Universidade de São Paulo
Instituto de Ciências Matemáticas e de Computação
Programa de Pós Graduação em Ciências de Computação e
Matemática Computacional

Mirador - Uma Ferramenta Para Monitoramento e Gerenciamento do SPP2

Luís Augusto Cassimiro de Araujo

ORIENTADOR: PROF. DR. ONOFRE TRINDADE JR.

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

São Carlos
Dezembro de 1998

*Aos meus pais, irmãs,
avós (em memória do meu avô Luis)
e à Adriana, minha namorada.*

Agradecimentos

Ao Prof. Onofre Trindade Jr. pela amizade, orientação, incentivo, boas risadas e pela oportunidade de aprender tantos assuntos novos, não somente os relacionados a este trabalho.

À minha família por tudo que sou, pelo carinho e pelo inestimável apoio em minhas empreitadas.

À Adriana, um coração enorme, uma pessoa simplesmente maravilhosa.

Aos amigos Gustavo, Juliana, Nilda e Magnus, que tanto me ajudaram na concretização deste trabalho.

Ao Prof. Armando de Oliveira Fortuna pela amizade e colaboração em todo o desenvolvimento do trabalho.

À Profa. Renata Pontim de Matos Fortes, à Profa. Rosely Sanches e às minhas amigas da área de engenharia de software pelo apoio no desenvolvimento da ferramenta Mirador.

Aos amigos do LCAD e do ICMC pelos momentos de diversão e trabalho compartilhados.

Aos meus amigos grandes amigos, muitos a milhares de quilômetros, pela amizade, incentivo e ajuda prestativa em questões pertinentes a este trabalho.

À CNPQ pelo auxílio financeiro.

Resumo

O desenvolvimento da tecnologia de computadores tem facilitado a construção de máquinas paralelas MIMD com memória distribuída e de alto desempenho, a exemplo do SPP2. Exceto pelo maior desempenho apresentado, essas máquinas têm operação similar a de uma rede local de computadores, permitindo a utilização direta de uma ampla gama de ferramentas para o desenvolvimento de aplicações paralelas disponíveis para essa classe de sistemas. Por outro lado, a disponibilidade de ferramentas para o monitoramento e gerenciamento da máquina paralela é muito escassa e as poucas ferramentas disponíveis não satisfazem as necessidades dos usuários.

A ferramenta Mirador, apresentada neste trabalho, cobre essa lacuna. Essa ferramenta permite o monitoramento e o gerenciamento do SPP2 em particular e, com pequenas alterações, de uma rede de computadores que esteja sendo utilizada como uma máquina paralela. Para tornar seu uso mais flexível, a ferramenta Mirador usa como interface com o usuário um navegador (*browser*) Internet com suporte à linguagem Java, permitindo o monitoramento e o gerenciamento da arquitetura alvo independentemente da plataforma de hardware ou de sistema operacional que o usuário utilize.

A funcionalidade da ferramenta Mirador foi testada e os primeiros resultados de utilização mostram que ela pode simplificar bastante as tarefas diárias de programadores e administradores de sistemas paralelos.

Abstract

The technological evolution of computer components have made easier the development of MIMD parallel architectures based on a distributed memory model, such as the SPP2. This class of computers is very similar in operation to a conventional computer network although presenting a better performance. There are several tools for the development and monitoring of parallel programs and some can be found as a freeware. On the other hand, there are few freeware tools for the monitoring and management of parallel machines, all of them without the necessary functionality.

The Mirador tool, presented in this work, overcomes this lack of functionality. This tool is intend for the monitoring and management of the SPP2 and, with minor changes, can be applied to a computer network used as a parallel machine. The Mirador was developed to operate remotely, allowing for the monitoring and management of the target architecture through a computer network. Its user interface is based on the Java and WWW technology, making its use possible in a wide range of platforms and operating systems.

The Mirador was functionally tested and preliminary user feedback shows that it can make easier most of the daily tasks of programmers and administrators of parallel systems.

Índice

1 INTRODUÇÃO.....	1
1.1 ASPECTOS GERAIS	1
1.2 A FERRAMENTA MIRADOR	2
1.3 ORGANIZAÇÃO DO TRABALHO.....	3
2 REVISÃO BIBLIOGRÁFICA.....	4
2.1 CONSIDERAÇÕES INICIAIS	4
2.2 PROCESSAMENTO PARALELO	4
2.2.1 Tipos de Paralelismo	5
2.2.2 Concorrência e Paralelismo	8
2.2.3 Níveis de Paralelismo e Granulosidade.....	8
2.2.4 Speedup e Eficiência.....	9
2.3 ARQUITETURAS PARA PROCESSAMENTO PARALELO.....	10
2.3.1 Taxonomia de Flynn	10
2.3.2 Taxonomia de Duncan	14
2.3.3 Tópicos Adicionais sobre as Arquiteturas MIMD com Memória Compartilhada e Distribuída	18
2.3.4 A Arquitetura do SPP2.....	25
2.4 PROGRAMAÇÃO PARALELA	26
2.4.1 Ativação e Término de Processos Paralelos.....	26
2.4.2 Comunicação e Sincronização de Processos Paralelos.....	30
2.5 FERRAMENTAS PARA DESENVOLVIMENTO DE APLICAÇÕES PARALELAS	36
2.5.1 Ferramentas de Programação	36
2.5.2 Ferramentas de Auxílio à Paralelização	37
2.5.3 Ferramentas de Depuração	37
2.5.4 Ferramentas de Análise de Desempenho	38
2.6 FERRAMENTAS PARA MONITORAMENTO E GERENCIAMENTO DE MÁQUINAS PARALELAS	44
2.7 ANÁLISE COMPARATIVA DAS FERRAMENTAS REVISTAS	49
2.8 CONSIDERAÇÕES FINAIS	51
3. DEFINIÇÃO DA FERRAMENTA MIRADOR.....	52
3.1 CONSIDERAÇÕES INICIAIS	52
3.2 OBJETIVOS.....	52
3.3 ESCOPO.....	53
3.4 REQUISITOS FUNCIONAIS.....	54
3.5 REQUISITOS NÃO FUNCIONAIS.....	69
3.6 CONSIDERAÇÕES FINAIS	70
4 ESPECIFICAÇÃO DA FERRAMENTA MIRADOR	71
4.1 CONSIDERAÇÕES INICIAIS	71
4.2 FUNCIONALIDADE DA FERRAMENTA.....	71
4.2.1 Diagrama de Fluxo de Dados da Ferramenta.....	71
4.2.2 Descrição das Funções da Ferramenta	73
4.2.3 Descrição das Bases de Dados da Ferramenta	83
4.3 COMPORTAMENTO DA FERRAMENTA.....	85
4.3.1 Statechart da Ferramenta	85
4.3.2 Descrição dos Estados da Ferramenta.....	89
4.4 CONSIDERAÇÕES FINAIS	92
5 PROJETO, IMPLEMENTAÇÃO E TESTES DA FERRAMENTA MIRADOR	93
5.1 CONSIDERAÇÕES INICIAIS	93
5.2 PROJETO DA ARQUITETURA DA FERRAMENTA.....	93
5.2.1 Diretrizes do Projeto da Arquitetura da Ferramenta	94
5.2.2 Arquitetura da Ferramenta	94

5.2.3 Organização dos Módulos de Software e Hardware da Ferramenta	98
5.2.3 Protocolos de Comunicação entre os Módulos da Ferramenta.....	101
5.2.4 Bases de Dados da Ferramenta.....	104
5.3 PROJETO DA INTERFACE DA FERRAMENTA	104
5.3.1 Diretrizes do Projeto da Interface da Ferramenta	104
5.3.2 Interface da Ferramenta	105
5.4 IMPLEMENTAÇÃO DA FERRAMENTA	112
5.4.1 Aspectos Gerais de Implementação da Ferramenta	112
5.4.2 Implementação dos Módulos	112
5.4.3 Protocolos de Rede para a Comunicação entre Módulos.....	114
5.4.4 Implementação das Bases de Dados	114
5.5 TESTES E RESULTADOS OBTIDOS.....	114
5.6 CONSIDERAÇÕES FINAIS	118
6 CONCLUSÃO	120
6.1 CONSIDERAÇÕES INICIAIS	120
6.2 AVALIAÇÃO FINAL DA FERRAMENTA MIRADOR – DESENVOLVIMENTO E UTILIZAÇÃO.....	120
6.3 CONTRIBUIÇÕES DESTE TRABALHO	121
6.4 SUGESTÕES PARA TRABALHOS FUTUROS.....	123
REFERÊNCIAS BIBLIOGRÁFICAS	124
APÊNDICE A: DICIONÁRIO DE DADOS DO DFD DA FERRAMENTA MIRADOR	128
APÊNDICE B: DICIONÁRIO DE DADOS DO STATECHART DA FERRAMENTA MIRADOR	132

Índice de Figuras

FIGURA 2.1 - TIPOS DE PARALELISMO.....	6
FIGURA 2.2 - FUNCIONAMENTO DE UM <i>PIPELINE</i>	7
FIGURA 2.3 - COMPUTADOR DO TIPO SISD.....	11
FIGURA 2.4 - COMPUTADOR DO TIPO SIMD	12
FIGURA 2.5 - COMPUTADOR DO TIPO MISD	13
FIGURA 2.6 - COMPUTADOR DO TIPO MIMD.....	14
FIGURA 2.7 - CLASSIFICAÇÃO DE DUNCAN.....	15
FIGURA 2.8 - ARQUITETURA MIMD COM MEMÓRIA COMPARTILHADA	17
FIGURA 2.9 - ARQUITETURA MIMD COM MEMÓRIA DISTRIBUÍDA	17
FIGURA 2.10 - ARQUITETURA MIMD COM MEMÓRIA COMPARTILHADA E QUE UTILIZA MEMÓRIA <i>CACHE</i>	20
FIGURA 2.11 - REDES DE INTERCONEXÃO PARA MEMÓRIA COMPARTILHADA	22
FIGURA 2.12 - REDES DE INTERCONEXÃO PARA MEMÓRIA DISTRIBUÍDA	24
FIGURA 2.13: A ARQUITETURA DO SPP2.....	25
FIGURA 2.14 - USO DE COROTINAS	27
FIGURA 2.15 - EXEMPLO DA UTILIZAÇÃO DE <i>FORK/JOIN</i>	28
FIGURA 2.16 - EXEMPLO DA UTILIZAÇÃO DE <i>COBEGIN/COEND</i>	29
FIGURA 2.17 - EXEMPLO DA UTILIZAÇÃO DE <i>DOALL</i>	29
FIGURA 2.18 - FUNCIONAMENTO DE UMA COMUNICAÇÃO SÍNCRONA.....	32
FIGURA 2.19 - FUNCIONAMENTO DE UMA COMUNICAÇÃO ASSÍNCRONA.....	33
FIGURA 2.20 - A COMUNICAÇÃO NÃO BLOQUEANTE	34
FIGURA 2.21 - MECANISMO DE COMUNICAÇÃO <i>RENDEZVOUZ</i>	35
FIGURA 2.22 - O MECANISMO RPC	35
FIGURA 2.23 - VISUALIZAÇÃO DA EXECUÇÃO DE UMA APLICAÇÃO PARALELA NO XMPI.....	39
FIGURA 2.24 - VISUALIZAÇÃO DA EXECUÇÃO DE UMA APLICAÇÃO PARALELA NO XPVM	39
FIGURA 2.25 - MONITORAMENTO DA EXECUÇÃO DE TAREFAS PVM POR ELEMENTOS DE PROCESSAMENTO	40
FIGURA 2.26 - INTERFACE DE VISÕES DO VT	41
FIGURA 2.27 - INTERFACE DO VT	43
FIGURA 2.28 - VISÃO INTERPROCESSOR COMMUNICATION DO VT.....	43
FIGURA 2.29 - MONITOR DE DESEMPENHO DO VT.....	46
FIGURA 2.30 - VISÃO PROCESSOR UTILIZATION (3D) DO VT.....	46
FIGURA 2.31 - MONITORAMENTO DE UMA MÁQUINA SPP2 PELO PROGRAMA <i>bWATCH</i>	47
FIGURA 2.32 - MONITORAMENTO DE UMA MÁQUINA SPP2 PELO PROGRAMA <i>TOP</i> DO CONJUNTO DE PROGRAMAS <i>PROCPS-PARE</i>	48
FIGURA 3.1: ILUSTRAÇÃO DA SEMÂNTICA DE NÓS SELECIONADOS PELO MONITORAMENTO	55
FIGURA 3.2 - RELAÇÃO ENTRE A ATUALIZAÇÃO DE INFORMAÇÕES DO MONITORAMENTO E A OBTENÇÃO DE INFORMAÇÕES DOS NÓS	60
FIGURA 3.3 - RELAÇÃO ENTRE TODAS AS VARIÁVEIS QUE REGEM A OBTENÇÃO DE INFORMAÇÕES DOS NÓS E A ATUALIZAÇÃO DAS INFORMAÇÕES DO MONITORAMENTO	62
FIGURA 3.4 - EXEMPLOS DE ÍCONES DE NÓS OPERACIONAIS (ESCALA DE 10% PARA CARGA)	69
FIGURA 3.5 - ÍCONES DE NÓS OPERACIONAIS	69
FIGURA 4.1 - DIAGRAMA DE FLUXO DE DADOS DA FERRAMENTA <i>MIRADOR</i>	72
FIGURA 4.2 - DIAGRAMA DE FLUXO DE DADOS DA FUNÇÃO EXECUTAR FUNÇÕES/MONITORAR INFORMAÇÕES	74
FIGURA 4.3 - <i>STATECHART</i> DA FERRAMENTA <i>MIRADOR</i>	87
FIGURA 5.1 - A ARQUITETURA DA FERRAMENTA <i>MIRADOR</i>	95
FIGURA 5.2 - MAPEAMENTO DAS FUNÇÕES, ENTIDADES E BASES DE DADOS NA ARQUITETURA DA FERRAMENTA <i>MIRADOR</i>	96
FIGURA 5.3 - PROTOCOLOS DE COMUNICAÇÃO DA FERRAMENTA <i>MIRADOR</i>	102
FIGURA 5.4 - ORGANIZAÇÃO DA INTERFACE DA FERRAMENTA <i>MIRADOR</i>	106
FIGURA 5.5 - JANELA PRINCIPAL.....	107
FIGURA 5.6 - MENU <i>SESSION</i>	107
FIGURA 5.7 - MENU <i>REPORT</i>	107
FIGURA 5.8 - MENU <i>CONTROL</i>	107
FIGURA 5.9 - MENU <i>SETUP</i>	107
FIGURA 5.10 - JANELA <i>OPEN</i>	108

FIGURA 5.11 – JANELA NODE DATE AND TIME	108
FIGURA 5.12 – JANELA NODE ERROR MESSAGES	108
FIGURA 5.13 – JANELA NODE MEMORY USAGE	108
FIGURA 5.14 – JANELA NODE TASKS	108
FIGURA 5.15 – JANELA NODE UTILIZATION.....	108
FIGURA 5.16 – JANELA SPP2 CONFIGURATION	108
FIGURA 5.17 – JANELA LOAD RANGE CONVENTIONS (ESCALA DE 10%).....	109
FIGURA 5.18 – JANELA LOAD RANGE CONVENTIONS (ESCALA DE 20%).....	109
FIGURA 5.19 – JANELA LOAD RANGE CONVENTIONS (ESCALA DE 50%).....	109
FIGURA 5.20 – JANELA COMMAND CONSOLE.....	110
FIGURA 5.21 – MENU RESET NODE.....	110
FIGURA 5.22 – MENU USER MANEGAMENT	110
FIGURA 5.23 – JANELA MANAGEMENT SETTINGS	110
FIGURA 5.24 – JANELA MONITORING SETTINGS	110
FIGURA 5.25 – JANELA SERVICES TIMEOUT (<i>TAIMTIMEOUT</i>).....	111
FIGURA 5.26 – JANELA ADD USER	111
FIGURA 5.27 – JANELA REMOVE USER.....	111
FIGURA 5.28 – JANELA UPDATE USER DATA	111
FIGURA 5.29 – JANELA SPP2 USERS	111
FIGURA 5.30 – MENU ATIVADO SOBRE O ÍCONE DE UM NÓ.....	111
FIGURA 5.31 – MENU ATIVADO SOBRE INFORMAÇÕES DE TAREFAS	112
FIGURA 5.32 – JANELA CHANGE PRIORITY.....	112
FIGURA 5.33 – INSTANTE DE EXECUÇÃO DE UM TESTE NA FERRAMENTA MIRADOR.....	115

Índice de Tabelas

TABELA 2.1 - QUADRO COMPARATIVO DAS FERRAMENTAS DE MONITORAMENTO E GERENCIAMENTO DE MÁQUINAS PARALELAS	50
TABELA 3.1 – SEMÂNTICA DE NÓS SELECIONADOS PELO MONITORAMENTO.....	55
TABELA 3.2 – INTERVALOS DE CARGA ASSOCIADOS ÀS CORES DOS ÍCONES DOS NÓS OPERACIONAIS	68
TABELA 5.1 – CORRELAÇÃO ENTRE OS SERVIÇOS E FUNÇÕES DO MÓDULO SINGOG.....	99
TABELA 5.2 - QUADRO COMPARATIVO DAS FERRAMENTAS DE MONITORAMENTO E GERENCIAMENTO DE MÁQUINAS PARALELAS COM A FERRAMENTA MIRADOR	118

Capítulo 1

Introdução

1.1 Aspectos Gerais

A resolução de problemas utilizando-se computadores esteve, durante muito tempo, limitada à forma estritamente seqüencial de execução de instruções imposta pela arquitetura de von Neumann. Esse modelo arquitetural tem satisfeito muitas das necessidades computacionais existentes. Entretanto, apesar dos avanços tecnológicos, o modelo de von Neumann não fornece o poder computacional exigido por algumas aplicações específicas, tal como a simulação de fenômenos físicos complexos. A necessidade de maior poder computacional tem aumentado sistematicamente. Se, por exemplo, alguns anos atrás uma aplicação típica envolvia a solução numérica de um sistema linear simples, hoje algumas aplicações envolvem a solução de sistemas lineares com grande número de variáveis e em curto espaço de tempo. Essas aplicações necessitam de elevado poder computacional aliado a uma grande quantidade de memória.

Limites físicos também têm imposto restrições sobre a máxima velocidade de processamento que se pode obter da arquitetura de von Neumann. Apesar do aumento de desempenho das máquinas von Neumann estar crescendo a uma taxa exponencial, acredita-se que esse aumento somente possa continuar utilizando-se novos modelos computacionais e novas arquiteturas. Arquiteturas paralelas podem fornecer hoje o desempenho computacional necessário para a execução das aplicações que necessitam computação de alto desempenho.

A introdução de máquinas paralelas foi seguida da introdução de novos paradigmas de programação. Programar uma máquina paralela requer, além das construções normalmente disponíveis em uma linguagem de programação seqüencial, outras que possibilitem a ativação, interrupção e cooperação de tarefas (processos). As ferramentas para programação de aplicações paralelas tornam essas construções disponíveis para o programador. Durante o desenvolvimento de aplicações paralelas, ferramentas adicionais possibilitam a análise do desempenho e a depuração dessas aplicações.

Apesar do aparato disponível para o desenvolvimento de aplicações paralelas, executá-las, monitorá-las e gerenciá-las nem sempre é uma tarefa trivial, comparativamente à execução das mesmas atividades em relação a aplicações em máquinas von Neumann. Tal dificuldade é causada pela multiplicidade de recursos de uma máquina paralela, que pode ser identificada tanto na quantidade e configurações dos elementos de processamento quanto nos canais de comunicação que interligam esses elementos. Dessa forma, há a necessidade de um elemento que aglutine essa multiplicidade de recursos de forma a facilitar a interação, o gerenciamento e o monitoramento da máquina paralela e, conseqüentemente, o desenvolvimento, execução, monitoramento e gerenciamento de aplicações paralelas. O elemento destinado a fornecer uma maior coesão dos recursos da máquina paralela sob a ótica do usuário pode ser tanto o sistema operacional da máquina quanto ferramentas que simplifiquem operações que demandariam um razoável conhecimento dos recursos da máquina e de como utilizá-los. Essas ferramentas são denominadas ferramentas para monitoramento e gerenciamento de máquinas paralelas.

1.2 A Ferramenta Mirador

O presente trabalho destina-se a ampliar o leque de ferramentas para desenvolvimento, monitoramento e gerenciamento de aplicações paralelas no SPP2 [TRI95a, TRI95b]. O SPP2 é uma máquina paralela desenvolvida no Laboratório de Computação de Alto Desempenho (LCAD) do Instituto de Ciências Matemáticas e de Computação (ICMC), vinculado à Universidade de São Paulo (USP). Há disponível na Internet um conjunto considerável de ferramentas que permitem o desenvolvimento de aplicações paralelas no SPP2. Alguns exemplos dessas ferramentas são o PVM [GEI94] e o *p2d* [PER96]. Entretanto, as ferramentas de monitoramento e gerenciamento disponíveis para o SPP2, bWatch [BWA98, BEO98] e Procps-Pare [PRO98, BEO98], possuem recursos limitados e não apresentam muitas facilidades para o monitoramento e gerenciamento de aplicações paralelas.

A ferramenta proposta neste trabalho, denominada Mirador, fornece funções de monitoramento e gerenciamento do SPP2, que podem ser utilizadas através de uma interface gráfica baseada em um navegador (*browser*) Internet com suporte à linguagem Java [JAV98]. Dessa forma, pode-se acessar as funções da ferramenta remotamente e independentemente de plataforma de hardware ou de sistema operacional. Citam-se como exemplos de funções fornecidas pela ferramenta Mirador o monitoramento do estado (carga, tarefas e memória) dos nós, a eliminação de tarefas em execução nos nós e o reinício de nós.

Em todo o desenvolvimento da ferramenta procurou-se utilizar de forma extensiva conceitos e ferramentas de engenharia de software. O uso dessas ferramentas permite que o sistema desenvolvido atenda muito melhor as necessidades dos usuários, minimizando bastante as etapas de testes e manutenção. Como benefício adicional, mas igualmente importante, o sistema fica perfeitamente documentado facilitando-se bastante o trabalho de implementação em equipe e as manutenções evolutivas que se façam necessárias ou desejáveis.

1.3 Organização do Trabalho

O presente trabalho é composto por seis capítulos. O capítulo 2 apresenta a revisão bibliográfica dos tópicos necessários para o desenvolvimento do trabalho. O capítulo 3 apresenta a definição da ferramenta Mirador, abordando os usuários em potencial e os requisitos funcionais e não funcionais que a ferramenta deve atender. O capítulo 4 corresponde à especificação da ferramenta, apresentando sua funcionalidade e o seu comportamento. No capítulo 5 são apresentados o projeto, a implementação e os testes da ferramenta. As conclusões pertinentes a este trabalho são apresentadas no capítulo 6.

Existem, ainda, dois apêndices que complementam o texto do corpo do trabalho. Nos apêndices A e B são apresentados os Dicionários de Dados referentes ao Diagrama de Fluxo de Dados e ao *Statechart* da ferramenta, respectivamente.

Capítulo 2

Revisão Bibliográfica

2.1 Considerações Iniciais

Este capítulo tem por objetivo apresentar os diversos temas relacionados com o processamento paralelo, enfocando-se as ferramentas para o desenvolvimento/monitoramento de programas e gerenciamento da máquina paralela. A seção 2.2 expõe a necessidade e os principais conceitos de processamento paralelo, enquanto na seção 2.3 são apresentadas as classificações de arquiteturas de computadores, dando-se ênfase às arquiteturas de multicomputadores e multiprocessadores. A arquitetura e o princípio de funcionamento do SPP2 são também apresentados nessa seção. A seção 2.4 versa sobre os principais mecanismos de programação paralela. Na seção 2.5 são descritas as diversas ferramentas para o desenvolvimento de aplicações paralelas. Além das ferramentas de desenvolvimento, a utilização de uma arquitetura paralela é largamente facilitada pelo uso de ferramentas para monitoramento e gerenciamento, que são abordadas na seção 2.6. Algumas dessas ferramentas estão disponíveis para o SPP2. Na seção 2.7 é exibida uma comparação entre as ferramentas existentes procurando-se salientar suas qualidades e defeitos. Finalmente, na seção 2.8 são apresentadas as considerações finais relativas aos assuntos abordados neste capítulo.

2.2 Processamento Paralelo

O processamento paralelo, também referenciado como computação paralela, designa uma forma eficiente de processamento com ênfase na exploração de eventos paralelos no processo computacional através de uma coleção de elementos de processamento (processadores) que cooperam e comunicam-se entre si [ALM94, NAV89].

O surgimento do processamento paralelo remonta ao ano de 1920, quando Vanevar Bush, pesquisador do MIT (Massachusetts Institute of Technology), apresentou um computador analógico capaz de resolver equações diferenciais em paralelo. O próprio von Neumann sugeriu em seus artigos, nos anos 40, uma grade para resolver equações diferenciais em que os pontos

são atualizados em paralelo. Apesar de tais pesquisas, considera-se como marco inicial no surgimento de máquinas para processamento paralelo o ILLIAC IV, construído no fim da década de 60 na Universidade de Illinois e composto por 64 processadores [NAV89].

O principal incentivo à utilização do processamento paralelo é a obtenção de um desempenho maior na execução de aplicações que demandam grande poder computacional, poder esse que, apesar dos avanços tecnológicos, ainda não está disponível nas arquiteturas sequenciais de von Neumann. Devido a limitações físicas como a velocidade finita da luz, o poder computacional necessário para a execução dessas aplicações poderá não ser atingido pelas arquiteturas sequenciais. Além da busca por um desempenho maior na execução de determinadas aplicações, a computação paralela é um dos meios mais comuns de se obter tolerância a falhas, utilizando a redundância de elementos de hardware e recursos de software embutidos na execução de aplicações intrinsecamente paralelas [ALM94].

As áreas de aplicação da computação paralela são inúmeras e extremamente heterogêneas, variando desde o prognóstico do tempo até o controle de reatores nucleares, passando por modelagem de reservatórios de petróleo e decisão das operações e trajetos a serem feitos por um ônibus espacial ou um avião moderno.

O processamento paralelo envolve uma considerável gama de conceitos. Nas próximas seções são apresentados os principais conceitos necessários ao entendimento das questões básicas relacionadas ao processamento paralelo.

2.2.1 Tipos de Paralelismo

A ocorrência de eventos computacionais (processamento de tarefas ou instruções) em um computador pode apresentar, dependendo da arquitetura e do mecanismo de escalonamento de processos do sistema operacional desse computador, uma das seguintes formas:

- Seqüencial: um evento somente é iniciado após o término de um outro evento;
- Concorrente: em um determinado instante dois ou mais eventos foram iniciados, mas não foram terminados.

A ocorrência de eventos concorrentes pode ser, segundo Kirner [KIR91], proveniente de um dos seguintes tipos de paralelismo:

- Lógico: os eventos são executados intercaladamente por um único elemento de processamento, de modo que apenas um deles está ativo em cada instante. Esse tipo de paralelismo é exemplificado na figura 2.1(a), que apresenta um gráfico onde se demonstra a execução de três eventos em relação ao tempo;
- Físico: todos os eventos podem estar ocorrendo simultaneamente, pois há um elemento de processamento disponível para cada evento (figura 2.1 (b));
- Misto: combinação dos paralelismos lógico e físico, ou seja, os eventos podem estar sendo executados intercalada ou simultaneamente (figura 2.1(c)).

(a) Paralelismo lógico

(b) Paralelismo físico

(c) Paralelismo misto

Figura 2.1 - Tipos de paralelismo

O paralelismo físico pode ser dividido em três tipos: espacial, temporal (*pipeline*) ou combinado.

O paralelismo espacial corresponde à execução simultânea dos eventos no tempo, podendo ser síncrono ou assíncrono. No paralelismo espacial síncrono os elementos de processamento são ligados a uma única unidade de controle e, portanto, executam a mesma instrução sincronamente. No paralelismo espacial assíncrono os elementos de processamento executam instruções diferentes e interagem entre si quando necessário.

O paralelismo temporal, ou *pipeline*, implica a execução de eventos sobrepostos no tempo. Uma determinada tarefa é subdividida em uma seqüência de subtarefas, sendo cada uma delas executada por um estágio especializado de hardware que opera paralelamente com os outros estágios do *pipeline*.

A figura 2.2 exemplifica o funcionamento de um *pipeline*, composto por 4 estágios (E1, E2, E3 e E4), cada qual responsável pela execução de um determinado conjunto de operações sobre um certo conjunto de dados pertencente a uma tarefa. Os estágios são organizados seqüencialmente, de maneira que, a partir do estágio inicial E1, cada um deles é alimentado pelo estágio imediatamente anterior (figura 2.2 (a)), similarmente ao funcionamento de uma linha de produção industrial. A cada instante de tempo uma nova tarefa é iniciada. As tarefas são denominadas T1, T2, T3, ..., Tn. A figura 2.2 (b) apresenta as tarefas executadas pelos estágios de execução do *pipeline* em relação ao tempo transcorrido. Após 4 unidades de tempo, o fluxo de tarefas completas (as tarefas que completaram os 4 estágios) torna-se contínuo e uma tarefa termina a cada unidade de tempo, resultando no paralelismo de eventos sobrepostos.

Figura 2.2 - Funcionamento de um *pipeline*

O paralelismo combinado equivale a vários estágios *pipeline* sendo executados em paralelo.

2.2.2 Concorrência e Paralelismo

Apesar dos eventos em um sistema computacional ocorrerem basicamente de forma seqüencial ou concorrente, faz-se uso freqüentemente, muitas vezes de modo errado, do termo “processamento paralelo” ao invés do termo “processamento concorrente”. É útil, portanto, uma elucidação e distinção entre os significados desses dois conceitos:

- Processamento concorrente: num determinado instante, dois ou mais eventos foram iniciados, mas não foram terminados, não havendo necessariamente o uso de mais de um elemento de processamento (há o uso do mecanismo de tempo compartilhado quando há um único elemento de processamento);
- Processamento paralelo: é um caso particular da execução concorrente onde há a ocorrência simultânea de eventos, ou seja, os eventos são executados ao mesmo tempo por elementos de processamento diferentes.

Neste trabalho será utilizado o termo “processamento paralelo” para se referenciar arquiteturas, programação e ferramentas para processamento paralelo, enfatizando-se a natureza do projeto do SPP2. Entretanto, deve-se ressaltar que muitas arquiteturas (inclusive a do SPP2) e ferramentas para processamento paralelo também são utilizadas para processamento concorrente.

2.2.3 Níveis de Paralelismo e Granulosidade

O paralelismo pode ser encontrado em diferentes níveis e está intimamente relacionado ao hardware utilizado. Hwang [HWA93] propõe cinco níveis de paralelismo:

1. Instruções;
2. Iterações (*loops*);
3. Rotinas, subrotinas e corotinas;
4. Subprocessos;
5. Processos independentes.

Quanto mais elementar o nível de paralelismo, mais fina é a granulosidade do processamento, que representa o tamanho da tarefa destinada aos processadores medido em número de instruções executadas [ALM94].

Uma arquitetura de computador explora um nível ou uma combinação dos níveis de paralelismo: máquinas com poucos processadores, frequentemente genéricos, são mais indicadas para executar aplicações com paralelismo em nível de processos ou subprocessos; máquinas com inúmeros processadores, geralmente específicos, são indicadas para aplicações com paralelismo em nível de instruções ou operações; máquinas com processamento vetorial adequam-se a aplicações com paralelismo em nível de iterações.

2.2.4 *Speedup* e Eficiência

Um dos principais motivos para a utilização do processamento paralelo é o aumento do desempenho possível na execução de aplicações paralelas em relação às correspondentes versões sequenciais dessas aplicações. Esse acréscimo de desempenho é geralmente quantificado por duas métricas: *speedup* e eficiência [ALM94, FOS95].

O *speedup* é uma medida utilizada para quantificar o aumento de velocidade obtido na execução de um programa paralelo por p processadores em relação à sua execução sequencial, sendo expresso pela relação:

$$Sp = T_{seq}/T_{par},$$

onde T_{seq} é o tempo de execução do algoritmo paralelo utilizando apenas um processador e T_{par} é o tempo obtido utilizando p processadores. Idealmente, Sp deveria ser igual a p ($Sp = p$), ou seja, o aumento de desempenho deveria ser diretamente proporcional ao número de processadores utilizados na execução do programa paralelo. Entretanto, essa igualdade é dificilmente obtida na prática devido a três fatores limitadores: a sobrecarga de comunicação entre os processadores e tarefas, seções de código que não podem ser paralelizadas e a inadequação do grau de paralelismo apresentado pela aplicação ao grau de paralelismo oferecido pela máquina que a executa [ALM94]. Foster [FOS95] define ainda casos especiais em que o *speedup* sobrepuja o limite teórico p ($Sp > p$), sendo tal ocorrência denominada de anomalia de *speedup*.

A métrica eficiência serve para medir de forma mais acurada a qualidade de um algoritmo paralelo, quantificando a utilização dos processadores através da fração do tempo total de execução em que os processadores executam operações úteis:

$$Ep = Sp/p ,$$

onde Sp é o *speedup* obtido com p processadores. Nota-se que a eficiência na utilização dos processadores tem como limitante inferior o valor 0 (zero) e será tão melhor quanto mais se aproximar de 100% ($Ep = 1$).

2.3 Arquiteturas para Processamento Paralelo

A grande diversidade de arquiteturas de computadores é produto direto das necessidades de aplicações específicas e da tecnologia de elementos computacionais (processadores e redes de interconexão, por exemplo) disponível para a construção de arquiteturas que supram tais necessidades. Observa-se, portanto, que há uma dificuldade considerável em se classificar as arquiteturas de computadores devido à grande dinâmica das necessidades das aplicações e da tecnologia dos elementos computacionais.

Apesar de vários esquemas de classificação de arquiteturas terem sido propostos, o presente trabalho descreverá e adotará as taxonomias de Flynn, a mais utilizada, e a de Duncan, que estende a taxonomia de Flynn para acomodar arquiteturas mais recentes. As seções 2.3.1 e 2.3.2 versam sobre tais taxonomias. A seção 2.3.3 fornece uma descrição mais detalhada de dois tipos de arquitetura muito utilizados para processamento paralelo, com exemplos reais de computadores que apresentam essas arquiteturas. A seção 2.3.4 descreve a arquitetura do SPP2.

2.3.1 Taxonomia de Flynn

A taxonomia mais popular para arquiteturas de computadores foi proposta por Flynn em 1966 e classifica as arquiteturas segundo a multiplicidade do fluxo de dados e de instruções. Esse princípio de classificação determina as quatro seguintes classes de computadores: SISD (*Single Instruction Single Data*), SIMD (*Single Instruction Multiple Data*), MISD (*Multiple Instruction Single Data*) e MIMD (*Multiple Instruction Multiple Data*) [DUN90].

Computadores SISD

Os computadores seqüenciais convencionais, aqueles com uma única unidade de processamento, são classificados como SISD. Cada instrução inicia uma operação em dados obtidos de um único fluxo de dados (figura 2.3). As instruções são executadas seqüencialmente, mas podem ser sobrepostas nos seus estágios de execução através do uso de *pipelines*.

Pode-se citar como exemplos de computadores SISD os seqüenciais com um único processador, tais como a maioria dos modelos pertencentes às famílias Apple e IBM PC.

Figura 2.3 - Computador do tipo SISD

Computadores SIMD

As máquinas SIMD possuem uma única unidade de processamento de instruções, geralmente denominada de unidade de controle, e várias unidades de processamento de dados, denominadas elementos de processamento (figura 2.4).

A unidade de controle é responsável por buscar e interpretar instruções. Quando ela encontra uma instrução referente ao processamento de dados, ela difunde essa instrução para todos os elementos de processamento, que executam a mesma operação sobre um conjunto distinto de dados.

Alguns exemplos de máquinas SIMD são o ILLIAC-IV, o primeiro computador dessa classe, e os computadores mais recentes Thinking Machine CM-2 [HWA93] e MasPar MP [FOS95].

Figura 2.4 - Computador do tipo SIMD

Computadores MISD

Um computador dessa classe (figura 2.5) corresponde, ou corresponderia (vide discussão no parágrafo a seguir), a uma máquina com vários elementos de processamento recebendo instruções distintas das unidades de controle, porém processando o mesmo fluxo de dados. O resultado de um processador é a entrada do próximo.

Alguns autores como Navaux [NAV89], Ducan [DUN90] e Tanenbaum [TAN92] consideram que não existem exemplos físicos de computadores dessa classe. Já Almasi [ALM94] considera como legítimos representantes dessa classe os computadores que utilizam *pipelines* e as arquiteturas sistólicas.

FI - Fluxo de Instruções
UC - Unidade de Controle
UP - Unidade de Processamento
M - Memória

Figura 2.5 - Computador do tipo MISD

Computadores MIMD

A categoria de máquinas MIMD é caracterizada por arquiteturas de computadores que envolvem múltiplos processadores executando de forma autônoma diferentes instruções em conjuntos de dados distintos (figura 2.6).

Essa classe inclui máquinas com processadores e unidades de memória especificamente projetados para serem componentes de uma arquitetura paralela, máquinas paralelas construídas com processadores comerciais, máquinas paralelas, computadores simples como o IBM PC interconectados por uma rede local [FOS95] e sistemas distribuídos [TAN92].

Unidades de processamento independentes buscam instruções e operam sobre diferentes fluxos de dados.

FI - Fluxo de Instruções
 UC - Unidade de Controle
 UP - Unidade de Processamento
 M - Memória

Figura 2.6 - Computador do tipo MIMD

2.3.2 Taxonomia de Duncan

Duncan [DUN90] propôs uma classificação para as arquiteturas de computadores que pudesse acomodar arquiteturas mais recentes que não se encaixam adequadamente na taxonomia de Flynn. Na concepção de sua taxonomia, Duncan seguiu as seguintes diretrizes:

- Excluir arquiteturas que incorporam apenas mecanismos paralelos de baixo nível (*pipelines*, múltiplas unidades funcionais e processadores dedicados de entrada e saída, por exemplo), que se tornaram comuns nos computadores modernos e que não fornecem mecanismos para o desenvolvimento de soluções paralelas de alto nível;
- Manter os elementos da difundida arquitetura de Flynn, baseada em fluxos de instruções e de dados;
- Incluir os processadores vetoriais *pipeline* e outras arquiteturas que eram difíceis de acomodar no esquema de classificação de Flynn.

O resultado do trabalho de Duncan é expresso pela figura 2.7, segundo a qual as arquiteturas são classificadas basicamente em síncronas e assíncronas.

Figura 2.7 - Classificação de Duncan

Arquiteturas Síncronas

Apresentam a característica de coordenação das suas operações através de relógios globais, unidades de controle centralizadas ou controladores de unidades vetoriais.

Arquiteturas Vetoriais *Pipeline*

Apresentam múltiplas unidades funcionais que podem operar concorrentemente utilizando *pipelines*. Além disso, elas implementam operações aritméticas e lógicas para vetores e números escalares. Tais arquiteturas efetuam o processamento vetorial paralelo do seguinte modo: direcionam os elementos do vetor para o *pipeline* de uma unidade funcional e a saída do *pipeline* de uma unidade funcional para o *pipeline* de outra unidade funcional.

Exemplos de computadores que apresentam essa arquitetura são o Cray X-MP [ALM94] e o Fujitsu VP-200 [DUN90].

Arquiteturas SIMD

Essas arquiteturas apresentam geralmente uma unidade de controle central, vários processadores e uma rede de interconexão para comunicação entre processadores e/ou entre processadores e memória. A unidade de controle difunde uma única instrução para todos os processadores, que é executada sobre os dados locais. As arquiteturas dessa classe podem, ainda, ser diferenciadas em processadores matriciais e processadores de memória associativa, dependendo do método utilizado para acesso à memória. Um processador matricial é constituído por vários elementos de processamento que funcionam sincronizados, sob a supervisão de uma unidade de controle

[KIR91]. Um processador de memória associativa é similar a um processador matricial, mas emprega uma lógica especial de comparação para acessar vários dados armazenados na memória simultaneamente [DUN90].

Pode-se citar como exemplos de processador matricial e de processador de memória associativa os computadores Connection Machine e PEPE (Parallel Element Processing Ensemble), respectivamente [DUN90].

Arquiteturas Sistólicas

As arquiteturas sistólicas são caracterizadas por processadores organizados como um *pipeline* multi-dimensional no qual os dados fluem de modo rítmico da memória para a rede de processadores e retornam novamente para a memória.

Alguns exemplos de arquiteturas sistólicas são os computadores Warp CMU/GE da empresa General Electric [ALM94] e Warp da Universidade Carnegie Mellon [DUN90], cujas redes de processadores são programáveis, ou seja, os processadores podem ser organizados segundo uma distribuição adequada para um determinado algoritmo.

Arquiteturas Assíncronas

As arquiteturas assíncronas não possuem um mecanismo centralizado de controle de hardware, como é o caso das arquiteturas síncronas.

Arquiteturas MIMD

Empregam múltiplos processadores que operam de modo autônomo em relação à execução de instruções. Os computadores com arquitetura MIMD podem ser classificados ainda segundo o modo de organização de sua memória: memória compartilhada ou memória distribuída.

A arquitetura MIMD com memória compartilhada, denominada geralmente de multiprocessador [TAN92], é composta basicamente por uma rede de interconexão que liga os processadores à memória centralizada, como mostrado na figura 2.8. A comunicação e a sincronização entre os processadores são feitas através da memória.

$M_i, i \in I, 1 \leq i \leq n$ - Módulos da Memória Compartilhada

$P_j, j \in I, 1 \leq j \leq k$ - Processadores

Figura 2.8 - Arquitetura MIMD com memória compartilhada

Pode-se citar como exemplo de multiprocessadores os computadores Challenge da Silicon Graphics [FOS95] e Cray Y-MP [ALM94].

Na arquitetura MIMD com memória distribuída (figura 2.9), ou multicomputador [TAN92], há uma rede de interconexão que interliga os processadores, que possuem memória local.

$M_i, i \in I, 1 \leq i \leq n$ - Memória do Processador P_i

$P_i, i \in I, 1 \leq i \leq n$ - Processadores

Figura 2.9 - Arquitetura MIMD com memória distribuída

Exemplos de multicomputadores são o SP2 [WOL95] e o Cray T3D [FOS95], das empresas IBM e Cray, respectivamente.

Paradigma MIMD

Agrupar arquiteturas que possuem os princípios da arquitetura MIMD de operação assíncrona e operação concorrente de múltiplos fluxos de dados e instruções. Entretanto, essas arquiteturas também são baseadas em princípios de organização distintos da arquitetura MIMD, dificultando, portanto, a caracterização das mesmas como puramente MIMD.

Nas arquiteturas SIMD/MIMD grupos de processadores são organizados segundo uma arquitetura SIMD e são controlados por outros processadores organizados segundo uma arquitetura MIMD. Um exemplo dessa arquitetura é o computador Texas Array Re-configurable Computer, ou TRAC [DUN90].

O princípio operacional básico da arquitetura a fluxo de dados é a execução de instruções tão logo os dados necessários estejam disponíveis. Exemplifica-se essa arquitetura com o Manchester Data Flow Computer [DUN90].

Em uma arquitetura de redução uma instrução está pronta para ser executada tão logo os seus resultados sejam necessários como operandos para outra instrução também pronta para execução. Exemplos de computadores com essa arquitetura são o Newcastle Reduction Machine e o Utah Applicative Multiprocessing System [DUN90].

As arquiteturas de frente de onda possuem organização similar à das arquiteturas sistólicas, mas não possuem um relógio global que cadencia a execução das instruções. Quando um processador executa uma instrução e está pronto para passar os dados para o seu sucessor, ele informa o sucessor. Após o sucessor indicar que está apto para recebê-los, o processador envia-os e recebe um reconhecimento do sucessor ao envio dos dados. Cita-se como exemplo de computador com arquitetura frente de onda o computador Wavefront Array da Universidade Johns Hopkins [DUN90].

2.3.3 Tópicos Adicionais sobre as Arquiteturas MIMD com Memória Compartilhada e Distribuída

Conforme discutido nas seções 2.3.1 e 2.3.2, as arquiteturas MIMD empregam vários processadores que podem executar fluxos de instruções distintos sobre dados locais (vide figuras 2.6, 2.8 e 2.9). Essas arquiteturas suportam aplicações paralelas com alto grau de autonomia entre processos, caracterizando uma granulosidade grossa. Os processos de software em execução em uma arquitetura MIMD necessitam de sincronização e interação, através da troca de

mensagens em uma rede de interconexão ou acesso a uma memória compartilhada. Tais formas de sincronização e interação são feitas de modo assíncrono [DUN90], ou seja, todas as operações de sincronização e comunicação entre os processadores não obedecem a um ritmo ou cadência ditados por um mecanismo de controle de hardware centralizado. Em vez disso, elas são realizadas conforme a necessidade dos processos e a disponibilidade de elementos de hardware (um canal de comunicação livre entre dois processadores, por exemplo).

O ímpeto no desenvolvimento e utilização de arquiteturas MIMD pode ser atribuído a diversos fatores inter-relacionados. As máquinas MIMD permitem um paralelismo de alto nível (nível de subprogramas e tarefas), que pode ser explorado por algoritmos construídos segundo a técnica de “dividir para conquistar” (técnica segundo a qual divide-se, sucessivamente, um problema inicial em problemas menores até que se consiga resolvê-los, e assim compor a resolução do problema inicial). Os computadores MIMD são aplicáveis a computações de propósito geral, ao contrário de computadores SIMD e MISD, que são utilizados em aplicações específicas. Além disso, a relação custo/benefício de computadores MIMD é melhor que a exibida por computadores SIMD por dois motivos: primeiro, o custo de n processadores é menor que o custo de um único processador com poder computacional equivalente ou próximo ao desses n processadores; segundo, pode-se expandir, conforme a necessidade e a disponibilidade de recursos, uma arquitetura MIMD adicionando-se a mesma mais processadores. Isso não é possível em arquiteturas SIMD, a menos que se troque o processador por um mais potente, o que pode ser economicamente inviável [HWA93, DUN90].

Arquiteturas MIMD com Memória Compartilhada

Em um computador MIMD com memória compartilhada (figura 2.8), ou multiprocessador, todos os processadores compartilham uma memória comum e também periféricos (discos rígidos, por exemplo) [HWA93]. A coordenação de eventos paralelos, bem como a sincronização e a comunicação entre os processadores, é feita utilizando variáveis compartilhadas na memória compartilhada (vide seção 2.4.2).

O arranjo básico descrito anteriormente pode ser incrementado com o uso de uma memória *cache* em cada processador (figura 2.10), diminuindo-se o tempo de acesso à memória compartilhada e, conseqüentemente, o tempo médio de acesso à memória (o tempo de acesso a uma memória *cache* local a um processador é muito menor que o tempo de acesso a uma memória comum e compartilhada). Entretanto, a utilização de memória *cache* exige, necessariamente, o uso de mecanismos de hardware ou software [FOS95] para manter a sua

consistência. Sempre que um processador atualizar um dado em sua memória *cache*, o dado associado na memória compartilhada e nas memórias *cache* dos outros processadores deve ser atualizado.

Figura 2.10 - Arquitetura MIMD com memória compartilhada e que utiliza memória *cache*

A conexão entre os processadores e a memória compartilhada é feita por uma rede de interconexão, que pode apresentar uma das seguintes topologias:

- **Barramento:** todos os processadores e a memória compartilhada são conectados a um barramento comum (figura 2.11 (a)). Para ler uma palavra da memória, por exemplo, o elemento de processamento, após deter o direito de uso do barramento, coloca o endereço da palavra que ele deseja nas linhas de endereço do barramento e informa através das linhas de controle que se trata de uma leitura. A memória responde colocando o valor da palavra nas linhas de dados, que poderá ser lido pelo elemento de processamento que o requisitou. O funcionamento da escrita é similar, embora exija um mecanismo adicional para controlar a coerência da memória *cache*, se ela for utilizada pela arquitetura.

A principal vantagem do barramento em relação às outras topologias é o custo. A principal limitação está no número de processadores que podem ser utilizados. Com muitos processadores o barramento fica sobrecarregado, tornando-se

um gargalo para o sistema;

- *Crossbar*: utiliza uma malha composta por n^2 pontos de intersecção (*crosspoints*), que podem estar abertos ou fechados, para interligar n processadores a n módulos de memória (figura 2.11 (b)). Quando um processador quer ter acesso a um determinado módulo de memória, o ponto de intersecção que os conecta é momentaneamente fechado, permitindo que o acesso ocorra.

A interconexão por rede *crossbar* permite, em relação a por barramento, mais de um processador comunicando-se com módulos distintos de memória simultaneamente. Entretanto, essa vantagem implica uma desvantagem. O custo da rede *crossbar* é proporcional ao quadrado do número de processadores (n^2 pontos de intersecção), tornando-se proibitivo se o número de processadores for elevado;

- Multi-estágio: os processadores são interligados aos módulos de memória através de múltiplos estágios de chaveamento, compostos por várias chaves de conexão (*switches*) (figura 2.11 (c)). Quando o número N de processadores é uma potência de dois, utiliza-se geralmente $\log_2 N$ estágios com $N/2$ chaves cada um. Para acessar um módulo de memória, especificado por uma certa sequência de *bits*, um processador submete essa sequência à rede multi-estágio. Ao atravessar a rede multi-estágio, cada *bit* será responsável por uma decisão de roteamento em cada estágio. O roteamento de dados em cada estágio é feito por uma chave de conexão pertencente ao estágio. Esse roteamento permite a transferência de dados entre os elementos computacionais (chaves de conexão, processadores ou módulos de memória) que estão ligados à chave.

A interligação por multi-estágio constitui um compromisso entre as alternativas de preço/desempenho oferecidas pelas redes *crossbar* e pelo barramento por dois motivos. Primeiro, embora o preço seja superior ao de uma rede por barramento, ele é muito inferior ao de uma rede *crossbar* ($N/2 \times \log_2 N$ têm um preço menor que N^2 chaves de conexão). Segundo, o desempenho é superior ao do barramento por permitir que mais de um processador acesse módulos distintos de memória ao mesmo tempo, embora seja inferior ao de uma rede *crossbar*, pois essa permite um número maior de conexões simultâneas entre os processadores e os módulos de memória.

$M_i, i \in I, 1 \leq i \leq 4$ - Módulos da Memória Compartilhada

$P_j, j \in I, 1 \leq j \leq 4$ - Processadores

C - Elementos de Chaveamento

Figura 2.11 - Redes de interconexão para memória compartilhada

Arquiteturas MIMD com Memória Distribuída

A arquitetura MIMD com memória distribuída (figura 2.9), ou multicomputador, é constituída de múltiplos computadores, frequentemente denominados nós de processamento, interconectados por uma rede de passagem de mensagens, denominada rede de interconexão. Cada nó de processamento, ou simplesmente nó, é um computador autônomo composto por um processador, memória local e, algumas vezes, por discos ou periféricos de entrada/saída [HWA93]. Todas as memórias locais são privativas, significando que somente os processadores locais às mesmas podem acessá-las.

A rede de passagem de mensagens é o meio que os nós utilizam para enviar mensagens a

outros nós, possibilitando a comunicação e a sincronização de processos.

As principais topologias para as redes de passagem de mensagens são:

- Barramento: essa topologia é similar à da rede de interconexão de mesmo nome para memória compartilhada. A única diferença é que somente os nós estão ligados ao barramento (figura 2.12 (a)). Devido à comunicação entre os nós de um multicomputador ser menor que a comunicação entre memória e processadores em um multiprocessador, a topologia em barramento pode suportar, para uma mesma capacidade de transmissão do barramento, um número maior de elementos de processamento em multicomputadores que em multiprocessadores. Ou, sob a perspectiva dos elementos de processamento, a capacidade de transmissão do barramento pode ser, para o mesmo número de elementos de processamento, menor em um multicomputador que em um multiprocessador;
- Anel: cada nó em um anel é conectado a dois e somente dois outros nós vizinhos (figura 2.12 (b)). O caminho que uma mensagem entre dois nós quaisquer deve percorrer é, na média, igual a $N/2$, onde N é o número de processadores. Pode-se incrementar esse arranjo utilizando-se conexões extras entre nós, que proporcionam uma maior tolerância a falhas e um caminho menor entre dois nós, o que demanda um tempo menor de comunicação entre nós. A topologia em anel é a mais apropriada para um pequeno número de processadores executando algoritmos que não exijam muita comunicação [DUN90];
- Malha: os nós interconectados por uma rede em malha possuem quatro nós vizinhos, exceto os das extremidades, que têm apenas dois ou três nós vizinhos (figura 2.12 (c)). A comunicação entre os nós pode ser aumentada utilizando-se ligações entre os nós das extremidades ou barramentos para conectar os nós por linhas ou por colunas. O uso de uma rede em malha é muito apropriado para algoritmos que utilizam matrizes;
- Árvore: os nós estão conectados segundo uma árvore (figura 2.12 (d)). Todos os nós, menos os nós folhas, possuem ao menos dois nós vizinhos. O caminho de uma mensagem entre dois nós quaisquer pode ser encurtado utilizando-se conexões auxiliares entre os nós, como, por exemplo, conectando-se todos os nós pertencentes a um mesmo nível da árvore. A topologia em árvore é aconselhável para algoritmos

de busca e ordenação, “dividir para conquistar” e paradigmas de redução e de fluxo de dados [DUN90];

- **Hipercubo:** os $N = 2^n$ nós do multicomputador estão organizados segundo um cubo de dimensão n , onde cada nó possui $n = \log_2 N$ ligações com nós adjacentes (figura 2.12 (e)). A topologia em hipercubo possui grande aplicação em multicomputadores que devem executar aplicações científicas tridimensionais ou de dimensões espaciais superiores;
- **Reconfigurável:** uma rede de interconexão reconfigurável (figura 2.12 (f)) pode ser modificada, através de chaves (*switches*) de programação, para apresentar uma topologia que se adeque aos padrões de comunicação da aplicação do usuário.

Figura 2.12 - Redes de interconexão para memória distribuída

2.3.4 A Arquitetura do SPP2

A arquitetura do SPP2 pode ser classificada, segundo a taxonomia de Duncan [DUN90], como assíncrona, MIMD e com memória distribuída. Uma das principais características apresentadas por essa máquina é o emprego de partes componentes de computadores pessoais na sua construção, seguindo a tendência da maioria das arquiteturas MIMD com memória distribuída disponíveis no mercado [CRA95, ISP93]. A figura 2.13 apresenta uma visão geral da arquitetura do SPP2:

Figura 2.13: A arquitetura do SPP2

O SPP2 suporta até 256 nós de processamento, ou simplesmente nós, distribuídos em módulos de 8 nós de processamento cada. Cada nó é uma placa (*motherboard*) de computador pessoal contendo um ou mais processadores, memória e interfaces de comunicação. Em um computador SPP2 com N nós ($0 < N \leq 256$), os nós são nomeados como P_i , onde i pode variar de 0 a $N-1$. A interligação dos nós é feita por dois tipos de rede de comunicação. Uma rede Ethernet (rede de interconexão em barramento) proporciona o meio de comunicação necessário para cada nó carregar de forma remota o sistema operacional compatível com Unix, o Linux [WEL94]. Essa rede é utilizada também para o controle do sistema e para a submissão de tarefas (processos), sendo denominada rede de gerenciamento. Uma rede de comunicação de alta velocidade formada por um conjunto de *switches* e adaptadores Myrinet [BOD95] provê a comunicação entre os nós de processamento durante a execução de aplicações paralelas. A

submissão de tarefas componentes de aplicações paralelas e o gerenciamento de todo o sistema são realizados através de um computador pessoal, o *host* na figura 2.13.

Neste trabalho são utilizados os termos tarefas e processos indistintamente. Entretanto, quando se referenciar especificamente ao SPP2, somente o termo tarefas será utilizado.

2.4 Programação Paralela

A arquitetura de computador proposta por von Neumann baseia-se em um processador executando seqüências de instruções [FOS95]. Uma instrução pode especificar, adicionalmente às várias operações aritméticas, o endereço dos dados a serem lidos ou escritos na memória ou o endereço da próxima instrução a ser executada. A programação dessa arquitetura para realizar uma determinada tarefa é feita por uma linguagem de programação seqüencial, que pode ser de baixo nível (linguagem de máquina e *assembly*) ou ainda de nível mais elevado, tal como C e Pascal.

A programação paralela requer, além das construções normalmente disponíveis em uma linguagem de programação seqüencial, outras para definir a execução paralela de processos, iniciar e interromper sua execução e coordenar e especificar a interação entre eles [ALM94]. Essas construções podem ser agrupadas segundo seu princípio de funcionamento e adequação a um determinado nível de paralelismo. As construções básicas para essas finalidades são apresentadas na seção 2.4.1.

No que tange à cooperação entre os diversos processos componentes de uma tarefa, pode-se recorrer a construções que possibilitem a comunicação entre os processos e a sincronização de suas atividades. Tais construções são detalhadas na seção 2.4.2.

2.4.1 Ativação e Término de Processos Paralelos

A ativação e o término de processos paralelos podem ser estipulados por diferentes construções, que diferem entre si pela adequação de uso, estruturação e flexibilidade [ALM94].

Corotinas

As corotinas são semelhantes às subrotinas de processos comuns, diferindo apenas no modo de transferência de controle, que é não hierárquico: sempre que são ativadas (figura 2.14), as corotinas são executadas a partir do ponto onde foi feita a última chamada a *resume*, comando

utilizado para ativar uma corotina.

A construção corotina é adequada para a organização de programas paralelos que compartilham um único elemento de processamento (programas concorrentes). Nessa situação, cada corotina abstrai um processo, e as várias corotinas são executadas intercaladamente, havendo apenas uma corotina ativa em cada instante.

Figura 2.14 - Uso de corotinas

Fork/Join

O comando *fork* inicia a execução de um determinado conjunto de instruções (processo filho) paralelamente com outros processos em execução, incluindo o processo que o executou (processo pai). O comando *join* é utilizado para a sincronização dos processos ativados. As sintaxes dos comandos *fork/join*, ilustrados na figura 2.15, são:

- **Fork end:** executa o processo cuja primeira instrução está no endereço *end* paralelamente com os demais processos em execução;
- **Join num, end1, end2:** decrementa a variável *num*, que contém o número de processos que devem sincronizar. Se *num* = 0, desvia o fluxo de execução para o endereço *end1*. Caso contrário, desvia para *end2* (geralmente um comando *quit*, que termina a execução).

A construção *fork/join* constitui um mecanismo extremamente poderoso e flexível para a especificação de qualquer forma de processamento paralelo. Entretanto, a organização do código fonte em que ela aparece é desestruturada e suscetível a erros, pois a sua sintaxe permite saltos para qualquer trecho do código. Exige-se, portanto, que o uso dessa construção seja feito de modo disciplinado e sistemático.

Figura 2.15 - Exemplo da utilização de *fork/join*

Cobegin/Coend

A construção *Cobegin/Coend*, também denominada de *Parbegin/Parend*, permite a execução paralela das instruções ou processos delimitados pelos comandos *Cobegin* e *Coend* (figura 2.16), constituindo-se uma maneira altamente estruturada para a ativação de um conjunto de instruções que devem ser executadas paralelamente. Entretanto, devido à necessidade de término de todas as instruções ou processos que foram ativados antes que se possa especificar a ativação de novos processos, essa construção é menos flexível que a construção *Fork/Join*.

Figura 2.16 - Exemplo da utilização de *cobegin/coend*

Doall

Essa construção e seus clones, tais como *forall* e *doacross*, apresentam um princípio de funcionamento similar ao da *Cobegin/Coend*, onde as instruções executadas em paralelo são as diversas instâncias de um bloco de comandos dentro de um comando de repetição (*loop*). A figura 2.17 ilustra o seu funcionamento.

Figura 2.17 - Exemplo da utilização de *doall*

2.4.2 Comunicação e Sincronização de Processos Paralelos

O esforço de cooperação entre processos que interagem para a resolução de uma certa tarefa somente pode ser obtido mediante a troca de informações pelos mesmos e a sincronização de suas atividades, assim como ocorre na construção de uma casa: o servente de pedreiro (processo) prepara a argamassa (informação) e a envia (comunicação) para o pedreiro (processo), que então inicia o assentamento dos tijolos (o pedreiro não pode começar a assentar os tijolos antes que receba a argamassa, devendo haver uma sincronização entre as suas atividades e as do servente de pedreiro).

A comunicação, ou troca de informações, entre processos possibilita a permuta dos dados necessários às suas atividades. A sincronização, como decorrência da existência da comunicação, possibilita a coordenação das atividades dos processos (controle de seqüência) e o controle dos acessos a recursos compartilhados pelos mesmos (controle de acesso), a fim de se manter a consistência desses recursos [ALM94].

A comunicação e a sincronização de processos podem ser feitas basicamente por mecanismos de passagem de mensagens ou por variáveis compartilhadas, havendo uma maior ou menor adequação de um ou outro mecanismo dependendo do arranjo da memória (compartilhada ou distribuída) do computador em que os processos estão sendo executados.

Comunicação e Sincronização em Memória Compartilhada

O uso de variáveis compartilhadas pelos diversos processos paralelos que cooperam na execução de uma tarefa é o modo de se obter sincronização e comunicação entre processos em computadores com memória compartilhada.

O controle de acesso é implementado através de exclusão mútua de regiões críticas, trechos de código que devem ser executados indivisivelmente por um processo sem que outros também o executem concorrentemente.

A implementação de exclusão mútua e controle de seqüência através de variáveis compartilhadas pode ser feita por vários métodos, dentre os quais destacam-se os seguintes:

- *Busy-waiting*: através da execução contínua de uma primitiva indivisível que manipula (testa e modifica - *test-and-set*) o valor de uma variável compartilhada. Um processo pode obter acesso à região crítica controlada assim que a primitiva conseguir modificar a variável compartilhada, após gastar ciclos de CPU testando-a,

ou seja, realizando uma espera ocupada (*busy waiting*).

O método *Busy-waiting* possui duas grandes desvantagens: gasto supérfluo de tempo de CPU e dificuldade de entendimento, depuração e validação de programas, visto que a primitiva de manipulação supra citada é implementada em baixo nível;

- Semáforos: um semáforo é uma variável compartilhada inteira e não negativa que controla o acesso a uma região crítica. Duas operações atômicas (indivisíveis): **p** e **v**, também chamadas de *down* e *up* são utilizadas para o controle do semáforo. As semânticas de operação em relação a um semáforo *s* estão descritas a seguir.

<p>p(s): se ($s = 0$) então { bloqueia-se o processo até que $s > 0$ $s = s - 1$ } senão $s = s - 1$</p>	<p>v(s): $s = s + 1$</p>
---	--

Os semáforos, além de evitar o desperdício de CPU, constituem um meio de sincronização de nível mais elevado do que o *busy-waiting*;

- Monitores: um monitor é um conjunto de procedimentos, variáveis e estruturas de dados agrupados em um módulo de programação. As variáveis e estruturas de dados representam o estado de algum recurso compartilhado enquanto que os procedimentos implementam operações sobre esses recursos. Somente os procedimentos internos a cada monitor podem manipular suas variáveis e estruturas de dados.

Devido à estruturação dos monitores, esses oferecem um meio de se implementar exclusão mútua menos propenso a erros que os semáforos, que são desestruturados.

Comunicação e Sincronização em Memória Distribuída

A comunicação e a sincronização em arquiteturas de memória distribuída são obtidas pela passagem de mensagens entre os processos através de canais de comunicação.

Uma operação de comunicação através de passagem de mensagens é feita, de modo genérico, pela utilização das primitivas *send/receive* (transmita/receba), cujas sintaxes são:

- *Send* mensagem *to* processo_destino (*Transmita* mensagem *para* processo_destino);
- *Receive* mensagem *from* processo_fonte (*Receba* mensagem *de* processo_fonte).

A forma mais simples de comunicação através de passagem de mensagens é a comunicação ponto-a-ponto. Uma mensagem é transmitida do processo fonte para um processo receptor.

Há diversas variações no modo de interação entre a transmissão de uma mensagem e a execução do processo [MPI95]. A primeira diferença comum está entre transmissão (*send*) síncrona e assíncrona.

As transmissões síncronas retornam informações sobre a finalização da operação, como ilustrado pelo aviso de recebimento de um fax na terceira situação da figura 2.18.

Figura 2.18 - Funcionamento de uma comunicação síncrona

As transmissões assíncronas somente se preocupam se a mensagem foi enviada. Na figura 2.19, o remetente (processo que executou o *send*) envia a sua correspondência (dados) e vai fazer outras operações, não se importando com o recebimento da correspondência (dados).

Figura 2.19 - Funcionamento de uma comunicação assíncrona

Uma mensagem de fax ou carta registrada é uma operação síncrona. O remetente pode determinar se a mensagem foi transmitida.

Um cartão postal é uma mensagem assíncrona. O remetente somente sabe que ele foi colocado dentro da caixa postal, mas não tem nenhuma idéia se ele chegou, a menos que o destinatário envie uma resposta.

A outra diferença importante é operação bloqueante e operação não bloqueante. Operações bloqueantes somente retornam da chamada da subrotina associada (*send* ou *receive*) quando a operação for completada (figura 2.18). Um *send* bloqueante pode ser completado pela simples cópia dos dados a serem transmitidos em uma área (*buffer*) de transmissão para transmissão posterior pelo sistema operacional. Um *receive* bloqueante é completado assim que ele recebe os dados transmitidos por um processo.

Operações não bloqueantes retornam imediatamente e permitem que o processo continue a executar outras operações. Algum tempo mais tarde o processo pode verificar se a operação não bloqueante foi completada.

Figura 2.20 - A comunicação não bloqueante

Aparelhos comuns de fax fornecem comunicação bloqueante. O fax mantém-se ocupado até que a mensagem tenha sido enviada. Alguns aparelhos mais modernos de fax possuem uma memória. Isso permite carregar um documento na memória e, se o aparelho remoto estiver ocupado, realizar outra atividade até que se possa enviar o documento com sucesso. Essa é uma operação não bloqueante.

O recebimento de uma mensagem também pode ser uma operação não bloqueante. É possível, por exemplo, ligar um aparelho de fax e deixá-lo em *standby* (estado de espera), para que alguma mensagem possa ser recebida. Observando-se periodicamente o aparelho, pode-se determinar se uma mensagem chegou.

As primitivas básicas *send/receive* podem gerar mecanismos mais poderosos de comunicação [ALM94], dentre os quais destacam-se os seguintes:

- *Rendezvous*: utiliza um par de primitivas *send/receive* bloqueantes organizadas de modo a fornecer uma comunicação bidirecional e permitir que um processo inicie a execução de um trecho de código em outro processo (figura 2.21).
- *RPC (Remote Procedure Call - Chamada a Procedimento Remoto)*: utiliza uma sintaxe semelhante à chamada de procedimentos locais para chamar procedimentos remotos a um processo: assim que um processo realiza uma chamada a um procedimento remoto ele é bloqueado até que os resultados do procedimento invocado retornem (figura 2.22).

Figura 2.21 - Mecanismo de comunicação *rendezvous*

Figura 2.22 - O mecanismo RPC

2.5 Ferramentas para Desenvolvimento de Aplicações Paralelas

O desenvolvimento de aplicações paralelas não é uma tarefa simples. A complexidade da programação paralela aliada à busca por maior desempenho na execução das aplicações requer o uso de diferentes tipos de ferramentas. Essas ferramentas destinam-se às seguintes atividades: programação (seção 2.5.1), auxílio à paralelização (seção 2.5.2), depuração (seção 2.5.3) e análise de desempenho (Seção 2.5.4).

2.5.1 Ferramentas de Programação

A implementação de um programa paralelo requer, como visto na seção 2.4, mecanismos que permitam a definição, ativação, interrupção e coordenação da execução de tarefas paralelas, além do particionamento e distribuição de dados. Tais mecanismos diferem quanto à granulosidade do paralelismo explorado e à arquitetura utilizada. As ferramentas que oferecem esses mecanismos apresentam diferentes relações entre o esforço do programador e o desempenho obtido pelo programa implementado, podendo ser classificadas em: compiladores e pré-processadores paralelizadores, extensões paralelas para linguagens imperativas e linguagens paralelas [ALM94].

Os compiladores e pré-processadores paralelizadores (Paráfrase [ALM94] para código Fortran, por exemplo) produzem um programa paralelo a partir de um programa serial. Esse tipo de ferramenta requer um esforço mínimo do programador, uma vez que a conversão do código fonte para sua versão paralela é automática. Entretanto, tais compiladores e pré-processadores possuem duas limitações: o desempenho na execução do código obtido com o seu uso é modesto e eles nem sempre estão disponíveis, principalmente para arquiteturas de memória distribuída.

As extensões paralelas para as linguagens imperativas são bibliotecas que complementam as linguagens convencionais com os mecanismos para programação paralela. O uso dessas extensões, se comparado ao dos compiladores, fornece geralmente um desempenho maior, mas exigem um pequeno esforço adicional do programador. Tal esforço concentra-se basicamente no aprendizado das rotinas componentes da extensão paralela da linguagem serial. Pode-se citar como exemplo dessas extensões o Pascal Concorrente, para ambientes de memória compartilhada, o MPI (Message Passing Interface) [MPI95] e o PVM (Parallel Virtual Machine) [GEI94], para memória distribuída. O MPI e o PVM são ambientes para programação por

passagem de mensagens.

As linguagens paralelas podem ser classificadas em dois paradigmas de programação. No paradigma imperativo as linguagens destinam-se a um determinado modelo computacional e possuem comandos relacionados com os mecanismos de programação paralela (linguagem Occam para arquiteturas que utilizam processadores Transputer, por exemplo). No paradigma não imperativo as linguagens não são destinadas a um modelo computacional específico e caracterizam-se por expressar *o que* fazer e não *como* fazer (linguagem Haskell, por exemplo). As linguagens paralelas exigem um grande esforço do programador, se comparado ao exigido pelos outros dois tipos de ferramentas descritos anteriormente, pois ele deverá aprender uma linguagem ou um paradigma de programação completamente novos. Apesar de tal esforço, o uso de linguagens paralelas quase sempre produz programas com código bem estruturado e com desempenho melhor que o obtido com a utilização de compiladores e extensões paralelas.

2.5.2 Ferramentas de Auxílio à Paralelização

A paralelização automática de programas sequenciais é um processo complexo, envolvendo, entre outras tarefas, intensa análise do código original (investigação de dependência de dados e de controle, por exemplo). Entretanto, apesar de tal complexidade, nem sempre os resultados da paralelização automática são satisfatórios. Por exemplo, muitas vezes é preferível não paralelizar um trecho de código porque o tempo de carga em um processador remoto pode ser maior que o tempo gasto na execução sequencial.

Uma abordagem mais apropriada à paralelização de programas envolve uma combinação de intuição humana, análise automática e transformação de código. A intuição humana, auxiliada pelas ferramentas de análise automática de código, fornece a heurística necessária para uma boa paralelização do código [KEN92].

Alguns exemplos de ferramentas de auxílio à paralelização são as ferramentas ParaScope [CAL88], IDA [MER94] e *p2d* [PER96], voltadas para códigos fonte escritos em Fortran.

2.5.3 Ferramentas de Depuração

As ferramentas de depuração destinam-se à determinação e correção de erros nos programas em desenvolvimento. A depuração de programas paralelos apresenta algumas dificuldades, dentre as quais cita-se a não repetição de resultados (programas incorretos podem apresentar resultados

não determinísticos). Essas ferramentas podem ser agrupadas, segundo McDowell [DOW89], em duas categorias:

- Ferramentas interativas: permitem que a execução do programa seja feita passo a passo, isto é, suspensão periodicamente para que as variáveis possam ser inspecionadas ou até mesmo modificadas;
- Ferramentas *post-mortem*: são utilizadas após o término da execução do programa, geralmente após a ocorrência de um erro. A utilização delas permite observar apenas os efeitos finais do erro, tal como o valor de uma determinada variável.

2.5.4 Ferramentas de Análise de Desempenho

Essas ferramentas fornecem informações sobre a execução de programas paralelos. Tais informações podem ser analisadas e utilizadas para a modificação do código de um programa paralelo com o objetivo de aumentar a velocidade de sua execução. Citam-se como exemplos de ferramentas de análise de desempenho o XPVM [GEI94, KOH97], o XMPI [XMP97] e o VT (Visualization Tool) da IBM [PEA95]. O XPVM, o XMPI e o VT são descritos nos próximos tópicos.

XPVM e XMPI

O XPVM e o XMPI fornecem uma interface gráfica para os ambientes de passagem de mensagens PVM e MPI (seção 2.5.1), respectivamente. Dentre outras facilidades, essas ferramentas realizam a animação de um programa paralelo escrito em PVM ou MPI durante (visualização da execução) ou após a sua execução (simulação da execução), mostrando o tempo de execução de cada tarefa e o de comunicação entre tarefas.

A figura 2.23 apresenta uma sessão do XMPI onde é visualizada a execução de quatro tarefas componentes de uma aplicação paralela, representadas por barras horizontais coloridas. As cores representam o tempo que a tarefa gastou efetuando operações efetivas, o tempo que ela esteve esperando por uma mensagem ou o tempo que ela esteve envolvida com comunicação. As linhas entre as barras indicam a ocorrência de uma comunicação entre as tarefas.

Figura 2.23 - Visualização da execução de uma aplicação paralela no XMPI

A figura 2.24 apresenta uma sessão do XPVM onde é visualizada a execução de nove tarefas componentes de uma aplicação paralela por diferentes elementos de processamento. As tarefas são representadas por barras horizontais coloridas. As diferenças de coloração nas barras indicam, assim como no XMPI, o tempo que uma tarefa gastou efetuando computação efetiva, comunicação e espera por uma mensagem. As linhas mostram as mensagens trocadas entre as tarefas.

Figura 2.24 - Visualização da execução de uma aplicação paralela no XPVM

O XPVM permite também o monitoramento dos elementos de processamento que estão executando uma aplicação paralela (figura 2.25). Este monitoramento informa, através de diferentes cores, se um elemento está realizando processamento efetivo (Active), se está executando rotinas do PVM (System) ou se não está executando tarefa alguma (No Tasks).

Figura 2.25 - Monitoramento da execução de tarefas PVM por elementos de processamento

VT

O VT é uma das ferramentas componentes do ambiente paralelo PE (Parallel Environment) da IBM para máquinas paralelas RISC System6000 Scalable POWERparallel (SP) ou redes de máquinas RISC System6000. O PE oferece um conjunto de ferramentas para o desenvolvimento de aplicações paralelas e um ambiente para a execução dessas aplicações. Basicamente, o VT é um ambiente gráfico composto por um grupo de telas de informação, denominadas visões, que são utilizadas para a simulação da execução de um programa e para o monitoramento do desempenho do sistema. Essas visões são apresentadas em diversos formatos (gráfico de barra, de torta ou de malha, por exemplo), procurando facilitar a interpretação dos dados, que são apresentados conforme a sua natureza. A figura 2.26 apresenta a interface do VT contendo algumas das visões, divididas em categorias.

Figura 2.26 - Interface de visões do VT

As visões do VT fornecem, de acordo com a categoria a que pertencem, as seguintes informações:

- Communication/Program (Comunicação/Programa): informações sobre a passagem de mensagens entre os nós de processamento durante a execução de um programa em particular. Também mostram o código fonte durante a simulação da execução. As visões dessa categoria são empregadas apenas para a simulação da execução;
- Computation (Computação): dados sobre a utilização dos nós de processamento que estão executando um determinado programa. Essas visões podem ser utilizadas tanto para a simulação da execução de programas quanto para o monitoramento de desempenho;
- Disk (Disco): informações sobre o número de transferências em disco. As visões

- dessa categoria destinam-se à simulação da execução de programas e ao monitoramento de desempenho;
- Network (Rede): dados sobre o número de pacotes TCP/IP enviados ou recebidos pelos nós. Essas visões podem ser utilizadas para a simulação da execução de programas e para o monitoramento de desempenho;
 - System (Sistema): informações das atividades do sistema, tais como ausência de páginas na memória de um nó e chaveamento de contexto (interrupção da execução de um processo para que outro seja executado). Esse grupo de visões pode ser utilizado tanto para a simulação da execução de programas quanto para o monitoramento de desempenho.

No que tange especificamente à análise de desempenho de programas paralelos, assunto desta seção, o VT utiliza a simulação da execução do programa paralelo para essa tarefa. O monitoramento de desempenho é assunto da próxima seção.

A simulação da execução de um programa paralelo pelo VT é feita seguindo-se a seguinte seqüência de etapas:

1. Executar a ferramenta VT, que apresenta a interface mostrada na figura 2.27;
2. Informar ao VT o nome do arquivo contendo os registros (informações) de execução do programa cuja atividade se quer monitorar (campo Trace File da figura 2.27). Esse arquivo é criado configurando-se uma opção de compilação do programa a ser monitorado;
3. Selecionar na interface de visões do VT (figura 2.26) as visões necessárias. Pode-se selecionar, por exemplo, a visão Interprocessor Communication (Comunicação entre Processadores), que informa o tipo e a duração dos eventos de comunicação;
4. Iniciar a simulação da execução do programa paralelo selecionando o botão Play na interface do VT. Nesse ponto é apresentada a visão mostrada na figura 2.28.

Cada barra da visão Interprocessor Communication representa um nó onde o programa paralelo esteve em execução e a escala horizontal representa um intervalo de tempo. O rótulo à direita de cada barra designa um nó e é colorido de acordo com o estado do nó durante a execução. As barras possuem blocos que representam um evento de comunicação envolvendo o nó ao qual elas estão associadas. O tamanho do bloco representa a duração do evento e a cor o

tipo de evento, que pode ser, por exemplo, um *send* ou um *receive* bloqueante ou não bloqueante (vide Comunicação e Sincronização em Memória Distribuída na Seção 2.4.2). Quando mensagens são trocadas entre nós, uma linha é traçada entre as barras associadas aos nós que estão se comunicando. O início de uma dessas linhas corresponde ao início do evento que iniciou a comunicação. O fim corresponde ao evento que completou a comunicação. Exemplificando, uma linha pode ser traçada do início de um *send* bloqueante até o fim do *receive* bloqueante associado a esse *send*.

É interessante observar que a simulação e a visualização da execução de um programa paralelo no VT, no XMPI e no XPVM são similares à operação de um gravador de fita cassete ou de um aparelho de disco laser, apresentando botões relacionados ao avanço e retrocesso da visualização ou simulação.

Figura 2.27 - Interface do VT

Figura 2.28 - Visão Interprocessor Communication do VT

2.6 Ferramentas para Monitoramento e Gerenciamento de Máquinas Paralelas

Essas ferramentas destinam-se a supervisionar o funcionamento de máquinas paralelas, apresentando funções de monitoramento e gerenciamento, dentre as quais citam-se as seguintes:

- Verificação do estado de operação de cada elemento de processamento, obtendo, por exemplo, a sua temperatura (algumas máquinas devem ter a temperatura de seus elementos de processamento controlada) e a sua taxa de ocupação com a execução de determinadas tarefas dos usuários;
- Monitoramento da rede de interconexão dos elementos de processamento para verificar a carga de trabalho e possíveis falhas;
- Interação com os elementos de processamento permitindo que esses possam ser iniciados ou ter as variáveis do sistema operacional modificadas.

Citam-se como exemplos de ferramentas de monitoramento e gerenciamento de máquinas paralelas o módulo monitor do Victor [ALM94], o VT (seção 2.5.4) e os programas AIX PSSP (AIX Parallel System Support Programs) executados nas máquinas paralelas SP2 (Scalable POWERparallel 2) da IBM [PSS96]. Especificamente em relação ao monitoramento e gerenciamento de máquinas paralelas formadas por redes de estações Linux, como é o caso do SPP2, podem-se citar o programa bWatch [BEO98, BWA98, RAD98] e o conjunto de programas Procps-Pare [BEO98, RAD98, PRO98]. Os próximos tópicos descrevem cada uma dessas ferramentas.

Módulo Monitor do Victor

O módulo monitor do Victor destina-se a monitorar o funcionamento dessa máquina paralela, que apresenta uma arquitetura com memória distribuída organizada segundo uma malha de processadores Inmos Transputer T800. Esse módulo monitor oferece informações tais como o estado de cada processador e a atividade de cada canal de comunicação entre os processadores.

VT

O VT, conforme visto na seção 2.5.4, apresenta como funções a simulação da execução de programas paralelos e o monitoramento de desempenho de máquinas paralelas IBM ou de redes de máquinas IBM. Essa seção abordará apenas as funções do VT para o monitoramento de

desempenho.

O monitoramento de desempenho realizado pelo VT é feito através de visões, descritas na seção 2.5.4. Essas visões permitem a visualização de informações como a utilização da CPU dos nós de processamento, o número de pacotes TCP/IP enviados ou recebidos por um nó e o número de vezes que um nó teve que carregar uma página de memória *swap* em sua memória principal.

Para realizar o monitoramento de desempenho com o VT deve-se seguir os seguintes passos:

1. Executar a ferramenta VT, que apresenta a interface vista na figura 2.27;
2. Selecionar no menu File a opção Performance Monitor, que apresentará a interface para monitoramento de desempenho vista na figura 2.29. Essa interface apresenta uma malha de quadrados, onde cada quadrado representa um nó de processamento da máquina paralela ou da rede de estações. Também são apresentados os nomes dos nós de processamento e, se o sistema for o SP, a listagem das tarefas em execução;
3. Selecionar os nós cujas atividades serão monitoradas. Essa seleção é feita ou pelo quadrado correspondente ao nó, ou pelo nome do nó ou ainda pelo nome da tarefa. A seleção pelo nome da tarefa determina o monitoramento de todos os nós que estão executando a tarefa;
4. Selecionar na interface de visões do VT (figura 2.26) as visões necessárias ao monitoramento. Pode-se selecionar, por exemplo, a Processor Utilization 3D (Utilização dos Processadores em 3D), que informa a taxa de utilização da CPU de cada nó de processamento em um gráfico tridimensional;
5. Selecionar no menu File a opção Monitor para iniciar o monitoramento. A figura 2.30 apresenta um instante do monitoramento da CPU de alguns nós de processamento. Os nós são representados pelas barras e a taxa de utilização da CPU pela altura das barras.

Legenda:

	Nó não operante.		Nó executando uma tarefa.
	Nó em operação.		Nó selecionado para monitoramento.

Figura 2.29 - Monitor de desempenho do VT

Figura 2.30 - Visão Processor Utilization (3D) do VT

Em relação aos dados apresentados pela figura 2.31, pode-se destacar os seguintes:

- Num Users: número de usuários em um nó;
- Time: número de horas que um nó está ligado;
- 1 min Load: número médio de processos prontos para execução durante o minuto anterior à atualização das informações;
- Num procs: número de processos em execução pelo nó;
- Total Mem: quantidade total de memória principal disponível no nó;
- Free Mem: quantidade de memória principal não utilizada;
- Total Swap: quantidade total de memória *swap* disponível no nó;
- Free Swap: quantidade de memória *swap* não utilizada.

Conjunto de Programas Procps-Pare

O conjunto de programas Procps-Pare corresponde a programas disponíveis no Linux modificados para monitorar uma máquina paralela formada por uma rede de estações Linux.

Os programas que realizam o monitoramento devem contatar periodicamente *daemons* (processos que efetuam serviços) em execução em cada nó para obter as informações de que precisam. A figura 2.32 apresenta um console com o programa Top do conjunto de programas Procps-Pare.

```

5:45pm up 6 days, 23:55, 1 user, load average: 0.00, 0.00, 0.00
120 processes: 113 sleeping, 7 running, 0 zombie, 0 stopped
CPU states: 0.02 user, 1.42 system, 0.00 nice, 98.56 idle
Mem: 44432K av, 31040K used, 41328K free, 31880K shrd, 20K buff
Swap: 0K av, 0K used, 0K free, 17380K cached

  PID TID          PPID  PPID  TID  R   S  %C  %M  %T  CMD
  --- --
p2_0 63 root      0 356 356 292 R  0.2 0.0 0:10 procd
p3_0 63 root      0 356 356 292 R  0.2 0.0 0:10 procd
p5_0 63 root      0 356 356 292 R  0.2 0.0 0:10 procd
p7_0 56 root      0 356 356 292 R  0.2 0.0 0:14 procd
p6_0 63 root      0 356 356 292 R  0.2 0.0 0:10 procd
p1_0 63 root      0 356 356 292 R  0.1 0.0 0:10 procd
p4_0 63 root      0 356 356 292 R  0.1 0.0 0:10 procd
p2_0 1 root       0 340 340 272 S  0.0 0.0 0:00 init
p6_0 1 root       0 340 340 272 S  0.0 0.0 0:02 init
p4_0 1 root       0 340 340 272 S  0.0 0.0 0:00 init
p3_0 1 root       0 340 340 272 S  0.0 0.0 0:00 init
p7_0 1 root       0 340 340 272 S  0.0 0.0 0:00 init
p5_0 1 root       0 340 340 272 S  0.0 0.0 0:00 init
p1_0 1 root       0 340 340 272 S  0.0 0.0 0:00 init
p2_0 65 root      0 296 296 240 S  0.0 0.0 0:00 agetty
p3_0 65 root      0 296 296 240 S  0.0 0.0 0:00 agetty

```

Figura 2.32 – Monitoramento de uma máquina SPP2 pelo programa Top do conjunto de programas Procps-Pare

Em relação à figura 2.32 é interessante observar que:

1. A quantidade de processos (120 processes) refere-se a todos os processos em execução pelos nós operacionais;
2. As quantidades de memória principal (Mem) disponível (444324K) e utilizada (31040K used) são, respectivamente, a soma das quantidades de memória principal disponível e utilizada em cada nó operacional. Para a memória *swap* (Swap) a interpretação é similar;
3. A ociosidade de todos os nós (98.8% idle) representa a porcentagem de tempo que os nós não executaram instrução alguma desde a última atualização das informações de monitoramento;
4. O dado Host designa o nó em que o processo com nome `COMMANND`, identificador PID e proprietário USER está em execução;
5. O dado NI representa a prioridade de execução de um processo;
6. O dado %CPU apresenta a porcentagem de tempo em que um processo ocupou a CPU de um nó desde a última atualização das informações do monitoramento.

Em relação ao Top do conjunto de programas Procps-Pare, deve-se observar que o mesmo difere em um aspecto básico do programa Top comumente disponível em estações Linux: não permite a realização de atividades de gerenciamento de tarefas. No programa Top para estações Linux pode-se eliminar tarefas ou alterar a prioridade de execução de tarefas. No Top do conjunto de programas Procps-Pare, essas atividades não são permitidas.

2.7 Análise Comparativa das Ferramentas Revistas

As ferramentas apresentadas na seção anterior possuem características diferentes. O quadro apresentado pela tabela 2.1 compara essas ferramentas em relação às suas características principais.

Característica/Ferramentas	Módulo Monitor do Victor	AIX PSSP	VT	bWatch	Procps-Pare
Monitoramento de nós de processamento	√	√	√	√	√
Monitoramento de tarefas		√	√		√
Monitoramento da máquina paralela vinculado à execução de aplicações paralelas			√		
Gerenciamento de nós de processamento		√			
Gerenciamento de tarefas		√			
Interação com nós de processamento		√			
Disponível para o SPP2				√	√

Tabela 2.1 - Quadro comparativo das ferramentas de monitoramento e gerenciamento de máquinas paralelas

A análise da tabela 2.1 permite constatar que as únicas ferramentas disponíveis para o SPP2 não possuem uma característica importante: funções de gerenciamento para a máquina. A ausência de tal característica dificulta o desenvolvimento de aplicações paralelas, já que não permite a realização de atividades básicas como eliminar dos nós do SPP2 tarefas componentes de uma aplicação paralela. Essa atividade pode ser realizada através da interação via comando de linha com um nó, mas tal procedimento é incômodo para um usuário quando se trata de um SPP2 com um número elevado de nós. Também pode ser impossível realizar essa tarefa quando o SPP2 for configurado para não permitir que usuários comuns do sistema interajam diretamente com os nós.

Além de não permitirem a realização de atividades de gerenciamento, as ferramentas disponíveis para o SPP2 não possuem outra característica muito útil no desenvolvimento e execução de aplicações paralelas: monitoramento do SPP2 vinculado à execução de aplicações paralelas. Essa característica permitiria, por exemplo, avaliar o desempenho ou possíveis problemas de nós responsáveis pela execução de determinadas tarefas componentes de uma aplicação paralela.

2.8 Considerações Finais

O presente capítulo apresentou a revisão bibliográfica necessária ao desenvolvimento da ferramenta Mirador, descrevendo conceitos básicos de processamento paralelo, arquiteturas de máquinas paralelas, a arquitetura do SPP2, programação paralela, ferramentas para desenvolvimento de aplicações paralelas e ferramentas para monitoramento e gerenciamento de máquinas paralelas. Todos esses assuntos possuem relação direta com o desenvolvimento da ferramenta descrita neste trabalho.

A partir das informações apresentadas toma-se clara a necessidade de ferramentas de apoio no processo de desenvolvimento de aplicações paralelas, algumas das quais foram analisadas quanto à sua funcionalidade. Também foram apresentadas algumas ferramentas para o monitoramento e gerenciamento de máquinas paralelas, cujas características também foram discutidas. As funções básicas disponíveis nessas ferramentas estão sumarizadas na tabela 2.1.

As características das ferramentas de gerenciamento e monitoramento discutidas servem de base para a análise de requisitos e especificação preliminar do sistema de gerenciamento proposto para o SPP2, que são apresentadas no próximo capítulo.

Capítulo 3

Definição da Ferramenta Mirador

3.1 Considerações Iniciais

No capítulo 2 foram apresentados os conceitos e trabalhos pertinentes à concepção da ferramenta Mirador, destinada ao monitoramento e gerenciamento do SPP2. Tais conceitos e trabalhos fornecem os subsídios necessários para a definição da ferramenta, que é o tema alvo deste capítulo. A definição da ferramenta Mirador envolve a avaliação e a apresentação de seus objetivos, usuários e requisitos funcionais e não funcionais. Na seção 3.2 são descritos os objetivos que devem ser alcançados com a utilização da ferramenta pelos usuários cujo perfil é apresentado na seção 3.3. Os requisitos funcionais que a ferramenta deve satisfazer são apresentados na seção 3.4, enquanto os não funcionais, na seção 3.5. As considerações finais relativas à definição da ferramenta são apresentadas na seção 3.6.

3.2 Objetivos

As ferramentas disponíveis para monitoramento do SPP2, bWatch e Procps-Pare (seção 2.6), apresentam uma série de deficiências, mostrando apenas informações dos nós ou informações de tarefas dos nós, respectivamente. Muitas informações apresentadas não possuem utilidade para usuários de aplicações paralelas. Exemplificando, as informações de carga e memória global do SPP2 apresentadas pelo programa Top do conjunto de programas Procps-Pare apresentam pequena utilidade para os usuários na execução de suas tarefas. Para se executar uma tarefa deve-se poder avaliar a carga e a memória dos nós individualmente. Dessa maneira, pode-se optar por um nó com mais memória ou com uma carga menor para se executar a tarefa.

A única ferramenta de gerenciamento disponível para o SPP2 é a interação direta, via comando de linha, com cada um dos nós. E essa ferramenta pode não estar disponível, pois o SPP2 pode estar configurado para permitir que um usuário possa apenas submeter tarefas aos nós e não interagir com os mesmos. Tal medida visa reduzir a carga dos nós e da rede de gerenciamento e, conseqüentemente, o tempo para a execução de aplicações paralelas.

As limitações anteriormente enumeradas sustentam a implementação da ferramenta Mirador, cujos objetivos são:

- Permitir que qualquer usuário do SPP2 acesse informações a respeito do estado dos nós de processamento (carga, tarefas, mensagens de erro, data, horário e utilização de memória principal e memória *swap*), elimine tarefas suas de qualquer nó e veja a configuração dos nós (tipo e velocidade dos processadores e quantidade de memória principal e memória *swap* instaladas);
- Possibilitar que um usuário acesse informações a respeito de nós que executam tarefas específicas ou pertencentes a usuários específicos, facilitando o acompanhamento da execução de aplicações paralelas;
- Possibilitar que o administrador do SPP2, além de ter acesso às informações supra citadas, adicione ou remova usuários, altere dados de usuários, reinicie nós, interaja com nós através de um console, elimine quaisquer tarefas dos nós e altere a prioridade de execução de quaisquer tarefas presentes nos nós;
- Permitir que todo o monitoramento e gerenciamento do SPP2 possam ser realizados remotamente e independentemente de plataforma, tanto de hardware quanto de sistema operacional, bastando que o usuário da ferramenta Mirador disponha de um navegador (*browser*) Internet com suporte à linguagem Java e que o computador dele tenha acesso ao *host* do SPP2;
- Apresentar um ambiente gráfico que facilite, além da interação com as funções da ferramenta, a visualização e a comparação de informações, de forma a atingir um elevado índice de usabilidade [JOH92] e, conseqüentemente, produtividade.

3.3 Escopo

A ferramenta Mirador destina-se a programadores e usuários de aplicações paralelas, que podem não estar familiarizados com a utilização de ferramentas através da interação direta por comando de linha com o sistema operacional (Linux) de cada nó do SPP2, e também ao administrador do SPP2, usuário mais familiarizado com a interação com os nós. A ferramenta Mirador deve facilitar a interface entre o usuário e o SPP2, disponibilizando informações e operações comuns de uma forma gráfica, intuitiva e simples.

3.4 Requisitos Funcionais

De acordo com os objetivos propostos e o perfil dos usuários especificados anteriormente, e baseando-se em algumas características presentes nas ferramentas de monitoramento e gerenciamento de máquinas paralelas descritas em 2.6, a ferramenta Mirador deve satisfazer os seguintes requisitos funcionais:

1. A utilização da ferramenta deve ser feita apenas por usuários registrados no SPP2. O registro de usuários é realizado pelo administrador do SPP2 através da própria ferramenta Mirador ou por outro meio. Para ter acesso às funções da ferramenta, o usuário deve informar tanto a sua identificação quanto a sua senha, correspondentes respectivamente ao *login* e ao *password* do sistema operacional da máquina;
2. Um usuário deve poder selecionar um conjunto de nós para monitorá-los. A seleção de um conjunto de nós, denominada seleção para o monitoramento, é composta por uma lista de identificadores de nós (vide requisito 3) e, possivelmente, por qualquer uma ou por ambas as seguintes informações: identificação de um usuário específico e nome de uma tarefa específica (essas duas informações serão referenciadas adiante abreviadamente por “usuário específico” e “tarefa específica”, respectivamente). Caso o usuário não especifique os identificadores dos nós, a ferramenta deve assumir para tal informação os identificadores padrão (*default*) estipulados pelo administrador do SPP2. O monitoramento é realizado apenas nos nós operacionais que satisfaçam todas as informações fornecidas na seleção para o monitoramento, denominados nós selecionados pelo monitoramento. Os nós operacionais contidos na lista de identificadores e que não satisfazem a seleção para o monitoramento são denominados nós não selecionados pelo monitoramento;

A tabela 3.1 descreve a semântica de nós selecionados pelo monitoramento segundo as informações fornecidas pelo usuário na seleção para o monitoramento. A figura 3.1 ilustra o conceito de nó selecionado pelo monitoramento sob a forma de Diagramas de Venn.

Informações que Compõem a Seleção para o Monitoramento	Semântica de Nós Seleccionados pelo Monitoramento	Ilustração
Lista de identificadores de nós	Nós operacionais do SPP2 cujos identificadores estejam contidos na lista de identificadores de nós	Figura 3.1 (a)
Lista de identificadores de nós Usuário específico	Nós operacionais do SPP2 cujos identificadores estejam contidos na lista de identificadores de nós e executem qualquer tarefa pertencente ao usuário específico	Figura 3.1 (b)
Lista de identificadores de nós Tarefa específica	Nós operacionais do SPP2 cujos identificadores estejam contidos na lista de identificadores de nós e executem a tarefa específica	Figura 3.1 (c)
Lista de identificadores de nós Usuário específico Tarefa específica	Nós operacionais do SPP2 cujos identificadores estejam contidos na lista de identificadores de nós e executem a tarefa específica pertencente ao usuário específico	Figura 3.1 (d)

Tabela 3.1 – Semântica de nós seleccionados pelo monitoramento

Legenda

- Nós do SPP2
- Nós contidos na lista de identificadores de nós
- Nós que executam tarefas do usuário específico
- Nós que executam a tarefa específica
- Nós que executam a tarefa específica pertencente ao usuário específico

Figura 3.1: Ilustração da semântica de nós seleccionados pelo monitoramento

Pode-se constatar pela tabela 3.1 e pela figura 3.1 que as informações “tarefa específica” e “usuário específico” destinam-se a restringir o conjunto de nós selecionados pelo monitoramento àqueles nós que podem ser de interesse para um determinado usuário. Exemplificando essa propriedade da ferramenta, um usuário pode estar interessado em monitorar apenas os nós que estejam executando uma tarefa específica pertencente a ele, enquanto que para o administrador do SPP2 é extremamente útil poder monitorar qualquer conjunto de nós, quer seja apenas para manutenção, quer seja para auxiliar um usuário com problemas em relação a algum nó.

No que se refere ainda à seleção para o monitoramento, deve-se observar que o conjunto de nós selecionados pelo monitoramento não é absolutamente estático, podendo variar dinamicamente. Essa variação está subordinada ao dinamismo do estado de cada um dos nós cujos identificadores estejam contidos na lista de identificadores de nós da seleção para o monitoramento. Assim, um nó que em um instante é considerado selecionado pelo monitoramento pode não o ser em outro, podendo estar não operacional ou simplesmente não satisfazer a todas as exigências da seleção para o monitoramento (pode não executar mais a tarefa específica, por exemplo);

3. A especificação da lista de identificadores dos nós da seleção para o monitoramento deve poder ser realizada através de intervalos, satisfazendo qualquer formato que possa ser gerado a partir da seguinte gramática:

FORMATO $\rightarrow$ identificador

FORMATO $\rightarrow$ identificador + “-” + FORMATO

FORMATO $\rightarrow$ identificador + “,” + FORMATO

onde “identificador” é o identificador de um nó, um número positivo (maior ou igual a zero) e menor que N, o número de nós do SPP2 que a ferramenta gerencia. Na linguagem gerada pela gramática, a cadeia composta por dois identificadores separados por “-” representa o conjunto de nós cujos identificadores estão delimitados pelos dois identificadores, incluindo esses dois identificadores. Uma cadeia formada por dois identificadores separados por “,” designa os nós correspondentes aos dois identificadores. Exemplificando em relação a uma seleção

para o monitoramento, se o usuário estipulasse a lista de identificadores de nós “0-4,8” acrescida da restrição que o usuário específico deve ser “juliana” e a tarefa específica deve ser “simula_escoamento”, apenas os nós operacionais dentre os nós 0, 1, 2, 3, 4 e 8 e que possuam a tarefa “simula_escoamento” pertencente ao usuário “juliana” serão considerados nós selecionados pelo monitoramento e, portanto, monitorados;

4. A atividade de monitoramento dos nós selecionados pelo monitoramento deve fornecer periodicamente, segundo escolha do usuário, algumas das seguintes informações textuais: carga, tarefas, memória, mensagens de erro, data e horário de cada nó selecionado pelo monitoramento em um intervalo de tempo especificado pelo administrador do SPP2 (*Toin* - taxa de obtenção de informações dos nós). A carga revela quanto tempo a CPU de um nó esteve ocupada (*To*) executando instruções durante o intervalo de tempo *Toin*, sendo expressa por $(To/Toin) \times 100\%$. As informações de tarefas, memória, data e o horário descrevem, respectivamente, as tarefas, o estado da memória, a data e horário de um nó ao fim de cada intervalo de tempo *Toin*. As mensagens de erro são as emitidas pelo nó durante *Toin*.

Neste trabalho as referências a *Toin* podem expressar tanto o intervalo de tempo *Toin* quanto a taxa *Toin*, pois essa variável é simultaneamente uma taxa segundo a qual ocorre a atualização das informações obtidas dos nós e também um intervalo de tempo no qual as informações dos nós são obtidas. Quando há a necessidade de explicitar um dos dois significados, o termo associado é utilizado;

5. Uma tarefa deve ser especificada pelo identificador do nó selecionado pelo monitoramento que a executa e por dados referentes ao processo correspondente no sistema operacional Linux do nó em questão: nome, usuário proprietário, quantidade de memória em Kbytes que ocupa, prioridade de execução, *pid* (*process identifier* - identificador de processo) e porcentagem de tempo que ocupou a CPU do nó durante *Toin*. Embora esses parâmetros identifiquem univocamente¹ uma tarefa em execução no SPP2, faz-se necessário um pequeno ajuste no nome de um dos dados: “*pid*” deve ser substituído por “*tid*” (*task identifier* - identificador de tarefa), já que utiliza-se o termo tarefas para se referenciar aos programas em execução no SPP2 (seção 2.3.4);

¹ O identificador de processo (*pid*) no Linux é um número que identifica univocamente um processo pelo sistema operacional. Dois processos podem ter o mesmo nome, mas nunca o mesmo *pid*.

6. A data e o horário de um nó selecionado pelo monitoramento devem ser informados juntamente com a data e o horário vigentes no *host* quando ocorre a obtenção das informações do nó, no final do intervalo *Toin*. A diferença em segundos entre essas datas e horários também deve ser informada, expressando-se aproximadamente quanto tempo o relógio de um nó está defasado em relação ao relógio do *host*. Observa-se que essa diferença é apenas aproximada, pois pode haver pequenos atrasos entre a obtenção da data e horário de um nó e a obtenção da data e horário do *host*.

As informações descritas acima são úteis ao administrador para que ele possa tentar sincronizar a data e horário de um nó com a data e horário do *host* ou determinar se há problemas na alimentação ou no relógio de um nó. Esses tipos de problemas podem ter consequências indesejáveis, tal como a perda das informações de configuração do sistema básico de entrada e saída (BIOS – *Basic Input and Output System*);

7. A informação referente à memória de um nó selecionado pelo monitoramento deve revelar a quantidade, em Kbytes, de memória principal e memória *swap* disponível e utilizada pelo nó no fim do intervalo *Toin*;
8. Uma mensagem de erro de um nó deve ser composta pelo identificador do nó, data e horário em que ocorreu o erro e pela descrição do erro;
9. Além das informações textuais que deve poder apresentar eventualmente (carga, tarefas, memória, mensagens de erro, data e horário), atendendo requisições do usuário, a ferramenta deve sempre monitorar e representar, através de ícones, os nós selecionados pelo monitoramento e, caso o usuário especifique, os nós não selecionados pelo monitoramento ou os nós não operacionais ou ambos. Um ícone representativo de um nó selecionado pelo monitoramento ou de um nó não selecionado pelo monitoramento deve expressar, segundo escolha do usuário, a carga ou o estado (operacional) do nó. A semântica para a carga de um nó não selecionado pelo monitoramento é a mesma que para a carga de um nó selecionado pelo monitoramento (vide requisito 4);
10. A ferramenta deve utilizar escalas para a representação da carga de um nó operacional (nó selecionado ou não selecionado pelo monitoramento) através de ícones. A utilização de escalas permite a representação de nós com cargas dentro de

um mesmo intervalo de carga com ícones similares, facilitando a comparação de informações de carga. A ferramenta deve utilizar, segundo a escolha do usuário, uma das seguintes escalas para a representação de carga: 10%, 25% e 50%. Caso o usuário não escolha nenhuma escala, a ferramenta deve utilizar a escala padrão definida pelo administrador. Exemplificando o funcionamento de uma escala, seja um conjunto qualquer de nós operacionais. Se for utilizada a escala de 25%, então todos os nós operacionais com carga dentro do intervalo [0%, 25%) (o valor 0% está incluído, mas o 25% não) serão representados de modo similar. O mesmo ocorre com os nós com carga nos intervalos [25%, 50%), [50%, 75%) e [75%, 100%]. As escalas de 10% e 50% funcionam de modo análogo, variando-se apenas o tamanho do intervalo de carga;

11. Os ícones representativos dos nós selecionados pelo monitoramento e o conjunto de informações requeridas pelo usuário e resultantes do monitoramento compõem as informações do monitoramento do SPP2. É interessante sintetizar que o conjunto de informações requeridas pelo usuário é expresso como sendo qualquer conjunto que envolva algumas das seguintes informações: quaisquer informações textuais dos nós selecionados pelo monitoramento, os ícones representativos dos nós não selecionados pelo monitoramento e os ícones representativos dos nós não operacionais;
12. A atualização das informações do monitoramento do SPP2 deve ser realizada periodicamente, após intervalos de tempo de mesmo valor (*Taim* - taxa de atualização das informações do monitoramento), escolhido pelo usuário dentre alguns valores fixos (*Tca* - taxas constantes de atualização) estipulados pelo administrador. Se o usuário não especificar um valor para *Taim*, a ferramenta deve utilizar um valor padrão especificado pelo administrador. Um exemplo da periodicidade da atualização das informações do monitoramento é a atualização dessas informações a cada 30 segundos (*Taim* igual a 30 segundos), desde que esse valor tenha sido estipulado pelo administrador (*Taim* deve ser igual a algum valor de *Tca*). O usuário deve poder, eventualmente, requerer a atualização das informações antes que se alcance o fim do intervalo de tempo correspondente à taxa *Taim*. Após a atualização, a taxa de atualização permanece a mesma que antes da requisição da atualização pelo usuário.

A figura 3.2 ilustra a relação entre a atualização das informações do monitoramento e a obtenção de informações dos nós, cadenciadas respectivamente pelas taxas *Taim* e *Toin*.

Figura 3.2 – Relação entre a atualização de informações do monitoramento e a obtenção de informações dos nós

Observa-se pela figura 3.2 que, dado um intervalo de tempo genérico em que a ferramenta esteja em execução, expresso pela linha Tempo da figura, a taxa *Taim* determina a cadência da apresentação das informações do monitoramento do SPP2 ao usuário. A cadência de obtenção de informações de cada nó (a figura ilustra a cadência de obtenção de informações para os nós P0, P1 e PN) é ditada pela taxa *Toin*. Exemplificando a relação entre a taxa *Taim* e a taxa *Toin*, as informações do monitoramento apresentadas ao usuário em t_3 e que englobem dados relativos aos nós P0, P1 e PN foram obtidas, respectivamente, em t_0 , t_2 e t_1 .

A figura permite constatar ainda que não se exige sincronismo na atualização das informações de todos os nós, podendo haver diferenças significativas entre um instante qualquer de atualização de todas as informações de um nó e um instante sucessivo, subsequente, de atualização de todas as informações de qualquer outro nó.

Essa situação é ilustrada pelas diferenças de tempo $t_1 - t_0$, $t_2 - t_0$ e $t_2 - t_1$. Tal característica garante a obtenção de informações de um nó assim que ele estiver operacional. Se houvesse exigência de sincronismo na obtenção de informações dos nós, os dados de um nó seriam disponibilizados pela ferramenta somente quando o nó estivesse operacional e fosse o instante exato para obtenção de informações dos nós;

13. A obtenção de informações de um nó segundo *Toin* não deve necessitar apresentar uma cadência extremamente acurada, podendo haver um certo atraso na obtenção das informações. Esse atraso é limitado pelo administrador do SPP2, que deve especificar o atraso máximo tolerável para obtenção de informações de um nó pela ferramenta (*ToinTimeout*). Caso a ferramenta não consiga atualizar as informações de um nó em um instante de tempo situado entre o final do intervalo de tempo *Toin* e o final do intervalo de tempo *ToinTimeout* subsequente, o nó deve ser considerado não operacional até que se consiga atualizar as informações do mesmo.

Deve-se observar que se a variável *ToinTimeout* não fosse utilizada, o seguinte efeito indesejável poderia ocorrer: um nó seria considerado não operacional ao fim do intervalo de tempo *Toin* mesmo que estivesse operacional, já que pode haver um pequeno atraso na obtenção das informações do nó decorrente de carga elevada no nó, na rede de gerenciamento do SPP2 ou no módulo da ferramenta responsável pela obtenção das informações dos nós. A variável *ToinTimeout* deve ser utilizada para estipular o atraso máximo tolerável permitido em tais circunstâncias;

14. A atualização das informações de monitoramento do SPP2 segundo *Taim*, a exemplo da obtenção das informações de um nó, não deve necessitar apresentar uma cadência extremamente precisa. O atraso na obtenção dessas informações é limitado pelo usuário, que deve especificar o quanto está disposto a aguardar uma atualização (*TaimTimeout*). Se o usuário não especificar o valor de *TaimTimeout*, a ferramenta deve assumir o valor padrão para *TaimTimeout* estipulado pelo administrador do SPP2. A atualização das informações de monitoramento deve, portanto, acontecer em um instante de tempo situado entre o final do intervalo de tempo *Taim* e o final do intervalo de tempo para tolerar atrasos, *TaimTimeout*. Caso a atualização não ocorra dentro do intervalo em questão, a ferramenta interrompe o monitoramento das informações e informa o usuário da impossibilidade de atualizar as informações do monitoramento. O usuário poderá, nessa circunstância, optar entre requisitar o

reinício do monitoramento ou cancelá-lo. Deve-se observar que a ferramenta deve interromper apenas o monitoramento das informações. A obtenção de informações dos nós não sofre interrupção.

A Figura 3.3 ilustra um intervalo de tempo de funcionamento da ferramenta e todas as variáveis envolvidas na obtenção de informações dos nós (*Toin* e *ToinTimeout*) e atualização das informações do monitoramento (*Taim* e *TaimTimeout*).

Figura 3.3 – Relação entre todas as variáveis que regem a obtenção de informações dos nós e a atualização das informações do monitoramento

A observação da figura 3.3 permite identificar as diferentes situações possíveis durante a execução da ferramenta:

- a) A atualização das informações de monitoramento não ocorre no final do intervalo de tempo *Taim* após a última atualização, ocorrendo somente no intervalo de tempo *TaimTimeout*, conforme ilustrado na figura no instante t_1 ;
 - b) A ferramenta atualiza as informações de monitoramento no instante padrão t_3 mostrado na figura;
 - c) Não é possível à ferramenta atualizar as informações de monitoramento nem mesmo até o atraso máximo suportado pelo usuário (*TaimTimeout*), o que ocorre no instante t_5 da figura;
 - d) A ferramenta obtém as informações de um nó no instante padrão. Por exemplo, após a última obtenção de informações do nó P0, a ferramenta obtém informações atualizadas de P0 no instante t_0 ;
 - e) A obtenção de informações de um nó só é conseguida com um certo atraso, durante *ToinTimeout*. Por exemplo, uma obtenção de informações de PN ocorre somente no instante t_2 ;
 - f) A ferramenta não consegue obter informações de um nó nem mesmo aguardando um intervalo de tempo *ToinTimeout* após *Toin*. Na figura, a ferramenta não consegue obter informações do nó P1 até o instante t_4 .
15. A apresentação de informações do monitoramento deve poder ser realizada, a pedido do usuário, de forma ordenada, facilitando a interpretação das mesmas para possíveis atividades de gerenciamento ou não. O usuário deve poder, por exemplo, requisitar a ordenação das informações de carga dos nós a fim de selecionar os mais ociosos para a execução de suas tarefas;
16. Durante a apresentação das informações do monitoramento, o usuário pode atribuir a um nó a condição de selecionado para gerenciamento (nó que poderá sofrer atividades de gerenciamento) pressionando brevemente o botão do *mouse* sobre o ícone que representa o nó. A ferramenta deve também atribuir automaticamente, segundo configuração feita pelo usuário, a condição de selecionado para gerenciamento a alguns dos seguintes conjuntos de nós: nós selecionados pelo

monitoramento, nós não selecionados pelo monitoramento e nós não operacionais.

Observa-se, nesse ponto, que a ferramenta deve poder dissociar os nós que sofrem monitoramento dos nós que sofrem gerenciamento. Tal característica permite uma versatilidade maior de uso da ferramenta: pode-se, por exemplo, monitorar a carga, a memória e as tarefas de um conjunto de nós selecionados pelo monitoramento enquanto tenta-se reiniciar os nós não operacionais.

17. O usuário deve poder atribuir a um nó selecionado para gerenciamento a condição de não selecionado para gerenciamento também através do pressionamento do botão do *mouse* sobre o ícone que representa esse nó. O ícone de um nó selecionado para gerenciamento deve diferir, portanto, do ícone de um nó não selecionado para gerenciamento;
18. A atividade de monitoramento e representação gráfica, utilizando ícones, deve ser realizada durante toda a execução da ferramenta. O início dessa atividade deve ocorrer após a identificação do usuário. Para iniciá-la, a ferramenta deve utilizar valores padrões para as seguintes variáveis:
 - Seleção para o monitoramento;
 - *Taim*;
 - *TaimTimeout*;
 - Opção de apresentar os ícones dos nós não selecionados pelo monitoramento;
 - Opção de apresentar os ícones dos nós não operacionais;
 - Formato de apresentação dos ícones dos nós (apresentação de acordo com uma escala específica para carga ou de acordo com o estado, operacional ou não, dos nós).

A ferramenta deve utilizar como valores padrão para essas variáveis os valores escolhidos pelo usuário em outras sessões com a ferramenta. Se algum valor não foi definido pelo usuário, a ferramenta deve utilizar o valor padrão definido pelo administrador;

19. A ferramenta deve fornecer a um usuário comum apenas a função de gerenciamento que permite a exclusão de tarefas desse usuário dos nós selecionados pelo

monitoramento. A exclusão de tarefas deve ser feita selecionando-se com o *mouse*, dentre as tarefas apresentadas, aquelas que devem ser excluídas e executando-se em seguida um comando de exclusão das mesmas. A ferramenta, com base no *tid* e no identificador do nó que executa cada tarefa selecionada para exclusão, bem como na identificação do usuário, deve verificar se a tarefa de fato pertence ao usuário e, caso afirmativo, terminar a sua execução. A seleção de uma tarefa com o *mouse* deve ser feita pressionando-se o botão do *mouse* quando esse apontar alguns dos parâmetros que a descrevem. A ferramenta deve dispor também de um comando específico para a seleção de todas as tarefas;

20. O administrador deve ser capaz de excluir quaisquer tarefas de quaisquer nós selecionados pelo monitoramento, bastando selecionar as tarefas e requisitar a exclusão delas de modo análogo ao que é feito por um usuário comum. Ainda em relação a tarefas, o administrador deve ser capaz de alterar a prioridade de execução de um conjunto de tarefas. O procedimento de seleção das tarefas cujas prioridades de execução devem ser modificadas é igual ao utilizado para excluir tarefas. Após a seleção, deve-se executar o comando adequado informando o novo nível de prioridade, que no sistema Linux situa-se entre -20 (prioridade mais alta) e 20 (prioridade mais baixa) [JHN97];
21. Adicionalmente, o administrador deve ser capaz de interagir com os nós selecionados para gerenciamento através de um console, reiniciá-los ou sincronizar a data e o horário de cada um com a data e o horário do *host*. O administrador deve poder também realizar as mesmas atividades de gerenciamento em um nó qualquer de forma independente dos nós selecionados para gerenciamento;
22. A interação com um nó deve ser baseada no protocolo Telnet [COM95], onde o administrador envia ao nó comandos que devem ser executados pelo mesmo e recebe como resposta o resultado da execução desses comandos;
23. O reinício de um nó pode ser por software, através de uma chamada ao sistema operacional (*system call*) do nó, ou por hardware, utilizando um mecanismo apropriado para esse fim, já que o reinício por software pode falhar se o nó não estiver respondendo a comandos;
24. A sincronização da data e do horário de um nó com a data e o horário do *host* deve ser disponibilizada pela ferramenta para aproximar a data e o horário do nó com a

data e o horário do *host*, conforme descrito no requisito 6;

25. A ferramenta deve fornecer, atendendo a uma requisição do usuário, a configuração dos nós do SPP2. A configuração de um nó deve ser composta pelo tipo e velocidade, em Mhz, do processador do nó e pela quantidade, em Kbytes, de memória principal e de memória *swap* instaladas no nó. As informações de configuração de um nó são importantes em diversas atividades, tais como a execução de tarefas e a determinação do nível de utilização da memória disponível.

Na execução de tarefas, se os nós do SPP2 não forem homogêneos, ou seja, não possuírem a mesma configuração para processador e memória, as informações de configuração permitem ao usuário optar por um processador mais rápido para executar uma tarefa. Deve-se observar que isso não seria necessário se todos os nós possuísem a mesma configuração, pois as informações de carga dos nós selecionados pelo monitoramento seriam suficientes para comparar o desempenho de diferentes nós. Entretanto, para nós com diferentes configurações somente a informação de carga não é suficiente, já que um nó com um processador rápido pode ter uma carga superior à de um nó com um processador mais lento e, no entanto, ser mais rápido para executar uma determinada tarefa.

No que tange à determinação do nível de utilização da memória de um nó, a comparação entre as informações de configuração e as de memória de um conjunto de nós selecionados pelo monitoramento permite determinar se há problemas físicos ou de configuração da memória. Pode-se citar como exemplos de origem desses problemas um dispositivo de memória ou um disco rígido com defeito, no caso de problemas físicos, e erro na configuração da memória *swap* de um nó, para o caso de problemas de configuração;

26. A ferramenta deve fornecer, associado ao ícone de um nó, um atalho para atividades de gerenciamento do nó, verificação da configuração do nó e inclusão ou exclusão do identificador do nó da lista de identificadores de nós da seleção para o monitoramento. Esse atalho deve ser ativado através de um pressionamento ininterrupto do botão do *mouse* sobre o ícone do nó, que deve apresentar um menu com opções para as atividades de interação com o nó, reinício do nó por hardware, reinício do nó por software, sincronização da data e o horário do nó com a data e o horário do *host*, apresentação da configuração do nó e inclusão ou exclusão do nó da

lista de identificadores da seleção para o monitoramento. Deve-se observar que, caso o nó esteja operacional, todas as opções são apresentadas ao administrador, enquanto que para um usuário comum são mostradas apenas as opções para informar a configuração do nó e incluir ou excluir o nó da lista da seleção para o monitoramento;

27. Os ícones utilizados pela ferramenta devem ser capazes de representar as seguintes informações:

- Estado operacional do nó;
- Carga do nó;
- Nó selecionado pelo monitoramento;
- Nó selecionado pelo gerenciamento.

28. Um ícone associado a um nó deve ser composto pela combinação de três elementos: cor, linha delimitadora e formato;

29. A cor de um ícone associado a um nó deve informar se o nó está operacional ou não e pode informar também o nível de carga desse nó. Se um ícone representar somente que o nó está operacional, então a cor do mesmo deve ser branca. Se representar carga, então as cores que pode assumir variam de acordo com a escala utilizada. A tabela 3.2 associa a cor dos ícones aos intervalos de carga das três escalas diferentes que devem poder ser utilizadas pelo usuário.

Cor do Ícone Representativo de um Nó Operacional		Intervalo em que a Carga do Nó Operacional está Situada		
Cor	Descrição	Escala de 10%	Escala de 20%	Escala de 50%
Preto	[0%, 10%)	[0%, 20%)	[0%, 50%)
Marrom	[10%, 20%)
Vermelho	[20%, 30%)
Laranja	[30%, 40%)	[20%, 40%)
Amarelo	[40%, 50%)
Verde	[50%, 60%)	[40%, 60%)
Azul	[60%, 70%)
Violeta	[70 %, 80%)	[60%, 80%)
Cinza	[80 %, 90 %)		
	Branco	[90 %, 100%]	[80%, 100%]	[50%, 100%]

Tabela 3.2 – Intervalos de carga associados às cores dos ícones dos nós operacionais

Quanto às cores utilizadas, elas seguem a sequência do padrão de cores utilizado para codificar componentes eletrônicos [LOU82]. Outros esquemas de codificação de cor poderiam ser utilizados desde que permitissem a fácil diferenciação da carga dos nós. Por exemplo, o uso de uma mesma cor com diferentes tonalidades não seria apropriado, pois poderia dificultar a comparação das cargas para a escala de 10%, que necessitaria de 10 tonalidades distintas para uma mesma cor;

30. A presença de uma linha delineadora em um ícone deve indicar que ele está selecionado pelo monitoramento (figuras 3.4(a), 3.4(b) e 3.4(f));
31. O formato de um ícone associado a um nó deve indicar se ele está selecionado ou não para o gerenciamento. Um ícone associado a um nó selecionado para gerenciamento deve possuir o formato de um “botão pressionado” (figuras 3.4 (b), 3.4 (d) e 3.4(f)), enquanto que um ícone associado a um nó não selecionado para gerenciamento deve possuir o formato de um “botão não pressionado” (figuras 3.4 (a), 3.4 (c) e 3.4 (e));

(a) Nó com carga entre 50% e 60%, selecionado pelo monitoramento e não selecionado para gerenciamento.

(b) Nó com carga entre 50% e 60%, selecionado pelo monitoramento e selecionado para gerenciamento.

(c) Nó com carga entre 50% e 60%, não selecionado pelo monitoramento e não selecionado para gerenciamento.

(d) Nó com carga entre 50% e 60%, não selecionado pelo monitoramento e selecionado para gerenciamento.

(e) Nó com carga entre 40% e 50%, não selecionado pelo monitoramento e não selecionado para gerenciamento.

(f) Nó com carga entre 20% e 30%, selecionado pelo monitoramento e selecionado para gerenciamento.

Figura 3.4 – Exemplos de ícones de nós operacionais (escala de 10% para carga)

32. Um ícone associado a um nó não operacional deve possuir um formato próprio com um “x” no seu interior. Essa peculiaridade é destinada a facilitar a distinção entre nós não operacionais e nós operacionais. A figura a seguir ilustra as duas possíveis apresentações para um nó não operacional.

(a) Nó não operacional não selecionado para gerenciamento.

(b) Nó não operacional selecionado para gerenciamento.

Figura 3.5 – Ícones de nós operacionais

3.5 Requisitos Não Funcionais

A ferramenta Mirador deve satisfazer os seguintes requisitos não funcionais:

1. O módulo da ferramenta que interage diretamente com o usuário deve ser implementado na linguagem Java, que está integrada aos principais navegadores (*browsers*) Internet, garantindo a característica multiplataforma da ferramenta, ao menos para um considerável conjunto de arquiteturas de computador e de sistemas operacionais. Os outros módulos da ferramenta podem ser implementados em uma linguagem qualquer;
2. A interface com o usuário deve ser gráfica e utilizar elementos que permitam a comparação de informações, tais como tabelas e cores;
3. Os nós do SPP2 não podem sofrer uma sobrecarga significativa proveniente do funcionamento dos módulos da ferramenta destinados ao seu controle e monitoramento;

4. Os módulos destinados ao monitoramento e ao gerenciamento do SPP2 devem ser construídos de modo a minimizar a quantidade de informações a serem transportadas pela rede de baixa velocidade, a rede de gerenciamento do SPP2. Uma quantidade menor de informações implica um número menor de pacotes na rede, um número menor de interrupções nos nós para tratamento dos pacotes e um tempo menor para se completar as atividades de monitoramento e gerenciamento.

3.6 Considerações Finais

Neste capítulo foram apresentados os diversos requisitos funcionais e não funcionais que devem nortear o desenvolvimento da ferramenta Mirador.

Foi enfatizada a necessidade do uso deste tipo de ferramenta, não somente pelo administrador do sistema, mas também pelos seus usuários, visto que algumas das informações disponibilizadas por essa ferramenta são importantes no desenvolvimento de aplicações paralelas. Nesse sentido, foi delineado o perfil de um usuário típico do sistema e identificado que o mesmo pode não ter familiaridade com o sistema operacional Unix e ter, conseqüentemente, dificuldade para interação com esse sistema via comandos de linha. A ferramenta deve apresentar uma interface gráfica de fácil utilização para contornar esse problema.

Um requisito importante para a Mirador é a sua operação remota, uma vez que os usuários do SPP2 poderão não estar nas proximidades do equipamento. Seguindo uma tendência atual no desenvolvimento desse tipo de aplicativo, a operação da Mirador deve ser baseada no sistema WWW.

Todos os requisitos definidos neste capítulo são utilizados no capítulo seguinte, que trata da especificação da ferramenta Mirador. Nessa especificação são definidos o comportamento e a funcionalidade da ferramenta.

Capítulo 4

Especificação da Ferramenta Mirador

4.1 Considerações Iniciais

O capítulo 3 apresentou a definição da ferramenta Mirador, descrevendo os objetivos, usuários e requisitos funcionais e não funcionais da ferramenta. Através da análise dos requisitos é realizada a especificação da ferramenta, assunto deste capítulo. A especificação da ferramenta envolve a especificação de sua funcionalidade e a especificação de seu comportamento, apresentadas nas seções 4.2 e 4.3, respectivamente. As considerações finais relativas à especificação da ferramenta são apresentadas na seção 4.4.

4.2 Funcionalidade da Ferramenta

O Diagrama de Fluxo de Dados (DFD) [PRE92] que modela a funcionalidade da ferramenta Mirador é apresentado na próxima seção. A descrição das funções apresentadas no DFD é feita na seção 4.2.2 e a das bases de dados, na seção 4.2.3.

4.2.1 Diagrama de Fluxo de Dados da Ferramenta

A figura 4.1 apresenta o DFD da ferramenta Mirador. O Dicionário de Dados referente ao DFD encontra-se no apêndice A.

Figura 4.1 – Diagrama de Fluxo de Dados da ferramenta Mirador

No que tange às entidades, funções e dados apresentados pelo DFD, uma observação se faz necessária e é extensível às demais figuras deste capítulo: as entidades e funções que se referem a P_i designam entidades e funções referentes ao nó de processamento P_i do SPP2, onde i é um valor entre 0 e $N-1$ ($N-1$ é o maior identificador de nó para uma máquina SPP2 com N nós).

4.2.2 Descrição das Funções da Ferramenta

A descrição de cada uma das funções da ferramenta apresentadas pelo DFD da figura 4.1 é realizada a seguir.

Executar Funções/Monitorar Informações

Essa função coordena toda a interação do usuário com a ferramenta e o monitoramento das informações requeridas pelo usuário. Na coordenação da interação do usuário com a ferramenta, essa função exibe, dependendo do usuário e das informações do monitoramento em apresentação, as funções da ferramenta disponíveis ao usuário e gerencia a execução das mesmas. No monitoramento, essa função realiza a atualização periódica das informações requeridas pelo usuário de acordo com a taxa *Taim* e o atraso *TaimTimeout*. O DFD da figura 4.2 apresenta uma visão mais detalhada da organização dessa função.

Figura 4.2 – Diagrama de Fluxo de Dados da função Executar Funções/Monitorar Informações

O usuário inicia a interação com a ferramenta Mirador informando a identificação e a senha dele (dado Usuário no DFD da figura 4.2), que são analisadas pela função Restringir Acesso a Usuários Cadastrados. A função Restringir Acesso a Usuários Cadastrados determina, mediante consulta à função Verificar Idoneidade do Usuário, se o usuário está cadastrado no SPP2 e pode, portanto, utilizar a ferramenta. Se o usuário não estiver cadastrado ou a senha fornecida for incorreta, a função Restringir Acesso a Usuários Cadastrados informa ao usuário o erro e aguarda novo fornecimento de identificação e senha. Caso o usuário esteja cadastrado e a senha fornecida seja correta, é iniciada a sessão do usuário com a ferramenta.

No início da sessão é realizada a configuração da ferramenta que é realizada pela função Configurar Mirador para Uso pelo Usuário, que obtém a configuração padrão para uso da ferramenta através de consulta à função Ler Configuração de Uso da Mirador. A configuração padrão para uso da ferramenta abrange diferentes configurações padrão, as quais especificam valores padrão definidos pelo administrador ou pelo usuário. As configurações padrão que formam a configuração padrão para uso da ferramenta são as seguintes:

1. Configuração padrão de seleção para o monitoramento:
 - Lista de identificadores de nós;
 - Usuário específico;
 - Tarefa específica.
2. Configuração padrão para apresentação dos ícones dos nós:
 - Apresentar os ícones dos nós não selecionados pelo monitoramento;
 - Apresentar os ícones dos nós não operacionais;
 - Formato de apresentação dos ícones dos nós.
3. Configuração padrão para o monitoramento:
 - *Taim*;
 - *TaimTimeout*.
4. Configuração padrão para a determinação automática dos nós selecionados para gerenciamento:
 - Atribuir automaticamente aos nós selecionados pelo monitoramento a condição de selecionado para gerenciamento;

- Atribuir automaticamente aos nós não selecionados pelo monitoramento a condição de selecionado para gerenciamento;
 - Atribuir automaticamente aos nós não operacionais a condição de selecionado para gerenciamento.
5. Configuração padrão para entrada de dados do usuário e apresentação de dados utilizados por outras funções da ferramenta:
- Número de nós da máquina SPP2;
 - *Tca*;
 - *Toin*.

Quanto à configuração padrão para uso da ferramenta, apresentada anteriormente, duas observações são necessárias. A primeira refere-se à necessidade de se garantir consistência no fornecimento de dados pelo usuário: o usuário não pode, por exemplo, colocar um identificador de nó na lista de identificadores da seleção para o monitoramento que não seja compatível com o número de nós da máquina. Além disso, se o usuário estipular para *Taim* um valor, esse valor deve ser um dos valores de *Tca*. A segunda observação diz respeito a *Toin*, utilizada apenas para fim informativo: o usuário deve saber com que taxa ocorre a obtenção e atualização das informações dos nós para avaliar dados como a carga de um nó. O atraso máximo para obtenção de informações de um nó, *ToinTimeout*, não é apresentado, pois é utilizado apenas para determinar a inatividade de um nó, o que é visível através dos ícones dos nós não operacionais.

Após a configuração da ferramenta para uso, é iniciado o monitoramento das informações requeridas pelo usuário e a apresentação das funções da ferramenta disponíveis ao usuário.

O monitoramento das informações requeridas pelo usuário é realizado pela função Monitorar Informações Requeridas pelo Usuário, que obtém da função Obter Informações do Monitoramento as informações requeridas pelo usuário segundo *Taim* e *TaimTimeout*. As informações do monitoramento são enviadas pela função Monitorar Informações Requeridas pelo Usuário à função Apresentar Informações/Executar Funções para que sejam apresentadas ao usuário.

O acesso às funções da ferramenta e a apresentação das informações do monitoramento são realizados pela função Apresentar Informações/Executar Funções. As funções da ferramenta que podem ser acessadas por essa função podem ser agrupadas em dois conjuntos: funções acessíveis

por qualquer usuário e funções com acesso dependente do usuário e das informações do monitoramento em apresentação.

As funções acessíveis por qualquer usuário englobam aquelas destinadas a incluir ou a excluir um nó da seleção para o monitoramento, selecionar um nó para gerenciamento, selecionar tarefas, alterar e armazenar a configuração do usuário, apresentar a legenda para os ícones dos nós em apresentação e atualizar as informações do monitoramento. Todas essas funções, com exceção da destinada a salvar a configuração do usuário, são executadas por módulos da própria função Apresentar Informações/Executar Funções. A função responsável por salvar a configuração do usuário, Armazenar Configuração do Usuário, é contatada quando o usuário altera algum valor padrão de sua configuração e opta por salvá-la.

As funções com acesso dependente do usuário e das informações do monitoramento em apresentação englobam as funções destinadas a interagir com um conjunto de nós (Interagir com Nós), gerenciar usuários (Gerenciar Usuários), gerenciar tarefas (Gerenciar Tarefas), gerenciar nós (Gerenciar Nós) e obter a configuração de um nó ou de todos os nós do SPP2 (Ler Configuração dos Nós do SPP2). Se o usuário for o administrador, a função permite o acesso a todas funções, com exceção daquelas referentes a informações não presentes nas informações do monitoramento em apresentação (o usuário não pode, por exemplo, eliminar uma tarefa se as informações de tarefas não estiverem presentes nas informações do monitoramento em apresentação). Se o usuário não for o administrador, apenas a função relacionada à eliminação de tarefas (funcionalidade disponibilizada pela função Gerenciar Tarefas) e a função para obtenção da configuração de um nó ou de todos os nós do SPP2 podem ser utilizadas.

Verificar Idoneidade do Usuário

A verificação da idoneidade de um usuário é realizada por essa função. Com a identificação e a senha do usuário, a função verifica, através de consulta ao sistema operacional do *host* (Sistema Operacional do *Host*), se o identificador do usuário está cadastrado no SPP2 e, caso esteja, se a senha fornecida corresponde à senha cadastrada. Se o usuário estiver cadastrado e a senha corresponder à senha cadastrada, é retornada para a função Executar Funções/Monitorar Informações a informação que o usuário pode utilizar a ferramenta. Caso o usuário não esteja cadastrado ou a senha fornecida não corresponda à senha cadastrada, é retornada a informação que o usuário não pode utilizar a ferramenta.

Gerenciar Nós

As atividades de gerenciamento de um conjunto de nós são coordenadas por essa função, que recebe como entrada os identificadores dos nós a serem gerenciados e a operação de gerenciamento a ser realizada sobre esse conjunto de nós: reinício dos nós por hardware, reinício dos nós por software ou sincronização da data e horário de cada nó com a data e horário do *host*.

Se a operação de gerenciamento a ser realizada for o reinício dos nós por software ou a sincronização da data e horário de cada nó com a data e horário do *host*, essa função contata a função responsável pelo gerenciamento de cada um dos nós (Gerenciar *Pi*) e solicita que a mesma efetue a operação. No DFD da figura 4.1 o dado Operação Nó que flui entre as funções Gerenciar Nós e Gerenciar *Pi* representa a solicitação de reinício do nó *Pi* ou a solicitação de sincronização acompanhada pela data e horário do *host*.

Caso a operação de gerenciamento a ser realizada for o reinício dos nós por hardware, o Mecanismo de Reinício de Nós por Hardware é ativado.

Em todas as operações de gerenciamento a base de dados contendo as informações dos nós (Informações dos Nós) é atualizada para refletir a situação de cada nó. Se um nó foi reiniciado, então as informações do nó na base de dados devem ser removidas, significando que o nó está momentaneamente não operacional. Se um nó teve sua data e horário sincronizados com a data e horário do *host*, a data e o horário do nó na base de dados devem ser atualizados.

Uma operação de gerenciamento de um nó pode falhar se o nó não estiver operacional ou houver problemas na comunicação entre as funções envolvidas no gerenciamento. Em tais circunstâncias, essa função deve retornar para a função Executar Funções/Monitorar Informações a listagem de erros em relação a cada um dos nós onde a operação de gerenciamento não pôde ser efetuada. Se não ocorreu erro algum em relação a uma operação de gerenciamento sobre um nó, a função Executar Funções/Monitorar Informações deve atualizar as informações do monitoramento em apresentação referentes ao nó (substituir o ícone do nó, se ele foi reiniciado, por um ícone indicando que o nó não está operacional, por exemplo).

Gerenciar *Pi*

As operações de reinício do nó *Pi* por software ou de sincronização da data e horário do nó são efetuadas por essa função através de chamadas ao sistema operacional do nó. Quando alguma das

operações anteriormente descritas é concluída, essa função informa a função Gerenciar Nós sobre a conclusão da operação.

Gerenciar Tarefas

As atividades de gerenciamento de uma seleção de tarefas são coordenadas por essa função, que recebe como entrada o identificador (*tid*) e o nó de cada tarefa presente na seleção e a operação de gerenciamento a ser realizada sobre essa seleção: eliminar as tarefas ou alterar a prioridade de execução das tarefas. Deve-se observar que se a operação for a alteração da prioridade de execução das tarefas, a nova prioridade de execução das tarefas também faz parte do conjunto de dados de entrada da função (no DFD da figura 4.1 o dado Operação Tarefa engloba também a nova prioridade de execução).

Uma operação de gerenciamento de tarefas pertencentes a um nó é realizada por essa função contatando a função responsável por gerenciar tarefas do nó (Gerenciar Tarefas de *Pi*) e fornecendo a ela o *tid* de cada tarefa e a operação a ser realizada.

Após o término da operação de gerenciamento de tarefas de um nó, a presente função deve atualizar na base de dados Informações dos Nós os dados relacionados às tarefas envolvidas na operação de gerenciamento. Os eventuais erros ocorridos na operação de gerenciamento de tarefas de cada um dos nós (um nó pode não estar operacional, por exemplo) devem ser reportados à função Executar Funções/Monitorar Informações para que sejam informados ao usuário. Se não houve erro na operação de gerenciamento de tarefas de um determinado nó, a função Executar Funções/Monitorar Informações deve atualizar os dados das tarefas do nó envolvidas na operação e em apresentação (suprimir os dados de uma tarefa, se ela foi eliminada, por exemplo).

Gerenciar Tarefas de *Pi*

As operações de gerenciamento de tarefas pertencentes ao nó *Pi* são efetuadas por essa função, que recebe como entrada a operação de gerenciamento a ser realizada e os identificadores (*tids*) das tarefas. Através de chamadas ao sistema operacional do nó *Pi* com parâmetros correspondentes aos identificadores das tarefas, a função realiza a operação de gerenciamento sobre cada uma das tarefas. Quando a operação for concluída, essa função informa a função Gerenciar Tarefas sobre a conclusão da operação.

Gerenciar Usuários

O cadastramento e a remoção de usuários, bem como a listagem e a alteração de dados de usuários cadastrados no SPP2, são realizados pelo administrador por intermédio dessa função. Para efetuar tais operações, a função deve receber como entrada a operação a ser realizada e os dados relativos à operação. O cadastramento de um usuário requer os dados do usuário. A remoção, o identificador do usuário. A alteração de dados, os novos dados do usuário. A listagem, diferentemente das operações anteriores, não requer nenhuma informação, já que a função deve retornar os dados de todos os usuários cadastrados no SPP2.

Se houver algum erro na operação realizada pela função (tentativa de remover um usuário não cadastrado, por exemplo), essa função deve retornar à função Executar Funções/Monitorar Informações o erro ocorrido para que o mesmo seja informado ao usuário.

Interagir com Nós

A interação do administrador com um nó é efetuada por intermédio dessa função, que recebe da função Executar Funções/Monitorar Informações o nó com o qual ocorrerá a interação e as sucessivas seqüências de caracteres associadas à interação. As seqüências de caracteres são enviadas à função responsável pela interação com o nó (Interagir com P_i). As seqüências de caracteres do resultado da interação com o nó e eventuais erros (nó não operacional, por exemplo) são enviadas ao administrador por intermédio da função Executar Funções/Monitorar Informações.

Interagir com P_i

A interação com o nó P_i é realizada por intermédio dessa função, que recebe uma seqüência de caracteres como entrada e transmite-a ao sistema operacional do nó. Se o sistema operacional do nó responder, a resposta, uma seqüência de caracteres, é transmitida à função Interagir com Nós.

Ler Configuração de Nós do SPP2

Essa função recebe como entrada o identificador do nó cuja configuração é requerida pelo usuário e retorna para a função Executar Funções/Monitorar Informações a configuração desse nó, armazenada na base de dados Configuração dos Nós do SPP2, para que seja apresentada ao usuário. Além de poder retornar a configuração de um único nó, essa função pode também retornar a configuração de todos os nós. Tal operação é realizada quando a entrada da função for a solicitação da configuração de todos os nós.

Se houver algum erro na leitura da configuração dos nós (o administrador pode não ter registrado as configurações dos nós), essa função retorna à função Executar Funções/Monitorar Informações o erro ocorrido para que o mesmo seja informado ao usuário.

Ler Configuração de Uso da Mirador

Essa função retorna a configuração padrão para uso da ferramenta Mirador (vide descrição da função Executar Funções/Monitorar Informações no início desta seção). A configuração padrão para uso da ferramenta é obtida através dos seguintes passos:

1. A função obtém a configuração padrão do usuário através de consulta à base de dados Configuração do Usuário;
2. A configuração padrão do usuário é comparada à configuração padrão definida pelo administrador, armazenada na base de dados Configuração da Mirador. Se houver discrepância entre valores, o valor definido pelo administrador é utilizado na configuração padrão para uso da ferramenta. Exemplificando, se o usuário possui um valor para a taxa *Taim* que não esteja presente nas taxas *Tca*, então o valor padrão definido pelo administrador para a taxa *Taim* é utilizado;
3. Se a base de dados Configuração do Usuário não existir ou alguma configuração padrão do usuário não foi estipulada pelo mesmo durante outras sessões com a ferramenta, a configuração não estipulada é substituída pela configuração padrão associada definida pelo administrador. Caso não haja uma configuração padrão associada definida pelo administrador, a configuração não estipulada não fará parte da configuração padrão para uso da ferramenta. Nesse caso, a ferramenta não efetuará as operações associadas à configuração;
4. A função adiciona à configuração padrão obtida pelos dois passos anteriores a configuração padrão para entrada de dados do usuário e apresentação de dados de uso no processamento de outras funções da ferramenta. Essa configuração é composta por dados armazenados na base de dados Configuração da Mirador (*Tca* e *Toin*) e também na base de dados Configuração do SPP2 (números de nós da máquina SPP2).

Armazenar Configuração do Usuário

O armazenamento das configurações padrão que compõem a configuração padrão do usuário na base de dados Configuração do Usuário é realizada por essa função.

Durante a operação de armazenamento da configuração do usuário pode haver algum erro (falta de espaço em disco para a armazenagem, por exemplo), que deve ser informado ao usuário através da função Executar Funções/Monitorar Informações.

Obter e Processar Informações de Pi

É a função responsável pela obtenção e processamento de informações do nó *Pi* segundo *Toin*. Através de chamadas ao sistema operacional do nó *Pi*, a função obtém informações de tarefas, memória, data, horário e mensagens de erro do nó. Com base nas informações de tarefas, a função determina a carga do nó, correspondente à fração de tempo que as tarefas estiveram em execução pelo processador durante *Toin*.

Após o cálculo da carga do nó, essa função contata a função Atualizar Informações dos Nós para que a mesma atualize as informações do nó na base de Informações dos Nós. Se não puder contatar aquela função, o ciclo de processamento e atualização de informações continua normalmente segundo *Toin*.

Atualizar Informações dos Nós

É a função responsável por armazenar as informações de um nó. Ao receber as informações de um nó, essa função armazena-as na base de dados Informações dos Nós juntamente com a data e o horário vigentes no *host*. A data e o horário do *host* são armazenados para permitir a detecção de um nó inoperante pela função Obter Informações do Monitoramento.

Obter Informações do Monitoramento

A obtenção das informações do monitoramento é realizada por essa função com base nas informações requeridas pelo usuário e nas configurações do monitoramento e da seleção para o monitoramento. Através de consultas à base de dados Informações dos Nós e com base na data e horário do *host*, essa função determina os nós selecionados pelo monitoramento e as informações requeridas pelo usuário associadas aos mesmos. Se o usuário também requisitou a apresentação dos nós não selecionados pelo monitoramento e os nós não operacionais, a função adiciona essas informações às informações referentes aos nós selecionados pelo monitoramento, compondo as informações do monitoramento. As informações do monitoramento são retornadas ao usuário por intermédio da função Executar Funções/Monitorar Informações.

Para determinar se um nó contido na lista de identificadores da seleção para o monitoramento é um nó selecionado pelo monitoramento, a função realiza os seguintes passos:

1. A função consulta a base de dados Informações dos Nós e obtém a data e horário do nó quando ocorreu a última atualização de informações do nó. Se não houver dados referentes ao nó, ele é considerado um nó não operacional;
2. A data e o horário do nó quando ocorreu a última atualização de informações do nó é comparada com a data e horário vigentes no *host*. Se a diferença entre esses instantes for superior a *ToinTimeout*, o nó é considerado não operacional. Se for menor ou igual, o nó é considerado candidato a nó selecionado pelo monitoramento;
3. Se a seleção para o monitoramento não possui as restrições usuário específico e tarefa específica, o nó candidato a nó selecionado pelo monitoramento é considerado um nó selecionado pelo monitoramento;
4. Se a seleção para o monitoramento estipular as restrições “usuário específico” e “tarefa específica”, a função analisa os dados do nó e determina se ele possui alguma tarefa que satisfaça as restrições fornecidas (vide requisito 1 na seção 3.4). Ao encontrar uma tarefa que satisfaça as restrições fornecidas, o nó é considerado um nó selecionado pelo monitoramento. Caso nenhuma tarefa satisfaça as restrições, o nó é considerado um nó não selecionado pelo monitoramento.

Nos passos listados anteriormente, ao determinar que um nó é um nó selecionado pelo monitoramento, a função adiciona às informações do monitoramento as informações referentes ao nó e requeridas pelo usuário. Ao determinar que um nó é um nó não selecionado pelo monitoramento ou um nó não operacional, a função pode adicionar às informações do monitoramento a carga do nó ou o identificador do nó, respectivamente. Deve-se notar que essas informações são adicionadas às informações do monitoramento apenas se o usuário as requisitou.

4.2.3 Descrição das Bases de Dados da Ferramenta

Os próximos tópicos descrevem as bases de dados utilizadas pelas funções da ferramenta Mirador, presentes no DFD da figura 4.1.

Configuração da Mirador

A configuração padrão para uso da ferramenta definida pelo administrador é armazenada por essa base de dados, sendo composta pelas seguintes configurações padrão e dados associados:

1. Configuração padrão de seleção para o monitoramento:
 - Lista de identificadores de nós.
2. Configuração padrão para apresentação dos ícones dos nós:
 - Formato de apresentação dos ícones dos nós.
3. Configuração padrão para o monitoramento:
 - *Taim*;
 - *Tca*;
 - *TaimTimeout*.
4. Configuração padrão para a obtenção de informações dos nós:
 - *Toin*;
 - *ToinTimeout*.

Configuração do Usuário

A configuração padrão para uso da ferramenta definida por um usuário específico é armazenada por essa base de dados. Tal configuração é composta pelas seguintes configurações padrão e valores associados:

1. Configuração padrão para a seleção para o monitoramento:
 - Lista de identificadores de nós;
 - Usuário específico;
 - Tarefa específica.
2. Configuração padrão para apresentação dos ícones dos nós:
 - Formato de apresentação dos ícones dos nós;
 - Apresentar os ícones dos nós não selecionados pelo monitoramento;
 - Apresentar os ícones dos nós não operacionais.

3. Configuração padrão para o monitoramento de informações dos nós:
 - *Taim*;
 - *TaimTimeout*.
4. Configuração padrão para a determinação automática dos nós selecionados para gerenciamento:
 - Atribuir automaticamente aos nós selecionados pelo monitoramento a condição de selecionado para gerenciamento;
 - Atribuir automaticamente aos nós não selecionados pelo monitoramento a condição de selecionado para gerenciamento;
 - Atribuir automaticamente aos nós não operacionais a condição de selecionado para gerenciamento.

Configuração do SPP2

É a base de dados que armazena os dados referentes à configuração da máquina SPP2. Os dados da configuração da máquina SPP2 são o número de nós da máquina e a configuração de cada nó (tipo e velocidade do processador e memória principal e de memória *swap* instaladas no nó).

Informações dos Nós

As informações de cada nó do SPP2 (carga, tarefas, memória, data, horário e mensagens de erro) são armazenadas por essa base de dados juntamente com a data e horário no *host* em que ocorreu a atualização dessas informações.

4.3 Comportamento da Ferramenta

O *Statechart* [PRE92, UML97] que modela o comportamento da ferramenta Mirador é apresentado na próxima seção. A descrição de cada estado do *Statechart*, que corresponde a um estado da execução da ferramenta, é feita na seção 4.3.2.

4.3.1 *Statechart* da Ferramenta

O *Statechart* da ferramenta (figura 4.3) apresenta os diversos estados em que a ferramenta pode se encontrar durante a sua execução. Para a análise do *Statechart* é interessante recorrer ao DFD da

ferramenta (figura 4.1), já que o *Statechart* destina-se a modelar o aspecto comportamental das funções da ferramenta. Dessa maneira, analisando-se o *Statechart* e o DFD simultaneamente, pode-se avaliar quando uma função tem a sua execução iniciada, condição que não é modelada convenientemente por um DFD. Os eventos e dados apresentados no *Statechart* estão descritos no apêndice B.

Figura 4.3 – Statechart da ferramenta Mirador

No *Statechart*, o estado Processador de Informações do Nó Pi designa um estado onde ocorre o processamento de informações do nó Pi. Deve-se observar que tal estado é uma abstração correspondente a todos os estados destinados ao processamento de informações dos nós do SPP2.

A análise do *Statechart* da ferramenta permite identificar quatro conjuntos de estados muito distintos funcionalmente entre si, a saber:

1. Conjunto de estados onde ocorre a interação da ferramenta com o usuário: abriga apenas o estado *Interface_com_o_Usuário* (IU).

Esse estado é responsável basicamente por ativar estados em que ocorrem operações requisitadas pelo usuário. Por exemplo, se o usuário requisitar a interação com um nó (condição *interagir_com_nó*), a ferramenta assumirá também um estado onde ocorre a interação com um nó (estado *Console*);

2. Conjunto de estados onde ocorre o monitoramento das informações requisitadas pelo usuário: engloba somente o estado *Monitor*.

Quando a ferramenta estiver nesse estado, a atualização das informações do monitoramento estará ocorrendo segundo *Taim* e *TaimTimeout*;

3. Conjunto de estados onde ocorre a execução de operações relacionada a funções requisitadas pelo usuário: composto pelos estados *Gerenciador_de_Nós*, *Gerenciador_de_Tarefas*, *Armazenador_da_Configuração_do_Usuário*, *Console*, *Gerenciador_de_Usuários* e *Leitor_de_Configuração_de_Nós*.

É interessante observar que a ferramenta só pode assumir algum desses estados se já não estiver no mesmo. Exemplificando, o estado *Armazenador_da_Configuração_do_Usuário* só poderá ser assumido pela ferramenta se a mesma já não estiver nesse estado (vide no *Statechart* a condição $\text{fim_arm} = 1$ em $[\text{armazenar_configuração} \wedge \text{fim_arm} = 1]$. *Armazenador_da_Configuração_do_Usuário.iniciar*(Conf. Usuário)). Essa restrição garante que não se realize uma operação associada a um estado em concorrência com uma outra operação associada ao mesmo estado, o que pode determinar inconsistências. Não se pode, por exemplo, requisitar a eliminação de uma tarefa e ao mesmo tempo, antes que a tarefa seja eliminada, requerer a alteração da prioridade da mesma (se uma tarefa está sendo eliminada então ela não pode ter sua prioridade alterada). Poderiam ser permitidas operações de gerenciamento simultâneas, mas a interface do sistema ficaria carregada devido à necessidade de

diferenciação entre os dados associados a cada uma das operações de gerenciamento. Essa providência garantiria que um mesmo dado não fosse associado a duas ou mais operações de gerenciamento simultaneamente.

A concorrência entre operações pertinentes a estados distintos é permitida. Pode-se, portanto, eliminar tarefas (estado Gerenciador_de_Tarefas) e sincronizar a data e horário de um nó com a data e horário do *host* (estado Gerenciador_de_Nós) concorrentemente;

4. Conjunto de estados onde ocorre a obtenção e o processamento de informações dos nós: composto apenas pelo estado Processador_de_Informações_do_Nó_Pi. Nesse estado ocorrem a obtenção, processamento e atualização das informações do nó Pi segundo *Toin*.

A descrição de cada um dos estados da ferramenta encontra-se na próxima seção.

4.3.2 Descrição dos Estados da Ferramenta

Interface com o Usuário (IU)

Nesse estado a ferramenta, após garantir a idoneidade do usuário, é configurada para uso pelo usuário, inicia o monitoramento das informações requeridas pelo usuário e o processo de execução de funções de acordo com as escolhas do usuário. Deve-se observar que as informações requeridas pelo usuário são compostas inicialmente apenas pelos valores presentes na configuração padrão para apresentação dos ícones dos nós (vide descrição da configuração do usuário na Seção 4.2.2).

A verificação da idoneidade de um usuário é realizada nos estados Lendo Identificação e Senha do Usuário e Verificando Idoneidade do Usuário.

A configuração da ferramenta para uso pelo usuário é feita no estado Configurando Mirador para Uso pelo Usuário.

O monitoramento das informações requeridas pelo usuário, realizado no estado Monitor, é iniciado logo após o término da configuração da ferramenta para uso pelo usuário.

A coordenação da execução de funções de acordo com as escolhas do usuário é realizada no estado Apresentando Informações/Executando Funções. Exemplificando a execução de uma função, se o usuário determinou a operação de reinício de um conjunto de nós (condição *gerenciar_nós*) e já não houver operação associada aos nós (condição *fim_nós* = 1), o estado Gerenciador_de_Nós

será ativado. Quando uma operação é finalizada, o estado associado à execução da operação informa o estado Apresentando Informações/Executando Funções para que o mesmo atualize as informações envolvidas na operação. No caso do reinício de um nó, por exemplo, o estado Gerenciador_de_Nós dispara o evento atualizar_nós (Erros Operação Nó) para que ocorra a atualização das informações associadas ao nó.

A atualização das informações do monitoramento é realizada através do estado Monitor disparando o evento atualizar_informações (Inf. do Monit.).

Monitor

A obtenção e a requisição da atualização das informações do monitoramento segundo a taxa *Taim* e o atraso *TaimTimeout* são realizadas nesse estado. Após a obtenção das informações do monitoramento no estado Obtendo Informações do Monitoramento, a ferramenta requisita a atualização dessas informações disparando o evento atualizar_informações(Inf. do Monit.) no estado Interface_com_o_Usuário. Finalizada a atualização (evento atualização_encerrada disparado pelo estado Interface_com_o_Usuário), a ferramenta permanece no estado Fim_da_Atualização por *Taim* segundos e então retorna ao estado Obtendo Informações do Monitoramento.

A permanência da ferramenta no estado Obtendo Informações do Monitoramento não pode ser superior a *TaimTimeout* segundos, que é o atraso máximo permitido pelo usuário para obter as informações do monitoramento. Se a obtenção das informações do monitoramento suplantam *TaimTimeout* segundos, a ferramenta irá para o estado Erro Atualização, onde o usuário poderá optar entre reiniciar e finalizar o monitoramento.

Durante a interação com a ferramenta, o usuário pode requisitar a atualização imediata das informações do monitoramento em apresentação. Essa operação é efetuada através do estado Interface_com_o_Usuário requisitando o reinício da obtenção das informações do monitoramento ao estado Monitor (evento iniciar(Conf. Sel. Monit., Conf. Monit., Inf. Requeridas)).

Console

A interação com um nó de processamento ocorre nesse estado e só é terminada quando a sequência de caracteres da interação corresponder ao comando de fim de interação.

Gerenciador de Nós

Nesse estado a ferramenta executa uma operação de gerenciamento sobre um conjunto de nós. O resultado da operação é informado ao estado Interface_com_o_Usuário através do evento atualizar_nós(Erros Operação Nó).

Gerenciador de Tarefas

A execução de uma operação de gerenciamento sobre um conjunto de tarefas é realizada nesse estado. Após a execução da operação esse estado dispara o evento atualizar_tarefas(Erros Operação Tarefas) no estado Interface_com_o_Usuário para que ocorra a atualização das informações das tarefas.

Gerenciador de Usuários

As operações de gerenciamento de usuários são realizadas quando a ferramenta estiver nesse estado. O resultado de uma operação de gerenciamento é informado ao estado Interface_com_o_Usuário através do evento atualizar_usuários(Dados Usuários ou Erro).

Armazenador da Configuração do Usuário

O armazenamento da configuração do usuário para uso da ferramenta é efetuado nesse estado. O resultado da operação de armazenamento é informado ao estado Interface_com_o_Usuário através do evento atualizar_armazenagem(Erro Armazen.).

Leitor de Configuração de Nós

A leitura da configuração de nós é feita nesse estado. A configuração dos nós ou o erro proveniente da leitura são informados ao estado Interface_com_o_Usuário através do evento atualizar_conf_nós(Conf. Nós).

Processador de Informações do Nó Pi

A obtenção, o processamento e a atualização das informações do nó *Pi* segundo *Toin* são realizados nesse estado.

Após a obtenção e o processamento das informações do nó *Pi* no estado Obtendo e Processando Informações de *Pi*, ocorre a atualização das informações do nó no estado Atualizando Informações de *Pi*. Finalizada a atualização das informações (condição atualização_encerrada), a ferramenta permanece no estado Fim da Atualização das Informações de *Pi* durante *Toin* segundos.

4.4 Considerações Finais

A especificação da ferramenta Mirador, que abrange a especificação da sua funcionalidade e do seu comportamento, foi apresentada neste capítulo.

Na especificação da funcionalidade utilizou-se um DFD, que apresenta as funções da ferramenta, o relacionamento entre elas e também os diversos elementos (entidades) que interagem com a ferramenta. Para a especificação do comportamento da ferramenta foi utilizado um *Statechart*, o qual representa os diversos estados em que a ferramenta pode se encontrar em sua execução.

A especificação da funcionalidade e do comportamento que a ferramenta deve apresentar é a base necessária ao seu projeto e subsequente implementação, assuntos do próximo capítulo.

Capítulo 5

Projeto, Implementação e Testes da Ferramenta Mirador

5.1 Considerações Iniciais

No capítulo 4 foram apresentados a funcionalidade e o comportamento da ferramenta Mirador através de um Diagrama de Fluxo de Dados (DFD) e de um *Satechart*. O projeto dos módulos de software e hardware, das bases de dados, dos protocolos de comunicação e da interface da ferramenta são feitos no presente capítulo respeitando-se as informações contidas no capítulo anterior. A implementação e os testes da ferramenta também são abordados neste capítulo.

Na seção 5.2 é apresentado o projeto da arquitetura da ferramenta, que descreve os diversos módulos de software, hardware, bases de dados e protocolos de comunicação da ferramenta. A seção 5.3 apresenta o projeto da interface da ferramenta com o usuário, descrevendo a organização da interface e os principais menus e janelas componentes da mesma. A seção 5.4 descreve alguns detalhes da implementação da ferramenta, apresentando a implementação dos módulos de software, dos módulos de hardware, das bases de dados e também os protocolos de rede utilizados para a comunicação entre os componentes da arquitetura da ferramenta. Os testes da ferramenta e os resultados obtidos com a sua utilização são descritos na seção 5.5. As considerações finais relacionadas ao projeto, implementação e testes da ferramenta são apresentadas na seção 5.6.

5.2 Projeto da Arquitetura da Ferramenta

Esta seção apresenta o projeto da arquitetura da ferramenta Mirador, destacando-se as diretrizes que foram seguidas para o projeto da arquitetura e seus componentes.

5.2.1 Diretrizes do Projeto da Arquitetura da Ferramenta

O projeto dos módulos de software e hardware, dos protocolos de comunicação entre esses módulos e das bases de dados que compõem a arquitetura da ferramenta é baseado no DFD, no *Statechart* e nos requisitos não funcionais da ferramenta. Também foi considerada uma organização dos módulos de software e hardware de modo a oferecer, sempre que possível, funções que podem ser utilizadas por outras ferramentas.

As próximas seções descrevem a arquitetura da ferramenta Mirador, apresentando os módulos de software e hardware, os protocolos de comunicação e as bases de dados que a compõem.

5.2.2 Arquitetura da Ferramenta

Esta seção apresenta o resultado da análise realizada para projetar os componentes da arquitetura da ferramenta Mirador e a alocação dos mesmos na arquitetura do SPP2. Tal análise buscou garantir a menor troca de informações entre os componentes e o maior desempenho da ferramenta com um mínimo de interferência no desempenho do SPP2.

Os módulos de software, hardware e bases de dados componentes da arquitetura da ferramenta Mirador são apresentados pela figura 5.1. Nessa figura, o termo *internet* [COM95] designa uma única rede ou uma interconexão de redes, podendo variar de uma simples rede local até a Internet. Deve-se observar ainda que o computador do usuário pode abstrair o próprio *host* (um usuário pode interagir com a ferramenta Mirador utilizando o *host*).

Figura 5.1 – A arquitetura da ferramenta Mirador

A associação da arquitetura da ferramenta com as funções, bases de dados e entidades apresentadas na especificação da ferramenta, descrita no capítulo 4, é ilustrada na figura 5.2.

Figura 5.2 – Mapeamento das funções, entidades e bases de dados na arquitetura da ferramenta Mirador

A análise da figura 5.2 permite observar que:

1. A função da ferramenta responsável pela interface com o usuário (Executar Funções/Monitorar Informações) é implementada pelo Módulo de Interação com o Usuário (MIU);
2. As funções responsáveis pela atualização das informações dos nós (Atualizar Informações dos Nós), obtenção das informações do monitoramento (Obter Informações do Monitoramento), leitura da configuração de nós (Ler Configuração de Nós do SPP2), armazenamento da configuração do usuário (Armazenar Configuração do Usuário) e pelas operações de gerenciamento (Gerenciar Tarefas, Gerenciar Nós e Gerenciar Usuários) são disponibilizadas pelo módulo Servidor de Informações dos Nós e Gerenciador de Operações de Gerenciamento (SINGOG);
3. As funções de gerenciamento do nó *Pi* (Gerenciar *Pi*) e de tarefas do nó *Pi* (Gerenciar Tarefas de *Pi*) são disponibilizadas pelo módulo Gerenciador do Nó *Pi* (GN*Pi*);
4. A função de obtenção e processamento de informações do nó *Pi* (Obter e Processar Informações de *Pi*) é implementada pelo módulo Monitor de Informações do Nó *Pi* (MIN*Pi*);
5. A função responsável pelo gerenciamento da interação com os nós (Interagir com Nós) é implementada pelo módulo Gerenciador de Interações com os Nós (GIN);
6. A função que permite a interação com o nó *Pi* (Interagir com *Pi*) é implementada pelo módulo Gerenciador de Interação com o Nó *Pi* (GIN*Pi*);
7. O reinício de nós por hardware é feito pelo Mecanismo de Reinício de Nós por Hardware (MRNH);
8. A comunicação entre algumas funções de módulos distintos, ou simplesmente comunicação entre módulos, exige um dado adicional aos que já estavam expressos no DFD da ferramenta (figura 4.1) devido à organização da arquitetura e a aspectos de segurança. O dado em questão é o **Usuário**, destacado em negrito na figura 5.2, e engloba não somente a identificação do usuário como também a sua senha de acesso.

Em relação à arquitetura da ferramenta, deve-se observar ainda que ela satisfaz o *Statechart* da ferramenta (seção 4.3). Cada estado do *Statechart* corresponde ao funcionamento de um conjunto de módulos. A comunicação entre estados corresponde à comunicação entre os módulos.

A justificativa para a organização feita dos módulos de software e hardware e das bases de dados em relação aos elementos componentes da arquitetura do SPP2 e ao computador do usuário (figura 5.1) encontra-se na próxima seção.

Os protocolos de comunicação entre os módulos e a necessidade de uso do dado adicional **Usuário** são abordados na seção 5.2.4.

A organização das bases de dados utilizadas pela ferramenta é tema da seção 5.2.5.

5.2.3 Organização dos Módulos de Software e Hardware da Ferramenta

Os próximos tópicos descrevem os módulos da ferramenta e a motivação para a localização de cada um deles como expresso pela figura 5.1.

Módulo de Interação com o Usuário (MIU)

Esse é o módulo que realiza a interação com o usuário, necessitando, portanto, ser executado pelo computador do usuário. Através de chamadas ao módulo SINGOG esse módulo obtém a configuração padrão de uso da ferramenta, armazena a configuração do usuário, obtém as informações do monitoramento, a configuração de nós e requisita a execução de operações de gerenciamento. A interação do usuário com um determinado nó é realizada por esse módulo contatando o módulo GIN.

Servidor de Informações dos Nós e Gerenciador de Operações de Gerenciamento (SINGOG)

É o módulo que implementa as funções de obtenção da configuração padrão de uso da ferramenta, armazenamento da configuração do usuário, obtenção das informações do monitoramento, atualização das informações dos nós e execução de operações de gerenciamento sobre um usuário, um conjunto de nós ou um conjunto de tarefas.

A localização desse módulo no *host* e não como parte do módulo MIU deve-se a dois fatores:

1. Se o módulo SINGOG fosse uma parte constituinte do módulo MIU, haveria necessidade de se colocar em execução no *host* um módulo próprio destinado a permitir que as chamadas realizadas pelo módulo SINGOG aos módulos localizados nos nós pudessem ser efetuadas. Tal necessidade acontece porque uma das funções do *host* na arquitetura do SPP2 é a de manter o tráfego da rede que interliga o *host* a outros computadores isolado da rede de gerenciamento dos nós;
2. A localização do módulo SINGOG no *host* permite que funções implementadas por ele possam ser acessadas por outras ferramentas, o que não seria possível se ele fosse parte do módulo MIU.

As funções disponibilizadas por esse módulo para uso por módulos da ferramenta Mirador, ou por outras ferramentas, implementam serviços de uso restrito a usuários do SPP2. A tabela 5.1 apresenta a correlação entre cada serviço e a função que o implementa.

Serviço	Função que Implementa o Serviço
Armazenar configuração do usuário para uso da Mirador	Armazenar Configuração do Usuário
Ler configuração de nós do SPP2	Ler Configuração de Nós do SPP2
Ler configuração de uso da Mirador	Ler Configuração de Uso da Mirador
Gerenciar um conjunto de nós	Gerenciar Nós
Gerenciar um conjunto de tarefas	Gerenciar Tarefas
Gerenciar usuários	Gerenciar Usuários
Obter informações de um conjunto de nós	Obter Informações do Monitoramento

Tabela 5.1 – Correlação entre os serviços e funções do módulo SINGOG

Gerenciador do Nó P_i (GNP $_i$)

As funções relacionadas a operações de gerenciamento sobre um determinado nó (gerenciamento do nó e gerenciamento de tarefas do nó) devem ser realizadas por um módulo (GNP $_i$) que esteja em execução no próprio nó. Essa necessidade acontece pela necessidade de uso de chamadas ao sistema operacional pelas operações de gerenciamento.

Monitor de Informações do Nó Pi (MINPi)

A função de obtenção e processamento de informações de um nó necessita ser implementada por um módulo de software que esteja em execução no nó pois envolve chamadas ao sistema operacional do nó. Devido ao fato de que apenas a obtenção de informações necessitar de chamadas ao sistema operacional do nó, seria possível desmembrar a função de obtenção e processamento em duas funções distintas: uma função para obter as informações do nó e outra função para processar as informações do nó. A primeira seria implementada por um módulo em execução no próprio nó. A outra, por um módulo em execução no *host*. Entretanto, tal arranjo não seria conveniente pois a determinação da carga de um nó com base nas informações obtidas do nó não representa uma sobrecarga significativa no processamento do nó. Deve-se observar que a determinação da carga de um nó não envolve muitas operações aritméticas associadas às tarefas do nó e é realizada apenas periodicamente, de acordo com *Toin*. Além disso, o processamento das informações pelo módulo em execução no *host* exigiria o envio delas através da rede de gerenciamento, o que poderia despendar um tempo não desprezível dependendo da carga da mesma.

Ainda em relação à localização desse módulo, deve-se observar que o mesmo interage com o módulo SINGOG para atualizar suas informações na base de dados contendo as informações dos nós. Esse arranjo representa uma inovação em relação ao funcionamento das ferramentas bWatch e Procps-Pare (seção 2.6). Nessas ferramentas as informações dos nós são requisitadas aos nós sempre que necessárias. Na ferramenta Mirador as informações dos nós são sempre atualizadas, segundo *Toin*, em uma base de dados, independentemente se são necessárias ou não. Quando necessárias, elas são obtidas dessa base de dados, que fica armazenada no *host*. Esse modo de funcionamento, se comparado ao das ferramentas anteriormente mencionadas, gera um tráfego menor na rede de gerenciamento quando a ferramenta é usada por mais de um usuário simultaneamente.

Gerenciador de Interações com os Nós (GIN)

A função de gerenciamento da interação do usuário com um nó deve ser implementada por um módulo de software em execução no *host* por dois motivos, similares aos apresentados para justificar a presença do módulo SINGOG no *host*:

1. Se o módulo GIN fosse parte integrante do módulo MIU, o contato entre o módulo GIN e o módulo de software que interage com um determinado nó (GINPi) não seria

- possível. Tal limitação é decorrente do *host* isolar o tráfego da rede externa aos nós do SPP2 da rede de gerenciamento dos nós;
2. A localização do módulo GIN no *host* permite que outras ferramentas tenham acesso às suas funções.

Gerenciador de Interação com o Nó Pi (GINPi)

A função responsável pela interação com um nó em particular deve ser implementada por um módulo de software (GINPi) em execução nesse nó. Tal localização é justificada pela necessidade de envio de comandos (seqüências de caracteres) ao sistema operacional do nó e de se receber respostas (seqüências de caracteres) dele.

Mecanismo de Reinício de Nós por Hardware (MRNH)

O Mecanismo de Reinício de Nós por Hardware deve ter acesso aos nós do SPP2 e também poder ser acionado pelo módulo SINGOG, em execução no *host*. Tais necessidades determinam a sua localização como expressa pela Figura 5.1.

5.2.3 Protocolos de Comunicação entre os Módulos da Ferramenta

O protocolo de comunicação entre dois módulos x e y da ferramenta é constituído basicamente por uma requisição de serviço (execução de uma função) de x para y e uma resposta de y , após a realização do serviço, para x . As únicas exceções são a ativação do módulo MRNH e a comunicação entre módulos para interação com um nó. No primeiro caso, há apenas o envio dos identificadores dos nós ao módulo MRNH. No segundo caso, utiliza-se o protocolo Telnet [COM95] e o protocolo Telnet com uma pequena modificação. A figura 5.3 ilustra os protocolos de comunicação entre os módulos da ferramenta.

Figura 5.3 – Protocolos de comunicação da ferramenta Mirador

Os próximos tópicos descrevem os três protocolos apresentados anteriormente.

Protocolo Requisição de Serviço/Resposta

Uma requisição de serviço de um módulo a outro é composta basicamente pelos seguintes campos:

1. Identificação do serviço a ser executado: um módulo pode implementar vários serviços (funções), sendo necessária a identificação de qual serviço deve ser executado. A figura 5.2 não apresenta essa identificação porque já a abstrai: uma função pertencente a um módulo está ligada através de um fluxo de dados diretamente a uma função específica de um outro módulo;
2. Código de acesso ao serviço: é o dado **Usuário** na figura 5.2, composto pela identificação e pela senha do usuário. O acesso aos serviços do módulo SINGOG de uso restrito a usuários do SPP2 requer um código de acesso para não se comprometer a segurança de todo o sistema SPP2. Exemplificando, se não houvesse um código de acesso, uma pessoa qualquer, não necessariamente um usuário do SPP2, poderia implementar um programa que reiniciasse um conjunto de nós sabendo apenas como é a interface do serviço de reinício de nós.

A utilização do dado **Usuário** como código de acesso é feita por três motivos: primeiro, o dado usuário restringe o acesso dos serviços aos usuários do SPP2. O segundo motivo é a garantia de acesso aos serviços do módulo SINGOG por qualquer

usuário do SPP2, divulgando-se apenas a interface desses serviços. O terceiro e último motivo é que a verificação da idoneidade do usuário é um processo eficiente pois o módulo SINGOG está localizado no *host*, possuindo acesso direto ao sistema operacional do mesmo, responsável por essa verificação.

Ainda em relação ao uso do dado **Usuário**, deve-se observar que o mesmo somente é utilizado quando necessário. Exemplificando, o dado Usuário não é necessário e não é utilizado em pedidos de serviços de uso restrito da ferramenta Mirador;

3. Operação a ser executada pelo serviço: alguns serviços permitem a execução de mais de uma operação, sendo necessária a especificação da operação a ser executada. Por exemplo, o serviço de gerenciamento de tarefas do módulo SINGOG pode eliminar tarefas ou alterar a prioridade de execução de tarefas. Na figura 5.2 a operação que um serviço deve executar é estipulada por qualquer dado que envolva o termo Operação fluindo entre funções pertencentes a módulos distintos;
4. Dados necessários à execução do serviço: são os dados de que um serviço precisa para ser executado e correspondem aos dados na figura 5.2 que são entrada de uma função correspondente a um serviço, com exceção do dado **Usuário** e dos dados que envolvam o termo Operação.

A resposta de um serviço, quando existente, é composta apenas pelos dados resultantes da execução do serviço. Na figura 5.2 esses dados correspondem àqueles dados que são resultado do processamento de uma função.

Protocolo de Ativação do Mecanismo de Reinício de Nós por Hardware (MRNH)

A ativação do mecanismo MRNH é realizada informando-se ao mesmo apenas a sequência dos identificadores de nós que se quer reiniciar.

Protocolo Telnet

A interação com um nó é realizada através de uma pequena modificação no protocolo Telnet. A modificação é a inclusão do identificador do nó com o qual se quer interagir, sendo aplicável apenas na comunicação entre os módulos MIU e GINPi. A comunicação entre os módulos GIN e GINPi utiliza o protocolo Telnet sem modificação.

5.2.4 Bases de Dados da Ferramenta

As bases de dados da ferramenta são armazenadas pelo *host* pois esse é o elemento computacional onde o módulo SINGOG está localizado. Deve-se destacar que o módulo SINGOG abriga as funções responsáveis por acessar ou gravar dados nas bases de dados, muitas vezes ativadas com grande frequência. Exemplificando, a função Atualizar Informações dos Nós é ativada por um único nó segundo a taxa *Toin*. Para um número maior de nós, a função é ativada com frequência maior, pois cada nó requisita a atualização de suas informações com uma taxa *Toin*.

A descrição da organização de cada uma das bases de dados utilizadas pela ferramenta encontra-se na seção 4.2.3.

5.3 Projeto da Interface da Ferramenta

Esta seção apresenta o projeto da interface da ferramenta Mirador com o usuário, destacando-se as diretrizes que foram seguidas para o projeto e apresentando-se os principais menus e janelas.

5.3.1 Diretrizes do Projeto da Interface da Ferramenta

O projeto da interface da ferramenta seguiu as seguintes diretrizes:

1. Organizar a interface de acordo com o padrão de interface (formato de menus e posicionamento de mensagens, por exemplo) de programas amplamente utilizados, tais como o Word [WOR98] e o Netscape [NET98];
2. Apresentar dados de tarefas, mensagens de erros, carga e memória em tabelas, facilitando a comparação e a ordenação de informações. A ordenação de informações é feita através do pressionamento do *mouse* sobre a coluna da tabela em relação a qual deve ocorrer a ordenação;
3. Apresentar mensagens sobre a execução de operações (imprimir que está ocorrendo uma atualização de informações, por exemplo);
4. Apresentar mensagens de erro em janelas próprias para alertar o usuário;

5. Organizar funções de mesma natureza em um mesmo menu. Por exemplo, as funções para requisição da apresentação de informações de tarefas e da carga dos nós selecionados pelo monitoramento devem estar no mesmo menu.

5.3.2 Interface da Ferramenta

A interface da ferramenta está organizada de acordo com o diagrama apresentado na figura 5.4. As janelas e menus componentes da interface da ferramenta são apresentados em seguida juntamente com explicações, quando necessárias.

Organização Principal dos Menus e Interfaces

Menu Ativado sobre o Ícone de um Nó

Menu Ativado sobre Informações de Tarefas

Figura 5.4 – Organização da interface da ferramenta Mirador

Figura 5.5 – Janela principal

Na janela, a área em branco é para a apresentação dos ícones dos nós. Logo abaixo, em cinza, há uma área para impressão de mensagens e também para a apresentação de duas informações: os segundos que faltam para a próxima atualização (Time to refresh) e a taxa *Toin* (Nodes data update rate).

Figura 5.6 – Menu Session

Figura 5.7 – Menu Report

Figura 5.8 – Menu Control

Figura 5.9 – Menu Setup

Figura 5.10 – Janela Open

Figura 5.11 – Janela Node Date and Time

Figura 5.12 – Janela Node Error Messages

Figura 5.13 – Janela Node Memory Usage

Figura 5.14 – Janela Node Tasks

Figura 5.15 – Janela Node Utilization

Figura 5.16 – Janela SPP2 Configuration

Figura 5.17 – Janela Load Range Conventions (escala de 10%)

Figura 5.18 – Janela Load Range Conventions (escala de 20%)

Figura 5.19 – Janela Load Range Conventions (escala de 50%)

Figura 5.20 – Janela Command Console

Figura 5.21 – Menu Reset Node

Figura 5.22 – Menu User Manegament

Figura 5.23 – Janela Management Settings

Figura 5.24 – Janela Monitoring Settings

Figura 5.25 – Janela Services Timeout (*Taim Timeout*)

Figura 5.26 – Janela Add User

Figura 5.27 – Janela Remove User

Figura 5.28 – Janela Update User Data

Figura 5.29 – Janela SPP2 Users

Figura 5.30 – Menu ativado sobre o ícone de um nó

Figura 5.31 – Menu ativado sobre informações de tarefas

Figura 5.32 – Janela Change Priority

5.4 Implementação da Ferramenta

Esta seção apresenta os aspectos mais importantes da implementação da ferramenta Mirador. Apesar da ferramenta não ter sido completamente implementada, a seção fornece algumas diretrizes para a implementação de alguns módulos.

5.4.1 Aspectos Gerais de Implementação da Ferramenta

Na implementação dos módulos da ferramenta exigiu-se apenas a utilização da linguagem Java para o módulo que interage com o usuário (MIU), garantindo-se o acesso à ferramenta independentemente de plataforma de hardware ou de sistema operacional. Na implementação dos demais módulos utilizou-se a linguagem C.

Para a comunicação entre os módulos que envolvem o protocolo requisição de serviço/resposta, utilizou-se o mecanismo de comunicação RPC [STE90], descrito na seção 2.4.2. Não foram utilizados outros mecanismos (*sockets* [STE90], por exemplo) pois o mecanismo RPC é de uso simples, abstrai o protocolo requisição de serviço/resposta e incorpora representação única de dados para arquiteturas diferentes (XDR). Deve-se observar que o módulo de interface com o usuário deve poder ser executado por diferentes arquiteturas.

5.4.2 Implementação dos Módulos

Os próximos tópicos descrevem os aspectos mais importantes da implementação de cada módulo da ferramenta. Os módulos não implementados são também apresentados juntamente com alternativas para implementá-los.

Módulo de Interação com o Usuário (MIU)

Foi implementado no formato de um *applet* Java [JAV98], um conjunto de instruções que pode ser executado por um navegador Internet com suporte à linguagem Java.

Esse módulo é composto basicamente por dois *threads* [JAV98] (linhas de execução distintas para um mesmo programa). Um *thread* é responsável pela interação com o usuário e o outro *thread* executa o monitoramento de informações. A comunicação entre os dois *threads* ocorre através de variáveis compartilhadas (seção 2.4.2). A sincronização para acesso a essas variáveis é feita através de monitores e semáforos (Seção 2.4.2).

Servidor de Informações dos Nós e Gerenciador de Operações de Gerenciamento (SINGOG)

Esse módulo foi implementado como um servidor RPC. A única função que não foi implementada por esse módulo é a de gerenciamento de usuários.

Gerenciador do Nó Pi (GNPi)

Assim como o módulo SINGOG, esse módulo também foi implementado como um servidor RPC.

Monitor de Informações do Nó Pi (MINPi)

Para a implementação desse módulo efetuou-se pequenas alterações no código do programa Top [NUX98], necessárias ao envio de alguns de seus dados (informações do nó utilizadas pela ferramenta) ao módulo SINGOG.

Gerenciador de Interações com os Nós (GIN)

Esse módulo não foi implementado. Entretanto, para implementá-lo pode-se alterar o código do *daemon* de Telnet do Linux (*in.telnetd*) [NUX98] para que seja aceito o identificador de um nó com o qual se quer interagir. Com base nesse identificador o *daemon* contataria o módulo GINPi.

Gerenciador de Interação com o Nó Pi (GINPi)

A funcionalidade desse módulo já é implementada pelo *daemon* de Telnet do Linux, não sendo necessária a sua implementação.

Mecanismo de Reinício de Nós por Hardware (MRNH)

Esse módulo encontra-se em fase de implementação. É composto basicamente por uma rede que permite a qualquer nó reiniciar, através de sua porta serial, quaisquer outros nós.

5.4.3 Protocolos de Rede para a Comunicação entre Módulos

Na comunicação entre módulos através da rede de gerenciamento do SPP2 utilizou-se o protocolo UDP [STE90, COM95]. Por ser uma rede local sem roteadores, uma rede isolada e também bem administrada, a rede de gerenciamento possui uma baixa taxa de erros e não exige controle de sequência de pacotes, sendo possível a utilização do UDP.

Na comunicação entre módulos através de uma *internet* (figura 5.1) utilizou-se o protocolo TCP [STE90, COM95], que fornece controle de erro e de sequência de pacotes. O uso do protocolo TCP é justificado pela possibilidade de erros de transmissão.

5.4.4 Implementação das Bases de Dados

As bases de dados foram implementadas como arquivos. A única exceção é a base que mantém as informações dos nós, que por ser muito utilizada foi implementada para ser armazenada em memória principal. Em sua implementação foram utilizadas estruturas com alocação dinâmica para se minimizar a quantidade de memória necessária para mantê-la.

5.5 Testes e Resultados Obtidos

Os testes da ferramenta Mirador foram realizados no SPP2 do LCAD-ICMC-USP, que possui 8 nós de processamento. A figura 5.33 apresenta um instante de execução de um teste na ferramenta.

Figura 5.33 – Instante de execução de um teste na ferramenta Mirador

Os testes realizados destinaram-se a determinar o nível de atendimento dos requisitos da ferramenta (capítulo 3) e o correto funcionamento da mesma. A seguir são apresentados os principais testes a que a ferramenta foi submetida:

- Execução do módulo MIU em diferentes arquiteturas (PC, Sun Sparcstation e Macintosh) e em diferentes sistemas operacionais (Windows 95, Windows NT, Solaris, Linux e Mac OS);
- Execução do módulo MIU em computadores remotos, não pertencentes à rede local do LCAD;
- Monitoramento de informações requisitadas por um usuário segundo diferentes valores para *Taim* e *TaimTimeout*;
- Obtenção de informações dos nós segundo diferentes valores para *Toin* e *ToinTimeout*;
- Gerenciamento de tarefas e nós;
- Desligamento de nós para avaliar a detecção de inatividade dos mesmos pela ferramenta;

- Geração de carga nos nós, através da execução de programas, para avaliar a carga deles pela ferramenta;
- Comparação das informações de memória da configuração dos nós com as de memória obtidas pelo monitoramento para avaliar nós com memória subutilizada.

A ferramenta Mirador foi validada funcionalmente na execução dos testes mencionados acima. Entretanto, avaliou-se que sua operação depende do uso de valores adequados para *Toin*, *ToinTimeout*, *Taim* e *TaimTimeout* de acordo com o ambiente de uso da ferramenta e da máquina sendo monitorada.

Os valores de *Toin* e *ToinTimeout* devem ser especificados pelo administrador do SPP2 levando-se em consideração dois fatores: a taxa com que se deseja obter informações atualizadas dos nós e a carga média da rede de gerenciamento do SPP2.

Na especificação da taxa de obtenção das informações dos nós, *Toin*, deve-se ponderar que quanto menor o seu valor, maior a frequência de atualização das informações dos nós, maior o tráfego na rede de gerenciamento decorrente do uso da ferramenta e menor é o intervalo de tempo utilizado para a determinação da carga dos nós. No último caso deve-se observar que *Toin* é também um intervalo de tempo para a obtenção de uma estatística (carga) de um nó. Desse modo, valores pequenos para *Toin* (um segundo, por exemplo) tendem a avaliar a carga instantânea dos nós, o que pode não corresponder à situação dos nós durante um intervalo de tempo maior. Quanto maior o valor de *Toin* (20 segundos, por exemplo), menor é a frequência de atualização das informações dos nós, menor é o tráfego na rede de gerenciamento e mais acurada é a avaliação da carga média dos nós.

A determinação do valor de *ToinTimeout* deve ser feita com base no tráfego da rede de gerenciamento. Quanto maior o tráfego na rede de gerenciamento, maior deve ser o valor de *ToinTimeout* para que a ferramenta não indique a inatividade de nós que estão operacionais. Quanto menor o tráfego na rede de gerenciamento, menor pode ser o valor de *ToinTimeout* para se tolerar atrasos na atualização das informações dos nós.

Os valores de *Taim* e *TaimTimeout* devem ser especificados pelo usuário ponderando-se três fatores: a taxa com que se deseja atualizar as informações do monitoramento, a carga média da *internet* que liga o computador do usuário ao *host* e a frequência de execução de funções da ferramenta. A determinação de *Taim* e *TaimTimeout* ponderando-se os dois primeiros fatores possui as mesmas implicações da determinação de *Toin* e *ToinTimeout*, excetuando-se a que se refere a avaliação da carga dos nós. Quanto à determinação baseada na frequência de execução

de funções, deve-se observar que valores menores para *Taim* determinam uma dificuldade maior de se acessar funções que dependam de informações do monitoramento, atualizadas segundo a taxa *Taim*. A taxa *Taim* deve ser compatível com o tempo máximo que um usuário leva para acessar uma função. Exemplificando, se *Taim* for igual a um segundo, é difícil o usuário conseguir selecionar tarefas e executar operações de gerenciamento sobre elas durante esse intervalo de tempo pois as informações de tarefas já teriam sido atualizadas.

A experiência de uso da ferramenta demonstrou que ela preenche uma lacuna funcional existente entre as ferramentas da sua classe disponíveis para o SPP2. Mais que isso, a ferramenta possibilita, através de uma interface com o usuário de fácil utilização, operação muito mais simples que as outras ferramentas disponíveis. O cuidado na distribuição dos módulos de software entre os diversos elementos de hardware do sistema e a escolha adequada dos mecanismos e técnicas de implementação permitem grande flexibilidade de operação, possibilitando que a Mirador possa ser utilizada na maioria das máquinas paralelas do tipo MIMD com memória distribuída ou em redes de computadores.

Baseando-se nas funções da ferramenta Mirador já implementadas, torna-se oportuno compará-la com as ferramentas de monitoramento e gerenciamento de máquinas paralelas apresentadas neste trabalho. A tabela 5.2 apresenta essa comparação em relação às principais características de cada uma.

Característica/Ferramentas	Módulo Monitor do Victor	AIX PSSP	VT	bWatch	Procps-Pare	Mirador
Monitoramento de nós de processamento	√	√	√	√	√	√
Monitoramento de tarefas		√	√		√	√
Monitoramento da máquina paralela vinculado à execução de aplicações paralelas			√			√
Gerenciamento de nós de processamento		√				√
Gerenciamento de tarefas		√				√
Interação com nós de processamento		√				
Disponível para o SPP2				√	√	√
Interface Gráfica	√		√	√		√
Monitoramento e gerenciamento remotos e independentes de plataforma de hardware ou de sistema operacional						√

Tabela 5.2 - Quadro comparativo das ferramentas de monitoramento e gerenciamento de máquinas paralelas com a ferramenta Mirador

5.6 Considerações Finais

Neste capítulo foram descritos o projeto, a implementação e os testes da ferramenta Mirador. Uma comparação da ferramenta Mirador com as ferramentas de monitoramento e gerenciamento de máquinas paralelas abordadas neste trabalho também foi apresentada.

O projeto da ferramenta requereu a análise da arquitetura do SPP2 de modo que a organização da ferramenta pudesse fornecer a mínima interferência no funcionamento do SPP2.

A implementação da ferramenta baseou-se no desenvolvimento completo de módulos de software e também na modificação de programas disponíveis na Internet. Algumas alternativas de implementação para os módulos da ferramenta ainda não implementados também foram apresentadas.

Os testes da ferramenta validaram-na quanto à satisfação dos seus requisitos e demonstraram a necessidade de configurá-la adequadamente, de acordo com o ambiente de uso do SPP2 e do usuário, para que seu funcionamento seja eficiente.

A comparação da ferramenta Mirador com outras ferramentas apresentadas neste trabalho demonstrou que ela apresenta a maioria das características dessas e também incorpora uma característica própria: monitoramento e gerenciamento da máquina paralela independentemente da plataforma de hardware ou de sistema operacional que seu usuário utilize.

O próximo capítulo apresenta as conclusões pertinentes ao desenvolvimento da ferramenta Mirador. Nele são descritas as principais contribuições deste trabalho e também são propostos trabalhos relacionados à ferramenta Mirador implementada.

Capítulo 6

Conclusão

6.1 Considerações Iniciais

No capítulo 2 foram apresentados os principais conceitos e trabalhos relacionados com o processamento paralelo, arquiteturas paralelas, a arquitetura do SPP2, ferramentas para desenvolvimento de aplicações paralelas e ferramentas de monitoramento e gerenciamento de máquinas paralelas, algumas disponíveis para o SPP2. Fundamentada nesse estudo, foi feita no capítulo 3 a definição da ferramenta Mirador, abrangendo-se seus usuários e requisitos.

A especificação da ferramenta com base na análise de seus requisitos foi apresentada no capítulo 4. O projeto da ferramenta, atendendo à especificação da ferramenta, aos requisitos não funcionais e a algumas diretrizes adicionais, foi descrito no capítulo 5. Esse capítulo também descreveu a implementação da ferramenta e os testes utilizados para validá-la, acompanhados de uma comparação da ferramenta com outras ferramentas de monitoramento e gerenciamento abordadas neste trabalho.

Neste capítulo são apresentadas a avaliação final da ferramenta Mirador, as contribuições feitas e sugestões para trabalhos futuros complementares a este trabalho.

Na seção 6.2 é apresentada a avaliação final da ferramenta Mirador, descrevendo-se o grau de atendimento dos objetivos que a ferramenta deveria alcançar, as dificuldades encontradas em seu desenvolvimento e a experiência de uso da ferramenta.

Na seção 6.3 são sumarizadas as principais contribuições deste trabalho, enquanto que na seção 6.4 são descritas as extensões propostas para trabalhos futuros.

6.2 Avaliação Final da Ferramenta Mirador – Desenvolvimento e Utilização

Os objetivos da ferramenta Mirador, com exceção dos correspondentes a funções não implementadas, foram plenamente atendidos. Um usuário, utilizando a Mirador, pode monitorar

e gerenciar remotamente e independentemente de plataforma o SPP2 através de uma interface gráfica e de um navegador (*browser*) Internet com suporte a linguagem Java.

A experiência de uso da ferramenta, embora restrita até o momento, permitiu validar não somente o seu funcionamento, mas também a sua adequação aos propósitos iniciais, principalmente quanto à facilidade de uso e atendimento das necessidades do usuário do SPP2.

O desenvolvimento da ferramenta Mirador exigiu um trabalho substancial em todas as suas fases de desenvolvimento, devido principalmente à sua arquitetura distribuída, forte interação com detalhes de baixo nível do sistema operacional, operação remota e uso de novas tecnologias na sua implementação. Na definição da Mirador, foi feita a análise de ferramentas de monitoramento e gerenciamento de máquinas paralelas e também das necessidades dos usuários de aplicações paralelas. Alguns requisitos e necessidades foram incorporados à definição da ferramenta em outras fases do seu desenvolvimento, exigindo-se alterações em muitos dos requisitos iniciais.

O uso de técnicas de engenharia de software, embora possa poupar tempo de manutenção e testes do sistema, exigiu que o autor aprendesse como colocar em prática muitos conceitos teóricos aprendidos em sala de aula. Uma tarefa não muito simples de ser realizada pela primeira vez. A documentação do sistema gerada nesse processo é extremamente importante para a realização das funções ainda não implementadas e para a manutenção evolutiva do sistema. As funções não implementadas (gerenciamento de usuários e interação com os nós através do protocolo Telnet, por exemplo), podem ser facilmente integradas ao sistema com base no projeto da ferramenta e nas alternativas de implementação propostas (capítulo 5). O esforço para a sua consecução se resume ao esforço de programação necessário.

6.3 Contribuições deste Trabalho

O desenvolvimento da ferramenta Mirador resultou em algumas contribuições significativas, ressaltando-se as seguintes:

- A ferramenta permite, se comparada a outras ferramentas de sua classe, que usuários não familiarizados com um sistema operacional em particular ou com a multiplicidade de recursos de uma máquina paralela possam realizar o monitoramento e o gerenciamento de suas aplicações paralelas ou dos recursos dessa máquina através de uma interface gráfica e de uso simplificado;

- A ferramenta apresenta uma característica não disponível nas outras ferramentas de monitoramento e gerenciamento de máquinas paralelas abordadas no trabalho: o usuário pode monitorar e gerenciar uma máquina paralela, no caso o SPP2, remotamente e independentemente de plataforma de hardware e de sistema operacional;
- A implementação de módulos da ferramenta Mirador que oferecem serviços possibilita a construção de outras ferramentas que utilizem esses serviços, característica não apresentada pelas demais ferramentas apresentadas neste trabalho;
- Para a implementação de funções complexas da ferramenta foram realizadas apenas pequenas modificações em programas disponíveis na Internet, não se exigindo a completa implementação das mesmas. Essa técnica de implementação sugere que no processo de desenvolvimento de um programa deve-se avaliar também os custos de mudanças em programas já disponíveis. Muitas vezes esses custos são ínfimos se comparados ao completo desenvolvimento do programa;
- A ferramenta Mirador apresenta uma característica inovadora se comparada às outras ferramentas de monitoramento e gerenciamento disponíveis para o SPP2: as informações dos nós são enviadas freqüentemente pelos nós a um módulo de software no *host*, responsável por fornecê-las quando são requisitadas. As outras ferramentas exigem que as informações sejam buscadas nos nós sempre que são necessárias. Essa característica da ferramenta Mirador determina um uso mais racional da rede de gerenciamento do SPP2;
- A ferramenta Mirador pode ser utilizada também para monitorar e gerenciar multicomputadores e redes de computadores com sistema operacional Linux, não necessariamente o SPP2. Se for um multicomputador ou uma rede de computadores com outro tipo de UNIX, requer-se apenas alterações nos códigos dos módulos escritos na linguagem C.

6.4 Sugestões para Trabalhos Futuros

Várias modificações evolutivas podem ser feitas na ferramenta para aumentar sua funcionalidade, gerando-se novas versões da mesma:

- Modificar a ferramenta para fornecer a taxa de ocupação da memória principal e da memória *swap* de um nó. A apresentação dos ícones dos nós pode, dessa maneira, ser realizada também em relação a uma dessas informações, assim como já é feito em relação às informações de carga dos nós;
- Implementar na ferramenta uma função destinada a fornecer informações ligadas ao hardware de um nó, tal como temperatura do processador, voltagem da bateria que alimenta o BIOS ou rotação do ventilador que refrigera o processador;
- Desenvolver um módulo que monitore constantemente as informações dos nós e envie mensagens (*e-mails*, por exemplo) a usuários avisando sobre a ocorrência de um evento. Exemplificando, esse módulo poderia avisar ao administrador que um nó não está operacional ou ainda informar a um usuário que uma determinada tarefa já foi executada.

As modificações acima podem ser feitas facilmente no sistema, exigindo-se apenas esforço adicional de programação. As sugestões seguintes envolvem um trabalho adicional intenso e podem ser base para futuros trabalhos de mestrado/doutorado:

- Integrar na Mirador um sistema para a execução de tarefas nos nós baseado em estatísticas de utilização desses nós, promovendo o balanceamento de carga de forma estática ou dinâmica.
- Integrar o sistema acima com as principais bibliotecas de comunicação (PVM, MPI) que não possuem mecanismos intrínsecos para o balanceamento de carga.
- Estender a funcionalidade da Mirador para incorporar as funções realizadas pela interface XPVM, por exemplo, aquelas relacionadas com o conceito de máquina paralela virtual. Com o uso de módulos de interface adequados, a Mirador poderia ser utilizada como uma interface gráfica universal para qualquer biblioteca de passagem de mensagens.

Referências Bibliográficas

- [ALM94] ALMASI, G.S.; GOTTLIEB, A. **Highly parallel computing**. 2.ed. The Benjamin Cummings Publishing Company, Inc., 1994.
- [BEO98] **Beowulf Software**. <http://www.beowulf.org/software/software.html>, 1998.
- [BOD95] BODEN, N. J. **Myrinet – A Gigabit-per-Second Local Area Network**. Myricom Inc., <http://www.myri.com>, 1995.
- [BWA98] **bWatch 1.0.2**. <http://www.sci.usq.edu.au/staff/jacek/bWatch/>, 1998.
- [CAL88] CALLAHAN, D.; COOPER, K.; TOREZON, L. Parascopes, a parallel programming environment. **J. Supercomput. Appl.**, v. 5, Winter 1988.
- [CAR89] CARRIERO, N.; GELERTER, D. How to write parallel programs. A guide to perplexed. **Parallel Computing**, v. 21, p. 323-357, Setembro 1989.
- [COM95] COMER, D. E. **Internetworking with TCP/IP**, V.I principles, protocols, and architecture; 3.ed. Prentice Hall, Inc. 1995.
- [CRA95] **Cray T3D system architecture overview**. Cray Research Inc., <http://www.cray.com>, 1995.
- [DOW89] MCDOWELL, C; HELMBOLD, D. Debugging concurrent programs. **ACM Computing Magazine**, v. 21, p. 593-622, Dezembro 1989.
- [DUN90] DUNCAN, R. A survey of parallel computer architectures. **IEEE Comput. Mag.**, v. 23, p. 5-16, Fevereiro 1990.
- [FOS95] FOSTER, Y. **Designing and building parallel programs: concepts and tools for parallel software engineering**. Addison-Wesley Publishing Company Inc., 1995.
- [GEI94] GEIST, G.; BEGUELIN, A.; DONGARRA, J.; JIANG, W.; MANCHECK, R., SUNDERAM, V. **PVM : Parallel Virtual Machine - A user's guide and tutorial for networked parallel computing**. The MIT Press, 1994.

- [HWA93] HWANG, K. **Advanced computer architecture: parallelism, scalability, programmability**. MacGraw-Hill, Inc., 1993.
- [ISP93] **IBM solutions: Scalable Power Parallel System 1**. IBM Corporation, <http://www.rs6000.ibm.com>, 1993.
- [JAV98] **The Source of Java Technology**. Sun Microsystems, <http://www.javasoft.com>, 1998.
- [JHN97] JOHNSON, M. K. **The Linux Kernel Hackers' Guide**. <http://khg.redhat.com/HyperNews/get/khg.html>, 1997.
- [JOH92] JOHNSON, P. **Human Computer Interaction**. 2.ed., McGraw Hill, 1992.
- [KEN92] KENNEDY, K. Software for supercomputers of the future. **Journal of Supercomputing**, v. 5, p. 251-262, Setembro 1992.
- [KIR91] KIRNER, C. Arquiteturas de sistemas avançados de computação. In: Anais da Jornada EPUSP/IEEE em Sistemas de Computação de Alto Desempenho, p. 307-353, 1991.
- [KOH97] KOHL, J. **XPVM: A graphical console and monitor for PVM**. <http://www.netlib.org>, 1997.
- [LOU82] LOUREIRO, H.A.; FERNANDES, L.E.P. **Laboratório de dispositivos eletrônicos**, Editora Guanabara Dois S.A. Rio de Janeiro – RJ, 1982.
- [MER94] MERLIN, J. **IDA - User's guide**, Southampton: University of Southampton, 1994.
- [MPI95] **EPCC MPI course**. EPCC Training and Education Center, <http://www.epcc.ed.ac.uk/epco-tec/package.html>, 1995.
- [NAV89] NAVAUX, P.O.A. Introdução ao processamento paralelo. **RBC- Revista Brasileira de Computação**, v. 5, n. 2, p.31-43, Outubro 1989.
- [NET98] **Netscape**. Netscape Netcenter. <http://www.netscape.com>, 1998.
- [NUX98] **Linux Online** – <http://www.linux.org/>, 1998.
- [PEA95] **IBM parallel environment for AIX version: operation and use**. v. 2, p. 111-190, Agosto 1995.

- [PER96] PEREZ, P.A.S. **P2D - Um ambiente de auxílio à paralelização de aplicações Fortran**. São Carlos: Universidade de São Paulo, 1996. 156p.
- [PRE92] PRESSMAN, R. S. **Software Engineering – A Practitioner’s Approach**. 3. ed., McGraw Hill, 1992.
- [PRO98] **Procps-Pare**. <http://www.sc.cs.tu-bs.de/pare/results/procps.html>, 1998.
- [PSS96] **IBM parallel system support programs for AIX: installation and migration guide**. Novembro 1996.
- [RAD98] RADAJEWSKI, J.; EADLINE, D. **Beowulf Howto**.
<http://www.sci.usq.edu.au/staff/jacek/beowulf/HOWTO/Beowulf-HOWTO-5.html#ss5.4>, 1998.
- [STE90] STEVENS, W.R., **Unix network programming**, Prentice Hall , Inc. Englewood Cliffs, New Jersey, 1990.
- [TAN92] TANENBAUM, A.S. **Modern operating systems**. 2.ed., Prentice Hall International, 1992.
- [TRI95a] TRINDADE, O.; MARQUES, E.; JEUKENS, I. A parallel architecture based on personal computers - requirements and definitions. In: Simpósio Nipo-Brasileiro de Ciência e Tecnologia, p. 203-212, Agosto 1995.
- [TRI95b] TRINDADE, O., MARQUES, E., JEUKENS, I., A parallel architecture based on personal computers - requirements and definitions - an overview. In: XV International Conference of the Chilean Computer Society, p. 479-490, Novembro 1995.
- [XMP97] **UML Notation Guide**.
http://193.174.33.104:7776/UMT/notation_guide_ch8.html, 1997.
- [WEL94] WELSH, M. **Linux installation and getting started**. LDP - LINUX Documentation Project , <http://www.sunsite.unc.edu>, 1994.
- [WOL95] WOLF, S. **A beginners’s guide to the IBM SP2**. Knoxville: University of Tennessee - Joint Institute of Computational Science, Setembro 1995.
- [WOR98] **Word for Windows 97**. Microsoft,
http://www.microsoft.com/products/prodref/188_ov.htm, 1998.

- [XMP97] **XMPI: A run/debug GUI for MPI.** Ohio Supercomputer Center,
<http://www.osc.edu/LAM/lam/xmpi.html>, 97.

Apêndice A

Dicionário de Dados do DFD da Ferramenta Mirador

Os dados relativos ao DFD da ferramenta Mirador são apresentados a seguir em ordem alfabética:

- Acesso Não Permitido: usuário não pode acessar o sistema;
- Apresentar Informações/Executar Funções: apresenta informações do monitoramento e executa funções da ferramenta;
- Armazenar Configuração do Usuário: armazena a configuração do usuário;
- Atualização Encerrada: fim da atualização das informações;
- Atualizar Informações dos Nós: atualiza informações dos nós na base de dados com informações dos nós;
- Chamadas ao S.O. para Obter Inf.: chamadas ao S.O. do nó para obter informações a serem processadas;
- Chamadas ao S.O./Dados Usuários: chamadas ao S.O. do *host* com os dados dos usuários;
- Chamadas ao S.O./Tids das Tarefas: chamadas ao S.O. do nó com os identificadores de tarefas; Chamadas ao S.O./Usuário: chamadas ao S.O. do *host* com a identificação do usuário;
- Conf. Administrador: configuração definida pelo administrador;
- Conf. Mirador ou Erro Leit.: configuração de uso da ferramenta ou erro na leitura;
- Conf. Monit.: configuração para o monitoramento;
- Conf. Nós: configuração dos nós;
- Conf. Sel. Monit.: configuração da seleção para o monitoramento;
- Conf. Sel. Monit./Inf.Requeridas: configuração da seleção para o monitoramento e

informações requeridas pelo usuário;

- Conf. Uso: configuração de uso da ferramenta;
- Conf. Usuário: configuração do usuário;
- Configuração da Mirador: configuração da ferramenta definida pelo administrador;
- Configuração do SPP2: configuração dos nós do SPP2;
- Configuração do usuário: configuração definida pelo usuário;
- Configurar Mirador para Uso pelo Usuário: configura a ferramenta para uso;
- Dados usuários ou Erro: dados dos usuários ou usuário não cadastrado;
- Dados Usuários: dados dos usuários;
- Data e Horário do Host: data e horário em vigência no *host*;
- Erro Armazen.: erro no armazenamento da configuração;
- Erro Leit.: erro de leitura;
- Erros da Operação Nós: lista de identificadores de nós e erros associados com a Operação Nós em cada nó;
- Erros da Operação Tarefas: identificadores de nós e erros associados ao gerenciamento das tarefas;
- Erros da Operação Tarefas: lista de identificadores de tarefas e erros associados com a Operação Tarefas em cada nó;
- Erros: erros da operação;
- Executar Funções/Monitorar Informações: executa funções e realiza o monitoramento de informações requeridas pelo usuário;
- Função/Dados: função a ser executada e dados associados;
- Gerenciar Nós: gerencia um conjunto de nós;
- Gerenciar Pi: gerencia o nó Pi;
- Gerenciar Tarefas de Pi: gerencia tarefas do nó Pi;
- Gerenciar Tarefas: gerencia tarefas de um conjunto de nós;

- Gerenciar Usuários: gerencia dados de usuários;
- Identif. Usuário: identificador do usuário;
- Inf. de tarefas, Memória, Data, Horário e Mensagens de Erro: informações do nó;
- Inf. do Monit.: informações do monitoramento;
- Inf. do Nó: informação do nó;
- Inf. dos Nós/ Data e Horário do Host: informação do nó e data e horário do *host*;
- Inf. Requeridas: informações requeridas pelo usuário;
- Informações : informações dos nós;
- Informações de Tarefas: informações de tarefas;
- Informações dos Nós: informações dos nós com data e horário de atualização;
- Informações: informações dos nós;
- Interagir com Nós: gerencia a interação com um conjunto de nós;
- Interagir com Nós: interage com um nó;
- Interagir com P_i : interage com nó P_i ;
- Ler Configuração de Nós do SPP2: lê a configuração dos nós do SPP2;
- Ler Configuração de Uso da Mirador: lê a configuração de uso da Mirador;
- Mecanismo de Reinício de Nós por Hardware: reinicia nó por hardware;
- Monitorar Informações Requeridas pelo Usuário: monitora as informações requeridas pelo usuário segundo a configuração da seleção para monitoramento e a configuração para monitoramento;
- Não Permitir Acesso: usuário não pode acessar a ferramenta;
- Nó ou Todos os Nós: obter a configuração de um nó ou de todos os nós;
- Nós : lista de identificadores de nós;
- Nós das Tarefas: nós em que as tarefas são executadas;
- Nós ou Todos os Nós: obter a configuração de um nó ou de todos os nós;

- Nós: lista de identificadores dos nós;
- Número de Nós: número de nós do SPP2;
- Obter e Processar Informação de *Pi*: obtém informações de *Pi* e calcula sua carga;
- Obter Informações do Monitoramento: obtém as informações do monitoramento;
- Operação Nó Encerrada: fim de uma operação de gerenciamento sobre nós;
- Operação Nó: operação de gerenciamento de nós a ser realizada;
- Operação Nós: operação a ser realizada sobre um conjunto de nós;
- Operação Tarefas Encerrada: fim de uma operação de gerenciamento de tarefas;
- Operação Tarefas: operação a ser realizada sobre um conjunto de tarefas;
- Operação Usuários: operação de gerenciamento de usuários;
- Permitir Acesso ou Não Permitir Acesso: possibilitar ou não o uso da ferramenta pelo usuário;
- Permitir Acesso: usuário pode acessar a ferramenta;
- Restringir Acesso a Usuários Cadastrados: permite que apenas os usuários do SPP2 acessem a ferramenta;
- Sequência de Caracteres: caracteres representativos da interação;
- Sistema Operacional do Host: sistema operacional do *host* do SPP2;
- Sistema Operacional do Nó *Pi*: sistema operacional do nó *Pi*;
- *Tids*: identificadores de tarefas;
- *Toin*: taxa de obtenção de informações dos nós;
- Usuário: identificação e senha do usuário;
- Verificar Idoneidade do Usuário: verifica se o usuário pode acessar a ferramenta.

Apêndice B

Dicionário de Dados do *Statechart* da Ferramenta Mirador

Os dados relativos ao *Statechart* da ferramenta Mirador são apresentados a seguir em ordem alfabética:

- Acesso Não Permitido: usuário não pode acessar o sistema;
- Acesso Permitido: usuário pode acessar o sistema;
- Apresentando Informações e Executando Funções: estado onde ocorre a apresentação de informações e o disparo de funções;
- Armazenador_de_Configuração: estado onde ocorre o armazenamento da configuração do usuário;
- Armazenagem Encerrada: fim do armazenamento da configuração do usuário;
- Armazenando Configuração: estado onde ocorre o armazenamento da configuração do usuário;
- armazenar_configuração: processar armazenamento da configuração do usuário;
- Atualização Encerrada: fim da atualização das informações;
- Atualizando Informações de P_i : estado onde ocorre a atualização das informações do nó P_i ;
- Atualizando Informações do Monitoramento: estado onde ocorre a atualização das informações do monitoramento;
- atualizar_informações: atualizar informações do monitoramento;
- atualizar_armazenagem: informar resultado da operação de armazenamento;
- atualizar_conf_nós: atualizar informações das configurações dos nós;
- atualizar_nós: atualizar informações dos nós;
- atualizar_tarefas: atualizar informações das tarefas;

- atualizar_usuários: atualizar informações dos usuários;
- Conf . Nós: configuração dos nós;
- Conf. Monit.: configuração para o monitoramento;
- Conf. Sel. Monit.: configuração da seleção para o monitoramento;
- Conf.Usuário: configuração do usuário;
- configuração encerrada: fim da configuração da ferramenta para uso pelo usuário;
- Configurando Mirador para uso pelo Usuário: estado onde ocorre a configuração da ferramenta para uso pelo usuário;
- Console: estado onde ocorre a interação com o usuário;
- Dados Usuários ou Erro: dados dos usuários ou erro da configuração dos usuários;
- Dados Usuários: dados dos usuários;
- Erro Armazen.: erro no armazenamento da configuração do usuário;
- Erro Atualização: erro na atualização da configuração do usuário;
- Erro leit.: erro de leitura;
- Erros Operação Nó: erros da operação de gerenciamento de nós;
- Erros Operação Tarefas: erros da operação de gerenciamento de tarefas;
- Fim Armazen.: fim do armazenamento;
- Fim da Atualização das Informações de Pi: estado onde a ferramenta aguarda *Toin* segundos até o próximo processamento de informações do nó Pi;
- Fim da Atualização: estado onde a ferramenta aguarda *Taim* segundos até nova obtenção de informações do monitoramento;
- Fim Gerenc. Nós: estado fim do gerenciamento dos nós;
- Fim Gerenc. Tarefas: estado fim do gerenciamento de tarefas;
- Fim Gerenc. Usuários: estado fim do gerenciamento de usuários;
- Fim Interação: estado fim da interação do usuário;
- Fim Leit.Conf.Nós: estado fim da leitura da configuração dos nós;

- Fim Monitoramento: estado fim do monitoramento;
- Fim: finalizar sessão com a tarefa;
- fim_arm: fim da operação do armazenamento da configuração do usuário;
- fim_int: fim da interação;
- fim_leit: fim da leitura de configuração de nós;
- fim_nós: fim do gerenciamento de nós;
- fim_tarefas: fim do gerenciamento de tarefas;
- fim_usuários: fim do gerenciamento de usuários;
- finalizar: fim do monitoramento;
- Gerenciador_de_Nós: estado onde ocorre o gerenciamento de nós;
- Gerenciador_de_Tarefas: estado onde ocorre o gerenciamento de tarefas;
- Gerenciador_de_Usuários: estado onde ocorre o gerenciamento de usuários;
- Gerenciando Nós: estado onde ocorre o gerenciamento de nós;
- Gerenciando Tarefas: estado onde ocorre o gerenciamento de tarefas;
- Gerenciando Usuários: estado onde ocorre o gerenciamento de usuários;
- gerenciar_nós: gerenciar um conjunto de nós;
- gerenciar_tarefas: gerenciar um conjunto de tarefas;
- gerenciar_usuários: gerenciar usuários;
- Inf. do Monit.: informações do monitoramento;
- Inf. Requeridas: informações requeridas pelo usuário;
- informações_atualizadas: fim da atualização das informações do monitoramento;
- Interagindo com Nó: estado onde ocorre a interação com um nó;
- interagir_com_nó: interagir com um nó;
- Interface_com_o_Usuário: estado onde ocorre a interação com o usuário;

- Leitor_de_Configuração_de Nós: estado onde ocorre a leitura da configuração de usuários;
- leitura de identificação e senha encerrada: fim do fornecimento da senha e da identificação do usuário;
- Leitura Encerrada: fim da leitura;
- Lendo Configuração: estado onde ocorre a leitura da configuração de nós;
- Lendo Identificação e Senha do Usuário: estado onde ocorre o fornecimento da senha e identificação do usuário;
- ler_conf_nós: ler configuração dos nós;
- Monitor: estado onde ocorre o monitoramento das informações requeridas pelo usuário;
- Nó: identificador do nó;
- Nós das Tarefas: nós em que as tarefas são executadas;
- Nós ou Todos os Nós: obter a configuração de um nó ou de todos os nós;
- Nós: lista de identificadores de nós;
- Obtendo e Processando Informações P_i : estado onde há o processamento de informações de P_i ;
- Obtendo Informações do Monitoramento: estado onde ocorre a obtenção das informações do monitoramento;
- Operação Nó Encerrada: operação de gerenciamento de nós encerrada;
- Operação Nós: operação a ser realizada sobre um conjunto de nós;
- Operação Tarefas Encerrada: operação de gerenciamento de tarefas encerrada;
- Operação Tarefas: operação a ser realizada sobre um conjunto de tarefas;
- Operação Usuários Encerrada: operação de gerenciamento de usuários encerrada;
- Operação_Nós: operação a ser realizada sobre um conjunto de nós;
- Operação_Usuários: operação de gerenciamento de usuários;

- Processador de Informações do Nó P_i : estado onde ocorre o processamento de informações do nó P_i ;
- *Processamento_Encerrado*: processamento de informações encerrado;
- *Seqüência de Caracteres*: caracteres representativos da interação;
- *Taim*: taxa de atualização das informações do monitoramento;
- *TaimTimeout*: atraso máximo para atualização das informações do monitoramento;
- *Tids*: identificadores de tarefas;
- Todos os Nós ou Nó: ler configuração de um nó ou de todos os nós;
- *Toin*: taxa de obtenção de informações dos nós;
- Verificando Idoneidade do Usuário: estado onde ocorre a verificação da idoneidade do usuário.