, 1994.

SANTOS, R. R. dos. *Escalonamento de Aplicações Paralelas*: Interface amigo-corba. Dissertação (Mestrado) — ICMC-USP, São Carlos, 2001.

SENGER, L. J. *Escalonamento de Processos*: Uma abordagem dinâmica e incremental para a exploração de características de aplicações paralelas. Tese (Doutorado) — ICMC-USP, São Carlos, ago. 2005.

SEVCIK, K. C. Application scheduling and processor allocation in multiprogrammed parallel processing systems. *Performance Evaluation*, Amsterdam, Netherlands, v. 19, n. 2-3, p. 107–140, 1994.

SHIM, Y.-C. Performance evaluation of scheduling schemes for NOW with heterogeneous computing power. *Future Generation Computer Systems*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 20, n. 2, p. 229–236, 2004.

SHIVARATRI, N. G.; KRUEGER, P.; SINGHAL, M. Load distributing for locally distributed systems. *IEEE Computer*, v. 25, n. 12, p. 33–44, 1992.

SIVASUBRAMANIAM, A. Execution-driven simulators for parallel systems design. In: *Winter Simulation Conference*. [S.l.: s.n.], 1997. p. 1021–1028.

SNIR, M.; OTTO, S.; HUSS-LEDERMAN, S.; WALKER, D.; DONGARRA, J. *MPI*: The complete reference. Cambridge, MA: MIT Press, 1996. (Scientific and Engineering Computation). Disponível em: <http://www.netlib.org/utk/papers/mpi-book/>. Acesso em: 16 jan. 2006.

SONG, B.; ERNEMANN, C.; YAHYAPOUR, R. Parallel computer workload modeling with Markov chains. In: FEITELSON, D. G.; RUDOLPH, L.; SCHWIEGELSHOHN, U. (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer Verlag, 2004, (Lecture Notes in Computer Science, v. 3277). p. 47–62.

SOUZA, J. A. M. de. *Avaliação de Desempenho de Políticas de Escalonamento de Processos*. Dissertação (Mestrado) — ICMC-USP, São Carlos, 2000.

SOUZA, M. A. de. *Avaliação das Rotinas de Comunicação Ponto-a-Ponto do MPI*. Dissertação (Mestrado) — ICMC-USP, São Carlos, Junho 1997.

SOUZA, M. A. de. *Uma abordagem para a avaliação do escalonamento de processos em sistemas distribuídos baseada em monitoração*. Tese (Doutorado) — ICMC-USP, São Carlos, 2004.

SOUZA, P. S. L.; SANTANA, M. J.; SANTANA, R. H. C. A new scheduling environment for near-optimal performance. In: *International Conference on Parallel and Distributed Processing Techniques and Applications -PDPTA'99*. Las Vegas, Nevada, U.S.A.: [s.n.], 1999.

SOUZA, P. S. L. de. *AMIGO: Uma Contribuição para a Convergência na Área de Escalonamento de Processos*. Tese (Doutorado) — IFSC–USP, São Carlos, 2000.

STALLINGS, W. *Arquitetura e Organização de Computadores*: Projeto para o desempenho. 5. ed. São Paulo: Prentice Hall, 2003. 786 p. Tradução: Carlos Camarão de Figueiredo e Lucília Camarão de Figueiredo.

STREIT, A. *Self-Tuning Job Scheduling Strategies for the Resource Management of HPC Systems and Computational Grids*. Dissertação (Mestrado) — Fakultät fur Elektrotechnik, Informatik und Mathematik der Universität Paderborn, Paderborn, 2003.

SUNDERAM, V. S.; GEIST, G. A. Heterogeneous parallel and distributed computing. *Parallel Computing*, Elsevier Science Publishers B. V., Amsterdam, The Netherlands, n. 25, p. 1699–1721, 1999.

TANENBAUM, A. S. *Distributed Operating Systems*. New Jersey: Prentice Hall, 1995. 614 p.

TANENBAUM, A. S. *Sistemas Operacionais Modernos*. Rio de Janeiro: LTC, 1995. 493 p. Tradução Nery Machado Filho.

TANENBAUM, A. S. *Redes de Computadores*. 3. ed. Rio de Janeiro: Campus, 1997. 923 p. Tradução da terceira edição.

TANENBAUM, A. S. *Structured Computer Organization*. 4. ed. Upper Saddle River, NJ: Prentice Hall, 1999.

TANENBAUM, A. S. *Modern Operating Systems*. 2. ed. Upper Saddle River, N.J.: Prentice-Hall, Inc., 2001. 951 p.

TANENBAUM, A. S.; STEEN, M. van. *Distributed Systems*: Principles and paradigms. Upper Saddle River, NJ: Prentice Hall, 2002. 803 p.

TSAFRIR, D.; ETSION, Y.; FEITELSON, D. G. Modeling user runtime estimates. In: FEITELSON, D. G.; FRACHTENBERG, E.; RUDOLPH, L.; UWE (Ed.). *Job Scheduling Strategies for Parallel Processing*. Berlim: Springer Verlag, 2005. (Lect. Notes Comput. Sci., v. 3834), p. 1–35.

TSAFRIR, D.; FEITELSON, D. G. *Workload flurries*. Jerusalem, 2003.

VETTER, J. S.; MUELLER, F. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. *J. Parallel Distrib. Comput.*, Academic Press, Inc., v. 63, n. 9, p. 853–865, 2003.

VOORSLUYS, W. *Avaliação de índices de carga de memória em sistemas computacionais distribuídos*. Dissertação (Mestrado) — ICMC-USP, São Carlos, abril 2006.

WEICKER, R. Benchmarking. In: CALZAROSSA, M.; TUCCI, S. (Ed.). *Performance Evaluation of Complex Systems: Techniques and Tools : Performance 2002. Tutorial Lectures*. London, UK: Springer-Verlag, 2002, (Lecture Notes in Computer Science, v. 2459). p. 179–207.

WONG, A. T.; OLIKER, L.; KRAMER, W. T. C.; KALTZ, T. L.; BAILEY, D. H. ESP: A system utilization benchmark. In: *Supercomputing '00: Proceedings of the 2000 ACM/IEEE conference on Supercomputing (CDROM)*. Washington, DC, USA: IEEE Computer Society, 2000. p. 15.

XU, M. Q. Effective metacomputing using lsf multicluster. In: *CCGRID '01: Proceedings of the 1st International Symposium on Cluster Computing and the Grid*. Washington, DC, USA: IEEE Computer Society, 2001. p. 100.

YANG, M.-T.; KASTURI, R.; SIVASUBRAMANIAM, A. A pipeline-based approach for scheduling video processing algorithms on NOW. *IEEE Trans. Parallel Distrib. Syst.*, IEEE Press, v. 14, n. 2, p. 119–130, 2003.

ZAKI, M. J.; LI, W.; CIERNIAK, M. *Performance Impact of Processor and Memory Heterogeneity in a Network of Machines*. Rochester, NY, USA, Março 1995.

ZHANG, Y.; SIVASUBRAMANIAM, A.; MOREIRA, J.; FRANKE, H. A simulation based study of scheduling mechanisms for a dynamic cluster environment. In: *Proceedings of the 2000 international conference on Supercomputing*. Santa Fe, New Mexico, USA: ACM, 2000. p. 100–109.

ZHANG, Y.; SIVASUBRAMANIAM, A.; MOREIRA, J.; FRANKE, H. Impact of workload and system parameters on next generation cluster scheduling mechanisms. *IEEE Trans. Parallel Distrib. Syst.*, IEEE Press, Piscataway, NJ, USA, v. 12, n. 9, p. 967–985, 2001.

ZHOU, M.; VENKATESH, K. *Modeling, Simulation, and Control of Flexible Manufacturing Systems: A Petri Net Approach*. [S.l.]: World Scientific Pub Co., 1999.

ZHOU, S.; FERRARI, D. *An Experimental Study of Load Balancing Performance*. California, 1987. UCB/CSD-87-336.

ZHOU, S.; ZHENG, X.; WANG, J.; DELISLE, P. Utopia: a load sharing facility for large, heterogeneous distributed computer systems. *Softw. Pract. Exper.*, John Wiley & Sons, Inc., New York, NY, USA, v. 23, n. 12, p. 1305–1336, 1993.

ZILBER, J.; AMIT, O.; TALBY, D. What is worth learning from parallel workloads?: a user and session based analysis. In: *ICS '05: Proceedings of the 19th annual international conference on Supercomputing*. New York, NY, USA: ACM Press, 2005. p. 377–386.

ZOMAYA, A. Y. (Ed.). *Parallel and Distributed Computing Handbook*. New York: McGraw-Hill, 1996. 1198 p. (Computer Engineering Series).

Apêndice

A

Logs do Parallel Workload Archive

Grande parte dos ambientes de produção mantêm logs da execução de aplicações paralelas com o objetivo de usá-los para avaliar a utilização do ambiente. Alguns desse logs são disponibilizados para uso por pesquisadores. Um repositório de logs é o Parallel Workload Archive (PWA, 2005). Este apêndice descreve os logs contidos no Parallel Workload Archive.

Os dados coletados podem diferir de log para log. Entretanto, alguns estão disponíveis em todos, como momento de chegada, início e fim da execução de um job, número de recursos requeridos, usuário. Alguns dados dependem do escalonador utilizado, como a estimativa de tempo de execução.

O Parallel Workload Archive contém logs bastante detalhados. Eles foram coletados em ambientes de produção de larga escala de vários países do mundo. Os logs foram convertidos para o formato padrão para cargas de trabalho (*vide* seção 5.2.2). Além disso eles foram tratados para retirar atividades não representativas, como repetidas submissões de um mesmo job por parte de um usuário. O trabalho de Tsafrir e Feitelson (2003) detalha as informações que devem ser retiradas e seus efeitos sobre a avaliação de desempenho.

A tabela A.1, na próxima página, resume as principais características contidas nos logs do Parallel Workload Archive.

Tabela A.1: Resumo dos logs disponíveis no *Parallel Workload Archive* (PWA, 2005)[a]

LOG	MÁQ. PARALELA	UCPs	DURAÇÃO	JOBS	USUÁRIOS	UTIL. (%)	INFORMAÇÃO ADICIONAL
NASA Ames iPSC/860	iPSC/860 hipercubo	128	10/1993 12/1993	42264	69	46.7	Número de nós requisitados limitado a potência de 2. Usa NQS.
SDSC 1995	Intel Paragon	416	12/1994 12/1995	76872	98	68.2	Apenas 352 nós para aplicações paralelas via NQS.
SDSC 1996	Intel Paragon	416	12/1995 12/1996	38719	60	72.2	Apenas 352 nós para aplicações paralelas via NQS.
CTC SP2	IBM SP2	512	06/1996 05/1997	79302	679	55.6	EASY e LoadLeveler. Nós heterogêneos.
KTH SP2	IBM SP2	100	07/1996 08/1997	28490	214	69.0	Escalonador EASY. Nós heterogêneos.
LANL CM-5	Connection Machine CM-5 from Thinking Machines	1024	10/1994 09/1996	201387	213	66.1	Particionamento fixo, potência de 2. Menor partição 32 nós
LLNL T3D	Cray T3D	256	06/1996 09/1996	21323	153	61.6	Suporta *gang-scheduling*.
SDSC SP2	IBM SP2	128	04/1998 04/2000	59725	437	83.2	
LANL Origin 2000	Cluster de 16 máquinas Origin 2000 com 128 proc. cada	2048	11/1999 04/2000	121989	337	64.0	LSF é o ambiente de escalonamento
OSC Cluster	Cluster Linux de 32 *quad-processor* e 25 *dual-processor*	178	01/2000 11/2001	80696	210	50.5	Usa software de escalonamento Maui 3.0.7 (aplicações batch)
SDSC Blue Horizon	IBM SP, com 8 processadores por nó	1152	04/2000 01/2003	243314	468	76.2	
DAS2 fs0	Um dos 5 clusters Linux/dual P3 1GHz de um Grid de Pesquisa	144	01/2003 01/2004	225711	102	14.9	Localizado na Vrije Univ. Amsterdam
DAS2 fs1	Cluster Linux/dual P3 1GHz	64	01/2003 12/2003	40315	36	12.1	Localizado na Univ. Leiden
DAS2 fs2	Cluster Linux/dual P3 1GHz	64	01/2003 12/2003	66429	52	19.1	Localizado na Univ. de Amsterdan
DAS2 fs3	Cluster Linux/dual P3 1GHz do Grid	64	01/2003 12/2003	66737	64	10.7	Localizado na Delft Univ. of Tecnhonology
DAS2 fs4	Cluster Linux/dual P3 1GHz do Grid	64	01/2003 12/2003	33795	40	14.5	Localizado na Univ. Utrecht
LPC	Cluster de 70 Linux duais 3GHz P4 Xeon	140	08/2004 05/2005	242695	56	21.6	Particionado em dois clusters de 28 e 42 nós. Parte do EGEE Grid

[a]Conforme recomendado em Tsafrir e Feitelson (2003) os dados foram limpos para omitir atividades não representativas.

Apêndice

B

Intervalos de Confiança

Para amostras pequenas, em que o desvio padrão amostral é uma estimativa menos confiável do desvio padrão populacional, o intervalo de confiança é calculado com base na distribuição t, conforme a fórmula B.1.

$$(\overline{x} - t_{(n-1,0.05)} * \frac{s}{\sqrt{n}}, \overline{x} + t_{(n-1,0.05)} * \frac{s}{\sqrt{n}}) \tag{B.1}$$

onde $\overline{x}$ é a média amostral; $t_{(n-1,0.05)}$ é o valor obtido da tabela de distribuição t, com n-1 graus de liberdade para um intervalo de confiança de 95%; s é o desvio padrão amostral e n é o número de amostras.

As tabelas a seguir apresentam as médias e os intervalos de confiança dos experimentos conduzidos neste trabalho.

Tabela B.1: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Feit96, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	04,70 ± 00,60	04,17 ± 00,81	03,49 ± 00,01	03,48 ± 00,02	03,48 ± 00,01	03,49 ± 00,03	08,06 ± 00,81	03,96 ± 00,77	03,49 ± 00,01	03,49 ± 00,02
25	12,55 ± 02,06	11,68 ± 03,13	14,22 ± 00,74	12,84 ± 01,34	12,98 ± 01.13	13,86 ± 00,89	16,30 ± 01,64	13,92 ± 04,08	14,24 ± 00,64	14,54 ± 00,13
50	26,64 ± 06,20	21,73 ± 03,99	20,75 ± 00,38	20,30 ± 00,88	20,54 ± 00,68	20,37 ± 00,56	38,91 ± 07,21	21,53 ± 01,77	20,11 ± 00,96	20,34 ± 00,67
75	52,23 ± 09,21	30,22 ± 02,35	28,86 ± 00,60	28,76 ± 00,45	28,70 ± 00,58	28,81 ± 00,52	82,24 ± 04,26	36,82 ± 02,08	35,26 ± 00,55	35,25 ± 00,36
100	63,83 ± 07,18	31,53 ± 01.17	32,06 ± 00,16	32,20 ± 00,45	32,14 ± 00,46	31,81 ± 00,25	136,66 ± 11,35	43,78 ± 01,20	43,33 ± 00,51	44,60 ± 00,74
150	51,83 ± 04,05	29,17 ± 01,36	29,10 ± 00,26	29,20 ± 00,17	29,35 ± 00,35	28,03 ± 00,47	101,48 ± 06,37	37,82 ± 00,76	38,90 ± 00,79	39,14 ± 00,87
200	43,18 ± 02,87	24,24 ± 01,01	25,08 ± 00,17	25,18 ± 00,22	25,05 ± 00,28	24,20 ± 00,48	88,78 ± 06,56	33,35 ± 01,22	33,16 ± 00,66	33,65 ± 00,72
250	38,80 ± 02,94	20,63 ± 01,09	22,20 ± 00,14	22,23 ± 00,14	22,15 ± 00,14	21,67 ± 00,35	74,02 ± 03,58	28,48 ± 00,88	28,88 ± 00,53	28,27 ± 00,27
300	34,56 ± 03,47	17,96 ± 00,61	19,14 ± 00,19	19,08 ± 00,09	19,23 ± 00,21	18,53 ± 00,24	64,74 ± 02,21	23,83 ± 00,39	24,67 ± 00,29	24,56 ± 00,29

Tabela B.2: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Feit96, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	06,09 ± 02,08	03,26 ± 00,11	03,48 ± 00,01	03,50 ± 00,02	03,49 ± 00,02	03,48 ± 00,01	05,81 ± 00,06	03,93 ± 00,74	03,50 ± 00,02	03,49 ± 00,01
25	25,07 ± 23,48	12,33 ± 04,07	14,09 ± 00,76	14,05 ± 00,79	14,68 ± 00,10	14,40 ± 00,19	10,95 ± 00,06	14,04 ± 04,18	13,92 ± 01,04	13,58 ± 01,05
50	24,98 ± 05,03	25,12 ± 04,91	20,68 ± 00,44	21,52 ± 01,22	20,14 ± 00,85	20,73 ± 00,62	14,51 ± 00,17	21,53 ± 01,73	20,82 ± 00,47	20,68 ± 00,66
75	52,27 ± 10,35	29,51 ± 02,11	28,98 ± 00,68	28,95 ± 00,39	29,18 ± 01,04	28,73 ± 00,54	46,69 ± 00,15	36,87 ± 02,08	35,93 ± 00,99	35,45 ± 00,87
100	61,87 ± 05,28	32,03 ± 00,98	32,00 ± 00,26	32,56 ± 00,66	32,20 ± 00,28	32,02 ± 00,17	69,64 ± 08,80	44,25 ± 01.15	46,31 ± 01,79	44,87 ± 00,99
150	52,28 ± 04,48	29,22 ± 01,06	29,18 ± 00,25	29,26 ± 00,51	29,21 ± 00,25	28,05 ± 00,30	59,04 ± 08,10	38,90 ± 01,85	38,62 ± 00,37	39,60 ± 00,79
200	45,49 ± 04,62	24,65 ± 01.11	25,22 ± 00,15	25,05 ± 00,32	25,15 ± 00,18	24,27 ± 00,31	55,51 ± 11,05	33,35 ± 01.18	33,68 ± 00,77	34,37 ± 01,36
250	40,23 ± 05,48	21,31 ± 00,72	22,14 ± 00,16	22,21 ± 00,12	22,17 ± 00,17	21,96 ± 00,34	42,30 ± 05,12	28,56 ± 01,24	29,06 ± 00,68	28,66 ± 00,52
300	35,06 ± 03,89	19,04 ± 00,65	19,30 ± 00,07	19,09 ± 00,12	19,12 ± 00,12	18,73 ± 00,27	35,24 ± 04,75	24,64 ± 00,81	24,74 ± 00,66	24,86 ± 00,55

Tabela B.3: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Feit96, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	04,70 ± 00,74	04,30 ± 00,79	03,60 ± 00,03	03,59 ± 00,03	03,56 ± 00,04	03,57 ± 00,03	05,82 ± 00,06	04,04 ± 00,77	03,57 ± 00,03	03,60 ± 00,02
25	17,72 ± 05,14	12,02 ± 03,50	14,22 ± 00,91	14,24 ± 01,06	14,49 ± 00,93	13,99 ± 01.10	11,06 ± 00,07	14,37 ± 04,37	14,21 ± 00,79	14,06 ± 00,96
50	29,47 ± 12,68	21,44 ± 03,76	21,31 ± 00,39	20,95 ± 00,58	20,90 ± 00,86	20,76 ± 00,66	14,67 ± 00,04	22,11 ± 01,76	21.15 ± 00,57	20,98 ± 00,60
75	45,67 ± 08,01	31,56 ± 03,00	30,07 ± 00,94	29,31 ± 00,42	29,67 ± 00,51	28,90 ± 01,03	47,32 ± 00,58	37,43 ± 02,01	35,76 ± 00,48	35,82 ± 00,49
100	63,93 ± 05,40	32,55 ± 01.11	32,60 ± 00,18	32,60 ± 00,45	32,93 ± 00,33	32,01 ± 00,42	70,87 ± 09,89	46,42 ± 01,92	45,77 ± 00,71	46,07 ± 03,13
150	58,03 ± 09,49	28,64 ± 01.14	29,82 ± 00,37	29,91 ± 00,32	29,71 ± 00,50	28,77 ± 00,30	66,90 ± 11,27	39,72 ± 02,68	39,86 ± 01,20	40,34 ± 00,69
200	46,46 ± 03,54	25,15 ± 00,65	25,58 ± 00,09	25,62 ± 00,25	25,47 ± 00,22	24,60 ± 00,18	52,50 ± 06,96	33,09 ± 00,86	33,18 ± 00,32	34,21 ± 00,95
250	38,47 ± 03,02	21,82 ± 00,82	22,56 ± 00,18	22,86 ± 00,38	22,74 ± 00,18	22,25 ± 00,34	38,71 ± 00,14	28,73 ± 01,06	29,11 ± 00,36	29,18 ± 00,63
300	36,10 ± 03,61	19,83 ± 00,84	19,46 ± 00,23	19,53 ± 00,25	19,61 ± 00,27	19,13 ± 00,22	35,80 ± 04,35	24,96 ± 00,59	25,16 ± 00,42	25,34 ± 00,56

Tabela B.4: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-ASCI, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	03,49 ± 00,24	02,80 ± 00,17	02,66 ± 00,01	02,67 ± 00,01	02,67 ± 00,01	02,67 ± 00,01	05,53 ± 00,25	02,76 ± 00,16	02,67 ± 00,01	02,68 ± 00,01
25	03,62 ± 00,37	02,86 ± 00,11	03,09 ± 00,01	03,09 ± 00,01	03,10 ± 00,01	03,10 ± 00,01	05,48 ± 00,21	02,86 ± 00,10	03,09 ± 00,01	03,11 ± 00,01
50	03,75 ± 00,19	02,84 ± 00,03	02,88 ± 00,00	02,88 ± 00,00	02,88 ± 00,01	02,88 ± 00,00	06,78 ± 00,16	02,92 ± 00,09	02,95 ± 00,00	02,95 ± 00,00
75	03,89 ± 00,14	02,94 ± 00,08	03,06 ± 00,00	03,06 ± 00,01	03,05 ± 00,00	03,05 ± 00,00	06,97 ± 00,07	02,94 ± 00,04	03,02 ± 00,00	03,01 ± 00,00
100	03,73 ± 00,06	02,84 ± 00,04	02,94 ± 00,00	02,94 ± 00,00	02,94 ± 00,00	02,94 ± 00,00	06,61 ± 00,09	02,89 ± 00,05	02,91 ± 00,00	02,91 ± 00,00
150	03,83 ± 00,05	02,92 ± 00,04	03,32 ± 00,01	03,32 ± 00,00	03,32 ± 00,00	03,32 ± 00,00	06,77 ± 00,14	02,94 ± 00,06	02,97 ± 00,00	02,97 ± 00,00
200	03,89 ± 00,06	03,03 ± 00,03	03,34 ± 00,00	03,34 ± 00,00	03,34 ± 00,00	03,33 ± 00,00	06,94 ± 00,13	03,03 ± 00,05	03,07 ± 00,00	03,07 ± 00,00
250	03,93 ± 00,05	03,05 ± 00,06	03,34 ± 00,00	03,34 ± 00,00	03,34 ± 00,00	03,34 ± 00,00	06,92 ± 00,07	03,06 ± 00,03	03,11 ± 00,00	03,11 ± 00,00
300	04,03 ± 00,11	03,09 ± 00,04	03,31 ± 00,00	03,31 ± 00,00	03,31 ± 00,00	03,31 ± 00,00	06,84 ± 00,06	03,14 ± 00,07	03,12 ± 00,00	03,12 ± 00,00

Tabela B.5: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-ASCI, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	03,78 ± 00,51	02,75 ± 00,15	02,68 ± 00,01	02,68 ± 00,01	02,68 ± 00,01	02,67 ± 00,01	04,63 ± 00,08	02,77 ± 00,16	02,67 ± 00,01	02,68 ± 00,01
25	03,93 ± 00,35	02,91 ± 00,14	03,11 ± 00,01	03,11 ± 00,01	03,11 ± 00,01	03,11 ± 00,01	04,05 ± 00,01	02,87 ± 00,10	03,11 ± 00,02	03,11 ± 00,01
50	03,83 ± 00,13	02,85 ± 00,04	02,89 ± 00,00	02,88 ± 00,00	02,89 ± 00,01	02,89 ± 00,01	05,97 ± 00,01	02,93 ± 00,08	02,95 ± 00,00	02,95 ± 00,01
75	03,87 ± 00,15	02,98 ± 00,06	03,06 ± 00,01	03,06 ± 00,00	03,06 ± 00,00	03,06 ± 00,00	06,11 ± 00,01	02,95 ± 00,05	03,03 ± 00,00	03,02 ± 00,00
100	03,82 ± 00,10	02,85 ± 00,05	02,95 ± 00,00	02,95 ± 00,00	02,95 ± 00,00	02,95 ± 00,00	05,92 ± 00,02	02,89 ± 00,05	02,92 ± 00,00	02,92 ± 00,00
150	03,80 ± 00,10	02,92 ± 00,03	03,33 ± 00,00	03,33 ± 00,00	03,33 ± 00,00	03,33 ± 00,00	06,15 ± 00,01	02,95 ± 00,06	02,98 ± 00,00	02,98 ± 00,00
200	03,95 ± 00,11	02,99 ± 00,05	03,35 ± 00,00	03,35 ± 00,01	03,35 ± 00,00	03,35 ± 00,00	06,18 ± 00,05	03,03 ± 00,05	03,08 ± 00,00	03,08 ± 00,00
250	03,99 ± 00,09	03,11 ± 00,07	03,35 ± 00,00	03,35 ± 00,00	03,35 ± 00,00	03,35 ± 00,00	06,04 ± 00,01	03,07 ± 00,02	03,12 ± 00,00	03,12 ± 00,00
300	04,02 ± 00,10	03,10 ± 00,03	03,32 ± 00,00	03,32 ± 00,00	03,32 ± 00,00	03,32 ± 00,00	06,04 ± 00,02	03,15 ± 00,07	03,13 ± 00,00	03,13 ± 00,00

Tabela B.6: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-ASCI, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	03,70 ± 00,23	02,70 ± 00,10	02,75 ± 00,02	02,76 ± 00,02	02,76 ± 00,02	02,75 ± 00,02	04,65 ± 00,03	02,85 ± 00,15	02,74 ± 00,02	02,75 ± 00,01
25	03,82 ± 00,23	02,93 ± 00,14	03,19 ± 00,01	03,18 ± 00,02	03,18 ± 00,01	03,19 ± 00,01	04,14 ± 00,02	02,95 ± 00,10	03,18 ± 00,01	03,18 ± 00,01
50	03,82 ± 00,14	02,96 ± 00,08	02,96 ± 00,01	02,96 ± 00,01	02,96 ± 00,01	02,96 ± 00,01	06,02 ± 00,01	03,01 ± 00,09	03,03 ± 00,01	03,03 ± 00,01
75	03,99 ± 00,12	02,98 ± 00,02	03,13 ± 00,01	03,13 ± 00,00	03,14 ± 00,01	03,13 ± 00,00	06,18 ± 00,01	03,02 ± 00,04	03,10 ± 00,00	03,10 ± 00,01
100	03,85 ± 00,13	02,92 ± 00,06	03,02 ± 00,00	03,02 ± 00,01	03,02 ± 00,00	03,02 ± 00,01	05,99 ± 00,01	02,96 ± 00,05	03,00 ± 00,00	03,00 ± 00,01
150	03,92 ± 00,10	02,96 ± 00,06	03,40 ± 00,00	03,41 ± 00,00	03,41 ± 00,01	03,40 ± 00,00	06,21 ± 00,01	03,02 ± 00,05	03,05 ± 00,01	03,05 ± 00,00
200	03,95 ± 00,05	03,13 ± 00,05	03,42 ± 00,01	03,42 ± 00,00	03,42 ± 00,00	03,42 ± 00,01	06,22 ± 00,01	03,11 ± 00,05	03,15 ± 00,00	03,16 ± 00,01
250	04,06 ± 00,07	03,20 ± 00,09	03,43 ± 00,00	03,43 ± 00,00	03,43 ± 00,00	03,43 ± 00,00	06,09 ± 00,01	03,15 ± 00,03	03,20 ± 00,00	03,20 ± 00,00
300	04,14 ± 00,07	03,17 ± 00,06	03,40 ± 00,00	03,40 ± 00,00	03,40 ± 00,00	03,40 ± 00,00	06,08 ± 00,01	03,23 ± 00,07	03,21 ± 00,00	03,21 ± 00,00

Tabela B.7: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-CTC, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	219,37 ± 36,18	108,46 ± 03,81	111,89 ± 00,63	112,44 ± 00,58	111,70 ± 01,33	85,34 ± 02,30	418,12 ± 92,08	134,96 ± 16,55	140,02 ± 01,20	140,00 ± 01.13
25	132,95 ± 16,64	84,21 ± 06,57	90,46 ± 00,47	90,31 ± 00,85	91,30 ± 00,69	73,05 ± 01.14	362,97 ± 86,36	102,06 ± 09,04	109,90 ± 00,91	109,77 ± 00,66
50	131,24 ± 19,69	69,14 ± 02,44	71,00 ± 00,26	70,70 ± 00,40	71.11 ± 00,38	58,57 ± 01,01	210,31 ± 36,85	94,70 ± 14,52	84,90 ± 00,42	84,36 ± 00,61
75	100,76 ± 27,18	60,83 ± 04,02	62,01 ± 00,30	62,01 ± 00,33	61,88 ± 00,30	54,96 ± 00,20	157,21 ± 19,72	77,42 ± 11,02	69,95 ± 00,45	70,13 ± 00,42
100	87,88 ± 14,00	57,35 ± 03,88	54,44 ± 00,29	54,50 ± 00,24	54,51 ± 00,22	48,22 ± 00,73	142,70 ± 22,75	62,77 ± 02,75	60,82 ± 00,18	60,73 ± 00,30
150	81,31 ± 09,81	50,24 ± 02,65	47,23 ± 00,13	47,26 ± 00,15	47,39 ± 00,13	43,24 ± 00,45	134,08 ± 13,92	61,42 ± 02,43	52,07 ± 00,18	51,89 ± 00,16
200	83,56 ± 05,38	58,17 ± 01,84	55,81 ± 00,14	55,69 ± 00,11	55,75 ± 00,16	47,70 ± 00,81	136,12 ± 14,08	65,35 ± 02,93	59,58 ± 00,23	59,69 ± 00,14
250	91,76 ± 08,50	52,39 ± 02,12	54,17 ± 00,15	54,27 ± 00,19	54,16 ± 00,14	46,46 ± 00,62	170,65 ± 14,24	68,85 ± 04,39	67,52 ± 00,20	67,47 ± 00,21
300	107,00 ± 15,41	55,72 ± 01,87	58,84 ± 00,22	58,73 ± 00,09	58,92 ± 00,05	49,44 ± 00,72	169,57 ± 17,08	68,43 ± 02,91	70,69 ± 00,20	70,56 ± 00,18

Tabela B.8: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-CTC, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	192,06 ± 32,08	109,77 ± 06,08	112,24 ± 00,89	111,78 ± 00,96	112,29 ± 01.14	86,63 ± 01,76	227,23 ± 01.12	134,64 ± 16,91	140,26 ± 00,51	139,54 ± 01,06
25	178,99 ± 80,04	85,95 ± 04,22	90,48 ± 00,57	91,22 ± 00,50	90,67 ± 00,79	72,52 ± 01,46	173,46 ± 01,64	102,49 ± 08,94	110,61 ± 00,83	109,75 ± 00,54
50	170,21 ± 98,51	68,16 ± 04,44	70,96 ± 00,51	71,00 ± 00,56	71.18 ± 00,51	58,38 ± 01,07	123,72 ± 01,99	95,03 ± 14,74	84,55 ± 00,71	84,14 ± 00,28
75	94,43 ± 10,07	63,51 ± 05,96	62,08 ± 00,26	62,14 ± 00,21	62,08 ± 00,44	54,94 ± 00,53	87,96 ± 01.13	77,40 ± 11,01	70,39 ± 00,37	70,16 ± 00,27
100	108,06 ± 21,75	55,39 ± 02,62	54,59 ± 00,31	54,58 ± 00,21	54,65 ± 00,20	48,31 ± 00,77	73,25 ± 00,76	62,94 ± 02,77	61.11 ± 00,34	61,36 ± 00,21
150	76,51 ± 04,03	50,76 ± 03,29	47,40 ± 00,17	47,45 ± 00,17	47,39 ± 00,11	43,74 ± 00,37	77,06 ± 02,06	61,39 ± 02,42	51,99 ± 00,13	52,04 ± 00,20
200	86,22 ± 08,57	56,42 ± 03,47	55,91 ± 00,17	55,82 ± 00,10	55,85 ± 00,16	48,35 ± 00,83	73,48 ± 01,68	65,64 ± 03,06	59,72 ± 00,14	59,73 ± 00,21
250	96,54 ± 12,91	52,57 ± 01,89	54,30 ± 00,12	54,24 ± 00,14	54,34 ± 00,13	45,91 ± 00,72	91,30 ± 01.12	68,98 ± 04,49	67,76 ± 00,19	67,79 ± 00,16
300	101,90 ± 14,70	56,67 ± 02,20	58,93 ± 00,13	58,90 ± 00,12	59,02 ± 00,15	49,90 ± 00,75	87,15 ± 01.18	68,76 ± 03,07	70,67 ± 00,19	70,77 ± 00,18

Tabela B.9: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Jann97-CTC, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	205,00 ± 45,27	112,25 ± 02,19	113,71 ± 00,90	114,22 ± 00,69	114,62 ± 01,00	87,71 ± 02,18	228,49 ± 01.11	137,70 ± 17,39	142,06 ± 01,38	142,38 ± 00,99
25	146,16 ± 18,69	85,31 ± 03,52	92,69 ± 00,62	92,85 ± 00,61	92,02 ± 01.10	71,87 ± 01,46	173,83 ± 00,59	104,39 ± 09,52	111,66 ± 01,37	110,92 ± 00,79
50	137,30 ± 37,17	69,10 ± 03,89	72,47 ± 00,59	72,58 ± 00,50	72,29 ± 00,72	60,26 ± 00,74	125,08 ± 01,22	96,46 ± 14,89	85,95 ± 01.10	86,28 ± 00,87
75	96,89 ± 25,41	63,33 ± 03,92	63,61 ± 00,28	63,10 ± 00,29	62,97 ± 00,42	55,68 ± 00,73	88,68 ± 00,70	78,74 ± 11,20	71,53 ± 00,20	71,67 ± 00,45
100	93,28 ± 16,76	59,00 ± 05,47	55,80 ± 00,27	55,58 ± 00,19	55,63 ± 00,43	48,40 ± 00,77	73,77 ± 00,27	64,15 ± 02,84	61,87 ± 00,39	61,92 ± 00,38
150	82,17 ± 07,89	50,61 ± 04,15	48,40 ± 00,20	48,37 ± 00,16	48,35 ± 00,23	43,87 ± 00,61	77,05 ± 01,70	62,34 ± 02,49	52,97 ± 00,25	52,98 ± 00,26
200	92,73 ± 16,85	55,22 ± 02,29	56,75 ± 00,14	56,93 ± 00,11	56,96 ± 00,19	48,21 ± 00,87	74,94 ± 00,33	66,53 ± 02,78	60,91 ± 00,27	60,90 ± 00,21
250	93,89 ± 10,45	54,13 ± 02,12	55,39 ± 00,20	55,38 ± 00,14	55,30 ± 00,18	47,32 ± 00,79	92,16 ± 00,78	69,78 ± 04,61	68,87 ± 00,23	68,80 ± 00,24
300	101,44 ± 06,36	56,63 ± 01.19	60,30 ± 00,13	60,13 ± 00,13	60,13 ± 00,09	51,08 ± 00,43	87,35 ± 01,01	70,03 ± 02,92	71,89 ± 00,21	71,85 ± 00,22

Tabela B.10: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Lub99, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	12,68 ± 04,29	19,20 ± 06,53	12,60 ± 00,12	12,50 ± 00,21	12,54 ± 00,16	12,70 ± 00,16	20,91 ± 05,45	14,03 ± 06,63	12,49 ± 00,15	12,57 ± 00,15
25	11,42 ± 02,22	08,75 ± 02,21	07,40 ± 00,07	07,35 ± 00,03	07,33 ± 00,07	07,38 ± 00,04	17,01 ± 03,78	08,73 ± 02,10	08,32 ± 00,09	08,34 ± 00,05
50	65,19 ± 13,74	33,23 ± 02,18	32,43 ± 00,28	32,52 ± 00,21	32,64 ± 00,15	30,40 ± 00,97	134,99 ± 16,89	45,76 ± 01,99	44,86 ± 00,24	44,99 ± 00,28
75	56,67 ± 13,54	22,53 ± 00,97	23,22 ± 00,10	23,16 ± 00,16	23,11 ± 00,15	21,92 ± 00,76	88,27 ± 11,01	31,74 ± 01,08	31,73 ± 00,14	31,82 ± 00,13
100	29,62 ± 05,59	17,17 ± 00,76	17,27 ± 00,08	17,31 ± 00,05	17,30 ± 00,08	16,53 ± 00,51	64,75 ± 07,68	23,17 ± 01,03	23,15 ± 00,11	23,01 ± 00,12
150	18,64 ± 02,06	11,82 ± 00,45	11,85 ± 00,08	11,88 ± 00,07	11,85 ± 00,05	11,42 ± 00,29	37,79 ± 04,38	15,87 ± 00,99	15,38 ± 00,08	15,40 ± 00,08
200	18,09 ± 04,51	09,73 ± 00,34	09,72 ± 00,04	09,74 ± 00,04	09,72 ± 00,05	09,53 ± 00,24	30,19 ± 02,46	12,53 ± 00,47	12,56 ± 00,07	12,53 ± 00,04
250	17,36 ± 02,92	08,50 ± 00,34	08,57 ± 00,02	08,58 ± 00,03	08,53 ± 00,05	08,44 ± 00,18	28,34 ± 03,44	10,91 ± 00,39	10,92 ± 00,05	10,92 ± 00,04
300	14,81 ± 05,47	07,31 ± 00,16	07,55 ± 00,05	07,53 ± 00,04	07,55 ± 00,03	07,23 ± 00,15	25,21 ± 03,21	09,61 ± 00,48	09,49 ± 00,02	09,48 ± 00,02

Tabela B.11: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Lub99, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	14,26 ± 06,37	14,44 ± 06,64	12,51 ± 00,17	12,53 ± 00,12	12,70 ± 00,14	12,71 ± 00,11	07,48 ± 00,05	13,91 ± 06,63	12,61 ± 00,16	12,56 ± 00,19
25	12,06 ± 06,64	08,09 ± 02,17	07,35 ± 00,05	07,44 ± 00,05	07,40 ± 00,09	07,38 ± 00,05	07,20 ± 00,40	08,70 ± 02,02	08,42 ± 00,04	08,41 ± 00,04
50	64,04 ± 17,23	32,27 ± 01,34	32,37 ± 00,24	32,67 ± 00,12	32,50 ± 00,15	31,63 ± 00,94	70,46 ± 00,17	45,86 ± 02,38	45,04 ± 00,34	44,99 ± 00,32
75	45,87 ± 13,57	23,01 ± 01,23	23,17 ± 00,19	23,29 ± 00,14	23,27 ± 00,15	21,61 ± 00,58	50,05 ± 00,10	31,77 ± 01.15	31,84 ± 00,22	31,75 ± 00,20
100	31,85 ± 08,33	16,84 ± 00,54	17,33 ± 00,09	17,34 ± 00,13	17,35 ± 00,06	16,52 ± 00,50	36,54 ± 00,08	23,10 ± 00,99	23,08 ± 00,15	23,03 ± 00,11
150	24,05 ± 05,32	11,51 ± 00,41	11,94 ± 00,06	11,94 ± 00,03	11,95 ± 00,05	11,77 ± 00,27	24,63 ± 00,24	15,93 ± 00,93	15,41 ± 00,08	15,48 ± 00,11
200	18,08 ± 03,54	09,57 ± 00,29	09,79 ± 00,03	09,76 ± 00,03	09,74 ± 00,06	09,37 ± 00,22	19,83 ± 00,06	12,50 ± 00,43	12,56 ± 00,06	12,53 ± 00,05
250	15,65 ± 01,86	08,49 ± 00,30	08,60 ± 00,03	08,62 ± 00,02	08,58 ± 00,04	08,23 ± 00,19	17,23 ± 00,13	10,95 ± 00,40	10,94 ± 00,05	10,91 ± 00,03
300	12,02 ± 01.13	07,38 ± 00,26	07,57 ± 00,03	07,56 ± 00,03	07,57 ± 00,02	07,10 ± 00,03	15,06 ± 00,03	09,56 ± 00,46	09,50 ± 00,03	09,49 ± 00,02

Tabela B.12: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 32 nós, Lub99, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	18,80 ± 07,07	14,53 ± 05,71	12,83 ± 00,14	12,87 ± 00,19	12,87 ± 00,18	12,69 ± 00,24	07,67 ± 00,05	14,06 ± 06,38	12,90 ± 00,09	12,71 ± 00,11
25	11,30 ± 02,44	06,11 ± 00,44	07,48 ± 00,12	07,52 ± 00,09	07,51 ± 00,07	07,55 ± 00,06	07,12 ± 00,04	08,74 ± 01,94	08,47 ± 00,07	08,55 ± 00,07
50	61,41 ± 10,28	31,74 ± 01,07	33,13 ± 00,21	33,14 ± 00,31	33,32 ± 00,24	30,88 ± 01,07	70,80 ± 00,27	45,91 ± 02,22	45,56 ± 00,41	45,43 ± 00,34
75	43,49 ± 09,81	24,39 ± 01,65	23,58 ± 00,14	23,47 ± 00,19	23,63 ± 00,13	22,81 ± 00,86	50,16 ± 00,12	32,05 ± 01,04	32,24 ± 00,23	32,21 ± 00,27
100	30,13 ± 05,85	16,62 ± 00,45	17,68 ± 00,08	17,60 ± 00,09	17,59 ± 00,13	17,09 ± 00,52	36,73 ± 00,14	23,46 ± 01,03	23,30 ± 00,17	23,42 ± 00,23
150	20,69 ± 02,36	11,56 ± 00,17	12,07 ± 00,04	12,11 ± 00,10	12,12 ± 00,09	11,67 ± 00,34	24,73 ± 00,10	16,09 ± 00,96	15,66 ± 00,11	15,58 ± 00,11
200	16,65 ± 01,99	09,99 ± 00,49	09,95 ± 00,05	09,93 ± 00,05	09,89 ± 00,06	09,65 ± 00,20	19,89 ± 00,04	12,69 ± 00,48	12,75 ± 00,07	12,73 ± 00,09
250	14,60 ± 03,15	08,62 ± 00,30	08,71 ± 00,07	08,77 ± 00,03	08,72 ± 00,04	08,37 ± 00,19	17,32 ± 00,05	11.13 ± 00,35	11,08 ± 00,05	11,07 ± 00,06
300	15,28 ± 03,14	07,86 ± 00,30	07,72 ± 00,03	07,72 ± 00,03	07,69 ± 00,04	07,44 ± 00,17	15,14 ± 00,03	09,75 ± 00,44	09,69 ± 00,08	09,67 ± 00,04

Tabela B.13: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Feit96, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	05,86 ± 01,23	09,13 ± 06,01	03,49 ± 00,02	03,47 ± 00,02	03,48 ± 00,02	03,48 ± 00,02	08,38 ± 01,06	07,44 ± 05,86	03,48 ± 00,02	03,48 ± 00,02
25	16,58 ± 04,09	27,98 ± 09,33	13,10 ± 01.11	13,43 ± 00,97	13,77 ± 00,92	13,91 ± 00,63	21,57 ± 04,87	18,38 ± 08,32	12,49 ± 01,21	13,85 ± 00,57
50	59,96 ± 43,30	62,97 ± 24,20	20,16 ± 00,63	20,81 ± 01,58	20,27 ± 00,72	19,72 ± 00,65	37,49 ± 08,46	50,90 ± 40,69	20,99 ± 01,56	20,55 ± 00,56
75	83,34 ± 29,97	69,13 ± 19,85	29,35 ± 01,00	29,07 ± 00,37	29,05 ± 00,34	27,09 ± 00,87	86,88 ± 09,83	59,17 ± 03,48	28,66 ± 00,52	29,04 ± 00,51
100	131.13 ± 37,41	66,23 ± 09,84	32,48 ± 00,26	32,09 ± 00,24	32,52 ± 00,54	31.12 ± 00,70	122,83 ± 06,89	68,69 ± 08,71	32,30 ± 00,53	32,68 ± 00,56
150	75,81 ± 14,72	56,12 ± 03,93	29,34 ± 00,24	29,72 ± 00,40	29,33 ± 00,42	27,80 ± 00,56	106,60 ± 08,03	61,22 ± 10,44	29,19 ± 00,34	29,86 ± 00,52
200	69,17 ± 10,24	54,04 ± 13,24	25,06 ± 00,15	25,40 ± 00,48	25,55 ± 00,40	23,92 ± 00,43	87,99 ± 05,99	50,25 ± 07,77	25,23 ± 00,62	24,95 ± 00,37
250	57,96 ± 11,61	34,97 ± 03,32	22,41 ± 00,20	22,51 ± 00,44	22,57 ± 00,38	21,53 ± 00,56	74,18 ± 04,10	51,65 ± 07,70	22,61 ± 00,44	22,40 ± 00,20
300	47,22 ± 07,16	36,19 ± 03,01	19,36 ± 00,38	19,28 ± 00,19	19,32 ± 00,19	18,65 ± 00,39	63,81 ± 04,39	39,08 ± 03,08	19,27 ± 00,19	19,35 ± 00,39

Tabela B.14: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Feit96, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	07,75 ± 06,40	04,95 ± 01,45	03,50 ± 00,03	03,50 ± 00,02	03,48 ± 00,02	03,50 ± 00,01	05,81 ± 00,04	07,42 ± 05,76	03,48 ± 00,02	03,49 ± 00,03
25	36,15 ± 35,27	57,16 ± 51,06	13,83 ± 00,78	14,15 ± 00,13	14,09 ± 00,17	13,31 ± 01,02	11,60 ± 01.13	18,55 ± 08,43	13,22 ± 01,06	14,18 ± 00,12
50	69,74 ± 34,93	61,99 ± 33,06	20,18 ± 00,58	20,39 ± 00,65	20,19 ± 00,64	20,36 ± 00,64	14,58 ± 00,26	50,94 ± 41,27	20,74 ± 00,24	20,57 ± 00,51
75	94,02 ± 26,90	69,38 ± 15,00	29,55 ± 01.16	28,58 ± 00,84	29,02 ± 00,64	26,76 ± 01.10	47,76 ± 01,28	59,01 ± 03,91	29,18 ± 00,27	28,80 ± 00,36
100	115,77 ± 28,96	58,23 ± 09,31	33,07 ± 01,08	32,60 ± 00,34	32,34 ± 00,25	31,28 ± 00,85	69,19 ± 09,48	66,30 ± 08,25	32,70 ± 00,36	32,39 ± 00,32
150	76,96 ± 17,82	60,31 ± 08,95	29,59 ± 00,35	29,49 ± 00,45	29,76 ± 00,53	28,22 ± 00,54	56,52 ± 04,51	60,48 ± 07,99	29,49 ± 00,25	29,69 ± 00,74
200	73,42 ± 17,48	50,55 ± 06,38	25,71 ± 00,53	25,25 ± 00,16	25,32 ± 00,22	24,18 ± 00,54	48,93 ± 05,57	51,37 ± 08,45	25,28 ± 00,31	25,35 ± 00,33
250	61,86 ± 09,80	40,40 ± 03,41	22,46 ± 00,21	22,48 ± 00,18	22,32 ± 00,28	21.11 ± 00,53	42,13 ± 05,25	51,00 ± 06,97	22,47 ± 00,18	22,50 ± 00,33
300	49,86 ± 05,46	35,02 ± 03,42	19,40 ± 00,26	19,48 ± 00,21	19,23 ± 00,20	18,19 ± 00,34	32,91 ± 00,31	38,80 ± 04,87	19,35 ± 00,14	19,32 ± 00,18

Tabela B.15: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Feit96, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	09,34 ± 06,53	05,21 ± 01,53	03,58 ± 00,03	03,59 ± 00,02	03,59 ± 00,03	03,57 ± 00,03	05,86 ± 00,05	07,65 ± 05,97	03,58 ± 00,03	03,59 ± 00,03
25	48,22 ± 42,67	45,78 ± 46,72	14,11 ± 00,59	14,35 ± 00,16	13,63 ± 01,04	13,61 ± 00,82	11,06 ± 00,05	19,08 ± 08,69	13,09 ± 01,20	13,91 ± 00,84
50	59,94 ± 26,95	44,94 ± 25,04	20,56 ± 00,71	20,70 ± 00,72	20,56 ± 00,79	20,50 ± 00,70	14,73 ± 00,04	52,07 ± 41,75	20,80 ± 00,52	20,13 ± 00,85
75	100,65 ± 50,07	71,32 ± 19,52	29,74 ± 00,48	29,71 ± 00,89	29,31 ± 01,09	26,92 ± 01,02	47,21 ± 00,48	61,21 ± 03,95	29,41 ± 00,43	29,28 ± 00,44
100	83,00 ± 11,60	55,79 ± 07,45	33,39 ± 00,36	33,68 ± 01,03	33,02 ± 00,31	32,30 ± 00,42	74,91 ± 12,46	67,83 ± 08,51	33,08 ± 00,31	33,36 ± 00,52
150	100,80 ± 33,06	55,63 ± 07,61	30,09 ± 00,45	30,23 ± 00,35	30,11 ± 00,29	27,99 ± 00,54	58,03 ± 05,36	62,38 ± 08,47	30,46 ± 00,28	29,91 ± 00,50
200	66,90 ± 10,97	47,13 ± 08,51	25,99 ± 00,21	25,85 ± 00,39	26,00 ± 00,20	24,75 ± 00,46	46,87 ± 00,63	51,36 ± 08,27	26,05 ± 00,44	26,21 ± 00,29
250	64,95 ± 10,74	47,99 ± 08,46	23,20 ± 00,41	22,92 ± 00,21	22,96 ± 00,25	21,61 ± 00,48	40,37 ± 03,86	52,79 ± 08,15	23,10 ± 00,37	23,00 ± 00,20
300	55,49 ± 08,72	39,52 ± 05,05	19,72 ± 00,25	19,83 ± 00,29	19,78 ± 00,20	19,00 ± 00,42	35,30 ± 03,73	40,18 ± 04,42	19,79 ± 00,37	19,86 ± 00,17

Tabela B.16: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-ASCI, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	04,18 ± 00,78	05,04 ± 01,57	02,66 ± 00,01	02,65 ± 00,01	02,66 ± 00,01	02,65 ± 00,01	05,80 ± 00,39	04,21 ± 00,74	02,66 ± 00,01	02,65 ± 00,01
25	04,41 ± 00,43	04,62 ± 00,83	03,10 ± 00,01	03,11 ± 00,01	03,10 ± 00,01	03,09 ± 00,01	05,59 ± 00,57	04,62 ± 00,53	03,09 ± 00,01	03,09 ± 00,01
50	04,17 ± 00,30	04,31 ± 01.19	02,58 ± 00,01	02,58 ± 00,00	02,58 ± 00,00	02,58 ± 00,00	06,86 ± 00,19	03,84 ± 00,23	02,58 ± 00,01	02,58 ± 00,00
75	04,71 ± 00,67	03,78 ± 00,52	02,77 ± 00,00	02,77 ± 00,00	02,77 ± 00,00	02,77 ± 00,00	06,90 ± 00,08	03,93 ± 00,28	02,65 ± 00,00	02,65 ± 00,00
100	04,22 ± 00,57	03,51 ± 00,22	02,68 ± 00,00	02,68 ± 00,00	02,68 ± 00,00	02,68 ± 00,00	06,67 ± 00,12	03,60 ± 00,15	02,59 ± 00,00	02,59 ± 00,00
150	04,12 ± 00,35	03,49 ± 00,09	03,03 ± 00,00	03,03 ± 00,01	03,03 ± 00,00	03,03 ± 00,00	06,84 ± 00,12	03,57 ± 00,12	02,62 ± 00,00	02,62 ± 00,00
200	04,50 ± 00,31	03,92 ± 00,41	03,04 ± 00,00	03,04 ± 00,00	03,04 ± 00,00	03,04 ± 00,00	06,89 ± 00,07	03,86 ± 00,25	02,71 ± 00,00	02,71 ± 00,00
250	04,46 ± 00,38	04,48 ± 00,37	03,06 ± 00,00	03,06 ± 00,00	03,05 ± 00,00	03,06 ± 00,00	06,85 ± 00,08	04,72 ± 00,77	02,77 ± 00,00	02,77 ± 00,00
300	04,95 ± 00,48	04,53 ± 00,42	03,02 ± 00,00	03,02 ± 00,00	03,02 ± 00,00	03,02 ± 00,00	06,89 ± 00,11	04,33 ± 00,25	02,77 ± 00,00	02,77 ± 00,00

Tabela B.17: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-ASCI, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	03,71 ± 00,34	04,55 ± 01.10	02,67 ± 00,01	02,67 ± 00,01	02,66 ± 00,01	02,66 ± 00,01	04,55 ± 00,02	04,20 ± 00,72	02,67 ± 00,01	02,67 ± 00,01
25	05,56 ± 01,63	04,17 ± 00,71	03,12 ± 00,01	03,10 ± 00,01	03,11 ± 00,01	03,11 ± 00,01	04,10 ± 00,02	04,68 ± 00,55	03,11 ± 00,01	03,11 ± 00,01
50	04,15 ± 00,23	03,55 ± 00,31	02,59 ± 00,01	02,59 ± 00,00	02,59 ± 00,00	02,59 ± 00,00	05,98 ± 00,02	03,86 ± 00,25	02,59 ± 00,00	02,59 ± 00,01
75	05,09 ± 00,96	04,11 ± 00,76	02,78 ± 00,00	02,78 ± 00,00	02,78 ± 00,01	02,78 ± 00,00	06,12 ± 00,02	03,94 ± 00,28	02,66 ± 00,00	02,66 ± 00,00
100	04,18 ± 00,30	04,01 ± 00,51	02,69 ± 00,00	02,69 ± 00,00	02,69 ± 00,00	02,69 ± 00,00	05,93 ± 00,01	03,62 ± 00,16	02,60 ± 00,00	02,60 ± 00,00
150	04,14 ± 00,25	03,65 ± 00,23	03,04 ± 00,00	03,04 ± 00,00	03,04 ± 00,00	03,04 ± 00,00	06,15 ± 00,01	03,58 ± 00,12	02,62 ± 00,00	02,62 ± 00,00
200	04,44 ± 00,32	03,85 ± 00,19	03,05 ± 00,00	03,05 ± 00,00	03,05 ± 00,00	03,05 ± 00,00	06,16 ± 00,01	03,88 ± 00,25	02,72 ± 00,00	02,72 ± 00,00
250	04,92 ± 00,39	03,98 ± 00,24	03,07 ± 00,00	03,07 ± 00,00	03,06 ± 00,00	03,06 ± 00,00	06,04 ± 00,01	04,74 ± 00,82	02,78 ± 00,00	02,78 ± 00,00
300	05,03 ± 00,68	04,35 ± 00,76	03,02 ± 00,00	03,02 ± 00,00	03,03 ± 00,00	03,03 ± 00,00	06,03 ± 00,01	04,34 ± 00,26	02,78 ± 00,00	02,78 ± 00,00

Tabela B.18: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-ASCI, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	06,49 ± 02,33	04,21 ± 00,97	02,74 ± 00,01	02,74 ± 00,02	02,74 ± 00,02	02,74 ± 00,02	04,65 ± 00,02	04,34 ± 00,76	02,73 ± 00,02	02,75 ± 00,01
25	05,68 ± 01,22	06,35 ± 03,22	03,18 ± 00,01	03,19 ± 00,01	03,18 ± 00,02	03,20 ± 00,02	04,15 ± 00,02	04,74 ± 00,50	03,18 ± 00,02	03,18 ± 00,01
50	04,20 ± 00,36	03,95 ± 00,36	02,66 ± 00,01	02,67 ± 00,01	02,67 ± 00,01	02,66 ± 00,01	06,04 ± 00,01	03,96 ± 00,25	02,66 ± 00,01	02,67 ± 00,01
75	04,33 ± 00,33	04,11 ± 00,39	02,86 ± 00,01	02,86 ± 00,00	02,86 ± 00,01	02,86 ± 00,01	06,16 ± 00,01	04,05 ± 00,29	02,74 ± 00,00	02,74 ± 00,01
100	04,43 ± 00,42	03,90 ± 00,47	02,77 ± 00,00	02,77 ± 00,01	02,77 ± 00,01	02,78 ± 00,00	05,98 ± 00,01	03,72 ± 00,16	02,68 ± 00,00	02,68 ± 00,01
150	04,25 ± 00,28	03,86 ± 00,23	03,12 ± 00,01	03,12 ± 00,01	03,12 ± 00,00	03,12 ± 00,00	06,20 ± 00,01	03,68 ± 00,12	02,70 ± 00,00	02,70 ± 00,00
200	04,90 ± 00,55	03,96 ± 00,20	03,13 ± 00,00	03,13 ± 00,00	03,13 ± 00,01	03,12 ± 00,00	06,22 ± 00,01	03,98 ± 00,25	02,80 ± 00,00	02,80 ± 00,00
250	04,72 ± 00,46	04,29 ± 00,30	03,14 ± 00,01	03,14 ± 00,00	03,14 ± 00,00	03,14 ± 00,00	06,09 ± 00,01	04,84 ± 00,79	02,86 ± 00,00	02,86 ± 00,00
300	04,91 ± 00,28	04,44 ± 00,24	03,10 ± 00,00	03,10 ± 00,00	03,10 ± 00,00	03,10 ± 00,00	06,08 ± 00,01	04,45 ± 00,26	02,86 ± 00,00	02,86 ± 00,00

Tabela B.19: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-CTC, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	321,39 ± 132,11	220,66 ± 112,31	116,43 ± 00,93	115,47 ± 00,85	116,47 ± 00,88	89,97 ± 02,03	458,83 ± 92,26	292,43 ± 110,65	115,14 ± 00,79	116,00 ± 00,69
25	412,80 ± 437,41	177,44 ± 73,85	93,69 ± 00,52	93,77 ± 00,71	94,44 ± 00,72	76,51 ± 00,44	265,76 ± 37,13	209,54 ± 66,26	93,83 ± 00,88	93,36 ± 00,54
50	211,25 ± 76,47	137,64 ± 50,14	72,99 ± 00,29	73,38 ± 00,38	73,32 ± 00,46	59,83 ± 01,74	248,63 ± 53,50	200,89 ± 80,52	72,93 ± 00,40	73,13 ± 00,40
75	216,17 ± 110,52	147,23 ± 34,35	63,06 ± 00,33	63,47 ± 00,24	62,98 ± 00,44	55,63 ± 01,02	163,45 ± 29,54	176,83 ± 53,84	63,22 ± 00,42	63,35 ± 00,24
100	134,78 ± 43,89	119,49 ± 28,94	55,63 ± 00,26	55,45 ± 00,26	55,29 ± 00,17	49,82 ± 00,51	140,99 ± 13,29	162,12 ± 89,28	55,46 ± 00,22	55,32 ± 00,22
150	151,44 ± 36,56	111,31 ± 16,90	48,13 ± 00,16	48,09 ± 00,12	47,95 ± 00,19	43,94 ± 00,44	137,14 ± 18,16	142,87 ± 53,07	47,98 ± 00,19	47,91 ± 00,15
200	278,90 ± 138,94	118,09 ± 21,24	56,32 ± 00,15	56,35 ± 00,14	56,27 ± 00,19	48,31 ± 01,09	144,85 ± 12,79	153,57 ± 40,12	56,30 ± 00,12	56,28 ± 00,22
250	173,86 ± 31,95	112,99 ± 17,06	55,93 ± 00,19	55,93 ± 00,20	56,01 ± 00,17	46,03 ± 00,55	164,18 ± 11,98	149,05 ± 45,55	55,93 ± 00,11	55,86 ± 00,11
300	187,30 ± 37,98	117,28 ± 06,58	60,40 ± 00,12	60,54 ± 00,21	60,42 ± 00,16	49,85 ± 00,61	165,05 ± 15,56	161,63 ± 35,09	60,40 ± 00,10	60,45 ± 00,18

Tabela B.20: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-CTC, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	231,95 ± 54,25	219,66 ± 88,72	116,72 ± 00,99	116,58 ± 00,93	116,78 ± 01.13	88,14 ± 02,86	227,81 ± 00,52	295,89 ± 113,18	116,77 ± 00,94	115,92 ± 01,03
25	554,46 ± 804,12	204,93 ± 56,93	93,93 ± 00,78	94,15 ± 00,65	93,71 ± 00,48	76,00 ± 00,90	173,52 ± 00,51	210,93 ± 69,01	93,55 ± 00,57	93,56 ± 00,35
50	279,28 ± 306,34	181,90 ± 65,69	73,61 ± 00,44	72,97 ± 00,49	73,35 ± 00,37	59,52 ± 01,30	124,49 ± 01,52	201,91 ± 81.14	73,20 ± 00,53	73,36 ± 00,52
75	251,85 ± 147,96	112,88 ± 26,91	63,25 ± 00,35	63,29 ± 00,28	63,31 ± 00,20	55,82 ± 00,62	88,68 ± 00,38	175,69 ± 53,40	63,30 ± 00,34	63,31 ± 00,28
100	237,03 ± 93,72	120,88 ± 31,20	55,53 ± 00,19	55,60 ± 00,25	55,42 ± 00,29	49,39 ± 00,73	73,61 ± 01,90	162,05 ± 87,58	55,52 ± 00,30	55,47 ± 00,14
150	124,74 ± 23,04	123,57 ± 18,44	48,24 ± 00,19	48,21 ± 00,10	48,20 ± 00,11	44,49 ± 00,38	78,07 ± 02,08	142,39 ± 51,42	48,02 ± 00,12	48,06 ± 00,16
200	155,93 ± 31,77	132,53 ± 25,17	56,45 ± 00,15	56,54 ± 00,16	56,58 ± 00,20	47,80 ± 00,73	74,98 ± 00,88	153,33 ± 38,97	56,46 ± 00,19	56,43 ± 00,15
250	151,67 ± 28,40	106,55 ± 11,34	56,03 ± 00,13	56,14 ± 00,15	56,20 ± 00,12	46,89 ± 01,01	91,33 ± 00,86	149,44 ± 46,20	56,03 ± 00,15	56,01 ± 00,13
300	184,32 ± 35,91	125,17 ± 11.10	60,48 ± 00,10	60,59 ± 00,12	60,50 ± 00,11	50,08 ± 00,70	89,31 ± 03,16	162,54 ± 35,97	60,46 ± 00,18	60,55 ± 00,15

Tabela B.21: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Jann97-CTC, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	306,15 ± 111.16	170,64 ± 36,77	118,27 ± 01,57	120,08 ± 01.16	118,64 ± 01,77	93,67 ± 01.14	227,84 ± 01,46	302,93 ± 117,60	118,35 ± 01,00	120,11 ± 01,66
25	191.12 ± 81,05	168,36 ± 44,08	95,74 ± 01,00	95,88 ± 00,76	95,48 ± 00,61	78,36 ± 00,78	172,84 ± 01,76	213,60 ± 68,69	95,77 ± 00,87	95,84 ± 00,60
50	216,02 ± 105,03	136,52 ± 38,30	74,79 ± 00,71	75,23 ± 00,65	74,83 ± 00,80	62,33 ± 00,86	124,81 ± 01,92	204,10 ± 79,01	74,87 ± 00,46	74,07 ± 00,68
75	171,42 ± 90,51	135,77 ± 37,29	64,62 ± 00,58	64,39 ± 00,55	64,82 ± 00,44	57,07 ± 00,69	88,60 ± 00,84	182,69 ± 55,57	64,09 ± 00,45	64,61 ± 00,36
100	141,95 ± 29,55	97,01 ± 22,30	56,38 ± 00,23	56,30 ± 00,38	56,48 ± 00,40	49,83 ± 00,96	72,88 ± 00,92	163,32 ± 87,11	56,60 ± 00,26	56,73 ± 00,28
150	177,78 ± 52,41	112,73 ± 18,29	49,08 ± 00,28	48,97 ± 00,21	49,20 ± 00,17	45,15 ± 00,42	77,27 ± 01,65	146,58 ± 55,10	49,32 ± 00,19	49,16 ± 00,23
200	164,98 ± 45,90	129,57 ± 16,79	57,36 ± 00,11	57,61 ± 00,23	57,58 ± 00,22	49,35 ± 01,20	75,03 ± 00,24	156,80 ± 41,26	57,66 ± 00,21	57,66 ± 00,13
250	175,26 ± 37,18	119,38 ± 16,88	57,24 ± 00,11	57,21 ± 00,19	57,15 ± 00,15	47,98 ± 00,64	91,78 ± 01,22	151,62 ± 46,45	57,27 ± 00,13	57,21 ± 00,19
300	248,95 ± 64,61	132,39 ± 18,35	61,85 ± 00,19	61,77 ± 00,17	61,79 ± 00,22	50,87 ± 00,63	88,03 ± 00,91	164,84 ± 35,99	61,74 ± 00,20	61,81 ± 00,22

Tabela B.22: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Lub99, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	13,99 ± 03,07	69,77 ± 82,00	12,56 ± 00,14	12,49 ± 00,09	12,64 ± 00,16	12,60 ± 00,16	15,79 ± 01,66	50,83 ± 53,46	12,54 ± 00,15	12,55 ± 00,20
25	24,72 ± 24,90	12,47 ± 03,14	07,62 ± 00,05	07,69 ± 00,08	07,71 ± 00,08	07,65 ± 00,05	16,04 ± 03,98	13,55 ± 05,37	07,68 ± 00,05	07,67 ± 00,07
50	103,22 ± 47,10	50,96 ± 13,02	33,29 ± 00,21	33,32 ± 00,20	33,27 ± 00,24	27,81 ± 01,85	131,83 ± 23,13	54,26 ± 15,21	33,33 ± 00,29	33,24 ± 00,16
75	46,15 ± 11.15	42,33 ± 09,97	23,75 ± 00,10	23,59 ± 00,15	23,73 ± 00,14	20,70 ± 01,69	82,75 ± 11,91	37,77 ± 10,18	23,67 ± 00,18	23,53 ± 00,13
100	47,12 ± 19,95	26,23 ± 05,42	17,53 ± 00,08	17,56 ± 00,09	17,55 ± 00,13	14,51 ± 00,53	65,81 ± 05,75	35,37 ± 05,72	17,34 ± 00,06	17,51 ± 00,05
150	30,83 ± 15,43	20,83 ± 05,68	11,88 ± 00,06	11,92 ± 00,08	11,91 ± 00,06	10,27 ± 00,49	41,48 ± 04,06	20,39 ± 04,57	11,88 ± 00,06	11,87 ± 00,05
200	21,80 ± 08,27	19,88 ± 05,53	09,79 ± 00,03	09,76 ± 00,04	09,78 ± 00,04	08,37 ± 00,25	34,99 ± 04,63	17,00 ± 03,67	09,73 ± 00,05	09,77 ± 00,03
250	19,79 ± 06,26	15,04 ± 02,45	08,60 ± 00,04	08,62 ± 00,04	08,60 ± 00,03	07,51 ± 00,17	28,82 ± 03,69	16,79 ± 04,35	08,57 ± 00,04	08,56 ± 00,05
300	15,52 ± 05,31	11,04 ± 02,59	07,55 ± 00,03	07,55 ± 00,02	07,57 ± 00,04	06,71 ± 00,28	23,28 ± 02,40	12,15 ± 03,77	07,51 ± 00,02	07,54 ± 00,02

Tabela B.23: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Lub99, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	59,50 ± 50,95	50,79 ± 74,99	12,42 ± 00,15	12,57 ± 00,13	12,55 ± 00,20	12,60 ± 00,19	07,55 ± 00,05	49,15 ± 51,80	12,48 ± 00,13	12,61 ± 00,19
25	26,17 ± 19,01	12,30 ± 02,86	07,67 ± 00,06	07,76 ± 00,07	07,67 ± 00,04	07,68 ± 00,03	07,03 ± 00,04	13,64 ± 05,47	07,68 ± 00,05	07,69 ± 00,05
50	82,68 ± 30,51	51,57 ± 12,20	33,36 ± 00,17	33,42 ± 00,22	33,37 ± 00,20	28,87 ± 01,78	70,77 ± 00,33	54,47 ± 15,19	33,33 ± 00,18	33,37 ± 00,11
75	59,27 ± 24,21	56,70 ± 14,80	23,77 ± 00,12	23,84 ± 00,11	23,72 ± 00,11	19,16 ± 00,60	49,92 ± 00,19	37,58 ± 10,15	23,84 ± 00,14	23,60 ± 00,14
100	50,12 ± 26,27	30,88 ± 12,36	17,60 ± 00,09	17,60 ± 00,11	17,54 ± 00,08	14,29 ± 00,47	37,33 ± 01,98	35,43 ± 05,59	17,49 ± 00,11	17,46 ± 00,07
150	25,42 ± 12,33	16,32 ± 03,15	11,92 ± 00,05	11,95 ± 00,10	11,98 ± 00,08	10,51 ± 00,56	24,65 ± 00,16	20,06 ± 04,33	11,95 ± 00,07	11,89 ± 00,05
200	29,55 ± 11,55	18,11 ± 02,97	09,84 ± 00,05	09,81 ± 00,06	09,82 ± 00,04	08,98 ± 00,53	19,85 ± 00,08	17,04 ± 03,84	09,77 ± 00,03	09,80 ± 00,04
250	17,87 ± 03,36	14,69 ± 02,22	08,63 ± 00,05	08,63 ± 00,03	08,60 ± 00,04	07,65 ± 00,28	17,30 ± 00,10	16,79 ± 04,27	08,60 ± 00,03	08,59 ± 00,03
300	18,22 ± 06,93	12,38 ± 02,34	07,57 ± 00,04	07,59 ± 00,03	07,59 ± 00,02	06,85 ± 00,28	15,09 ± 00,17	12,06 ± 03,54	07,57 ± 00,02	07,56 ± 00,03

Tabela B.24: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Cluster 128 nós, Lub99, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	18,89 ± 09,12	19,76 ± 15,40	12,72 ± 00,15	12,81 ± 00,21	12,78 ± 00,16	12,84 ± 00,16	07,61 ± 00,05	52,21 ± 56,08	12,78 ± 00,21	12,85 ± 00,11
25	16,00 ± 05,40	29,19 ± 24,81	07,90 ± 00,05	07,85 ± 00,07	07,86 ± 00,07	07,88 ± 00,08	07,10 ± 00,03	13,99 ± 05,56	07,91 ± 00,07	07,92 ± 00,08
50	79,02 ± 31,23	51,90 ± 14,53	34,24 ± 00,36	34,31 ± 00,29	34,50 ± 00,47	28,38 ± 02,38	70,82 ± 00,22	55,79 ± 15,71	34,01 ± 00,40	34,20 ± 00,30
75	65,80 ± 26,03	45,01 ± 09,43	24,32 ± 00,18	24,27 ± 00,29	24,21 ± 00,15	20,64 ± 01,25	50,19 ± 00,25	38,75 ± 10,63	24,18 ± 00,19	24,34 ± 00,30
100	44,99 ± 20,40	28,26 ± 06,36	18,06 ± 00,20	17,98 ± 00,11	18,02 ± 00,13	14,91 ± 00,95	36,67 ± 00,12	36,17 ± 05,67	17,90 ± 00,18	18,06 ± 00,11
150	32,66 ± 08,91	24,18 ± 07,23	12,25 ± 00,11	12,25 ± 00,08	12,20 ± 00,09	10,23 ± 00,35	24,80 ± 00,15	20,71 ± 04,58	12,14 ± 00,13	12,15 ± 00,09
200	19,35 ± 03,25	17,68 ± 05,27	10,02 ± 00,07	10,00 ± 00,04	10,01 ± 00,07	08,61 ± 00,44	19,90 ± 00,05	17,39 ± 03,88	10,07 ± 00,09	09,96 ± 00,09
250	26,76 ± 10,82	16,26 ± 02,33	08,79 ± 00,05	08,90 ± 00,05	08,87 ± 00,05	07,96 ± 00,37	17,38 ± 00,12	17,14 ± 04,38	08,85 ± 00,05	08,79 ± 00,06
300	16,63 ± 05,31	14,95 ± 04,09	07,81 ± 00,05	07,74 ± 00,05	07,77 ± 00,04	07,13 ± 00,33	15,16 ± 00,07	12,30 ± 03,63	07,74 ± 00,04	07,72 ± 00,05

Tabela B.25: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Feit96, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	05,74 ± 03,30	04,60 ± 01,41	03,49 ± 00,02	03,50 ± 00,02	03,48 ± 00,02	03,48 ± 00,01	07,69 ± 01,01	07,98 ± 05,81	03,49 ± 00,02	03,47 ± 00,03
25	33,75 ± 33,96	21,54 ± 11,02	13,77 ± 00,54	13,43 ± 00,99	13,56 ± 00,92	14,03 ± 00,20	18,63 ± 02,39	19,13 ± 09,53	13,20 ± 01,04	13,30 ± 01.13
50	30,99 ± 09,31	47,95 ± 23,62	20,15 ± 00,55	21,03 ± 01,33	20,10 ± 00,64	20,38 ± 00,47	44,06 ± 08,95	51,69 ± 29,16	21,48 ± 01,59	19,76 ± 00,67
75	55,00 ± 21,47	45,50 ± 04,86	29,42 ± 00,55	29,27 ± 00,95	28,69 ± 00,60	67,98 ± 25,34	104,92 ± 18,15	49,18 ± 04,88	28,93 ± 00,33	29,26 ± 01.19
100	51,01 ± 05,82	49,62 ± 11,53	32,48 ± 00,48	32,58 ± 00,54	32,10 ± 00,43	56,46 ± 11.18	134,35 ± 15,92	51,88 ± 08,74	32,42 ± 00,64	32,66 ± 00,79
150	44,60 ± 04,82	44,13 ± 06,38	29,54 ± 00,32	29,36 ± 00,40	29,76 ± 00,29	39,38 ± 10,15	106,47 ± 04,87	45,48 ± 05,26	29,31 ± 00,37	29,25 ± 00,40
200	39,56 ± 04,02	35,90 ± 05,68	25,44 ± 00,31	25,40 ± 00,36	25,23 ± 00,28	38,65 ± 10,19	86,10 ± 05,21	37,77 ± 03,43	25,20 ± 00,22	25,24 ± 00,34
250	35,26 ± 06,62	31,51 ± 03,42	22,38 ± 00,14	22,45 ± 00,18	22,35 ± 00,27	32,03 ± 07,12	79,82 ± 05,62	33,96 ± 03,37	22,57 ± 00,26	22,44 ± 00,25
300	31,60 ± 03,28	28,99 ± 02,97	19,39 ± 00,14	19,15 ± 00,17	19,44 ± 00,22	24,31 ± 06,05	61,89 ± 03,74	29,09 ± 03,11	19,31 ± 00,22	19,15 ± 00,16

Tabela B.26: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Feit96, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	423,67 ± 263,35	670,56 ± 809,91	238,45 ± 56,09	236,95 ± 51,69	259,24 ± 35,41	263,78 ± 55,02	189,33 ± 48,77	509,74 ± 463,36	258,28 ± 56,08	81,62 ± 13,10
25	920,35 ± 458,76	970,98 ± 654,61	525,77 ± 85,72	546,02 ± 79,09	570,72 ± 91,22	504,62 ± 112,19	276,87 ± 56,43	881,25 ± 354,98	607,98 ± 93,55	98,89 ± 21.19
50	1557,80 ± 378,52	3023,85 ± 1590,20	898,75 ± 139,38	1062,35 ± 118,18	958,52 ± 138,57	854,79 ± 158,19	527,52 ± 108,76	2655,34 ± 1557,08	1021,98 ± 189,58	139,08 ± 35,20
75	3642,93 ± 1505,90	2554,57 ± 838,18	1941,41 ± 267,34	1622,54 ± 402,68	1754,11 ± 434,63	1697,99 ± 322,93	737,04 ± 124,79	2567,07 ± 679,74	1636,22 ± 396,86	390,92 ± 25,16
100	3115,73 ± 428,17	2665,28 ± 438,77	1852,86 ± 203,25	2010,89 ± 171,41	1866,70 ± 274,92	1652,71 ± 249,93	822,35 ± 113,08	2720,21 ± 394,51	1950,47 ± 215,66	775,00 ± 111,31
150	2761,72 ± 466,02	3083,32 ± 592,32	1813,39 ± 188,74	1817,06 ± 150,12	1686,69 ± 99,28	1618,12 ± 191,97	714,01 ± 59,42	2482,81 ± 313,34	1904,19 ± 101,82	691,06 ± 91,45
200	2437,77 ± 402,65	2073,88 ± 322,82	1502,26 ± 205,10	1438,52 ± 142,90	1539,57 ± 127,31	1454,96 ± 145,14	649,63 ± 55,36	2222,95 ± 241,81	1510,35 ± 132,51	609,40 ± 55,26
250	2036,20 ± 384,11	1753,32 ± 159,84	1221,41 ± 127,44	1349,29 ± 101,71	1326,16 ± 95,41	1195,69 ± 120,89	572,09 ± 53,61	2169,75 ± 367,11	1246,65 ± 130,24	475,95 ± 69,96
300	1620,34 ± 345,15	1561,24 ± 190,03	1099,43 ± 110,37	1090,46 ± 74,66	1053,69 ± 93,24	1108,93 ± 61,39	498,45 ± 27,20	1587,05 ± 202,56	1147,86 ± 81,04	391,42 ± 55,58

Tabela B.27: Tempo de Resposta Médio±Intervalo de Confiança de 95% – Grid 512 nós, Feit96, 1000 Kpbs

Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	6247,75±6384,36	9086,61±8293,52	2746,31±557,20	2284,14±301,45	2622,85±646,31	2581,88±345,12	2058,55±458,27	5785,23±4636,71	2690,34±418,34	624,75±153,39
25	13877,76±15761.13	25001,45±35211,20	4804,67±1423,24	6038,55±1074,00	5759,26±1284,71	5786,54±1374,24	3339,35±627,10	12688,67±7167,71	5915,07±1136,17	299,62±26,02
50	20961,88±9941,82	25946,77±11762,30	9329,18±1700,04	10227,08±1186,46	8606,89±1249,52	9110,77±1381,66	6016,72±1078,63	22447,43±11609,67	9195,13±1461,78	242,47±58,52
75	24969,49±7568,86	22711,21±7218,01	16454,57±3981,74	18565,25±3558,94	12685,42±3847,46	14433,93±2014,27	7605,50±1403,73	32322,00±8500,32	16147,99±3271,26	534,51±105,82
100	28575,84±5249,05	25155,44±3027,69	19305,27±1898,07	18475,22±1357,16	20349,56±2028,32	19606,80±2348,88	7301,40±685,31	28328,44±2552,54	18186,20±2238,63	779,99±136,38
150	24578,52±2686,36	23421,37±2301.10	16438,28±1134,97	17551,57±1906,51	17514,98±1585,24	17073,61±2089,47	6364,57±403,61	28860,92±4865,28	18437,18±2068,82	793,31±158,79
200	21800,35±3813,91	20493,95±1614,93	14351.17±1529,17	14877,72±1131,30	15280,19±1576,28	14192,45±1118,25	5944,01±543,46	20432,08±2416,23	15591,02±1161.15	636,55±181,38
250	22284,95±3055,50	21828,29±3224,12	11930,72±947,46	12783,53±902,15	12394,62±1261.14	12074,02±982,48	5318,85±242,89	20215,36±3254,64	13624,79±982,79	494,44±67,10
300	18658,28±4265,96	17164,92±4067,53	11279,68±831.15	10767,00±1319,21	10728,17±1115,76	11411,52±1131,00	4851,53±423,53	17202,71±2561,33	10663,55±939,76	459,80±74,31

Tabela B.28: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-ASCI, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	03,86 ± 00,94	03,26 ± 00,45	02,65 ± 00,01	02,66 ± 00,01	02,65 ± 00,01	02,65 ± 00,01	06,17 ± 00,49	03,80 ± 00,58	02,66 ± 00,01	02,66 ± 00,01
25	03,67 ± 00,39	04,03 ± 00,69	03,10 ± 00,01	03,08 ± 00,01	03,10 ± 00,01	03,10 ± 00,01	05,38 ± 00,17	04,00 ± 00,46	03,10 ± 00,01	03,09 ± 00,01
50	03,30 ± 00,33	03,12 ± 00,13	02,58 ± 00,01	02,58 ± 00,00	02,58 ± 00,00	02,58 ± 00,00	06,74 ± 00,20	03,14 ± 00,10	02,58 ± 00,00	02,58 ± 00,00
75	03,29 ± 00,21	03,46 ± 00,54	02,77 ± 00,01	02,77 ± 00,00	02,77 ± 00,00	02,77 ± 00,00	07,02 ± 00,16	03,45 ± 00,20	02,65 ± 00,00	02,66 ± 00,00
100	03,55 ± 00,43	03,02 ± 00,10	02,68 ± 00,00	02,68 ± 00,00	02,68 ± 00,00	02,68 ± 00,00	06,64 ± 00,11	03,13 ± 00,16	02,60 ± 00,00	02,60 ± 00,00
150	03,39 ± 00,21	03,18 ± 00,12	03,03 ± 00,00	03,04 ± 00,01	03,04 ± 00,00	03,04 ± 00,00	06,73 ± 00,07	03,13 ± 00,08	02,62 ± 00,00	02,62 ± 00,00
200	03,57 ± 00,24	03,31 ± 00,19	03,04 ± 00,00	03,04 ± 00,00	03,04 ± 00,00	03,04 ± 00,00	06,86 ± 00,12	03,36 ± 00,16	02,71 ± 00,00	02,71 ± 00,00
250	03,54 ± 00,14	03,65 ± 00,22	03,05 ± 00,00	03,05 ± 00,00	03,06 ± 00,00	03,06 ± 00,00	06,95 ± 00,09	03,62 ± 00,14	02,77 ± 00,00	02,77 ± 00,00
300	03,57 ± 00,14	03,61 ± 00,13	03,02 ± 00,00	03,02 ± 00,00	03,02 ± 00,00	03,01 ± 00,00	06,87 ± 00,07	03,62 ± 00,17	02,77 ± 00,00	02,77 ± 00,00

Tabela B.29: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-ASCI, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	306,51 ± 154,89	264,07 ± 45,17	212,58 ± 34,12	204,05 ± 27,42	225,88 ± 33,36	199,43 ± 16,51	180,25 ± 28,69	266,22 ± 56,24	193,45 ± 36,03	50,36 ± 11,58
25	253,24 ± 59,09	264,96 ± 38,19	208,87 ± 27,99	215,01 ± 26,02	222,05 ± 25,58	209,46 ± 22,53	164,94 ± 17,07	322,81 ± 43,40	209,94 ± 24,33	34,28 ± 04,40
50	241,31 ± 35,09	230,58 ± 18,77	196,36 ± 20,85	193,77 ± 18,35	204,60 ± 15,23	208,54 ± 20,28	174,89 ± 13,58	240,73 ± 22,32	192,77 ± 14,59	39,14 ± 01,50
75	238,04 ± 15,97	240,38 ± 15,86	190,59 ± 07,60	206,22 ± 14,95	208,52 ± 13,40	202,29 ± 13,01	159,36 ± 11,30	256,87 ± 18,47	206,07 ± 11,59	39,44 ± 00,94
100	264,57 ± 26,98	234,14 ± 15,46	198,59 ± 11,08	206,83 ± 13,57	199,43 ± 11,74	200,23 ± 12,79	168,16 ± 07,79	234,86 ± 20,71	206,54 ± 07,00	38,19 ± 01,06
150	240,90 ± 12,20	246,02 ± 12,61	203,23 ± 07,40	207,32 ± 06,41	204,91 ± 06,50	195,45 ± 07,44	169,80 ± 09,13	236,74 ± 15,90	200,61 ± 10,97	38,69 ± 00,58
200	254,47 ± 11,83	250,20 ± 09,52	209,41 ± 10,29	210,64 ± 08,45	202,78 ± 08,05	204,35 ± 07,30	174,87 ± 09,85	248,33 ± 11.17	205,72 ± 05,38	38,25 ± 00,35
250	251,24 ± 10,62	269,10 ± 22,68	214,88 ± 07,91	209,35 ± 07,00	208,73 ± 11,80	211,72 ± 09,19	166,22 ± 05,58	259,15 ± 10,84	213,48 ± 06,47	37,93 ± 00,39
300	269,17 ± 12,66	263,70 ± 07,15	208,54 ± 05,34	210,35 ± 04,57	206,34 ± 07,06	211,92 ± 05,93	166,45 ± 06,78	247,26 ± 10,21	208,48 ± 06,29	37,68 ± 00,24

Tabela B.30: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-ASCI, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	3096,06 ± 1034,31	2664,52 ± 423,58	1776,08 ± 318,26	1761,23 ± 335,83	2099,29 ± 365,51	2188,15 ± 263,37	1551,84 ± 298,82	2508,64 ± 440,00	1976,83 ± 385,37	327,27 ± 123,32
25	2503,84 ± 466,75	2872,01 ± 483,65	2343,82 ± 217,68	2084,47 ± 231,41	2022,63 ± 257,29	2146,99 ± 184,34	1583,09 ± 176,64	2558,25 ± 222,00	2073,78 ± 178,12	172,33 ± 73,04
50	2552,96 ± 341,46	2482,81 ± 327,92	2068,40 ± 114,06	1937,62 ± 183,09	1899,06 ± 114,13	1976,55 ± 144,58	1555,93 ± 176,21	2410,68 ± 246,27	1876,38 ± 134,20	89,65 ± 14,34
75	2364,60 ± 97,21	2612,95 ± 179,26	1943,29 ± 136,29	2083,64 ± 185,21	1963,92 ± 167,92	1890,42 ± 150,67	1580,23 ± 55,83	2525,35 ± 201,81	2095,06 ± 117,63	58,69 ± 08,38
100	2529,15 ± 272,49	2356,99 ± 152,16	2072,86 ± 107,63	2078,81 ± 90,59	2020,07 ± 143,41	2111,96 ± 136,72	1648,01 ± 84,53	2417,61 ± 105,50	2017,85 ± 90,73	55,36 ± 05,85
150	2465,61 ± 155,02	2397,27 ± 122,97	2117,14 ± 115,20	1983,96 ± 115,08	2020,58 ± 79,70	2032,35 ± 59,69	1715,32 ± 87,38	2362,77 ± 97,49	2058,89 ± 92,73	54,19 ± 05,09
200	2696,72 ± 193,16	2492,67 ± 119,39	2020,85 ± 61,00	2016,71 ± 49,35	2095,66 ± 56,63	2118,13 ± 65,51	1657,34 ± 40,46	2461,46 ± 114,13	2118,72 ± 92,41	50,75 ± 02,80
250	2464,56 ± 109,14	2582,37 ± 173,53	2110,94 ± 90,37	2062,01 ± 100,76	2115,20 ± 79,43	2063,86 ± 66,02	1674,80 ± 50,43	2487,53 ± 111,95	2120,80 ± 45,02	45,53 ± 02,12
300	2556,28 ± 174,90	2602,66 ± 216,64	2058,81 ± 67,39	2100,57 ± 71,41	2078,65 ± 85,49	2004,06 ± 81,84	1652,20 ± 58,03	2531,69 ± 103,38	2055,65 ± 65,43	46,73 ± 03,07

Tabela B.31: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Jann97-CTC, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	290,75 ± 123,24	215,46 ± 72,06	115,53 ± 01,20	115,90 ± 00,56	116,13 ± 00,55	87,75 ± 02,43	448,08 ± 83,23	231,49 ± 97,98	116,64 ± 00,61	116,17 ± 00,84
25	188,11 ± 63,25	169,47 ± 53,63	93,64 ± 00,75	93,60 ± 00,40	94,26 ± 00,75	77,87 ± 03,96	300,84 ± 67,97	167,85 ± 62,91	93,61 ± 00,50	93,45 ± 00,94
50	137,98 ± 52,40	116,29 ± 24,79	73,14 ± 00,49	72,84 ± 00,35	73,30 ± 00,40	60,38 ± 01,09	200,24 ± 34,15	184,39 ± 70,94	72,70 ± 00,37	72,91 ± 00,35
75	108,26 ± 19,52	94,69 ± 12,94	63,16 ± 00,17	63,16 ± 00,34	63,20 ± 00,21	61,49 ± 11,76	170,09 ± 37,56	146,30 ± 35,54	63,22 ± 00,26	63,04 ± 00,23
100	103,89 ± 21.17	110,35 ± 24,99	55,45 ± 00,21	55,50 ± 00,15	55,40 ± 00,28	53,75 ± 09,14	164,35 ± 27,87	120,85 ± 42,99	55,41 ± 00,17	55,36 ± 00,25
150	139,61 ± 62,40	88,40 ± 15,55	48,11 ± 00,07	48,10 ± 00,10	48,24 ± 00,21	45,84 ± 01,56	139,26 ± 15,11	111,70 ± 28,68	47,99 ± 00,14	47,97 ± 00,15
200	122,17 ± 19,19	99,69 ± 20,65	56,41 ± 00,13	56,38 ± 00,23	56,28 ± 00,21	50,16 ± 00,74	138,20 ± 13,05	106,22 ± 20,84	56,28 ± 00,20	56,25 ± 00,20
250	101,68 ± 09,34	98,04 ± 16,90	55,92 ± 00,14	56,13 ± 00,14	56,06 ± 00,16	68,47 ± 08,14	167,20 ± 16,57	104,54 ± 24,39	55,91 ± 00,13	55,96 ± 00,08
300	115,60 ± 18,65	102,96 ± 07,89	60,37 ± 00,11	60,41 ± 00,17	60,34 ± 00,13	71,24 ± 10,55	173,04 ± 10,98	108,84 ± 14,27	60,39 ± 00,10	60,22 ± 00,15

Tabela B.32: Tempo de Resposta Médio±Intervalo de Confiança de 95% – Grid 512 nós, Jann97-CTC, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	15418,48±5820,31	16345,13±10605,67	6599,41±2228,14	6086,76±2534,07	9404,41±1986,87	6278,53±1883,31	3310,94±854,49	14917,57±8509,56	9489,37±2118,42	2941,65±385,62
25	16728,01±10356,15	6830,22±2309,25	7112,36±1500,30	4484,32±1997,39	5483,77±1751.19	4458,05±1504,03	2185,98±491,63	10027,41±5874,11	3088,66±1241,21	1953,11±316,40
50	5890,76±3437,10	9150,37±3776,72	5615,15±943,59	4342,99±1416,51	3406,95±1439,85	3360,51±1128,42	1592,33±620,79	8070,35±4185,69	4242,78±1602,47	1379,81±142,66
75	4845,39±1346,74	6028,74±1434,33	3249,34±868,55	3802,44±643,73	3376,96±647,37	2940,95±694,14	1668,74±328,58	10005,03±4628,43	3059,22±558,27	976,68±114,18
100	6005,12±2476,68	6326,46±1860,79	3445,72±618,70	2769,75±637,56	2801,68±609,01	2279,18±374,63	1302,80±236,11	6063,78±1790,71	3034,66±596,74	819,15±103,44
150	4379,02±797,93	4956,82±1202,62	2554,77±359,10	2428,03±366,30	2816,99±430,44	2193,72±338,13	1146,92±122,92	6279,05±2052,34	2732,05±450,72	808,15±54,31
200	5487,14±1714,95	4910,60±1085,89	2644,45±309,27	2894,58±370,92	2683,35±380,47	2460,70±211,25	1221,86±183,50	5539,17±1228,21	2777,14±379,18	750,62±61,42
250	5564,24±750,63	5833,34±1360,83	2844,12±272,18	3056,58±331,05	2947,30±493,74	2524,59±258,78	1334,38±169,60	5661,45±1728,44	3119,53±429,48	759,17±32,74
300	5751,20±649,64	5688,05±816,28	2987,95±297,81	3175,81±293,24	3057,51±303,28	2641,34±273,71	1429,33±164,11	5587,07±859,54	3188,44±261,26	725,27±38,21

Tabela B.33: Tempo de Resposta Médio±Intervalo de Confiança de 95% – Grid 512 nós, Jann97-CTC, 1000 Kpbs

Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	123199,13±69016,09	111336,96±54952,53	77920,93±22487,80	70775,93±24944,15	80615,25±21623,93	54814,52±20271,99	23303,72±9635,52	149295,85±97378,38	49017,77±13726,24	11602,54±4555,38
25	86707,23±19793,63	96467,20±43240,44	50594,86±18133,61	64153,31±19049,82	41926,37±14816,73	44000,19±12098,41	16840,06±6435,25	92289,49±65309,86	57292,75±18635,93	5763,47±2330,12
50	55990,05±26794,38	78092,23±58903,14	37630,16±16161,34	38917,01±14352,35	41163,71±13840,80	30680,86±8249,75	14462,93±4825,99	92394,33±52224,89	34373,73±11981,27	8506,97±1195,67
75	52853,31±15566,07	59690,08±33023,34	39012,26±7655,36	26915,03±7141,38	37628,98±9200,33	31902,78±4921,00	14004,01±3452,56	63251,29±20935,37	29376,70±7637,00	4405,95±1038,44
100	66428,14±25360,04	52538,75±13812,02	24742,01±5127,31	31198,01±5490,91	29382,55±7365,35	29965,69±4097,57	12904,63±1834,25	53891,26±12647,36	25613,45±5955,88	3299,25±715,27
150	70333,28±22056,19	46671,94±11163,07	27588,95±3191,64	25200,61±3353,46	21435,76±4349,92	22990,97±2933,22	11449,28±1350,98	56474,84±18953,45	28392,19±3588,95	2224,33±513,93
200	47047,20±8803,09	48997,12±12822,82	27785,64±2536,67	27049,62±2772,41	25518,82±4048,91	23265,30±3644,79	12650,57±1264,85	44350,90±11448,53	30476,54±1842,73	1901,77±468,10
250	61788,22±14869,76	49984,64±12950,26	27940,03±3070,63	30422,01±2729,77	30838,18±2667,70	24066,58±3308,18	12631.11±1218,48	55926,81±15903,44	30072,32±4203,24	1493,20±334,70
300	62128,03±10021,08	67095,31±15527,64	32607,45±2457,93	29565,36±4009,89	31419,23±2584,61	25405,10±1887,99	13901,76±1625,83	58913,41±11763,44	32806,81±3358,20	1475,17±225,50

Tabela B.34: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Lub99, s/ rede

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	16,81 ± 05,12	22,31 ± 14,46	12,57 ± 00,12	12,52 ± 00,19	12,38 ± 00,19	12,48 ± 00,19	19,10 ± 05,70	15,48 ± 03,39	12,49 ± 00,10	12,59 ± 00,22
25	15,14 ± 09,44	21,47 ± 18,37	07,73 ± 00,05	07,69 ± 00,04	07,70 ± 00,05	07,67 ± 00,05	17,64 ± 04,13	11,23 ± 03,30	07,66 ± 00,05	07,71 ± 00,04
50	64,56 ± 32,73	46,97 ± 12,80	33,20 ± 00,11	33,32 ± 00,24	33,34 ± 00,20	70,15 ± 29,03	124,62 ± 15,02	45,25 ± 08,34	33,27 ± 00,17	33,33 ± 00,27
75	29,41 ± 04,81	29,57 ± 05,00	23,73 ± 00,08	23,69 ± 00,13	23,61 ± 00,14	47,46 ± 23,76	90,92 ± 15,84	30,35 ± 09,10	23,53 ± 00,19	23,68 ± 00,13
100	26,50 ± 06,09	23,74 ± 05,41	17,49 ± 00,10	17,51 ± 00,10	17,49 ± 00,09	44,60 ± 17,20	60,49 ± 08,06	18,89 ± 01,79	17,45 ± 00,09	17,46 ± 00,10
150	18,45 ± 03,59	15,55 ± 02,44	11,89 ± 00,08	11,85 ± 00,04	11,85 ± 00,05	25,22 ± 09,02	45,18 ± 07,52	15,13 ± 03,25	11,84 ± 00,06	11,87 ± 00,05
200	15,11 ± 02,95	13,09 ± 02,49	09,87 ± 00,03	09,79 ± 00,04	09,78 ± 00,04	26,70 ± 07,17	34,62 ± 04,79	12,22 ± 02,10	09,73 ± 00,03	09,78 ± 00,04
250	12,98 ± 02,45	11,37 ± 01,44	08,60 ± 00,05	08,59 ± 00,03	08,63 ± 00,04	18,57 ± 06,00	26,59 ± 01,26	12,31 ± 02,68	08,56 ± 00,04	08,56 ± 00,03
300	10,22 ± 01,64	09,02 ± 01,36	07,59 ± 00,03	07,56 ± 00,03	07,59 ± 00,01	14,57 ± 04,95	26,73 ± 03,23	09,34 ± 02,47	07,54 ± 00,02	07,53 ± 00,03

Tabela B.35: Tempo de Resposta Médio ± Intervalo de Confiança de 95% – Grid 512 nós, Lub99, 100 Kbps

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	567,58 ± 205,97	873,78 ± 709,77	566,53 ± 102,54	454,74 ± 164,22	527,33 ± 143,13	441,76 ± 155,51	320,35 ± 52,06	889,92 ± 740,53	513,67 ± 188,83	151,49 ± 138,16
25	454,58 ± 154,23	395,93 ± 94,97	390,88 ± 74,52	355,16 ± 84,05	420,48 ± 86,00	360,55 ± 95,47	219,64 ± 26,69	638,38 ± 212,59	382,80 ± 83,65	58,28 ± 29,19
50	3286,86 ± 1304,95	3426,85 ± 1691,45	1827,18 ± 600,96	2396,06 ± 803,18	1137,49 ± 545,81	2040,90 ± 498,64	807,93 ± 213,77	2298,80 ± 688,86	1778,71 ± 619,95	553,04 ± 01,99
75	2247,42 ± 1009,70	2384,69 ± 956,80	1504,93 ± 511,92	996,24 ± 515,45	1015,32 ± 257,91	1248,55 ± 201,94	527,38 ± 144,80	2251,87 ± 820,57	1329,71 ± 616,76	400,37 ± 12,99
100	1546,44 ± 521,06	1660,59 ± 739,01	1351,79 ± 214,16	946,75 ± 343,03	930,67 ± 274,29	891,45 ± 256,90	432,98 ± 102,51	1172,21 ± 302,33	969,72 ± 314,83	282,01 ± 01.18
150	886,42 ± 383,13	1204,18 ± 290,71	566,61 ± 173,92	651,57 ± 175,47	569,63 ± 208,50	628,49 ± 160,84	373,98 ± 58,61	851,47 ± 328,54	820,30 ± 203,09	185,74 ± 00,43
200	723,20 ± 216,09	897,38 ± 341.15	513,96 ± 154,69	682,24 ± 119,80	626,99 ± 148,13	602,88 ± 129,86	333,34 ± 38,24	859,54 ± 226,19	680,78 ± 171,81	152,57 ± 03,97
250	909,94 ± 238,13	612,80 ± 137,55	568,92 ± 120,82	491,27 ± 100,40	644,71 ± 141,46	546,30 ± 66,23	271,68 ± 30,93	733,44 ± 287,21	473,49 ± 120,98	130,63 ± 02,47
300	713,38 ± 120,55	806,42 ± 288,33	526,13 ± 91,74	550,64 ± 69,53	475,19 ± 82,94	443,91 ± 103,04	275,58 ± 34,42	586,80 ± 153,51	445,69 ± 104,86	111.16 ± 00,44

Tabela B.36: Tempo de Resposta Médio±Intervalo de Confiança de 95% – Grid 512 nós, Lub99, 1000 Kpbs

Qtde, Jobs	Random	Lowest	Disted	Global	Central	Tlba	GAS	R,Robin	DPWPExt	NBSP
10	10191,60±8717,81	5560,98±3782,93	5263,08±1900,47	4192,66±1436,05	5175,96±1735,51	3722,52±1235,64	3158,40±526,18	7613,17±4399,76	4568,26±1368,40	254,20±134,32
25	8184,07±6479,10	7975,86±5794,94	4012,98±1100,18	4053,00±841,28	3990,23±569,99	3771.15±843,99	2007,46±395,74	5112,89±1817,39	4365,85±692,99	137,57±53,64
50	26583,07±10798,60	29688,20±8843,24	20567,86±8897,38	18249,07±6871,86	19185,78±6315,96	15841,56±4739,54	8328,79±2052,25	23623,77±6886,51	22025,13±7970,34	593,25±20,07
75	17531,84±5742,95	22251,33±13610,42	11343,03±3737,63	12389,28±4671,35	16781,41±4523,81	11572,42±3721.17	5485,77±1471,70	16192,36±8177,26	15803,13±3722,87	414,84±16,46
100	13447,94±6647,02	14253,71±3963,67	9098,42±3150,17	12040,66±2757,25	13044,97±2459,23	8535,63±2192,26	4657,17±1078,15	11264,96±3031,39	11696,19±3221,28	315,05±16,66
150	10677,66±2880,21	11133,25±4392,95	7454,71±1740,45	7530,78±2019,34	6298,93±1951,30	7266,76±1692,82	2706,00±675,14	9277,94±3859,12	6420,43±2052,51	219,59±53,86
200	8295,96±1936,60	7689,75±2266,73	5837,52±1147,89	6041,45±1521,33	5425,86±1753,62	5928,59±1008,90	2827,53±481,74	8360,77±3342,38	6597,15±1343,84	172,28±22,37
250	6443,03±2013,17	6295,05±1676,75	5501,29±1378,10	4164,04±669,81	5956,74±1386,64	5036,28±1230,43	2596,93±430,59	8027,27±2102,44	5019,40±1460,16	137,07±05,27
300	6272,67±1458,41	8323,76±3373,36	5264,41±1048,06	4571,43±1028,42	4271,83±1092,69	3831,64±613,98	2637,03±323,40	5447,42±1447,81	4340,26±1184,31	116,13±03,04