**Uma abordagem de predição da dinâmica
comportamental de processos para prover
autonomia a ambientes distribuídos**

Evgueni Dodonov

SERVIÇO DE PÓS-GRADUAÇÃO DO ICMC-USP

Data de Depósito:

Assinatura: _____

Uma abordagem de predição da dinâmica comportamental de processos para prover autonomia a ambientes distribuídos

Evgueni Dodonov

Orientador: *Prof. Dr. Rodrigo Fernandes de Mello*

Tese apresentada ao Instituto de Ciências Matemáticas e de Computação — ICMC-USP — para o Exame de Doutorado, como parte dos requisitos para obtenção do título de doutor em Ciências de Computação e Matemática Computacional.

USP – São Carlos
Maio de 2009

Agradecimentos

Antes de tudo, gostaria de agradecer ao Prof. Dr. Rodrigo Mello pela constante ajuda, orientação, dedicação e acompanhamento durante a realização deste Doutorado. Pela paciência infinita em tentar me transformar em um pesquisador de verdade. E, é claro, pela amizade.

Gostaria também de agradecer a meus amigos de Bauru, que conviveram comigo durante a realização deste doutorado, contribuíram imensamente para a minha dedicação a ele, e ajudaram em todas as etapas mais complicadas da minha vida: a Juan Falguera, Fernanda Sartori, Lilian Martins e Larisa Waldige. E a todas as pessoas de MStech, que conviveram comigo e me ajudaram a crescer profissionalmente e pessoalmente.

Gostaria de agradecer especialmente a Paulo Costa, que me ajudou em diversas etapas dessa jornada, que me apoiou na decisão de fazer o Doutorado, como também em todas as iniciativas, trabalhos, projetos e desafios.

Além disso, gostaria de agradecer a meus amigos de São Carlos: Terence, Samuel, Silvana, Elisangela, Lohan, Gi, Marcelo, Lu, Picé, Sandra, Thais, Raquel, e a todos os demais, que me apoiaram e ajudaram durante toda essa fase.

Também queria agradecer a meus amigos do ICMC/USP, que colaboraram para que este trabalho se tornasse possível: José Augusto Andrade Filho, Marcelo Keese Albertini, Matheus Lorenzo dos Santos, Renato Porfírio Ishii e, mais uma vez, Rodrigo Mello. Sem vocês este trabalho não seria possível!

E, é claro, gostaria de agradecer a Vanessa, que me acompanhou nas fases mais complicadas do desenvolvimento desta tese e, também, da minha vida; e a minha família, sem qual nada disso teria acontecido.

Além disso, gostaria de agradecer a todas as pessoas que auxiliaram direta ou indiretamente nesta tese.

Finalmente, queria agradecer ao Iron Maiden, Metallica, e outras bandas do meu *playlist*, por elaborarem composições musicais que colaboraram de forma indiscutível para a escrita de textos, artigos, realizações de experimentos e todas as outras atividades que precisam de concentração; ao *ViM*, por editar todos os textos escritos no decorrer desta tese de Doutorado; e ao *Linux*, sem o qual nada disso funcionaria direito.

Resumo

A evolução de sistemas distribuídos resultou em aumento significativo de complexidade para manutenção e gerenciamento, tornando pouco eficientes técnicas convencionais baseadas em intervenções manuais. Isso motivou pesquisas que deram origem ao paradigma de computação autônoma (*Autonomic Computing*), que provê aspectos de auto-configuração, auto-recuperação, auto-otimização e auto-proteção a fim de tornar sistemas auto-gerenciáveis. Nesse contexto, esta tese teve como objetivo prover autonomia a ambientes distribuídos, sem a necessidade de mudar o paradigma de programação e as aplicações de usuários. Para isso, propôs-se uma abordagem que emprega técnicas para compreensão e predição de dinâmicas comportamentais de processos, utilizando abordagens de sistemas dinâmicos, inteligência artificial e teoria do caos. Os estudos realizados no decorrer desta pesquisa demonstraram que, ao predizer padrões comportamentais, pode-se otimizar diversos aspectos de computação distribuída, suportando tomadas de decisão autônomas pelos ambientes. Para validar a abordagem proposta, foi desenvolvida uma política de escalonamento distribuído, denominada *PredRoute*, a qual utiliza o conhecimento sobre o comportamento de processos para otimizar, transparentemente, a alocação de recursos. Experimentos realizados demonstraram que essa política aumenta o desempenho em até 4 ordens de grandeza e apresenta baixo custo computacional, o que permite a sua adoção para escalonamento *online* de processos.

Abstract

The evolution of distributed systems resulted in a significant growth in management and support complexities, which uncovered the inefficiencies incurred by the usage of conventional management techniques, based in manual interventions. This, therefore, has motivated researches towards the concept of Autonomic Computing, which provides aspects of self-configuration, self-healing, self-optimization and self-protection, aiming at developing computer systems capable of self-management. In this context, this thesis was conceived with the goal of providing autonomy to distributed systems, without changing the programming paradigm or user applications. In order to reach this goal, we proposed an approach which employs techniques capable of modelling and predicting the dynamics of application behavior, using concepts introduced in dynamical systems, artificial intelligence, and chaos theory. The obtained results demonstrated that it is possible to optimize several aspects of distributed computing, providing support for autonomic computing capabilities to distributed environments. In order to validate the proposed approach, a distributed scheduling policy was developed, named *PredRoute*, which uses the knowledge about the process behavior to transparently optimize the resource allocation. Experimental results demonstrated that this policy can improve the system performance by up to a power of 4, and also requires a considerably low computational cost, which suggests its adoption for online process scheduling in distributed environments.

Sumário

1	Introdução	1
1.1	Contextualização do projeto MidHPC	2
1.2	Motivações e Objetivo	5
2	Computação Autônoma e Distribuída	7
2.1	Considerações iniciais	7
2.2	Conceitos sobre sistemas autônomos	8
2.3	Evolução da autonomia em ambientes distribuídos	11
2.4	Conhecimento sobre aplicações	17
2.5	Considerações finais	20
3	Abordagens para Predição de Comportamento de Processos	21
3.1	Considerações iniciais	21
3.2	Classificação de padrões comportamentais de processos	22
3.3	Abordagens de predição de comportamento	24
3.4	Considerações finais	30
4	Avaliação da dinâmica comportamental de processos	31
4.1	Considerações iniciais	31
4.2	Conceitos de sistemas dinâmicos	31
4.3	Avaliação de variações comportamentais	35
4.4	Desdobramento de séries	36
4.5	Considerações finais	42
5	Emprego da dinâmica comportamental de processos a fim de prover autonomia a ambientes distribuídos	43
5.1	Considerações iniciais	43
5.2	Objetivo	44
5.3	Metodologia	44
5.4	Estudos iniciais	45
5.5	Predição de acessos a dados como base para autonomia em ambientes distribuídos	48

5.6	Provendo auto-otimização a ambientes distribuídos por meio do emprego de dinâmicas comportamentais em escalonamento de processos	54
5.6.1	Problema de Escalonamento de Processos	54
5.6.2	Técnicas adotadas	57
5.6.3	<i>PredRoute</i> : política de escalonamento distribuído e dinâmico de processos	65
5.6.4	Experimentos e discussões	68
5.6.5	Avaliação da complexidade da abordagem proposta	77
5.7	Considerações finais	82
6	Conclusões	83
6.1	Contribuições do trabalho	84
6.2	Publicações	85
6.3	Trabalhos futuros	87
	Referências Bibliográficas	87

Lista de Figuras

2.1	Organização do comportamento dos processos	17
4.1	Ilustração de órbitas em sistemas dinâmicos	33
4.2	Cálculo de <i>Rescaled Range</i> do expoente de Hurst	36
4.3	Exemplo de variedade	37
4.4	Estudo de dimensões de separação e embutida: função logística e atrator de Lorenz	39
4.5	Ilustração do cálculo da dimensão de separação	40
4.6	Ilustração do cálculo da dimensão embutida	41
5.1	Arquitetura da rede ART-2A	46
5.2	Determinação da precisão adequada para agrupamento pela rede ART-2A	47
5.3	Modelo de Markov para <i>NAS Parallel Benchmark</i>	49
5.4	Predição com rede TDNN em função das características comportamentais de processos, extraídos por técnica de monitoramento periódico	50
5.5	Padrão de acesso de <i>NAS Parallel Benchmark</i>	52
5.6	Predição com rede TDNN em função das características comportamentais de processos, extraídos por técnica baseada em eventos	53
5.7	Exemplo de rede de interconexão e alocação de processos	56
5.8	Arquitetura da rede neural artificial RRBF	60
5.9	Arquitetura da rede neural artificial SONDE	61
5.10	Estrutura da rede SRRBF resultante	66
5.11	Arquitetura da política de escalonamento	66
5.12	Análise temporal caótica do <i>workload</i> de NERSC	70
5.13	Avaliações de latências de rede em 2005 e 2008	72
5.14	Tempo de execução médio: ambientes de <i>cluster</i>	74
5.15	Tempo de execução médio: ambientes de grade com 256 e 512 elementos	75
5.16	Tempo de execução médio: ambientes de grade com 1024 elementos	76
5.17	Custo de execução das políticas de escalonamento considerando conjunto de dados de NERSC	79
5.18	Custo de execução das operações de pré-processamento e treinamento da abordagem proposta	81
5.19	Custo de execução das operações de predição da abordagem proposta	82

Lista de Tabelas

4.1	Série temporal do atrator de Lorenz	42
4.2	Conjunto de dados do atrator de Lorenz reconstruído segundo a dimensão embutida e de separação encontradas ($m = 3$ e $\tau = 5$)	42
5.1	Valores de ρ adequados para aplicações de <i>NAS Parallel Benchmark</i>	49
5.2	Valores de lag adequados para aplicações de <i>NAS Parallel Benchmark</i>	50
5.3	Avaliação das propriedades caóticas de aplicações do <i>NAS</i>	51
5.4	Padrões de acesso do <i>NAS</i> , com $lag = 5$, predizendo 1 evento futuro	53
5.5	Comportamento dos processos das aplicações paralelas	55
5.6	Características dos computadores	55
5.7	Trecho do conjunto de dados GTC	69
5.8	Avaliação de expoente de Lyapunov e Hurst para aplicações do <i>NERSC</i>	69
5.9	Parametrização das aplicações de <i>NERSC</i>	71
5.10	Tempo médio de execução apresentado por diversas políticas de escalonamento (segundos)	77

Lista de Siglas

AMI	<i>Auto-Mutual Information</i>
ARMAX	<i>Auto-Regressive Moving Average Model with Exogeneous Inputs</i>
AR-SOM	<i>Auto-Regressive Self-Organizing Maps</i>
ART	<i>Adaptive Resonance Theory</i>
ATNN	<i>Adaptive Time-Delay Neural Network</i>
BGP	<i>Border Gateway Protocol</i>
BMU	<i>Best-Matching Unit</i>
CCLA	<i>Cascade-Correlation Learning Architecture (CCLA)</i>
CEC	<i>Constant Error Carrousel, conceito de circulação constante de erro utilizado pela rede LSTM</i>
CFS	<i>Completely Fair Scheduler</i>
CLAM	<i>Contextual Layered Associative Memory</i>
CPU	<i>Central Processing Unit</i>
DSM	<i>Distributed Shared Memory</i>
DSMSim	<i>DSM Simulator, simulador de políticas de DSM</i>
EGP	<i>Exterior Gateway Protocol</i>
ePCM	<i>Enhanced Partial Critical Path</i>
EP	<i>Elemento de Processamento</i>
FNN	<i>False Nearest Neighbors</i>
GCS	<i>Growing Cell Structure</i>
GNG	<i>Growing Neural Gas</i>
GPS	<i>Global Positioning System</i>
GTC	<i>Gyrokinetic Toroidal Code, aplicação do conjunto NERSC</i>
GSOM	<i>Growing Self-Organizing Maps</i>
GWR	<i>Grow When Required</i>
IBL	<i>Instance-Based Learning</i>
IP	<i>Internet Protocol</i>
HMM	<i>Hidden Markov Model</i>
HHMM	<i>Hierarchical Hidden Markov Model</i>
LAN	<i>Local Area Network</i>
LHMM	<i>Layered Hidden Markov Model</i>
LSTM	<i>Long Short-Term Memory</i>
MAN	<i>Metropolitan Area Network</i>
MIPS	<i>Million of Instructions per Second</i>
MPI	<i>Message Passing Interface</i>

MPLS	<i>Multiprotocol Label Switching</i>
MSE	<i>Mean Squared Error</i>
MidHPC	<i>Middleware for High Performance Computing</i>
NARX	<i>Nonlinear Auto-Regressive Exogeneous Model</i>
NAS	<i>NASA Advanced Supercomputing</i>
NERSC	<i>National Energy Research Scientific Computing Center</i>
NP	<i>Nondeterministically Polynomial</i>
OSPF	<i>Open Shortest Path First</i>
PAFS	<i>Parallel/distributed file system</i>
PGMS	<i>Prefetching Global Memory System</i>
PMEMD	<i>Particle Mesh Ewald Molecular Dynamics, aplicação do conjunto NERSC</i>
POSIX	<i>Portable Operating System Interface</i>
PBS	<i>Portable Batch System</i>
PredRoute	<i>Predictive Route, política de escalonamento que estende a RouteGA adicionando predição de eventos de comunicação entre processos</i>
QT	<i>Quality Threshold</i>
R/S	<i>Rescaled Range</i>
RBF	<i>Radial Basis Function</i>
RRBF	<i>Recurrent Radial Basis Function</i>
RCE	<i>Restricted Coloumb Energy</i>
RTT	<i>Round-Trip Time</i>
Route	<i>Política de escalonamento</i>
RouteGA	<i>Route with Genetic Algorithm Support, política de escalonamento que estende Route original</i>
RSDL	<i>Rotating Staircase Deadline Scheduler</i>
RSOM	<i>Recurrent Self-Organizing Maps</i>
SMP	<i>Symmetric Multi-Processing</i>
SOM	<i>Self-Organizing Maps</i>
SONDE	<i>Self-Organizing Novelty Detection</i>
SRNN	<i>Spiral Recurrent Neural Network</i>
SVCA	<i>Sinusoidally Varying Coefficients Autoregressive Model</i>
SVM	<i>Support Vector Machine</i>
SchedSim	<i>Scheduling Simulator, simulador de políticas de escalonamento de processos</i>
TDNN	<i>Time-Delay Neural Network</i>
TDRBF	<i>Time-Delay Radial Basis Function</i>
TISEAN	<i>Tlme SEries ANalysis, software para análise de séries temporais não lineares</i>
TKM	<i>Temporal Kohonen Maps</i>
TRBF	<i>Temporal RBF</i>
VOPP	<i>View-Oriented Parallel Programming</i>
WAN	<i>Wide Area Network</i>

Introdução

A disponibilidade de microprocessadores de baixo custo e a evolução das técnicas de comunicação motivaram o desenvolvimento de sistemas distribuídos, onde elementos de processamento (*EPs*) cooperam entre si para realizar uma mesma tarefa computacional (Buyya, 1999). Essa motivação demandou diversas pesquisas por novos temas, tais como escalonamento de processos, técnicas de comunicação, mecanismos de otimização de acesso a dados, e compartilhamento de informações (Baker, 2000).

Ao longo das últimas décadas, ambientes distribuídos evoluíram de máquinas paralelas para aglomerados de computadores (*clusters*), onde processos são alocados sobre *EPs* independentes e homogêneos, os quais executam o mesmo sistema operacional e são interconectados por meio de uma rede local de latência uniforme. Aglomerados motivaram o desenvolvimento de novas abordagens para a utilização eficiente de recursos compartilhados, resultando em pesquisas focadas em escalonamento distribuído, mecanismos de antecipação de acessos a dados (*prefetching*) e sistemas de memória compartilhada distribuída (Buyya, 1999).

O tratamento de problemas cada vez mais complexos levou à ampla adoção de aglomerados de computadores. Esses aglomerados foram, então, interconectados a fim de prover maior capacidade computacional, o que deu origem a uma nova arquitetura de computação distribuída denominada *grid computing*, ou grade computacional (Foster *et al.*, 2001). Ambientes de grade são compostos por recursos heterogêneos e geograficamente distribuídos, interconectados por meio de redes com latência variável.

A concepção de ambientes de grade, por sua vez, introduziu novos problemas relacionados ao uso eficiente de recursos distribuídos, considerando aspectos relativos à heterogeneidade, latência de acesso variável, possibilidade de falhas de comunicação, ampla distribuição geográfica dos elementos de processamento e a escalabilidade do sistema (Fitzgerald *et al.*, 1997; Foster

et al., 2001; Stockinger *et al.*, 2001; Allcock *et al.*, 2001). Todos esses fatores tornaram evidente a alta complexidade envolvida no gerenciamento e manutenção desses ambientes de grande porte. Dessa forma, técnicas de gerenciamento convencionais, baseadas em intervenções manuais e interações com usuários, mostraram-se pouco eficientes (Horn, 2001), motivando pesquisas com o objetivo de tornar o ambiente auto-gerenciável.

Diversas pesquisas foram, então, conduzidas a fim de prover auto-gerenciamento para ambientes distribuídos, as quais são baseadas no paradigma de Computação Autônoma (*Autonomic Computing*) (Horn, 2001; Salehie & Tahvildari, 2005; Arnautovic *et al.*, 2007; Parashar, 2007). Esse paradigma visa prover capacidade de auto-gerenciamento para aplicações e ambientes por meio de quatro aspectos básicos: auto-configuração, auto-recuperação de falhas, auto-otimização e auto-proteção (Horn, 2001; Parashar, 2007). Esses aspectos permitem ao sistema adaptar-se, de acordo com condições do ambiente, oferecendo maior desempenho e funcionamento contínuo, sem a necessidade de intervenções manuais.

As pesquisas conduzidas no sentido de prover autonomia para grades computacionais motivaram o projeto *MidHPC (Middleware for High-Performance Computing)*¹ (Andrade Filho *et al.*, 2008), que permite a execução de aplicações legadas sobre grades computacionais e automatiza aspectos de otimização, configuração, proteção e recuperação de falhas. Diversas pesquisas foram conduzidas dentro do projeto *MidHPC*, conforme apresentado a seguir.

1.1 Contextualização do projeto MidHPC

O projeto *MidHPC (Middleware for High Performance Computing)* visa oferecer autonomia a sistemas distribuídos. Para atingir tal objetivo, diferentes frentes de pesquisa foram abordadas para avaliar técnicas de comunicação entre processos, estudar e projetar mecanismos eficientes de escalonamento de processos e de acesso transparente a dados distribuídos (Ferreira & Mello, 2004; Dodonov *et al.*, 2005, 2006; Dodonov & Mello, 2006, 2007; Mello *et al.*, 2006a, 2007b; Ishii *et al.*, 2007; Andrade Filho *et al.*, 2008).

As primeiras pesquisas relacionadas ao projeto *MidHPC* avaliaram modelos de comunicação de processos, visando selecionar o mais adequado para ambientes distribuídos heterogêneos. Essas avaliações consideraram protocolos de rede e mecanismos de comunicação (Ferreira & Mello, 2004), comparando abordagens baseadas em troca de mensagens, chamadas remotas de procedimento e mecanismos de memória compartilhada distribuída. A partir desses estudos, concluiu-se que abordagem de memória compartilhada distribuída (*DSM, Distributed Shared Memory*) é a mais adequada para o projeto, uma vez que oferece transparência de acesso a dados e, ao mesmo tempo, considera aspectos de consistência e distribuição. Dessa forma, processos visualizam a memória compartilhada da mesma forma que a local, não havendo

¹Projeto de auxílio pesquisa individual *Fapesp* – processo número 04/02411 – 9, e projeto CPNq Universal 470739/2008 – 8 – Edital MCT/CNPq 14/2008.

necessidade de alterações de código e paradigma de programação.

Essas pesquisas motivaram, inicialmente, o estudo de técnicas aplicadas a sistemas de memória compartilhada distribuída, visando avaliar desempenho e viabilidade de sua utilização em ambientes de grade. No decorrer desses estudos, percebeu-se que a eficiência de acessos a dados distribuídos varia, significativamente, em função da localização geográfica, tipo e padrão de acesso, e estrutura das aplicações. Com isso, experimentos foram realizados para avaliar parâmetros funcionais (tais como latência de acesso, variações da taxa de transferência e perda de pacotes) de ambientes distribuídos reais. Esses experimentos consideraram redes locais (*LANs*), metropolitanas (*MANs*) e de longa distância (*WANs*), utilizando os protocolos de comunicação comumente adotados em sistemas distribuídos (*TCP* e *UDP*), e técnicas otimizadas para transferência de dados (*Zero-Copy* e *Sendfile*) (Dodonov *et al.*, 2005). A realização desses experimentos permitiu quantificar a influência da localização geográfica na variação de latência e vazão de dados entre *EPs* distribuídos. Resultados obtidos foram, posteriormente, utilizados para compor um modelo estocástico de comportamento de latência em diferentes redes. Esse modelo pode ser adotado em simulações comportamentais de sistemas de memória compartilhada distribuída sobre redes heterogêneas e, também, para simular comunicação entre aplicações.

Paralelamente à medição de parâmetros funcionais de aplicações, foram realizados estudos adicionais a fim de propor mecanismos de escalonamento de processos. Essas pesquisas resultaram na proposta e implementação de um modelo de simulação que permite avaliar o comportamento de processos em função da carga de elementos de processamento, operações de entrada e saída, e tráfego em redes heterogêneas (Mello & Senger, 2006). A partir disso, foi desenvolvido um simulador, denominado *SchedSim* (Mello & Senger, 2006), que auxilia no projeto e validação de diferentes abordagens de escalonamento de processos (Mello *et al.*, 2006a, 2007b; Ishii *et al.*, 2007). Foram, então, propostas abordagens que aplicam, essencialmente, conhecimentos sobre o comportamento de aplicações paralelas, a fim de distribuí-las de forma otimizada. Resultados permitiram comprovar que tais abordagens reduzem, significativamente, o tempo de execução de aplicações.

Conhecimentos sobre o comportamento de aplicações paralelas, empregados nas pesquisas anteriores foram, inicialmente, obtidos de um modelo de carga de trabalho (*workload*) proposto por Feitelson *et al.* (1997), o qual é baseado na execução de ambientes de produção. Esses comportamentos foram estimados utilizando uma técnica de aprendizado baseado em instâncias (*IBL*, *Instance-Based Learning*) (Aha *et al.*, 1991), a qual avalia tendências predominantes na execução de processos, com base em históricos.

No decorrer dessas pesquisas, observou-se que o modelo de carga de trabalho de Feitelson caracteriza somente alguns tipos de aplicações. Dessa forma, a fim de suportar uma maior gama de aplicações reais, observou-se a necessidade de um mecanismo de extração de comportamento de aplicações. Esse fato motivou pesquisas adicionais que resultaram na

proposta de abordagens para extração de comportamento de processos, baseadas em monitoração contínua (Senger *et al.*, 2005; Mello *et al.*, 2005) e interceptação de eventos (Dodonov *et al.*, 2006; Dodonov & Mello, 2007).

Durante estudos para extração de comportamento, observou-se que o acesso a dados gera grande influência no tempo de execução de aplicações, o que tende a degradar decisões de escalonamento. Visando aumentar o desempenho de aplicações, pesquisas sobre novas abordagens de distribuição e acesso a dados foram conduzidas, as quais empregaram conhecimento sobre aplicações e características de ambientes distribuídos (Mello *et al.*, 2007a).

Os resultados dos trabalhos anteriormente apresentados motivaram a implementação de um protótipo do *MidHPC* (Andrade Filho, 2008), o qual permite a execução automática de aplicações, compiladas no sistema *GNU/Linux*, em ambientes de grade. Esse protótipo converte, de forma automática, *threads* em processos distribuídos, e utiliza uma abordagem de *DSM* para comunicação transparente entre eles. O escalonamento de processos é realizado com base no conhecimento sobre o comportamento histórico de aplicações. Bons resultados foram obtidos (Andrade Filho *et al.*, 2008), contudo, experimentos adicionais com o *MidHPC* permitiram observar que o conhecimento extraído pela técnica de aprendizado baseado em instâncias (*IBL*) é insuficiente para situações onde processos apresentam variações comportamentais. Nessas circunstâncias o escalonador tomaria decisões baseadas em tendências históricas médias de processos, o que não permite otimizar o tempo de execução em função do comportamento instantâneo.

Para ilustrar esse problema, considere uma aplicação composta pelos processos $\{p_0, p_1\}$. Suponha que p_0 e p_1 processam um grande volume de instruções P , em seguida consomem uma grande quantidade de recursos de comunicação C , onde $P \gg C$. Utilizando *IBL*, obter-se-ia o comportamento médio dos processos e, dado que a quantidade de comunicação é menor que a de processamento, eles tenderiam a ser escalonados em recursos geograficamente distribuídos, visando utilizar as capacidades de processamento disponíveis. No entanto, essa decisão degradaria o desempenho da etapa de comunicação. Nesse caso, poder-se-ia adotar técnicas de predição *online* de comportamento de processos, as quais permitiriam realizar o escalonamento inicial em regiões esparsas da grade, aproveitando-se da capacidade dos recursos disponíveis, contudo, antes da etapa de comunicação, migrar-se-iam processos para locais com baixa latência de comunicação.

A predição *online* de comportamento de processos seria, ainda, útil para definir aspectos como distribuição, replicação e antecipação de acesso a dados em ambientes de grade computacional, detecção de falhas e intrusões, e configuração automática do ambiente.

1.2 Motivações e Objetivo

Os aspectos anteriormente apresentados motivaram esta tese de doutorado que apresenta um estudo sobre o comportamento de processos e propõe uma abordagem de compreensão e predição *online* de eventos. Essa abordagem é aplicada a fim de oferecer autonomia, ou seja, capacidade de auto-gerenciamento, para sistemas distribuídos heterogêneos. Sistemas distribuídos podem beneficiar-se dessa abordagem em aspectos tais como otimização de escalonamento, distribuição, replicação e antecipação de acesso a dados, detecção de falhas e intrusões e configuração automática de serviços.

A abordagem proposta contempla as etapas de extração e pré-processamento de conhecimento como fases auxiliares para predição. A primeira define técnicas para obter informações sobre o comportamento de processos e a segunda, métodos para caracterizar as informações mais relevantes a serem previstas. Nesse contexto, esta tese avaliou diversas técnicas estocásticas e baseadas em inteligência artificial, empregando-as em conjunto com conceitos de sistemas dinâmicos e teoria do caos para compreender, modelar e predizer acontecimentos futuros. Os resultados deste trabalho devem ser aplicados ao projeto *MidHPC*, a fim de auxiliar no desenvolvimento de técnicas de computação autônoma, tornando o ambiente auto-gerenciável.

Esta tese é subdividida nos seguintes capítulos:

- Cap. 2** – apresenta os principais conceitos de computação distribuída e sistemas autônomos, e realiza um estudo sobre a evolução das técnicas que visam prover autonomia a ambientes distribuídos;
- Cap. 3** – descreve as principais abordagens de classificação e predição de comportamento de processos propostas na literatura, utilizadas neste trabalho para extrair padrões representativos e realizar estimativas sobre eventos futuros;
- Cap. 4** – aborda conceitos de sistemas dinâmicos e teoria do caos, empregados neste trabalho para compreender e modelar tendências comportamentais de processos;
- Cap. 5** – descreve os estudos realizados no decorrer do desenvolvimento desta tese, propõe uma abordagem que utiliza predição de tendências comportamentais para prover autonomia a ambientes distribuídos, e avalia por meio uma política de escalonamento de processos;
- Cap. 6** – apresenta conclusões, contribuições decorrentes desta tese e planos para trabalhos futuros.

Computação Autônoma e Distribuída

2.1 Considerações iniciais

A área de computação distribuída tem evoluído, constantemente, ao longo das décadas (Hemker, 1984; Balram *et al.*, 1988; Luecke *et al.*, 1992; Yeo *et al.*, 2006; Hwang, 2008), motivada pela possibilidade de resolução de problemas computacionais cada vez mais complexos. Nessa evolução, duas abordagens essenciais para a construção de sistemas distribuídos têm sido estabelecidas (Hennessy & Patterson, 2002; Michael *et al.*, 2007): a expansão vertical (*scale-up*) e a horizontal (*scale-out*).

A abordagem vertical (Fet, 1995) visa a construção e expansão de sistemas centralizados. Ao longo do tempo, essa abordagem evoluiu de máquinas paralelas (Thelen, 2000; Kuehn & Smith, 1988) para processadores simétricos (*SMP*) (Hennessy & Patterson, 2002; Drosinos & Koziris, 2006), sistemas *multi-core* (Balakrishnan *et al.*, 2005; Creeger, 2005) e, recentemente, *many-core* (Seiler *et al.*, 2008). Além disso, outras técnicas têm sido utilizadas para aumentar o desempenho de elementos de processamento, tais como abordagens *Hyper-Threading* (Tian *et al.*, 2003) e virtualização de *hardware* (Smith & Nair, 2005; Aguiar *et al.*, 2008).

A expansão vertical oferece diversas vantagens, tais como baixa latência de comunicação entre elementos de processamento, acesso eficiente à memória e simplicidade na utilização de recursos, uma vez que tarefas são tratadas no nível de *hardware* e no núcleo do sistema (Tannenbaum, 2001; Hennessy & Patterson, 2002). As desvantagens, por sua vez, incluem o aumento significativo do custo, devido à complexidade de *hardware*, o que limita a escalabilidade da solução (Flynn, 1999; Matzke, 1997).

A abordagem horizontal (Cooper, 1987), por sua vez, tem como objetivo agrupar elementos de processamento independentes, interconectados por meio de redes de comunicação

(Tanenbaum & Steen, 2001). A escalabilidade é superior à abordagem vertical, uma vez que é possível agregar, dinamicamente, novos elementos ao sistema (del Rosario & Choudhary, 1994; Kotz, 2000). Duas principais arquiteturas se destacam nessa abordagem: aglomerados de computadores (*clusters*) e grades. Um aglomerado é definido por um conjunto de computadores independentes, geralmente homogêneos, interconectados por uma rede com latência uniforme, e agrupados para a realização de uma mesma tarefa computacional. A disponibilidade de recursos computacionais de alto desempenho, aliada à crescente complexidade dos problemas computacionais, motivaram a integração de *EPs* heterogêneos em larga escala, o que deu origem às grades computacionais (Foster *et al.*, 2001).

Grades são baseadas na suposição de que diversos computadores interconectados passam a maior parte do tempo ociosos, o que permite utilizá-los para executar tarefas distribuídas (Foster *et al.*, 2001). Essa abordagem pode aumentar, significativamente, a capacidade de processamento distribuído, sem nenhuma restrição teórica quanto ao número e arquitetura de elementos participantes. No entanto, o desenvolvimento de grades enfrenta diversos problemas, tais como configuração heterogênea de estações, possibilidade de operações desconectadas, grande quantidade de elementos de processamento e alta variação de latência de acessos (Dodonov *et al.*, 2005; Dodonov & Mello, 2006).

Pode-se destacar duas gerações de grades computacionais. A primeira visa a utilização de recursos distribuídos para a resolução de problemas específicos, e a segunda provê suporte para aplicações genéricas. Como exemplos da primeira geração pode-se citar os projetos *SETI@HOME* (Werthimer *et al.*, 2001) e *Distributed.net* (Distributed.net, 2006). Entre algumas das implementações da segunda geração estão o *Globus* (Foster & Kesselman, 1997), *XGRID* (Apple Computer, 2006), *OurGrid* (Andrade *et al.*, 2003), *InteGrade* (Goldchleger *et al.*, 2004), *EuroGrid* (Lesyng *et al.*, 2003; Wypychowski *et al.*, 2004) e *ProGrid* (Costa *et al.*, 2003).

Ao longo das últimas décadas, ambientes distribuídos passaram de dezenas para milhares de elementos de processamento independentes (Yeo *et al.*, 2006; Hwang, 2008). Esse crescimento resultou em considerável aumento de complexidade de gerenciamento e manutenção, tornando inviável a adoção de técnicas convencionais, baseadas em intervenções manuais. Esse fato motivou estudos sobre abordagens para tornar ambientes auto-gerenciáveis, dando origem à área de computação autônoma (Horn, 2001; Salehie & Tahvildari, 2005; Arnautovic *et al.*, 2007), a qual é apresentada a seguir.

2.2 Conceitos sobre sistemas autônomos

O conceito de computação autônoma foi introduzido por Horn (2001), que propôs a construção de sistemas auto-gerenciáveis, capazes de se manter em funcionamento sem a necessidade de intervenção manual. De acordo com Horn, o crescimento do poder computacional

nas últimas décadas, aliado à disponibilidade de dispositivos computacionais de baixo custo, aumentou de forma exponencial a complexidade envolvida no gerenciamento de sistemas (Tianfield & Unland, 2004). Com isso, abordagens tradicionais, baseadas em serviços e configurações manuais, tornaram-se inviáveis, requisitando cada vez mais recursos humanos e aumentando os custos operacionais envolvidos. A capacidade de auto-gerenciamento, por sua vez, permite esconder a complexidade interna do sistema, tratando, automaticamente, questões de configuração, otimização, proteção e recuperação de falhas.

Para ser considerado autônomo, um sistema computacional deve ser baseado em três princípios básicos (Horn, 2001):

1. **Flexibilidade** – o sistema deve ser capaz de processar requisições de usuário e realizar tarefas de forma independente da plataforma e configuração de hardware;
2. **Acessibilidade** – o sistema deve ser capaz de manter-se em funcionamento constante, protegendo-se de falhas e alocando recursos necessários para atender requisições;
3. **Transparência** – o funcionamento do sistema deve ser transparente ao usuário, o qual não precisa conhecer detalhes sobre a execução das operações.

Para isso, sistemas autônomos devem prover os seguintes aspectos (Parashar, 2007):

1. **Auto-configuração** – o sistema autônomo deve ser capaz de modificar sua configuração de forma automática, de acordo com alterações no ambiente. Conseqüentemente, ele se mantém em funcionamento mesmo com a ocorrência de mudanças drásticas ou inesperadas;
2. **Auto-otimização** – o sistema deve adaptar-se, constantemente, de forma a aumentar o seu desempenho;
3. **Auto-recuperação** – o sistema deve ser capaz de detectar e recuperar-se de falhas, minimizando prejuízos e retomando o fluxo normal de execução;
4. **Auto-proteção** – o sistema deve ter a capacidade de detectar, identificar e proteger-se de ataques e intrusões, mantendo sua segurança e integridade.

Além desses aspectos essenciais, sistemas autônomos eficientes devem, ainda, contemplar alguns tópicos complementares (IBM, 2001; Kephart & Chess, 2003), tais como *auto-identificação*, que indica que o sistema deve ter conhecimento sobre os seus componentes e suas inter-relações; e *auto-sustentação*, que sugere que, por mais complexo que seja o funcionamento do sistema, suas operações internas não devem ser expostas ao usuário, mantendo transparente sua complexidade.

O paradigma de computação autônoma tem demonstrado potencial para solucionar diversos problemas inerentes à computação distribuída (Parashar, 2007), tais como o gerenciamento de grande quantidade de elementos de processamento com configuração não-uniforme e ampla variedade de aplicações. Esses fatores motivaram diversas pesquisas de computação autônoma em ambientes de grade computacional (Parashar, 2007).

Agarwal *et al.* (2003) conceberam o projeto *AutoMate*, com o objetivo de investigar questões relacionadas à autonomia em ambientes de grade, abordando a complexidade, heterogeneidade e natureza dinâmica de aplicações distribuídas. As pesquisas realizadas nesse projeto resultaram na proposta de dois *frameworks*, denominados *Accord* (Liu *et al.*, 2005) e *Rudder* (Li & Parashar, 2006). O primeiro provê capacidade autônoma a aplicações, representando-as por conjuntos de componentes independentes, e gerenciando sua composição e interoperabilidade. O segundo utiliza agentes *peer-to-peer* para identificar e agrupar serviços computacionais presentes no ambiente, alocando-os de acordo com as necessidades das aplicações.

Champrasert & Farms (2006) propuseram uma arquitetura autônoma baseada em computação bio-inspirada, denominada *SymbioticSphere*. Nesse trabalho, conceitos de descentralização, autonomia, seleção natural, emissão de feromônios e simbiose são utilizados para prover autonomia a ambientes distribuídos, tendo como base colônias biológicas (por exemplo, colônias de abelhas). Dessa forma, aplicações ou serviços de *middleware* são vistos como entidades biológicas, e os conceitos de computação bio-inspirada são empregados para gerenciá-los de forma autônoma. Avaliações realizadas pelos autores demonstram que essa abordagem oferece bons resultados em termos de adaptabilidade e eficiência.

Questões de auto-recuperação de falhas foram estudadas por Deris *et al.* (2004), que propõem uma abordagem de alta disponibilidade para ambientes distribuídos, caracterizada por baixa utilização de largura de banda, alta escalabilidade e recuperação automática de falhas. Nesse trabalho, falhas de acessos são detectadas e evitadas automaticamente, utilizando acessos redundantes a dados localizados em elementos de processamento independentes.

Questões de alocação automática de recursos em grades foram abordadas por Omar *et al.* (2006). Nesse trabalho, a demanda por recursos é constantemente avaliada e, posteriormente, utilizada para estimar futuras requisições de aplicações. Um *middleware* de aprendizado, baseado em mapas auto-organizáveis (*SOM, Self-Organizing Maps*) (Kohonen *et al.*, 2000), é empregado para caracterizar o uso de recursos das aplicações, alocando-os de forma automática conforme necessário.

Além disso, diversas outras pesquisas, focadas em oferecer autonomia para ambientes computacionais, foram realizadas mesmo antes da concepção da área de computação autônoma. As pesquisas mais representativas são apresentadas a seguir.

2.3 Evolução da autonomia em ambientes distribuídos

Embora o conceito de computação autônoma tenha sido recentemente introduzido (Horn, 2001), diversas abordagens relacionadas ao tema têm sido propostas ao longo das últimas décadas. Entre tais abordagens destacam-se as de escalonamento de processos, otimização e antecipação de acesso a dados, auto-configuração e auto-identificação de ambientes, auto-proteção e auto-recuperação de falhas.

As técnicas de escalonamento de processos evoluíram no decorrer das últimas décadas partindo de abordagens estáticas, baseadas em compartilhamento de tempo, para formas mais sofisticadas, adequadas a diversas classes de processos e sistemas (Mello, 2003; Ganger *et al.*, 1993; Spinellis, 2008). O objetivo dessas abordagens é otimizar e sincronizar a utilização de recursos (Wong *et al.*, 2008), minimizando a ociosidade de sistemas e, conseqüentemente, reduzindo o tempo total de execução de processos.

As primeiras abordagens de escalonamento ainda têm sido utilizadas pela maioria dos sistemas operacionais, tais como *Linux*, *FreeBSD*, *Solaris* e *Microsoft Windows* (Spinellis, 2008). Essas abordagens raramente tratam questões relacionadas à autonomia na tomada de decisões. Recentemente, novas classes de escalonadores de processos foram introduzidas com o objetivo de melhorar a responsividade de sistemas. Essas focam, principalmente, na troca de contexto entre processos, e são empregadas em ambientes de *desktop* ou sistemas de tempo real (Roberson, 2003). O objetivo dessa família de escalonadores é diminuir a latência de acesso a recursos e o tempo necessário para realizar trocas de contexto. Nessas circunstâncias, o sistema determina, de forma autônoma, a seqüência de operações futuras a serem realizadas pelos processos e aloca os recursos disponíveis de maneira uniforme entre tarefas. Enquanto essa abordagem pode resultar em tempo de execução superior, aumenta-se a interatividade de processos individuais, o que é mais adequado para aplicações que priorizam a interatividade em detrimento do tempo total de execução. Como exemplos dessas abordagens pode-se citar os escalonadores *CFS* (*Completely Fair Scheduler*) (Wong *et al.*, 2008) e *RSDL* (*Rotating Staircase Deadline Scheduler*) (LWN, 2007), presentes em versões recentes do *Linux*.

A adoção em larga escala de ambientes distribuídos motivou pesquisas sobre técnicas mais avançadas de escalonamento distribuído, as quais utilizam parâmetros adicionais na tomada de decisões autônomas. Dentre esses parâmetros estão a utilização de rede, localização de elementos de processamento, ociosidade e capacidade disponíveis, e contexto de execução de processos (Kocjan, 2002; Mello *et al.*, 2006a).

A primeira geração de escalonadores distribuídos foi baseada em políticas fixas, as quais utilizam informações fornecidas por usuários (tais como o comportamento esperado e estimativas do tempo de execução, características de elementos de processamento desejados para a alocação de processos, etc). Essas informações são, geralmente, coletadas por meio de uma linguagem de alto nível, ou informadas através de um portal de acesso (Zorzatto, 2005). Essa abordagem

gera bons resultados quando parâmetros são corretamente especificados, no entanto, quando imprecisos, essa abordagem pode causar atrasos significativos na execução, ou sub-utilização de recursos distribuídos (Mello *et al.*, 2006a). Como exemplos de sistemas que utilizam essa abordagem estão *PBS* (Barrett *et al.*, 2003), *Cosmic* (Chung, 1997), *Globus* (Foster & Kesselman, 1997) e *ProGrid* (Costa *et al.*, 2003).

Com o objetivo de aumentar a eficiência de escalonamento, diversos ambientes foram propostos, tais como *Condor* (Tannenbaum *et al.*, 2001) e *Stealth* (Krueger & Babbar, 1993). Eles visam maximizar, de forma automática, a utilização de processamento de computadores ociosos. Seus escalonadores avaliam, continuamente, a carga de computadores do ambiente e atribuem cargas conforme esses tornam-se ociosos. Essas abordagens oferecem características autônomas de configuração e otimização ao ambiente, sem necessidade de intervenção explícita de usuários e administradores.

Outra abordagem que visa tornar autônomo um ambiente distribuído é empregada por sistemas como *Mosix* (Barak & La'adan, 1998), *Sprite* (Ousterhout *et al.*, 1988) e *ePCP* (Sudarsanam *et al.*, 2004). Essa abordagem busca distribuir, de maneira uniforme, o processamento entre todos os *EPs* e geralmente resulta em redução de latência e tempo de execução de aplicações. Ambientes são constantemente avaliados e seus processos migrados para recursos com maior ociosidade, oferecendo capacidades de auto-otimização e auto-configuração ao sistema.

Com a evolução do conceito de computação distribuída, o escalonamento de processos realizado por *Mosix* e *Sprite* deixou de ser eficiente, uma vez que suas estratégias de balanceamento de carga são focadas em ambientes homogêneos com latências uniformes. Tal abordagem não é eficiente para ambientes de grade e para redes *WAN* (Mello *et al.*, 2007a), uma vez que a configuração dos elementos de processamento, latência de acessos e disponibilidade de recursos apresentam grandes variações (Dodonov *et al.*, 2005). Esses fatores motivaram o desenvolvimento de escalonadores com capacidades autônomas, que distribuem processos de acordo com características individuais de elementos de processamento. Como exemplos dessas abordagens, pode-se citar o escalonamento realizado em ambientes heterogêneos gerenciados pelo *OurGrid* (Andrade *et al.*, 2003) e *MidHPC* (Andrade Filho *et al.*, 2008).

Ao longo da evolução de abordagens de escalonamento, observou-se que o comportamento de processos pode apresentar variações significativas (Devarakonda & Iyer, 1989; Feitelson *et al.*, 1997; Dodonov *et al.*, 2006; Dodonov & Mello, 2007), o que tende a comprometer a eficiência de abordagens autônomas (Mello *et al.*, 2006b, 2007b). Isso motivou pesquisas sobre comportamentos de processos, as quais buscam estudar e prever operações a fim de prover autonomia a escalonadores.

Uma das primeiras abordagens de escalonamento autônomo, baseado na predição de comportamento de processos, foi proposta por Devarakonda & Iyer (1989). Nesse trabalho,

a utilização de recursos de processamento, memória e entrada e saída é avaliada com o objetivo de identificar diferenças no comportamento de processos ao longo da execução, e escaloná-los de forma eficiente em função do comportamento estimado. A previsibilidade do comportamento de processos também foi abordada por Feitelson *et al.* (1997), que estudaram variações comportamentais de execuções repetidas de um mesmo conjunto de aplicações. Os autores concluíram que tais variações são pequenas, demonstrando que o histórico de execuções pode ser utilizado para estimar comportamentos e auxiliar no escalonamento de aplicações.

Questões relativas à auto-otimização em escalonamento, com base no comportamento de processos, também são discutidas por Gibbons (1997). Nesse trabalho, processos são classificados de acordo com suas características funcionais (tais como o nome do usuário que inicia a tarefa, nome do executável e fila de execução escolhida) e, posteriormente, avaliados, calculando o tempo de execução e utilização esperada de recursos. Resultados são empregados para configurar um mecanismo de escalonamento, que introduz aspectos de auto-otimização para o ambiente. Uma extensão dessa abordagem foi proposta por Smith *et al.* (1998), que apresenta uma técnica para predição do tempo de execução de aplicações similares.

O escalonamento autônomo, baseado em comportamento de processos, também foi estudado por Sevcik (1989), Harchol-Balter & Downey (1997), Brecht & Guha (1996) e Naik *et al.* (1997), resultando em diversas abordagens de otimização para auxiliar a tomada de decisões autônomas.

Técnicas de predição *online* de comportamento de processos foram estudadas em (Arpaci-Dusseau *et al.*, 1998; Silva & Scherson, 2000; Corbalan *et al.*, 2001). Esses trabalhos empregam modelo de Bayes (Cheng & Greiner, 2001) e lógica *fuzzy* (Freeman & Skapura, 1991) a fim de predizer o comportamento de processos durante execução. Uma abordagem similar é utilizada por Senger *et al.* (Senger *et al.*, 2003; Senger, 2004; Senger *et al.*, 2005), a qual prediz o comportamento futuro de processos com base em acontecimentos anteriores e propõe políticas de escalonamento baseadas em avaliação histórica (Senger *et al.*, 2003; Mello & Senger, 2004). Esses trabalhos provêm conhecimento que auxilia na tomada de decisões de escalonamento e dão suporte para demais mecanismos de auto-configuração e auto-otimização de ambientes. Conhecimentos sobre comportamentos de processos também são utilizados pelos escalonadores *Route* (Mello *et al.*, 2006a) e *RouteGA* (Mello *et al.*, 2007b), que tomam decisões de alocação de processos com base em estimativas obtidas por uma técnica de aprendizado baseado em instâncias.

Avaliações sobre latências entre elementos de processamento foram conduzidas por Ishii *et al.* (2007), em seu estudo sobre escalonamento de processos em grades. Resultados obtidos demonstraram melhor escalonamento com base na avaliação de comportamento de rede sobre técnicas convencionais. Essa abordagem necessita, contudo, de conhecimentos prévios sobre o comportamento de processos para tomar decisões eficientes de alocação.

Além do aspecto de escalonamento, diversas pesquisas apresentam a possibilidade de otimizar, automaticamente, o desempenho de operações de entrada e saída. Nesse contexto, visa-se antecipar acessos a dados (*prefetching*), pré-carregando-os em uma região de acesso rápido, tal como a memória primária ou *cache* (Dodonov & Guardia, 2002). A previsibilidade de acessos a dados foi estudada por diversos autores ao longo das últimas décadas (Kotz & Nieuwejaar, 1994; Nieuwejaar *et al.*, 1995; Dodonov, 2004), demonstrando que a maioria deles ocorre de forma repetitiva e seus padrões podem ser modelados. Esses padrões podem ser utilizados para prever e antecipar, automaticamente, requisições futuras, o que motivou o desenvolvimento de diversas abordagens analíticas (Cao *et al.*, 1995; Lei & Duchamp, 1997; Xi, 2001) para representar o comportamento e a dinâmica de acessos a dados.

Abordagens de *prefetching* foram aferidas na prática por diferentes autores (Kroeger & Long, 1999; Shah *et al.*, 2004; Byna *et al.*, 2004; Xi, 2001; Sakr *et al.*, 1996). Pesquisas realizadas por Kroeger & Long (1999) destacam que a maioria dos acessos a dados em sistemas distribuídos ocorre de forma seqüencial, sendo que previsões baseadas em histórico apresentam precisão entre 70 a 80%. Shah *et al.* (2004) também avaliaram a previsibilidade de acessos a arquivos independentes, obtendo precisão de até 80%. Outro estudo relacionado à previsão de acessos distribuídos foi realizado por Byna *et al.* (2004), com foco em acesso a dados realizados por aplicações *MPI*. O trabalho confirma que, ao determinar padrões de funcionamento de aplicações, pode-se prever de 80 a 90% dos acessos a dados. Além disso, Xi (2001) aplicou conceitos de previsão sobre bases de dados, concluindo que é possível prever até 80% dos acessos. Sakr *et al.* (1996) confirmam que se pode prever, de forma automática, até 90% dos acessos a páginas de memória. Esses resultados averiguaram a viabilidade das abordagens estudadas, e motivaram outras pesquisas (Barve *et al.*, 1997; Cortes & Labarta, 1999; Voelker *et al.*, 1998; Dodonov, 2004).

Tratando-se de estudos sobre o comportamento de processos, focados na previsão de acessos a dados, destacam-se os trabalhos de Cortes *et al.* (1996, 1997a,b). Esses utilizam um sistema de arquivos paralelo *PAFS* (Cortes *et al.*, 1997a) para estudar e avaliar diversas técnicas de previsão de acessos, e propõem abordagens de *cache* e *prefetching*. Além disso, Cortes & Labarta (1999) avaliaram diversos tipos de acessos a dados em *cache*, propondo a utilização de servidores dedicados (*cache servers*). Nesse trabalho, a previsão de acessos e o gerenciamento de *caches* são realizados de forma autônoma, aumentando, de forma transparente, o desempenho do ambiente.

Entre os trabalhos relacionados a técnicas de auto-otimização em ambientes paralelos, destacam-se os trabalhos de Kotz, que tratam da avaliação e previsão de comportamento de aplicações (Kotz, 1991; Kotz & Ellis, 1991; Kotz & Nieuwejaar, 1996). Técnicas de reconhecimento de padrões seqüenciais de acesso são discutidas por Kotz & Ellis (1991, 1993), resultando na proposta de um algoritmo que determina, automaticamente, o tipo e padrão de acesso da aplicação. Os trabalhos também avaliaram a eficiência da organização de dados em

caches locais e distribuídas, com o objetivo de diminuir a latência de acessos (Kotz & Ellis, 1993; Kotz & Nieuwejaar, 1996).

Tópicos relevantes para a área de antecipação de acessos a dados são apresentados em (Voelker *et al.*, 1998; Kimbrel & Karlin, 1996; Kimbrel *et al.*, 1996). Esses trabalhos definem o *prefetching* como um mecanismo para redução do tempo ocioso e o utilizam para otimizar, automaticamente, a latência de acessos a dados. De acordo com experimentos realizados pelos autores, a predição e a antecipação de acessos permitem aumentar, significativamente, o desempenho do sistema. Ao determinar o padrão de acesso da aplicação, a antecipação de acessos pode ser realizada de forma automática, introduzindo noções de auto-otimização ao ambiente. Os trabalhos introduzem o sistema de predição e antecipação automática de acessos *PGMS (Prefetching Global Memory System)*, a fim de integrar *caches locais* de servidores em um único *cache global*, compartilhado entre todos elementos de processamento.

Estudos sobre automação de acessos também foram realizados por Barve *et al.* (1997), propondo dois algoritmos de *prefetching*, denominados *NOM* e *GREED*. O algoritmo *NOM* avalia, constantemente, o *look-ahead* global do sistema, e o *GREED* utiliza *look-ahead* local para determinar próximos acessos. De acordo com a previsão do algoritmo, a sequência de acessos pode ser reordenada, de forma automática, diminuindo o número de requisições e reduzindo a latência. Com isso, o ambiente torna-se auto-otimizável, onde questões de organização de acesso a dados são, transparentemente, tratadas. Uma abordagem similar foi proposta por Kallahalla & Varman (2001), onde diferentes prioridades são associadas para requisições de leitura de dados. Esse trabalho avalia o histórico de acessos e determina padrões de comportamento, os quais auxiliam a estimar eventos futuros e otimizar operações de entrada e saída.

Comparações entre diversas abordagens autônomas de predição de acessos a dados distribuídos foram realizadas por diversos autores (Kimbrel *et al.*, 1996; Sakr *et al.*, 1996, 1997; Dodonov, 2004). Em (Kimbrel *et al.*, 1996), uma série de algoritmos de *prefetching* foi avaliada por meio de *traces* de aplicações reais. Estudos sobre predição de acessos a páginas de memória foram realizados por Sakr *et al.* (1996, 1997), as quais abrangem técnicas de aprendizado de máquina e demonstram que a predição correta permite reduzir em até 3 vezes o número de acessos à memória. Em (Dodonov, 2004), foi proposta uma abordagem para *prefetching* que escolhe, automaticamente, a estratégia mais eficiente, observando padrões de acesso a dados. Dessa forma, pode-se otimizar a latência de acessos em diversas ordens de grandeza, antecipando operações e armazenando os dados em uma região de acesso rápido (como uma *cache*).

Técnicas de predição e antecipação de acessos a dados também podem ser aplicadas a sistemas de memória compartilhada distribuída (Bianchini *et al.*, 1996), reduzindo a latência de acesso a páginas de memória e o *overhead* de tratamento de consistência de dados. A avaliação de padrões de acesso a páginas permite otimizar, transparentemente, o desempenho de sistemas, conforme estudado por Monnerat & Bianchini (1998), onde um *DSM* adapta-se a padrões de acesso observados, resultando em desempenho até 155% superior em relação a

abordagens tradicionais.

Tratando-se de questões de auto-configuração e auto-identificação, pode-se mencionar diversas abordagens utilizadas em engenharia de tráfego e em roteamento dinâmico de pacotes na Internet (Fortz, 2000). Enquanto os primeiros protocolos de roteamento intra-redes, como, por exemplo, *EGP (Exterior Gateway Protocol)* (Rosen, 1982), dependiam de uma infra-estrutura fixa e pré-configurada, com o crescimento da Internet tal abordagem mostrou-se pouco flexível e inadequada. Isso motivou o desenvolvimento de novos protocolos, tais como *OSPF (Open Shortest Path First)* (Fortz, 2000; Moy, 1998), *BGP (Border Gateway Protocol)* (Feamster & Rexford, 2007) e, recentemente, *MPLS (Multiprotocol Label Switching)* (Marchese *et al.*, 2006; Wang & Li, 2008). Esses protocolos visam manter uma visão consistente sobre o estado global de conexões entre redes individuais, determinando melhores caminhos para mensagens e evitando gargalos e rotas lentas de forma autônoma.

Questões sobre recuperação automática de falhas também têm sido estudadas por diversos pesquisadores (Wende *et al.*, 2002; Rilling & Morin, 2005). Wende *et al.* (2002) apresentaram uma abordagem baseada em transações para manter, de forma automática, a consistência de dados distribuídos. Rilling & Morin (2005) propuseram um mecanismo de compartilhamento de dados para ambientes de grade que detecta e gerencia, automaticamente, falhas de comunicação e inconsistências de dados, provendo capacidades de auto-recuperação ao ambiente.

Com a adoção, em larga escala, de sistemas distribuídos, questões de auto-configuração foram trazidas à tona, com o objetivo de estudar a melhor forma para alocar, gerenciar e manter o ambiente em funcionamento. Com isso, diversas abordagens foram propostas, tais como conjuntos de destino (Martin *et al.*, 2003), que visa agrupar *EPs* de forma a garantir consistência de dados trafegados, e programação orientada a visões (*VOPP, View-Oriented Parallel Programming*) (Huang *et al.*, 2005), que estabelece conjuntos de elementos de processamento inter-dependentes, cuja consistência deve ser mantida. Questões de auto-otimização e auto-configuração em sistemas distribuídos também foram estudadas por Spessoto (2003), onde um mecanismo dinâmico é utilizado para ajustar *timeouts* de acesso a dados distribuídos em função da latência e carga do ambiente.

Com isso, observa-se, que, ao longo do tempo, várias abordagens para oferecer aspectos de autonomia a ambiente distribuídos foram concebidas. No entanto, para oferecer capacidades autônomas ao ambiente, é necessário conhecer suas características e funcionamento. Esses fatores motivaram estudos mais detalhados sobre o comportamento de processos, abrangendo a sua classificação, extração e representação, conforme apresentado na seção a seguir.

Figura 2.1: Organização do comportamento dos processos

2.4 Conhecimento sobre aplicações

O conhecimento sobre comportamento de processos é essencial para identificar características de aplicações, analisar o fluxo de execução e prever acontecimentos futuros. Com isso, esta seção tem por objetivo descrever algumas das principais abordagens utilizadas para classificar o comportamento de processos, ferramentas que analisam esses comportamentos e abordagens que podem transformar o comportamento observado em uma representação formal, passível de análise.

O comportamento de processos pode ser classificado de acordo com diversos parâmetros que representam suas características fundamentais. De forma geral, o conhecimento sobre o comportamento de processos pode ser organizado nos níveis global e isolado, conforme demonstrado na figura 2.1.

O nível global tem como objetivo o agrupamento de todas as informações sobre aplicações em execução no sistema, tendo como características comuns o tempo de execução e a carga total do sistema. Essas características podem ser organizadas em classes, por exemplo, agrupadas por nome do usuário ou da aplicação. Essa abordagem é amplamente utilizada em ambientes de grades, sendo a base de diversos modelos de *grid economy* (Abramson *et al.*, 2002).

O nível isolado, por sua vez, representa o comportamento da aplicação em função da execução de tarefas específicas, tais como a interação com usuários, caracterizando aplicações interativas e não-interativas (também conhecidas como processamento em lote). Esse nível classifica, também, o comportamento em relação à utilização de recursos computacionais, abrangendo aplicações *CPU-Bound*, voltadas para cálculos e processamento de dados, *IO-Bound*, com predominância de transferências e acessos a dados, e as voltadas para comunicação (*Communication Intensive*), caracterizadas por troca excessiva de mensagens entre elementos de processamento.

Além disso, pode-se classificar aplicações de acordo com o grau de paralelismo, determinando sua capacidade de utilização de mais de um elemento de processamento em paralelo. Nesse contexto, o grau de paralelismo representa o número de *EPs* que maximiza a eficiência da execução da aplicação. Da mesma forma, o grau de paralelismo mínimo determina a quantidade mínima de *EPs* necessária para execução da aplicação, e o paralelismo máximo representa o número máximo de elementos de processamento que pode ser utilizado de forma eficiente.

A classificação por flexibilidade representa a capacidade das aplicações adaptarem-se a ambientes distintos, caracterizando sua eficiência em relação às decisões tomadas pelo escalonador (Feitelson *et al.*, 1997). Uma aplicação inflexível necessita de uma configuração de ambiente pré-determinada, sendo que sua execução em um sistema diferente é inviável ou difícil. Aplicações flexíveis, por sua vez, podem ser classificadas em moldáveis e maleáveis. As moldáveis podem, somente no início da execução, adaptar-se ao ambiente, e as maleáveis permitem realizar adaptações durante toda a execução.

Tratando-se de aplicações com características *a priori* desconhecidas, diferentes abordagens podem ser utilizadas para determinar comportamentos. A primeira abordagem consiste na avaliação estática, onde uma aplicação é analisada sem a necessidade de execução. Outra estratégia, também conhecida como avaliação dinâmica, consiste na extração de comportamento durante execução, ou em geração de *traces* para posterior análise (Fang *et al.*, 2004).

A abordagem de avaliação estática é uma das técnicas mais antigas, sendo concebida na década de 1930 (Fischer, 1965). Originalmente, essa técnica foi aplicada às máquinas de estados finitos e máquinas de Turing, com o objetivo de determinar possíveis falhas que levariam a um estado de empasse, finalizando a execução de maneira imprevista. Atualmente, ela é utilizada em ferramentas como *Lint* (Johnson, 1977), *Blast* (Henzinger *et al.*, 2003) e *Valgrind* (Nethercote & Seward, 2007).

Com a evolução de sistemas computacionais, novas técnicas de avaliação estática foram propostas. Atualmente, entre tais técnicas, pode-se citar o método de verificação de modelo (Müller-Olm *et al.*, 1999; Giunchiglia & Traverso, 1999; Déharbe, 2003; Schuster, 2003), que consiste na redução de aplicações para uma representação formal por meio de um autômato; e o método de interpretação abstrata (Codish & Taboch, 1997; Loiseaux *et al.*, 1995), que consiste na avaliação do sistema modelando-o na forma de uma máquina de estados, onde cada declaração é representada por um estado de execução.

A técnica de avaliação dinâmica, por sua vez, permite investigar o comportamento de aplicações durante a execução, geralmente com o auxílio de ferramentas de monitoramento ou depuração. É possível dividir as técnicas de avaliação dinâmica nas de monitoramento contínuo e baseadas em eventos (Jain, 1991).

Técnicas de monitoramento contínuo realizam avaliação periódica do estado de apli-

cações em execução para determinar variações comportamentais. Essa técnica é, geralmente, mais simples de ser utilizada, uma vez que aplicações não precisam ser alteradas. Contudo, sua desvantagem está relacionada ao fato de que o monitoramento é contínuo, o que introduz *overhead* na execução. Outra desvantagem dessa abordagem está relacionada ao fato de não permitir a extração de comportamento de aplicações em estados de execução específicos. A abordagem de monitoramento contínuo é utilizada por diversos autores (Keskar & Leibowitz, 2005; Senger *et al.*, 2005), e é integrada em grande parte dos sistemas operacionais existentes (Wilding & Behman, 2005).

A técnica baseada em eventos avalia estados de execução de aplicações em pontos específicos, tais como chamadas de sistema e pontos de sincronização, o que resulta na extração de comportamento mais precisa, uma vez que apenas ações relevantes são capturadas. Para isso, é necessário adaptar ou instrumentar as aplicações, ou ainda empregar uma técnica de interceptação de eventos (Dodonov *et al.*, 2004, 2006) para extrair operações desejadas. Geralmente, essa técnica apresenta maior complexidade de utilização, necessitando de conhecimento prévio sobre o funcionamento das aplicações avaliadas. Além disso, a necessidade de adaptar aplicações para o processo de interceptação de eventos torna essa técnica mais complexa para adoção. Como vantagem dessa abordagem pode-se citar a precisão na determinação do comportamento de aplicações em períodos de execução específicos.

A técnica de avaliação dinâmica é utilizada em ferramentas como *strace* (Kirshner *et al.*, 2004), *ltrace* (Wilding & Behman, 2005), *GridBox* (Dodonov *et al.*, 2004) e *AppArmor* (Bauer, 2006). Essa técnica também é empregada para avaliar o comportamento de aplicações distribuídas, baseadas em abordagens de troca de mensagens (Buyya, 1999), por meio de ferramentas como *Intel Trace Tools* (originalmente *Vampir*) (Nagel *et al.*, 1996), *MPE* (Chan *et al.*, 1998), *MPICL* (Huband & McDonald, 2001) e *mpiP* (Vetter & Chambreau, 2008).

Após extrair o comportamento de processos, pode-se transformá-lo em uma representação formal, passível de análise. Para isso, empregam-se conceitos sobre séries temporais, onde seqüências de eventos observados na execução de processos são traduzidos para valores numéricos relativos a instantes de tempo. Por exemplo, o comportamento de um processo pode ser representado por meio de uma série de transições entre estados **em execução** e **ocioso**. Ao representar o estado **em execução** como 1, e estado **ocioso** como 0, a série temporal de comportamentos durante $n = 10$ intervalos de amostragem pode ser definida como $X = \{X_1, X_2, \dots, X_n\} = \{0, 1, 0, 1, \dots, 1\}$. Séries temporais resultantes podem ser modeladas e utilizadas para predição de eventos, e análise de relações entre eventos subseqüentes (Dodonov & Mello, 2006, 2007, 2008).

Dessa forma, o comportamento de processos pode ser extraído e, transparentemente, convertido para uma representação formal ou matemática. Com isso, pode-se empregar técnicas estocásticas, matemáticas ou baseadas em inteligência artificial para prever acontecimentos futuros. Isso motivou este trabalho de doutorado a estudar e avaliar diversas técnicas para

predizer comportamentos de processos.

2.5 Considerações finais

No decorrer dos anos, ambientes distribuídos passaram de máquinas paralelas para aglomerados de computadores e grades computacionais. Com isso, técnicas convencionais de gerenciamento mostraram-se pouco flexíveis e eficientes, motivando a concepção do paradigma de computação autônoma, onde aspectos de configuração, otimização, proteção e recuperação de falhas são tratados de forma transparente, sem a necessidade de intervenções manuais.

Os trabalhos apresentados neste capítulo, visam oferecer autonomia a ambientes distribuídos, trazendo a tona questões de auto-configuração, auto-otimização, auto-recuperação e auto-identificação. No entanto, a eficiência dessas abordagens depende de conhecimento sobre o ambiente e sobre os processos em execução, necessitando de técnicas para extrair e representar o comportamento de processos.

Enquanto diversas pesquisas foram realizadas com o objetivo de predizer o comportamento de processos, a grande maioria desses trabalhos avalia padrões de acesso simples, apresentando limitações para predizer comportamentos mais complexos (Dodonov & Mello, 2007). Isso motivou um estudo sobre técnicas de extração, classificação e predição de comportamento de processos (Dodonov & Mello, 2008), abordado no capítulo 3.

Abordagens para Predição de Comportamento de Processos

3.1 Considerações iniciais

Pesquisas realizadas ao longo das últimas décadas, apresentadas no capítulo anterior, demonstram que conhecimentos sobre o comportamento de processos e uso de recursos são essenciais para prover capacidades autônomas a sistemas distribuídos. Esse fato motivou diversas abordagens a extrair e estudar padrões comportamentais com o objetivo de estimar operações futuras e otimizar a tomada de decisões. A possibilidade de representar o comportamento de processos por meio de uma série temporal, por sua vez, tornou viável a aplicação de diversas técnicas estatísticas e matemáticas, com o objetivo de realizar predições de eventos (Dodonov & Mello, 2006).

A existência de diversas abordagens e técnicas, que podem ser utilizadas para classificar e prever padrões comportamentais, têm motivado diversas comparações, tais como *K. U. Leuven competition* (Suykens & Vandewalle, 2000), *Eunite* (Chen *et al.*, 2004), *Santa Fe competition* (Weigend & Gershenfeld, 1994) e *CATS Benchmark* (Lendasse *et al.*, 2004). Essas comparações confirmam que diversas abordagens têm o potencial de realizar predições eficientes, sendo que cada técnica oferece melhores resultados para situações específicas, o que inviabiliza a escolha de uma única abordagem. Isso motivou, no contexto desta tese, um estudo detalhado sobre diferentes técnicas¹ para auxiliar a predição de comportamento de processos, apresentado nas seções a seguir.

¹A descrição detalhada das técnicas abordadas a seguir encontra-se em (Dodonov & Mello, 2008).

3.2 Classificação de padrões comportamentais de processos

Diversos trabalhos evidenciam que a extração do comportamento de processos resulta em uma grande quantidade de informações repetitivas e similares, o que torna complexa a sua análise (Madhyastha & Reed, 1997; Feitelson *et al.*, 1997; Pavlovic *et al.*, 1999; Thacker & Mayhew, 1990; Santos *et al.*, 2007; Mello *et al.*, 2005; Senger *et al.*, 2005; Dodonov *et al.*, 2006; Dodonov & Mello, 2007). Isso motivou o projeto de técnicas, baseadas em abordagens de classificação e agrupamento de dados, para reduzir a dimensionalidade de dados e determinar padrões mais representativos. Algumas das principais técnicas são apresentadas a seguir.

A abordagem de agrupamento iterativo engloba técnicas estocásticas de k -médias (Bradley *et al.*, 1998; Fayyad *et al.*, 1998; Farnstrom & Lewis, 2008), *fuzzy c*-médias (Liao *et al.*, 2003; Pham & Prince, 1998; Leski, 2008) e agrupamento controlado (*QT*, *Quality Threshold*) (Heyer *et al.*, 1999; Jiang *et al.*, 2004). Os modelos de k -médias e *fuzzy c*-médias visam agrupar elementos em partições distintas. Para isso, eles são inicialmente particionados em k grupos, por meio do cálculo iterativo da distância entre todos os pontos do mesmo grupo. O algoritmo é executado até que o centróide de cada grupo tenha variação mínima. A técnica de *c*-médias é similar ao modelo de k -médias, com a diferença de que, ao invés de pertencer a um único grupo, cada amostra tem um grau de semelhança em relação a todas as partições. Nesse modelo, o centróide é calculado como sendo a média de todos os pontos presentes no sistema.

O método *QT* foi inicialmente proposto para o agrupamento de genes de cromossomos humanos (Heyer *et al.*, 1999). Enquanto ele requer maior poder computacional para atingir o resultado, seu funcionamento não depende da pré-determinação do número de grupos. O algoritmo inicialmente escolhe o diâmetro de agrupamento máximo para cada grupo e, após isso, encontra o mais apropriado por meio de uma análise de proximidade entre todas as amostras cobertas pelo raio. O valor de proximidade entre uma amostra e um grupo é calculado considerando a distância entre todos os pontos nele contidos.

Outra abordagem existente é a *SVM* (*Support Vector Machines*), que realiza aprendizagem linear supervisionada e é voltada para classificação, regressão de padrões (Schölkopf & Smola, 2001) e predição de elementos em séries (Hirose *et al.*, 2007). Considerando um plano multi-dimensional, composto por diversos grupos de elementos, o objetivo dessa abordagem é encontrar uma função linear que resulte em separação máxima entre grupos. Para isso, vetores de entrada são mapeados em um espaço amostral com dimensão superior, por meio de construção de um hiper-plano que maximize a distância entre elementos. Conseqüentemente, somente um subconjunto de pontos é associado ao hiper-plano de separação máxima.

Técnicas baseadas em mapas auto-organizáveis (*SOM*, *Self-Organizing Maps*) foram introduzidas por Kohonen (Kohonen *et al.*, 2000; Freeman & Skapura, 1991) com o objetivo de resolver o problema de classificação de dados de acordo com níveis de similaridade (Kaski & Oja, 1999; Kohonen *et al.*, 2000). *SOM* utiliza treinamento não-supervisionado, onde o neurônio

mais apto a determinado padrão de entrada é utilizado para representá-lo por meio de um mapa de pontos. Após mapeamento inicial de dados, a técnica é aplicada sobre um vetor de entrada de dimensões arbitrárias, contendo todos os pontos do mapa, com o objetivo de agrupar vetores similares por meio do processo *winner-takes-all*. Nesse processo, o neurônio com vetor de peso mais similar à entrada é declarado vencedor (*BMU*, *Best Matching Unit*) e tem seu peso ajustado, de forma a tornar o neurônio mais próximo ao vetor de entrada. Essa técnica também considera ajustes proporcionais de pesos em neurônios vizinhos ao vencedor.

Enquanto técnicas como *SOM* permitem classificação automática e não-supervisionada de dados, elas apresentam limitações relacionadas à configuração prévia de parâmetros topológicos e de treinamento. Com o objetivo de permitir uma classificação adaptativa, diversas redes neurais artificiais auto-expansíveis, ou seja, redes que possibilitam a criação de novos neurônios sob demanda, foram propostas (Burzevski & Mohan, 1996; Cheng & Zell, 2000), tais como *Cascade-Correlation Learning Architecture (CCLA)* (Fahlman & Lebiere, 1990; Squires & Shavlik, 1991; Baluja & Fahlman, 1994), *Growing Cell Structure (GCS)* (Kunze & Steffens, 1995), *Probabilistic GCS* (Vlassis *et al.*, 1997), *Growing Neural Gas (GNG)* (Kunze & Steffens, 1995; Hamker & Heinke, 1997; Fritzke, 1995), *Growing Self-Organizing Maps* (Alahakoon *et al.*, 1998; Damminda *et al.*, 2000), e *CLAM (Contextual Layered Associative Memory)* (Thacker *et al.*, 1997). Redes auto-expansíveis são caracterizadas por não necessitarem de configuração topológica prévia, uma vez que elas avaliam padrões de entrada de forma incremental, criando, automaticamente, novos neurônios para representar padrões de entrada distintos em iterações fixas.

Além das redes auto-expansíveis, outras redes neurais artificiais foram propostas com o objetivo de detectar variações comportamentais entre padrões. Dentre elas estão as redes voltadas para detecção de novidades. Essas redes são bastante similares às redes auto-expansíveis, agrupando, automaticamente, padrões similares durante a execução. Após isso, variações nos neurônios da rede são detectados, indicando uma novidade. Diferentemente das redes auto-expansíveis, essa família de rede permite a criação de novos neurônios em qualquer momento de execução, e não em iterações pré-definidas. Como exemplos dessa família de redes estão a rede *GWR (Growing When Required)* (Marsland *et al.*, 2002) e *SONDE (Self-Organizing Novelty Detection)* (Albertini & Mello, 2007).

Outra abordagem é definida pela técnica *Restricted Coulomb Energy (RCE)*, a qual é uma rede neural não-supervisionada projetada para classificar padrões de entrada sem criar conexões entre neurônios vizinhos (Dokur & Ölmez, 2002; Yuan & Xie, 2002). Essa rede utiliza séries de vetores protótipo para classes particulares de entrada. Caso nenhum dos vetores existentes seja suficientemente próximo ao valor de entrada avaliado, uma nova classe é criada e inicializada com o valor de entrada atual. Assim, não há conexões entre grupos e os protótipos não são modificados após serem criados.

A abordagem empregada na família de redes neurais artificiais *ART (Adaptive Re-*

sonance Theory) (Carpenter & Grossberg, 1987) é caracterizada por adotar um modelo não-supervisionado de aprendizado. Durante a etapa de agrupamento, a camada de comparação recebe um vetor de entrada e o transmite para a camada de reconhecimento, onde o neurônio mais similar (i.e., cujo conjunto de pesos é mais próximo ao vetor de entrada) é encontrado. Cada neurônio da camada de reconhecimento alimenta um sinal negativo, proporcional à semelhança do neurônio encontrado com o padrão de entrada, para outros neurônios da camada de reconhecimento, conseqüentemente inibindo a sua saída. Dessa forma, a camada de reconhecimento apresenta inibição lateral, permitindo com que os seus neurônios representem diversas categorias nas quais os dados de entrada são classificados.

A família *ART* é composta por várias redes. As versões mais simples são *ART-1*, que aceita somente valores binários no vetor de entrada, e *ART-2*, caracterizada pela capacidade de suportar valores de entrada contínuos (Carpenter & Grossberg, 1987). Devido ao baixo desempenho de *ART-2*, a rede *ART-2A* foi proposta (Carpenter, 1989; He *et al.*, 2004), resultando em uma classificação mais precisa e eficiente. Além disso, outras extensões da rede foram propostas, tais como *ART-3* (Carpenter & Grossberg, 1990), caracterizada pela possibilidade de inibição parcial de neurônios por meio de um mecanismo de regulação neuro-transmitida, *Fuzzy ART* (Carpenter *et al.*, 1991b), que emprega lógica *fuzzy* durante o reconhecimento de padrões, *ARTMAP* (Carpenter *et al.*, 1991a) e *Fuzzy ARTMAP* (Carpenter *et al.*, 1992), que utilizam mecanismos de retro-alimentação a fim de melhorar a precisão da classificação.

Outra abordagem é empregada pela rede neural *RBF* (*Radial Basis Function Neural Network*), amplamente utilizada para classificação de padrões e previsões de séries temporais (Bors & Pitas, 1996; Yee & Haykin, 1998; Billings & Zheng, 1995). Essa rede calcula funções de regressão a fim de representar padrões por meio de um conjunto de funções radiais. A rede *RBF* é composta por três camadas: uma de entrada, uma escondida com funções radiais de ativação, e uma de saída. O valor de saída da rede é calculado combinando os resultados das funções radiais com os pesos das conexões, estabelecidos durante treinamento.

Todas as técnicas apresentadas podem ser utilizadas para classificar ou agrupar padrões, sendo que cada abordagem tem suas vantagens e desvantagens (Dodonov *et al.*, 2006; Dodonov & Mello, 2007, 2008). No contexto deste trabalho, essas abordagens são utilizadas na etapa de pré-processamento, para extrair padrões mais representativos. Após essa etapa, os resultados podem ser utilizados por diversas técnicas de predição.

3.3 Abordagens de predição de comportamento

Pesquisas têm confirmado que o conhecimento sobre o comportamento das aplicações é essencial para otimizar o desempenho de sistemas computacionais (Dodonov *et al.*, 2006; Dodonov & Mello, 2006, 2007, 2008). No contexto deste trabalho, tal conhecimento é utilizado

para realizar predições sobre eventos, a fim de auxiliar na tomada de decisões de otimização em sistemas distribuídos.

Conforme descrito na seção 2.4, o comportamento de processos pode ser representado por meio de séries temporais, passíveis de análise. Pode-se utilizar diversas abordagens para realizar predições sobre essas séries, estimando acontecimentos futuros.

Uma das abordagens de predição mais empregadas é a de cadeias de Markov (Bolch *et al.*, 1998). Cadeias de Markov são utilizadas para realizar predições sobre séries de variáveis aleatórias. Uma cadeia de Markov é definida por uma sequência $X_1, X_2, X_3, \dots, X_n$ de variáveis, sendo que seu escopo é denominado espaço de estados e X_n denota o estado do processo no instante n . Em uma cadeia de Markov, uma transição para um próximo estado X_{n+1} depende apenas do estado atual X_n . Essa propriedade é conhecida como propriedade de Markov (Bolch *et al.*, 1998).

As cadeias de Markov são, geralmente, representadas por meio de grafos dirigidos, cujos vértices representam diferentes estados e arestas são rotuladas com probabilidades de transição. As probabilidades de transição entre estados são armazenadas em uma matriz de incidência, que é utilizada para estimar os próximos estados a partir do estado de execução atual. Nesse modelo, apenas o estado atual é considerado para predição de um próximo, o que limita sua eficiência para comportamentos mais complexos. Para contornar essa limitação, a abordagem conhecida como Modelo Markov Escondido (*HMM, Hidden Markov Model*) foi proposta.

No *HMM* um sistema é visto como uma cadeia de Markov com parâmetros desconhecidos e observáveis, sendo que o objetivo é determinar os desconhecidos a partir dos observáveis (Meng, 2003). Diferentemente do modelo de Markov tradicional, onde o estado é visível ao observador e as probabilidades de transições entre estados são os únicos parâmetros do sistema, no *HMM* um estado não é diretamente acessado. A distribuição de probabilidade de cada estado é calculada indiretamente, avaliando a sequência de símbolos gerados pelo sistema. Dessa forma, *HMM* permite realizar predições mais complexas, levando em consideração uma sequência de estados anteriores. Diversas extensões foram propostas para *HMM*, tais como *HHMM (Hierarchical HMM)* (Fine *et al.*, 1998) e *LHMM (Layered HMM)* (Adibi & Shen, 2001).

Filtros de Kalman (Kalman, 1960; Doblinger, 1998) representam outra abordagem de predição que considera valores em espaço contínuo. Essa abordagem assume que o estado verdadeiro de um processo $x(k)$ no intervalo k é uma evolução a partir de seu estado no intervalo anterior $k - 1$, possivelmente distorcido por um ruído, e emprega duas fases de funcionamento: a de predição, e a de atualização. Durante a fase de predição, informações do estado atual são utilizadas para prever o próximo. Na fase de atualização, o erro de predição é calculado, e o filtro adaptado.

Diversas extensões foram propostas para o filtro de Kalman, tais como a técnica de *Kalman Smoother* (Cohn *et al.*, 1994; Evensen, 2000), que introduz um modelo auto-regressivo

na predição de curto alcance e compara os resultados estimados com observações passadas para melhorar a predição de longo alcance (Särkkä *et al.*, 2007); o filtro estendido de Kalman, que utiliza funções diferenciais ao invés de lineares (Sum *et al.*, 1996); e *Weighted Bidirectional Multi-stream Extended Kalman Filter*, que emprega técnicas de *back-propagation* (Freeman & Skapura, 1991) e separação de dados de entrada em múltiplos fluxos distintos (Hu *et al.*, 2007). Atualmente, o modelo estendido do filtro de Kalman é utilizado em casos onde há necessidade de realizar predições sobre observações com ruídos, tais como sistemas de navegação e *GPS* (Kim & Iltis, 2008; Ding *et al.*, 2006).

Outras abordagens buscam prever o comportamento empregando técnicas de auto-regressão (Jain, 1991; Yaffee & McGee, 2000; Box & Jenkins, 1990). A auto-regressão linear é uma das técnicas mais utilizadas para representação de comportamentos (Yule, 1927; Jaditz, 1995), e visa descrevê-los por meio de um modelo matemático, que calcula o próximo estado da série em função de uma soma ponderada de termos passados. Modelos mais avançados, tais como *SVCA* (*Sinusoidally Varying Coefficients Autoregressive Model*) (Palacios-González, 2007), *NARX* (*Nonlinear Auto-Regressive Exogeneous Model*) (Box & Jenkins, 1990) e *ARMAX* (*Auto-Regressive Moving Average Model with Exogeneous Inputs*) (Box & Jenkins, 1990), estendem a técnica original de regressão para atender séries *não-lineares*. Esses modelos são particularmente eficazes em casos onde termos da série não apresentam correlações facilmente observadas.

Há ainda uma outra família de técnicas, conhecida como redes neurais artificiais recorrentes (Zhao & Shawe-Taylor, 1994). Essas redes apresentam conexões entre unidades na forma de um grafo cíclico direcionado (Erñanský *et al.*, 2007; Zhao & Shawe-Taylor, 1994), utilizado para armazenar eventos recentes (memória de curto prazo – *short-term memory*). Dentre essa classe está a rede de Elman (Kremer, 1995; Steijvers & Grunwald, 1996), caracterizada por introduzir camadas especiais de cópia, empregadas para representar um número arbitrário de estados anteriores.

Outra rede neural artificial recorrente é a de Hopfield (1988), proposta para atuar como memória associativa entre dados de entrada e saída. A associação é realizada com base em uma função de energia, que tende a convergir para estados estáveis de um sistema. A ocorrência de alterações no sistema resulta na queda do valor da função, até que seja atingido um mínimo. A superfície da função de energia é armazenada nos pesos dos neurônios da rede. O uso dessa função inicialmente motivou a adoção da rede de Hopfield para a solução de problemas de otimização (Caparrós *et al.*, 1997; Guo *et al.*, 2007; Ishii *et al.*, 2007), nos quais se pretende obter a melhor solução para certa situação, e predição (Nakano-Miyatake & Perez-Meana, 1997a,b; Scott *et al.*, 2000).

Outra família de técnicas é conhecida como máquinas de Boltzmann, a qual considera componentes estocásticos (Freeman & Skapura, 1991; Coughlin & Baran, 1991), e é tipicamente utilizada para predição (Liu, 1997) e otimização combinatória (Aarts & Korst, 1987; Faassen,

1997). De forma similar à rede neural de Hopfield, máquinas de Boltzmann utilizam uma função de energia. Durante o processo de aprendizagem, seus neurônios processam entradas e atualizam a temperatura global do sistema. Em seguida, eles são ajustados a fim de manter o equilíbrio térmico do sistema. Essa rede tem duas fases de execução, sendo que na primeira, ou positiva, neurônios são treinados, e na segunda, ou negativa, a rede se adapta aos padrões anteriormente observados.

Redes recorrentes convencionais utilizam apenas o conceito de memória de curto prazo, o que tende a reduzir a eficiência de predições de eventos ocorridos em intervalos maiores. Para solucionar esse problema, foi proposta a rede *LSTM (Long-Short Term Memory)* (Hochreiter & Schmidhuber, 1997; Gers & Schmidhuber, 2000; Hammerton, 2003; Ribeiro & Alquézar, 2002), que combina memória de curto prazo com predição de longo alcance. A técnica introduz o conceito de circulação constante de erro (*CEC, Constant Error Carousel*) para controlar variações de valores de erro de neurônios. Mecanismos de proteção de entrada e saída são empregados para evitar modificações indesejáveis em neurônios, mantendo seu conteúdo intacto, e utilizando-os para realizar predições de longo prazo. Dessa maneira, o neurônio resultante torna-se mais complexo e passa a ser denominado de célula de memória. Diversas células podem ser agrupadas em blocos, que compartilham as mesmas unidades de proteção.

Entre extensões da rede *LSTM* está o trabalho de Pérez-Ortiz *et al.* (2003), que propõe a utilização do filtro de Kalman para auxiliar na etapa de aprendizado, resultando em maior eficiência e precisão em ambientes com ruídos e menor custo computacional. Outras extensões introduziram o conceito de esquecimento de dados irrelevantes por meio de *forget gates* (Gers *et al.*, 2000), e inter-conexões entre células independentes (*peephole connections*) (Gers *et al.*, 2002), utilizadas para aumentar a precisão no reconhecimento de seqüências similares.

Enquanto a rede neural *RBF*, descrita anteriormente, pode ser utilizada para realizar predições (Yee & Haykin, 1998; Dodonov & Mello, 2007), a sua eficiência é limitada, uma vez que não considera o contexto do estado de execução atual e o histórico de transições entre estados (Zemouri *et al.*, 2003). A rede *RRBF (Recurrent RBF)* foi proposta (Zemouri *et al.*, 2003) a fim de solucionar tais questões. Ela estende a *RBF* tradicional, adicionando uma camada de neurônios recorrentes que provêm capacidade de memória dinâmica. Esses neurônios são utilizados para determinar a relevância de valores históricos no cálculo de uma próxima saída.

Outra família de redes neurais preditoras é conhecida como redes competitivas, caracterizadas pela abordagem *winner-takes-all* (Kaski & Oja, 1999), introduzida pela rede *SOM*. Essa família é representada por técnicas tais como as redes neurais artificiais *TKM, RSOM, AR-SOM* e *VQTAM*. A rede *TKM (Temporal Kohonen Maps)* tem por objetivo separar padrões de acordo com a ordem e o contexto de processamento (Chappell & Taylor, 1993). Além de agrupar padrões de entrada de acordo com a vizinhança, essa abordagem permite classificar diversos padrões em função do tempo. Para isso, é introduzido o conceito de memória de curto prazo, utilizado para armazenar o histórico de comportamento para cada elemento do mapa. A principal

diferença em relação à *SOM* original está na camada de saída. Enquanto *SOM* zera os valores de neurônios da camada de saída após encontrar o vencedor, a *TKM* ajusta, gradativamente, os valores dos neurônios, reduzindo a importância dos estados anteriores em função do tempo. Essa abordagem é, entretanto, limitada por não existir nenhum mecanismo de propagação de erros (*backpropagation*), sendo eficiente apenas para padrões de entrada simples.

Koskela *et al.* (1997) propuseram a rede neural *RSOM* (*Recurrent SOM*), que ajusta, de forma recorrente, valores de entrada em função da classificação de padrões históricos. Com isso, calculam-se diferenças vetoriais entre os valores obtidos e esperados. Outro método, denominado *AR-SOM* (*Auto-Regressive SOM*), foi proposto por Lampinen & Oja (1989), e representa nós do mapa por vetores auto-regressivos que descrevem a evolução do aprendizado. Dessa maneira, cada nó do mapa mantém um histórico que pode ser utilizado para prever futuras transições. Uma outra abordagem foi proposta por Barreto & Araujo (2004), denominada *VQTAM* (*Vector-Quantized Temporal Associative Memory*), a qual utiliza memória associativa aliada à classificação vetorial de dados realizada pela *SOM*.

Um grupo de redes neurais, inicialmente introduzido por Waibel *et al.* (1989), é conhecido como *TDNN* (*Time-Delay Neural Networks*), e é caracterizado por utilizar uma topologia voltada ao reconhecimento e a predição de sub-padrões independentes, com foco em representação eficiente de relações entre eventos temporais. Para isso, *TDNN* introduz atrasos nos padrões de entrada e adota um procedimento de aprendizado recorrente (*backpropagation*) para determinar o erro global. Uma vez que cada conexão é associada a um atraso correspondente, elas atuam como um mecanismo de memória de curto prazo, o que permite relatar e comparar um vetor de entrada às observações prévias, atualizando os pesos das conexões correspondentes. O procedimento é repetido até a rede convergir para a saída desejada, o que resulta em predição eficiente somente após um treinamento repetitivo. Entre as extensões da *TDNN* pode-se citar a rede *ATNN* (*Adaptive Time-Delay Neural Network*), que adapta, automaticamente, os atrasos da rede (Day & Davenport, 1993).

A abordagem baseada em introdução de atrasos nos padrões de entrada também pode ser integrada à rede neural *RBF* (Berthold, 1994; Mesbahi *et al.*, 2004). Berthold (1994) propõe a rede *TDRBF* (*Time-Delay Radial Basis Function*) visando unir o desempenho oferecido pela rede *RBF* com o reconhecimento de padrões temporais da *TDNN*. A rede converte suas entradas em uma série de vetores, representando a evolução do comportamento ao longo do tempo. Esses vetores são deslocados em tempo, representando a memória de curto prazo, e processados pela *RBF*. Uma outra abordagem é proposta por Mesbahi *et al.* (2004), conhecida como *TRBF* (*Temporal RBF*), que introduz blocos de atrasos para representar dependências temporais entre eventos.

A família de técnicas conhecidas como *Reservoir Computing* foi recentemente introduzida (Jaeger, 2007), caracterizada pela existência de um sistema dinâmico recorrente, denominado reservatório (*reservoir*), e composta por um grande número de nós interconectados

com conteúdo aleatório. Uma função de supervisão é utilizada para treinar o sistema, mapeando resultados desejados de acordo com a dinâmica interna do reservatório. A atualização dos estados internos é supervisionada durante a etapa de treinamento, de forma a guiar um mecanismo de produção de saídas. Assume-se, com isso, que as conexões entre neurônios da camada escondida se auto-organizam, produzindo resultados desejáveis. A maior desvantagem de *Reservoir Computing* encontra-se em sua complexidade espacial, que cresce exponencialmente em função do número de padrões avaliados (Jaeger, 2007), e em seu comportamento instável, que nem sempre converge para resultados desejados. Entre técnicas de *Reservoir Computing* estão *Echo State Networks* (Venayagamoorthy, 2007), *Liquid-state machines* (Verstraten *et al.*, 2005) e *Backpropagation-Decorrelation* (Jaeger, 2007).

Com o objetivo de contornar limitações da abordagem de *Reservoir Computing*, tais como grande complexidade espacial e comportamento instável, propôs-se a rede neural artificial *SRNN* (*Spiral Recurrent Neural Network*) (Gao *et al.*, 2007). A estrutura dessa rede é similar a *Reservoir Computing*, com a diferença de que uma rede recorrente, completamente interconectada, é utilizada ao invés de um reservatório. A técnica evita problemas de estabilidade por meio da parametrização de pesos de sua camada escondida, a qual contém neurônios, interconectados de forma circular, que compartilham pesos com conexões vizinhas. O funcionamento da rede é similar a *Reservoir Computing*, e os valores da camada escondida são treinados por uma função de supervisão.

Uma outra abordagem é empregada em redes neurais Bayesianas (Bouckaert, 1994; Zhang & Poole, 1994; Cheng & Greiner, 2001; Guo & Hsu, 2002), as quais utilizam o conceito de aprendizado baseado no modelo estatístico de Bayes (1763), que relaciona probabilidades condicionais e marginais da distribuição de variáveis aleatórias para atualizar conhecimentos em função de novas evidências. Essas redes podem ser utilizadas para calcular a probabilidade de acontecimentos futuros, avaliando probabilidades condicionais de eventos antecedentes. Formalmente, redes Bayesianas são representadas por grafos acíclicos, cujos vértices representam variáveis e arestas codificam dependências condicionais e indicam relações de precedência entre variáveis.

Além das técnicas apresentadas, outras abordagens também podem ser utilizadas para auxiliar na predição de eventos, tais como algoritmos genéticos (Jacob *et al.*, 2005; Ooi & Tan, 2003; Lin *et al.*, 2002; Thomas & Sycara, 1999) e *Simulated Annealing* (Kirkpatrick *et al.*, 1983; Lee *et al.*, 2008). Essas abordagens, em conjunto com a avaliação detalhada das técnicas apresentadas neste capítulo, foram descritas no relatório técnico desenvolvido no decorrer desta tese (Dodonov & Mello, 2008).

3.4 Considerações finais

Os estudos apresentados no capítulo 2 averiguaram que o conhecimento sobre o comportamento de processos tem influência significativa na tomada de decisões em sistemas distribuídos. Isso motivou essa tese a realizar estudos mais detalhados sobre diversas abordagens que permitem estimar e prever eventos futuros. No contexto desta tese, as técnicas estudadas podem ser empregadas para reduzir a dimensionalidade de dados, obtidos na etapa de extração do comportamento, determinar padrões mais representativas, e realizar estimativas sobre eventos futuros.

Entretanto, uma vez que todas as técnicas apresentam vantagens e desvantagens (Dodonov & Mello, 2008), a escolha da abordagem mais adequada depende do problema considerado. Isso motivou estudos adicionais, com o objetivo de compreender as tendências comportamentais de processos por meio de conceitos de sistemas dinâmicos. Esses estudos são apresentados no capítulo a seguir.

Avaliação da dinâmica comportamental de processos

4.1 Considerações iniciais

O paradigma de computação autônoma tem potencial para solucionar diversos problemas de otimização e gerenciamento em sistemas distribuídos, determinando ações a serem realizadas em função do conhecimento sobre o ambiente e processos nele executados. As tendências comportamentais de processos podem ser estimadas por meio de diversas técnicas de predição, apresentadas no capítulo anterior, as quais são aplicadas sobre séries temporais que representam o comportamento de processos. Contudo, a eficiência da predição pode ser prejudicada por variações e mudanças inesperadas no comportamento de processos (Dodonov & Mello, 2006; Dodonov *et al.*, 2006). Ao conhecer essas variações, pode-se melhorar a eficácia das técnicas preditoras.

O comportamento de processos, representado por séries temporais, pode ser estudado por técnicas e ferramentas oriundas da área de sistemas dinâmicos, o que permite compreender, avaliar e estimar variações comportamentais. Dessa forma, este capítulo têm como objetivo apresentar noções de sistemas dinâmicos junto com as principais abordagens utilizadas para estudar variações em séries, com foco na sua aplicação no comportamento de processos.

4.2 Conceitos de sistemas dinâmicos

As primeiras pesquisas sobre sistemas dinâmicos foram concebidas na década de 1960, ao realizar estudos sobre comportamento de ar na atmosfera (Lorenz, 1963; Scheinerman, 1996). Ao modelar o movimento do ar, concluiu-se que mesmo pequenas variações nos valores iniciais

de variáveis podem levar a resultados divergentes, o que motivou estudos em diversas áreas da ciência (Lorenz, 1963; Alligood *et al.*, 1997; Clark, 1990), e deu origem a conceitos relativos a sistemas dinâmicos.

Um sistema dinâmico é composto por um conjunto de estados possíveis e uma regra que determina seu estado atual em função do passado (Alligood *et al.*, 1997). Para exemplificar, considere a equação $x_{n+1} = 2 \cdot x_n$ como representação do tamanho de uma população de indivíduos no tempo, onde n é um instante do tempo e x_n define o tamanho da população no momento n . Nesse caso, o sistema é denominado determinístico, pois essa equação define o próximo estado em função do passado. Existem, ainda, sistemas dinâmicos sujeitos a demais efeitos além do passado. Considerando um modelo de previsão de preços de ações do mercado em função do tempo, o valor atual é composto por uma equação que considera preços anteriores. Contudo, essa equação conta, também, com uma variável aleatória ν , que modifica seus resultados. Esse tipo de sistema é denominado dinâmico estocástico ou aleatório (Alligood *et al.*, 1997).

Para ilustrar, considere uma seqüência de observações da uma série temporal $X = \{x_0, x_1, \dots, x_{n-1}\}$. Os eventos podem ser monitorados de forma discreta ou contínua, oriundos de regras determinísticas ou de processos estocásticos. A fim de compreender os próximos estados desse sistema, deve-se aplicar modelos matemáticos capazes de representá-lo. Os comportamentos são simples de serem estudados caso se conheça sua regra (conjunto de equações que define o próximo estado em função do passado) e ele seja determinístico. Conhecendo a regra e sendo ele definido por processo estocástico, pode-se, também, compreender suas tendências comportamentais. Desconhecendo a regra, deve-se encontrar um conjunto de equações capaz de representá-la, o que se torna mais complexo para sistemas estocásticos, devido à existência de termos aleatórios.

Ao modelar o conhecimento embutido em um sistema dinâmico compreende-se a repetição de seus padrões, o que permite conhecer suas tendências, realizar estimativas futuras e classificar suas operações (Alligood *et al.*, 1997). Para exemplificar, considere um sistema dinâmico que representa a evolução de populações, onde o tamanho da população dobra a cada hora de observação. Esse sistema é representado pela equação $f(x) = 2 \cdot x$, e a evolução da população no tempo é definida, nesse caso, por $f^k(x)$, onde k representa o número de instantes de tempo futuros em que será feita a próxima observação. Dessa forma, uma saída produzida pelo sistema no instante t torna-se entrada em $t + 1$, e a população tem fator de crescimento exponencial igual a 2.

Para estudar a variação, ou a órbita, de valores produzidos por um sistema dinâmico pode-se utilizar um modelo denominado *Cobweb plot* (Alligood *et al.*, 1997). Segundo esse modelo, traça-se os valores das saídas produzidas em função das entradas, junto com uma reta diagonal $g(x) = x$. Para ilustrar o funcionamento do modelo, considere o sistema que representa evolução de uma população, definido por $f(x) = 2 \cdot x$, conforme apresentado anteriormente.

Figura 4.1: Ilustração de órbitas em sistemas dinâmicos

Tendo entrada inicial $x = 0,01$, têm-se a saída $f(0,01) = 0,02$, que, por sua vez, é considerada como próxima entrada, calculando $f(0,02) = 0,04$, e assim sucessivamente. O *Cobweb plot* representa esses cálculos por meio de segmentos de retas que tocam $f(x)$ e $g(x)$. Por exemplo, inicialmente traça-se um segmento horizontal do padrão entrada-saída $(0,01; 0,02)$ até tocar na reta diagonal $g(x)$. Em seguida traça-se outro segmento de reta vertical com origem no ponto que toca em $g(x)$ até chegar em $f(x)$, conforme apresentado na figura 4.1a. Os pares de valores entrada-saída obtidos, ou seja, os pares de pontos da curva $f(x)$, representam as saídas produzidas pelo sistema, ou a órbita.

O exemplo anterior apresenta a órbita de um sistema com características lineares e que tende a gerar saídas infinitas. Contudo, a fim de melhor caracterizar o tamanho de populações, indica-se a adoção de uma função que imponha limites de crescimento, e seja capaz de simular os recursos escassos do ambiente. Para exemplificar, considere a função $h(x) = 2 \cdot x \cdot (1 - x)$ onde x representa a população de entrada em milhões. Nesse caso, a população é o produto de x pelo termo $(1 - x)$, o que resulta em um efeito não linear no crescimento. Para estudar a órbita desse sistema aplica-se o mesmo método do *Cobweb plot*, conforme apresentado na figura 4.1b. Pode-se observar que, em contraste com o mapeamento linear anteriormente considerado, para quaisquer valores entre $[0,0, 0,5]$, a função $h(x)$ resulta em valores próximos de 0,5.

Considere agora outro exemplo que permite melhor exemplificar a órbita de uma função em relação a condições iniciais. Seja um sistema definido pela função $q(x) = \frac{3x-x^3}{2}$ com valores iniciais $x = 1,6$ e $x = 1,8$. Monta-se o *Cobweb plot* para as duas condições iniciais, conforme a figura 4.1c. Cabe ressaltar que, quando a curva estiver abaixo de $g(x)$, deve-se traçar linhas horizontais da órbita à esquerda e quando estiver acima, à direita, conforme realizado nos exemplos anteriores. Observa-se, nesse caso, que para a primeira condição inicial, tende-se ao ponto $x = 1$, enquanto para a segunda, a $x = -1$. Valores próximos ao ponto $x = 0$, embora diferentes, fazem a órbita mover-se em direção à -1 ou 1 . Nesse contexto, pontos $x = 1$ e $x = -1$ são definidos como pontos de atração, e os próximos de $x = 0$, como de repulsão, e são utilizados para estudos de estabilidade de sistemas dinâmicos.

Os exemplos anteriores apresentaram três tipos de sistemas dinâmicos, sendo que o primeiro é caracterizado por comportamento linear, os outros dois, não-lineares. A reta $g(x)$ foi utilizada para auxiliar na observação gráfica das órbitas, simplificando a geração de próximas entradas e cálculo de saídas. Além de auxiliar na visualização de órbitas, $g(x)$ permite definir alguns pontos relevantes para os sistemas em estudo. Percebe-se que todos os sistemas exemplificados, $f(x)$, $h(x)$ e $q(x)$ cruzam a função diagonal $g(x)$, ou seja, há pares de pontos, denominados fixos, onde a abscissa e a ordenada são iguais. No caso de $f(x)$ o único ponto fixo é $x = 0$, para $h(x)$ os dois pontos fixos são $x = 0$ e $x = 0,5$, e em $q(x)$ observa-se três pontos fixos $x = -1$, $x = 0$ e $x = 1$. Em $f(x)$, para qualquer população de entrada positiva diferente de zero, diverge-se do ponto fixo $x = 0$ e tende-se ao infinito. Para $h(x)$ a tendência é divergir de 0 e convergir para uma população máxima de $0,5$ milhão. Para $q(x)$ o sistema diverge de 0 e tende a -1 ou 1 , dependendo das condições iniciais (tais como $x = 1,6$ e $x = 1,8$).

Os pontos fixos de convergência são denominados estáveis, enquanto os de divergência, instáveis. Por exemplo, uma esfera no topo de uma montanha está sobre um ponto instável, qualquer pequena perturbação faz com que ela se mova até encontrar um vale, ou ponto estável, onde o movimento cessa. Assim, pontos estáveis atraem a órbita de um sistema, enquanto instáveis a repelem.

Conhecer os pontos fixos de um sistema permite concluir sobre suas tendências. Por exemplo, ao definir temperaturas globais da terra na forma de uma série temporal, os seus valores definem a órbita do sistema dinâmico em questão, que pode ser utilizada para encontrar sua regra de origem. Ao conhecer essa regra, é possível definir os pontos fixos e estudar, por exemplo, regiões estáveis de temperatura, que definiriam máximos ou mínimos de temperatura para o planeta. Além disso, pode-se encontrar regiões instáveis, onde temperaturas tendem a divergir. Ao compreender a regra geradora dessa órbita e seus pontos fixos, é possível, por exemplo, avaliar a influência do aquecimento global em outros sistemas, tais como fauna e flora (Clark, 1990).

Os pontos fixos estáveis e instáveis podem, ainda, ser utilizados para estudar o comportamento de processos distribuídos. Considerando os elementos de uma série temporal

que representa o comportamento de um processo, pode-se determinar a órbita do sistema e, após isso, encontrar a regra origem dessa órbita e seus pontos fixos. Esses pontos permitem avaliar, por exemplo, quando as tendências comportamentais de processos sofrem alterações, o que permite estimar pontos críticos de execução. Percebe-se, portanto, que, ao compreender a órbita e os pontos fixos de um sistema, pode-se tirar conclusões sobre o seu comportamento.

4.3 Avaliação de variações comportamentais

Diversos trabalhos visam compreender o comportamento de sistemas dinâmicos (Rosenstein *et al.*, 1993; Jones *et al.*, 1996; Kaplan, 2003), estudando mudanças e variações em relação às condições iniciais. Entre as abordagens mais utilizadas estão as baseadas no expoente de Lyapunov (Rosenstein *et al.*, 1993), que permite determinar a estabilidade de uma série, e as baseadas no expoente de Hurst (Jones *et al.*, 1996; Kaplan, 2003; Mielniczuk & Wojdylo, 2007), que avalia o quão aleatório é um sistema.

O expoente de Lyapunov (Edmonds, 1996) mede a taxa de variação de trajetórias vizinhas em relação a condições iniciais, considerando uma separação δZ_0 . Essa divergência é dada por $|\delta Z(t)| \approx e^{\lambda t} |\delta Z_0|$, onde λ é o expoente de Lyapunov. O expoente é definido na equação 4.1, e o seu valor determina a órbita do sistema, sendo que para $\lambda < 0$ a série é atraída para um ponto fixo ou uma órbita periódica, para $\lambda > 0$ o sistema apresenta uma órbita caótica e instável, e para $\lambda = 0$ a órbita do sistema tende para um ponto fixo neutro.

$$\lambda = \lim_{t \rightarrow \infty} \frac{1}{t} \ln \frac{|\delta Z(t)|}{|\delta Z_0|} \quad (4.1)$$

Além de permitir o estudo de tendências, esse expoente pode ainda ser utilizado para compreender o horizonte máximo de predição de sistemas dinâmicos, definido por $-\frac{\ln \eta}{\lambda}$, onde η é o erro medido no estado inicial. Para exemplificar, considere que qualquer técnica foi utilizada para prever uma série temporal, cuja divergência é dada pelo expoente de Lyapunov. Caso não haja nenhuma divergência, pode-se prever inúmeros eventos futuros com base no modelo inicial. Quanto maior for a divergência, menor é o número de elementos que podem ser previstos sem alterar o modelo. Dessa forma, até mesmo estimativas qualitativas são impossíveis para intervalos além desse horizonte (Elmer, 1998). Por exemplo, considerando que o erro de predição de um próximo valor é de 0,001, e o expoente de Lyapunov dessa série é de 0,692. Nesse caso, o horizonte de predição dessa série é igual a $-\frac{\ln 0,001}{0,692} = 9,98$. Isso significa que se pode prever, no máximo, o comportamento dos próximos 9,98 valores futuros.

O expoente de Hurst, por sua vez, permite avaliar o quanto o comportamento de um sistema dinâmico depende de valores passados, e pode ser utilizado para indicar como dados históricos podem ser empregados para estimar acontecimentos futuros. Esse expoente pode ser estimado por diversas técnicas, sendo a mais utilizada a de cálculo de *Rescaled Range* (R/S)

Figura 4.2: Cálculo de *Rescaled Range* do expoente de Hurst

(Kaplan, 2003). Para exemplificar, considere a série temporal $X(n) = \{x_0, x_1, \dots, x_{n-1}\}$. O primeiro passo consiste em computar a média Avg_{X_k} para todos os valores de $X(k)$, onde $k = n$. Em seguida, encontra-se o valor mínimo (min) e máximo (max) em relação à média, e o desvio padrão $STDEV_{X(k)}$ para todos os elementos de $X(k)$. Após isso, é calculado o primeiro valor de R/S , definido por $RS_0 = \frac{|max-min|}{STDEV_{X(k)}}$, sendo que o índice 0 de RS_0 significa a primeira iteração do cálculo. Após isso, a série temporal é dividida em dois subconjuntos: o primeiro onde $k \in [0, \frac{n}{2} - 1]$ e segundo com $k \in [\frac{n}{2}, n - 1]$. Para cada subconjunto, o mesmo procedimento é realizado, obtendo-se $RS_{1,[0, \frac{n}{2}-1]}$ e $RS_{1,[\frac{n}{2}, n-1]}$, e considerando RS_1 como sendo a média desses valores. O mesmo procedimento é realizado, recorrentemente, determinando os valores $RS_2, RS_3, \dots, RS_w$, onde w é a última iteração, conforme pode ser visto na figura 4.2.

Após calcular todos os valores de $RS = \{RS_0, RS_1, \dots, RS_w\}$, obtém-se o expoente de Hurst por meio de uma regressão linear dos pontos $(\log_2(|X(\frac{n}{2^y})|), \log_2(RS_y))$, onde $y = 0, 1, \dots, w$ representa a iteração do algoritmo, e $|X(\frac{n}{2^y})|$ é o número de elementos por partição da série na iteração y . Fazendo a regressão linear, obtém-se a equação $y = Hx + \alpha$, onde o ângulo da curva, representado por H , é o expoente de Hurst. O valor do expoente varia entre 0 e 1, sendo que valores próximos de 1 indicam comportamento persistente (ou seja, há correlação entre um evento e ocorrência de um outro no futuro), os próximos de 0 indicam comportamento anti-persistente (ou seja, um acréscimo no valor passado gera decréscimo em um valor futuro), e os próximos de 0,5 indicam que a série apresenta comportamento aleatório (*random walk*).

Dessa forma, os expoentes de Lyapunov e Hurst podem ser utilizados para compreender o comportamento de um sistema dinâmico, e ajustar os mecanismos de predição de forma apropriada.

4.4 Desdobramento de séries

Além dos expoentes de Lyapunov e Hurst, utilizados para avaliar as órbitas de sistemas dinâmicos, outras abordagens podem ser empregadas para estimar as regras, ou funções, que

Figura 4.3: Exemplo de variedade

definem o seu comportamento. Por exemplo, considerando uma função qualquer que gera uma série temporal caracterizada por expoentes de Hurst ou Lyapunov próximos de 0,5, o comportamento dessa função apresenta tendência caótica e instável, sendo que a aplicação direta de uma técnica de predição resultaria em resultados ruins. Nesse caso, são necessários estudos alternativos para compreender as regularidades internas e reconstruir a órbita.

Whitney (1936) aplicou variedades diferenciáveis como forma de reconstruir funções utilizando transformações para espaços Euclidianos multidimensionais. Variedades são espaços matemáticos onde cada ponto tem uma vizinhança a qual aparenta um espaço Euclidiano, contudo a estrutura global pode ser mais complexa. Para exemplificar, considere um triângulo pequeno na superfície de uma esfera. A soma dos ângulos internos desse triângulo totaliza um valor muito próximo de 180° , no entanto, para um triângulo maior, o somatório dos ângulos tenderá a valores maiores, dada a curvatura da esfera. Nessa situação, embora a esfera não esteja em espaço Euclidiano, leis da geometria Euclidiana apresentam uma boa aproximação local para avaliar distâncias entre pontos na superfície. Entretanto, essa aproximação é tendenciosa, uma vez que esconde as irregularidades observáveis em espaço com dimensão superior. Nesse caso, a esfera é definida como uma variedade, e essa pode ser expressa em termos de propriedades conhecidas em diferentes espaços multidimensionais (Whitney, 1936).

Para ilustrar, considere a figura 4.3 que apresenta um exemplo de parametrização de um plano $\mathbb{R}^{m-1} \times \mathbb{R}_+$ para outro $\mathbb{R}^k$. Dado um ponto q' pode-se, por meio de $\phi_q : H_0 \rightarrow H \cap M$, encontrar um ponto q em M correspondente. O mesmo ocorre com p' , contudo, nesse caso, obtém-se uma região na borda de M . Esse exemplo ilustra o mapeamento de um ponto e sua vizinhança em um plano com maior número de dimensões.

Observando essa possibilidade de mapear objetos mais complexos em outros espaços multidimensionais, Whitney (1936) propôs o teorema de imersão, segundo o qual qualquer variedade em n dimensões pode ser mapeada em espaço Euclidiano de $2n + 1$ dimensões, onde quaisquer dois pontos de uma variedade n dimensional não são mapeados para o mesmo ponto no espaço Euclidiano de $2n + 1$ dimensões. Esse mapeamento pode, ainda, ser estendido para sistemas dinâmicos, representando cada estado do sistema por um vetor de $2n + 1$ dimensões.

Estudos realizados por Takens (1980) provaram que, ao invés de mapear os estados de um sistema dinâmico em espaço Euclidiano de $2n+1$ dimensões, pode-se reconstruí-lo considerando deslocamentos no tempo. Dessa forma, uma série temporal $x_0, x_1, \dots, x_{n-1}$ pode ser reconstruída em espaço multidimensional $x_n(m, \tau) = (x_n, x_{n+\tau}, \dots, x_{n+(m-1)\tau})$, ou de coordenadas de atraso, onde m é a dimensão embutida e τ representa um *time delay* ou dimensão de separação. Essa técnica de mapeamento ou reconstrução pode ser utilizada para transformar saídas produzidas por sistemas dinâmicos, representadas por séries temporais unidimensionais, em um conjunto de pontos em espaço Euclidiano de m dimensões. Essa reconstrução auxilia na obtenção de regras de sistemas dinâmicos, simplificando o estudo de comportamentos, órbitas, e tendências do sistema (Alligood *et al.*, 1997).

Para exemplificar, considere a equação logística $x_{t+1} = b \cdot x_t \cdot (1 - x_t)$, com condições iniciais $t \in [0, 4000]$, $b = 3,8$ e $x_0 = 0,5$. A figura 4.4a apresenta as saídas, ou órbita, para essa regra, a qual tem expoente de Hurst $H = 0,40535$, que evidencia comportamento aleatório com pequeno grau de anti-correlação, e Lyapunov $\lambda = 0,4348$, que indica comportamento caótico e instável, onde a distância entre pontos da trajetória tende a divergir. Contudo, ao reconstruir a série em um espaço multidimensional com $m = 2$ e $\tau = 1$, obtém-se relação entre pares de pontos (x_t, x_{t+1}) (figura 4.4b). Dessa forma, o comportamento da função logística, que era aparentemente uma caminhada aleatória, pode ser estudado, compreendido e modelado de forma mais simples. Da mesma forma, ao avaliar o atrator de Lorenz (Eckmann & Ruelle, 1985), cujas saídas são apresentadas na figura 4.4c, pode-se reconstruir sua série utilizando dimensão embutida 2 (figura 4.4d) e, ao adicionar uma nova dimensão ($m = 3$), desdobra-se todo o comportamento da série, simplificando sua compreensão (figura 4.4e). Pára-se de adicionar dimensões caso não haja desdobramentos de novos comportamentos. Nesse caso, a dimensão $m = 3$ representa a melhor dimensão para o atrator de Lorenz. Ao realizar uma regressão dos pontos resultantes, pode-se obter a regra do sistema dinâmico e determinar seus comportamentos futuros. Tendo essa regra e um x_t , pode-se, por exemplo, definir o próximo valor da série, x_{t+1} , o qual pode ser retro-alimentado para gerar x_{t+2} , e assim sucessivamente.

A dimensão embutida define, basicamente, o número de eixos, do espaço de coordenadas de atraso, necessários para plotar o comportamento reconstruído da série. A dimensão de separação, por sua vez, auxilia na extração de comportamentos periódicos, ou a sazonalidade, de séries. Essa dimensão informa o deslocamento de valores históricos que devem ser avaliados a fim de estimar comportamento futuro. Por exemplo, ao estudar temperaturas de uma região do mundo na data de 12 de dezembro de 2007, pode-se observar medidas próximas das do ano anterior, o que sugere que o próximo ano também tenha temperaturas similares. Nesse exemplo, a dimensão de separação seria igual a 365 dias. Por outro lado, caso seja considerado o período de um mês, o comportamento das temperaturas não se repetiria da maneira esperada, sendo que as temperaturas de 12 de setembro, outubro e novembro de 2007, muito provavelmente, não auxiliariam na predição das temperaturas de 12 de dezembro de 2007.

Figura 4.4: Estudo de dimensões de separação e embutida: função logística e atrator de Lorenz

Para avaliar as dimensões embutida e de separação, pode-se aplicar diversas técnicas, tais como uma função de auto-correlação (Abarbanel *et al.*, 1993), que mensura a repetição de comportamento de um trecho da série em relação ao seu histórico, e a técnica de auto-informação mútua (*AMI*, *Auto Mutual Information*) (Fraser & Swinney, 1986), que considera diferentes deslocamentos no tempo. A técnica de auto-correlação é definida na equação 4.2, onde $E[.]$ é o valor esperado, μ é a média, k é o deslocamento no tempo e σ^2 , a variância, que auxilia na determinação da dimensão de separação de séries. Contudo, essa técnica é formulada para séries lineares, e, conseqüentemente, não é adequada para outros tipos de séries.

Figura 4.5: Ilustração do cálculo da dimensão de separação

$$ACF(k) = \frac{E[(X_i - \mu)(X_{i+k} - \mu)]}{\sigma^2} \quad (4.2)$$

A informação mútua média é definida pela equação 4.3, onde X e Y seguem, respectivamente, as funções de distribuição de probabilidades P_X e P_Y , e X e Y ocorrem em pares com distribuição conjunta P_{XY} (Kennel, 2002). A equação é aplicada iterativamente, considerando diferentes deslocamentos no tempo, e, em seguida, traça-se uma curva em função dos deslocamentos, adotando seu primeiro mínimo como dimensão de separação. Aplicando essa técnica sobre o conjunto de dados de Lorenz (Fraser & Swinney, 1986), obtém-se a figura 4.5, a partir da qual se encontra o primeiro mínimo, igual a 5 (Lorenz, 1963; Kennel *et al.*, 1992).

$$I(X; Y) = \int P_{XY}(x, y) \log_2 \frac{P_{XY}(x, y)}{P_X(x)P_Y(y)} dx dy \quad (4.3)$$

Após definir a dimensão de separação de uma série, deve-se encontrar a embutida. Takens (1980) e Mañé (1980) estudaram e confirmaram que o limite superior da dimensão embutida D_e pode ser estimado utilizando a dimensão fractal D_f , sendo $D_e > 2.0 \cdot D_f$. Contudo, a dimensão resultante é, em geral, maior que o necessário, o que adiciona complexidade à modelagem e análise do sistema. Uma forma alternativa para obter a dimensão embutida mínima é por meio do cálculo de invariantes do sistema, tais como o expoente de Lyapunov (Alligood *et al.*, 1997), para diferentes valores de dimensão, observando a saturação dos resultados. A complexidade dessa abordagem motivou Kennel *et al.* (1992) a propor o método de falsos vizinhos (FNN, *False Nearest Neighbors*), que calcula os vizinhos mais próximos de cada ponto, no espaço de coordenadas de atraso. Para isso, novas dimensões são adicionadas, iterativamente, ao sistema, calculando a distância entre vizinhos mais próximos. Caso haja acréscimo nessa distância, os pontos são considerados falsos vizinhos, o que evidencia a necessidade de mais dimensões para reconstruir o comportamento da série.

Kennel *et al.* (1992) consideram uma dimensão embutida d onde o r -ésimo vizinho mais

Figura 4.6: Ilustração do cálculo da dimensão embutida

próximo de $y(n)$ é dado por $y^{(r)}(n)$. A distância Euclidiana entre o ponto $y(n)$ e seu r -ésimo vizinho mais próximo é dada pela equação 4.4. Ao adicionar uma nova dimensão, reconstrói-se a série em $d + 1$ e adiciona-se a coordenada $(d + 1)$ em cada vetor $y(n)$, a qual é incluída na equação de distância Euclidiana (termo $x(n + Td)$ da equação 4.5). Dessa forma, o critério mede a variação de distância ao adicionar uma nova dimensão, conforme descrito pela equação 4.6. Segundo os autores, se $V_{n,r} > R_{tol}$ os pontos são considerados falsos vizinhos, onde R_{tol} é um limiar. Eles ainda concluem, empiricamente, que $R_{tol} \geq 10.0$ é um bom limite para a geração de resultados.

$$R_d^2(n, r) = \sum_{k=0}^{d-1} (x(n + kT) - x^{(r)}(n + kT))^2 \quad (4.4)$$

$$R_{d+1}^2(n, r) = R_d^2(n, r) + (x(n + dT) - x^{(r)}(n + dT))^2 \quad (4.5)$$

$$V_{n,r} = \sqrt{\frac{R_{d+1}^2(n, r) - R_d^2(n, r)}{R_d^2(n, r)}} = \frac{|x(n + Td) - x^{(r)}(n + Td)|}{R_d^2(n, r)} \quad (4.6)$$

Aplicando o método *FNN* sobre o conjunto de dados do atrator de Lorenz, utilizando a dimensão de separação 5 previamente obtida, obtém-se os resultados apresentados na figura 4.6. Essa figura traça a fração dos falsos vizinhos *versus* a dimensão embutida considerada. Quando a fração é igual a zero, encontra-se a melhor dimensão embutida. Nesse caso, a dimensão embutida encontrada é 3, o que confirma os resultados obtidos por Kennel *et al.* (1992).

Após definir as duas dimensões, aplica-se a teoria de imersão de Takens (1980), conforme apresentado anteriormente, onde a série temporal $x_0, x_1, \dots, x_{n-1}$ é reconstruída no espaço multidimensional, ou de coordenadas de atraso, $x_n(m, \tau) = (x_n, x_{n+\tau}, \dots, x_{n+(m-1)\tau})$, sendo que m representa a dimensão embutida e τ a de separação. A reconstrução desdobra, completamente, o comportamento da regra desse sistema dinâmico. Para exemplificar esse desdobramento,

considere o conjunto de dados de Lorenz apresentado na tabela 4.1. Depois de reconstruir esse conjunto com dimensão embutida 3 e separação 5, obtém-se a tabela 4.2. Essa reconstrução permite desdobrar o comportamento da série e obter sua regra, ou seja, a função que define sua órbita no tempo. Obtendo tal função, pode-se estudar seus pontos fixos estáveis e instáveis, compreender suas tendências e, também, predizer seu comportamento futuro.

Tabela 4.1: Série temporal do atrator de Lorenz

Tempo (t)	Valor ($f(t)$)	Tempo (t)	Valor ($f(t)$)
1	-9.6559617	8	-1.7784935
2	-6.9902085	9	-1.8834828
3	-4.9834927	10	-2.1397586
4	-3.5773619	11	-2.5521791
5	-2.6589215	12	-3.1453527
6	-2.1120568	13	-3.9638112
7	-1.8411753	14	-5.0733551

Tabela 4.2: Conjunto de dados do atrator de Lorenz reconstruído segundo a dimensão embutida e de separação encontradas ($m = 3$ e $\tau = 5$)

Dimensão 1	Dimensão 2	Dimensão 3
-9.655962	-2.112057	-2.552179
-6.990209	-1.841175	-3.145353
-4.983493	-1.778494	-3.963811
-3.577362	-1.883483	-5.073355

As técnicas apresentadas acima fazem parte da área de teoria do caos, e os seus resultados permitem compreender comportamentos internos de sistemas dinâmicos, mesmo quando esses não são observados por técnicas convencionais.

4.5 Considerações finais

Os estudos sobre o comportamento de processos, realizados no decorrer deste trabalho de doutorado, demonstraram que o conhecimento sobre processos pode influenciar de forma significativa no desempenho de um sistema distribuído. Diversas técnicas de predição podem ser empregadas para estimar as operações futuras, contudo, a sua eficiência depende de conhecimentos sobre características comportamentais de processos. Isso motivou estudos detalhados sobre a natureza das variações de comportamento.

Ao representar o comportamento de processos por meio de uma série temporal, esse pode ser visto como um sistema dinâmico, o que permite a aplicação de técnicas abordadas neste capítulo para compreender suas tendências. Dessa forma, pode-se entender a dinâmica comportamental de processos, e utilizá-la para guiar mecanismos de classificação e predição abordados nesta tese.

Emprego da dinâmica comportamental de processos a fim de prover autonomia a ambientes distribuídos

5.1 Considerações iniciais

Com o aumento da complexidade envolvida no gerenciamento e manutenção de ambientes distribuídos de grande porte, técnicas de gerenciamento convencionais, baseadas em intervenções manuais e interações com usuários, mostraram-se pouco eficientes (Horn, 2001; Arnautovic *et al.*, 2007). Essa complexidade motivou pesquisas com o objetivo de tornar ambientes auto-gerenciáveis. Isso levou à concepção da área de computação autônoma, que visa oferecer capacidade de auto-gerenciamento para aplicações e ambientes por meio de quatro aspectos básicos: auto-configuração, auto-recuperação, auto-otimização e auto-proteção (Horn, 2001; Parashar, 2007). Esses aspectos permitem ao sistema adaptar-se, de acordo com as condições do ambiente, e oferecem maior desempenho e funcionamento contínuo, sem a necessidade de intervenções manuais.

Motivados pelo objetivo de prover autonomia a ambientes distribuídos, diversos estudos foram realizados no desenvolvimento desta tese de doutorado, avaliando abordagens de extração, compreensão e predição de dinâmicas comportamentais de processos. O objetivo, metodologia e a evolução desses estudos são apresentados a seguir.

5.2 Objetivo

Esta tese de doutorado teve como o objetivo a realização de estudos sobre a dinâmica comportamental de processos, os quais resultaram na proposta de uma abordagem de predição *online* de eventos futuros. Essa abordagem é aplicada a fim de oferecer autonomia a ambientes distribuídos, em aspectos tais como otimização de escalonamento, distribuição, replicação e antecipação de acesso a dados, detecção de falhas e intrusões, e configuração automática de serviços.

A abordagem proposta também contempla as etapas de extração e classificação de conhecimento como fases de pré-processamento para predição. A primeira emprega técnicas para obter, transparentemente, informações sobre o comportamento de processos, e a segunda, métodos para reduzir a dimensionalidade e caracterizar as informações mais relevantes a serem previstas. A realização dessas etapas provê capacidade de auto-identificação ao sistema, e os conhecimentos obtidos podem ser empregados como base para outros aspectos de autonomia. Para validar a abordagem, foi desenvolvida uma política de escalonamento de processos, que emprega o conhecimento sobre as dinâmicas comportamentais de processos para prover capacidade de auto-otimização ao ambiente distribuído.

5.3 Metodologia

No decorrer desta tese de Doutorado, as seguintes etapas foram contempladas.

Inicialmente, buscou-se estudar e avaliar técnicas de predição para modelar e estimar tendências comportamentais de aplicações (capítulo 3). No decorrer desses estudos, observou-se que, embora diversas técnicas apresentam o potencial para realizar predições eficientes, cada uma tem suas particularidades (Dodonov & Mello, 2008). Dessa forma, decidiu-se adotar duas técnicas de predição para avaliação inicial: uma baseada em modelo estatístico de Markov, caracterizada por estimar tendências momentâneas de execução de processos, e outra que emprega a rede neural artificial *TDNN*, para realizar estimativas de acordo com o histórico. Para validar essas técnicas em ambiente real, foi adotado o conjunto de ferramentas *NAS Parallel Benchmark* (Bailey *et al.*, 1991), amplamente utilizado para experimentos em ambientes distribuídos (Dodonov *et al.*, 2006). Visando realizar uma avaliação mais abrangente, o comportamento de processos foi extraído por meio de duas técnicas: uma baseada em monitoração, e outra em interceptação de eventos.

Esses estudos, descritos na seção 5.4, apresentaram resultados promissores e demonstraram que a técnica de extração baseada em interceptação de eventos resulta em maior precisão, o que motivou sua adoção em trabalhos subseqüentes. Além disso, notou-se que técnicas de predição baseadas em histórico têm potencial de realizar predições mais adequadas, considerando

o contexto e tendências atuais de execução de processos (Dodonov *et al.*, 2006). Contudo, experimentos realizados demonstraram que as técnicas utilizadas na avaliação do *NAS Parallel Benchmark* não oferecem eficiência adequada para aplicações mais complexas, levando a estudos adicionais para escolher abordagens mais apropriadas para o objetivo do trabalho. Além disso, observou-se que o conhecimento sobre a dinâmica de processos tem papel fundamental na modelagem de seu comportamento, o que motivou estudos para sua compreensão e utilização em previsões mais precisas (capítulo 4).

Após isso, os conceitos abordados foram avaliados, visando prover capacidades de auto-configuração e auto-otimização por meio de escalonamento dinâmico de aplicações, considerando e estimando tendências comportamentais (seção 5.6). Nessa avaliação, foram realizados estudos a fim de escolher abordagens mais adequadas de compreensão e predição do comportamento, e foi desenvolvida uma política de escalonamento, que considera a dinâmica comportamental de processos e toma decisões em função de eventos previstos. Essa política foi avaliada com conjunto de aplicações do *NERSC* (Oliker *et al.*, 2008), caracterizadas por comportamento difícil de ser previsto. Após isso, a complexidade da abordagem proposta foi avaliada, estudando seus custos computacionais e a viabilidade de sua utilização de forma *online*.

Os resultados das etapas apresentadas anteriormente foram incorporados na proposta de uma abordagem dinâmica para prover autonomia a ambientes distribuídos, em função da avaliação e predição da dinâmica comportamental de processos.

5.4 Estudos iniciais

Para validar os conceitos de predição de comportamento de processos estudados neste trabalho, escolheu-se, inicialmente, o conjunto de ferramentas *NAS Parallel Benchmark* (NPB), frequentemente utilizado para avaliar o desempenho de sistemas distribuídos (Bailey *et al.*, 1991). O *NAS* é composto por um conjunto de aplicações (*EP*, *MG*, *CG*, *FT*, *IS*, *LU*, *SP* e *BT*), derivadas de sistemas de avaliação computacional de dinâmica de fluidos (*CFD*, *Computational Fluid Dynamics*) (Bailey *et al.*, 1991), e é disponível em diversas versões¹. Neste estudo de caso, adotou-se a versão *NAS-MPI* (Cappello & Etiemble, 2000), que utiliza o padrão de troca de mensagens *MPI* (Burns *et al.*, 1994; Aumage & Mercier, 2003) para comunicação entre processos independentes.

As aplicações do *NAS* podem ser compiladas em diferentes classes (A, B, C, W e S), caracterizadas por requisitos computacionais distintos, tais como o uso do processador, memória, tempo de execução, tamanho do problema e quantidade de comunicações. O comportamento de aplicações do *NAS* é bem conhecido (Bailey *et al.*, 1993), o que motivou sua adoção para validar as técnicas de predição estudadas no decorrer deste estudo.

¹Diferentes versões do *NAS Parallel Benchmark* são disponíveis em <http://www.nas.nasa.gov/Software/NPB/>

Figura 5.1: Arquitetura da rede ART-2A

Foram concebidas três etapas para a realização de previsões comportamentais: extração de comportamento, pré-processamento e previsão. A primeira etapa foi responsável por extrair, de forma automática e transparente, eventos de comunicação ocorridos entre aplicações. Foram avaliadas duas abordagens de extração (apresentadas na seção 2.4), sendo uma baseada em monitoração periódica, e outra em interceptação de eventos. A primeira abordagem utilizou um *kernel Linux* modificado (Senger *et al.*, 2005), no qual chamadas de sistema foram introduzidas para obter, periodicamente, informações sobre o número de mensagens trafegadas por protocolo de comunicação e a quantidade de dados recebida e transmitida. A segunda adotou o projeto *GridBox* (Dodonov *et al.*, 2004) para interceptar, transparentemente, chamadas de funções *MPI* e reconstruir o fluxo de execução de aplicações. O fluxo reconstruído foi utilizado para determinar a sequência de eventos de comunicação consecutivos, tempo de processamento entre esses eventos, localização e quantidade de dados acessados a cada comunicação.

A segunda etapa visou determinar os pontos relevantes de execução. Ela foi responsável por agrupar comportamentos de processos e reduzir sua dimensionalidade, de forma a simplificar previsões. Essa etapa empregou a rede neural artificial *ART-2A* (Mello *et al.*, 2005), a qual é composta por sistemas de atenuação e orientação (figura 5.1). O sistema de atenuação é composto pelas camadas de pré-processamento F_0 , de entradas F_1 , e de agrupamento F_2 . As camadas de entrada e de agrupamento são interconectadas por dois conjuntos de pesos denominados *bottom-up* ($F_1 \rightarrow F_2$) e *top-down* ($F_2 \rightarrow F_1$), onde a conexão entre i -ésimo neurônio da camada F_1 e o j -ésimo neurônio da camada F_2 é representada por w_{ij} , e a conexão inversa é representada por w_{ji} . Esses pesos são utilizados para armazenar conhecimentos da rede, e a interação entre as camadas F_1 e F_2 é controlada pelo sistema de orientação, que considera um parâmetro de vigilância ρ para controlar o processo de aprendizado.

O nível de similaridade entre os padrões de entrada é controlado pelo parâmetro de vigilância ρ , que influencia o número de agrupamentos a serem criados pela rede. O valor de ρ define o raio para cada agrupamento, dado por $\sqrt{2 \cdot (1 - \rho)}$. No caso de $\rho = 0$ todos os padrões de entrada são representados por um único agrupamento, e para $\rho = 1$ um agrupamento distinto é criado para cada entrada da rede. A definição do valor mais adequado para ρ , portanto,

Figura 5.2: Determinação da precisão adequada para agrupamento pela rede ART-2A

tem papel fundamental na etapa de agrupamento. Para isso, Mello *et al.* (2005) propuseram duas métricas para cálculo do valor ideal de ρ , considerando a relação entre a distância entre padrões armazenados em cada grupo (denominada distância *intra-cluster*) e distâncias entre centróides de agrupamentos distintos (distância *inter-cluster*).

O cálculo da distância *intra-cluster* é definido na equação 5.1, onde m é o número de agrupamentos criados pela rede. Essa métrica permite quantificar a distância média entre os padrões de um grupo i , e o valor de seu centróide c_i , calculando a diferença média entre ângulos dos vetores x_{ij} , que representam os padrões do agrupamento c_i , e valor de seu centróide. A medida *inter-cluster*, por sua vez, é calculada por meio da equação 5.2, que define a distância entre centróides de agrupamentos distintos.

$$Intra = \frac{\sum_{i=1}^m \sum_{j=1}^n ||x_{i,j}|| \times ||c_i||}{m \cdot n} \quad (5.1)$$

$$Inter = \frac{\sum_{i=1}^m \sum_{j=1}^m ||c_i|| \times ||c_j||}{m \cdot m} \quad (5.2)$$

Ao realizar o cálculo das distâncias *intra* e *inter-clusters* para diferentes valores de ρ , deve-se encontrar um ponto de intersecção das funções resultantes, ou seja, ponto onde a distância *intra-clusters* é mínima, e a *inter-clusters*, máxima, conforme apresentado na figura 5.2. Esse ponto define o valor ideal de ρ para o conjunto de dados e, conseqüentemente, o número adequado de grupos para realizar o agrupamento preciso pela rede ART-2A (Mello *et al.*, 2005).

Após a etapa de agrupamento, duas abordagens foram adotadas para realizar a predição do comportamento: a de cadeias de Markov (Bolch *et al.*, 1998), e a rede neural artificial TDNN (Waibel *et al.*, 1989). A primeira abordagem é utilizada para estimar tendências médias de processos, calculando probabilidades de transição entre estados de execução. A segunda, por sua vez, estuda tendências históricas repetitivas no comportamento da série, com o objetivo

de estimar operações futuras. Para ambas as abordagens, foram utilizados os resultados de agrupamento realizado pela rede neural *ART-2A*.

Na primeira abordagem de predição, uma matriz de incidência é construída, a qual contém todas as transições entre estados observadas durante a execução da aplicação e, em seguida, calcula-se a probabilidade de cada transição. Esse modelo não considera o histórico de execução da aplicação, e assume que a sequência de execução varia somente em função do estado atual.

A segunda abordagem avalia a série temporal, que representa as variações de comportamento da aplicação, por meio da rede neural *TDNN*. A série é re-estruturada em uma sequência de vetores de tamanho n , onde n representa o *lag*, ou atraso temporal do vetor v . Dessa forma, o comportamento de processos é representado conforme a equação 5.3, onde v é vetor de tamanho n no instante x , e e_k é o k -ésimo estado de execução, agrupado pela rede neural *ART-2A*.

$$v_x = \{e_{n-k}, e_{n-k+1}, \dots, e_n\} \quad (5.3)$$

Essa organização adota uma abordagem similar à de janela deslizante (Qiao *et al.*, 2003), onde o vetor $v_1 = \{e_1, \dots, e_n\}$ é utilizado para prever o estado e_{n+1} , o vetor $v_2 = \{e_2, \dots, e_{n+1}\}$ para prever e_{n+2} , e assim sucessivamente. Diferentemente da abordagem baseada em cadeias de Markov, *TDNN* permite realizar predições considerando o histórico de operações. Entretanto, o histórico deve ser constantemente atualizado para que a rede seja capaz de atualizar seu modelo de representação da série de forma eficiente, uma vez que ela atribui maior peso aos últimos elementos da série, funcionando como uma memória de curto prazo. A determinação do tamanho do *lag*, portanto, é essencial para predição correta.

Dessa forma, a abordagem descrita anteriormente permite extrair e agrupar padrões de entrada de forma transparente, sem necessidade de modificar ou adaptar a aplicação. Os padrões de acesso detectados, por sua vez, são utilizados para prever a probabilidade de transições entre estados (utilizando o modelo de Markov), e para prever a sequência de eventos futuros (utilizando a rede *TDNN*). Ambas abordagens foram avaliadas, conforme demonstrado a seguir.

5.5 Predição de acessos a dados como base para autonomia em ambientes distribuídos

Para avaliar as técnicas descritas anteriormente, o *NAS Parallel Benchmark* foi executado em um ambiente de *cluster* dedicado, composto por 5 nós com processadores *AMD Athlon 2 GHz*, com 2 GB de memória *RAM*, e interconectados por um *switch Gigabit*. Um nó do *cluster* foi responsável por executar o processo mestre, e outros quatro, os processos escravo. Para simplificar a compreensão de comportamento de aplicações do *NAS*, somente os resultados dos

Figura 5.3: Modelo de Markov para *NAS Parallel Benchmark*

workloads IS.A, IS.B, CG.A e CG.B são apresentados nesta seção por serem mais representativos (Dodonov *et al.*, 2006) ².

Tabela 5.1: Valores de ρ adequados para aplicações de *NAS Parallel Benchmark*

Aplicação	ρ	Aplicação	ρ
CG.A	0,997	CG.B	0,9999
IS.A	0,99	IS.B	0,999

Inicialmente, o conhecimento sobre o comportamento dessas aplicações foi extraído por meio da técnica baseada em monitoração periódica, e foi agrupado pela rede *ART-2A*, com o valor de ρ adequado para cada *workload*, conforme apresentado na tabela 5.1. Após isso, o comportamento agrupado foi transformado em uma série temporal, e a técnica de cadeias de Markov foi aplicada para calcular a matriz de incidência e as probabilidades de transições entre estados de execução (figura 5.3).

Como pode-se observar na figura 5.3, as transições entre os estados podem ser estimadas de forma probabilística. Para exemplificar, considere a cadeia de Markov apresentada na figura 5.3.a. Quando a aplicação está no estado de execução caracterizado por tráfego de 0 *Bytes*, tem-se probabilidade de 10,28% de transição para estado caracterizado por tráfego de 1.029 *Bytes*, de 10,28% para estado com tráfego de 2.106.432 *Bytes*, de 12,15% para estado caracterizado por tráfego de 1 *Byte*, ou de permanecer no mesmo estado com probabilidade de 33,64%. Da mesma maneira, a probabilidade de observar um fluxo de execução caracterizado por tráfego de 1 *Byte*, seguido por tráfego de 2.106.432 *Bytes* pode ser calculada como sendo $10,28\% \cdot 12,15\% = 0,1028 \cdot 0,1215 = 0,0124$, ou seja, esse fluxo de execução pode ser observado com probabilidade de 1,24%.

A partir desses valores, pode-se estimar a tendência de execução do sistema, alocando recursos necessários, ou antecipando operações futuras. Entretanto, essa abordagem é limitada por considerar a probabilidade média de transições entre estados, não levando em consideração o histórico de execução. Com isso, o modelo considera que a aplicação tem a mesma probabilidade de permanecer em estado de processamento intensivo (caracterizado por tráfego de 0 *Bytes*) durante toda a execução, o que não representaria, de forma correta, o comportamento ocorrido durante períodos de comunicação intensiva. Essa desvantagem motivou este trabalho a focar em

²Resultados para outros *workloads* podem ser encontrados em (Dodonov *et al.*, 2006) e (Mello *et al.*, 2009)

Figura 5.4: Predição com rede *TDNN* em função das características comportamentais de processos, extraídos por técnica de monitoramento periódico

técnicas que consideram o histórico e contexto de execução da aplicação.

Dessa forma, a próxima etapa considerou a rede *TDNN* para realizar predições. Empregou-se a técnica de auto-correlação (seção 4.4) para determinar o valor adequado do *lag* (número de estados passados necessários para realizar a predição). Os valores de *lag* correspondentes a cada *workload* são apresentados na tabela 5.2. Após calcular os *lags*, a rede *TDNN* foi construída e configurada de acordo com o valor de *lag* correspondente, tendo uma camada de entrada com $n = lag$ neurônios, duas escondidas, compostas, respectivamente, por $lag \cdot \frac{2}{3}$ e $lag \cdot \frac{1}{3}$ neurônios, e uma de saída, composta por 1 neurônio, utilizada para indicar o próximo elemento da série (Dodonov *et al.*, 2006; Mello *et al.*, 2009). A rede foi treinada com 80% das entradas e, após o fim do treinamento, as saídas da rede foram utilizadas como entradas sucessivas. Comparou-se os resultados previstos com os esperados, averiguando a precisão da predição realizada por meio do cálculo do valor de *MSE* (*Mean Squared Error*) (Shefler, 1988; Jain, 1991). Os resultados da predição para cada aplicação são apresentados na figura 5.4.

Tabela 5.2: Valores de *lag* adequados para aplicações de *NAS Parallel Benchmark*

Aplicação	lag	Aplicação	lag
CG.A	12	CG.B	48
IS.A	6	IS.B	72

Como observado, essa abordagem resultou em um erro médio inferior a 25% para todos os casos, sendo que ele tende a decrescer ao longo da execução. Dessa forma, os resultados obtidos são boas estimativas sobre comportamentos de processos, o que auxilia na tomada de decisões autônomas pelo ambiente. Entretanto, a divergência da precisão de predição para diversos tipos de *workload* (Mello *et al.*, 2009) motivou estudos detalhados sobre a natureza do comportamento de processos.

Ao estudar o comportamento extraído por meio de técnicas de monitoração contínua,

observou-se que essa abordagem tende a esconder a dinâmica interna do comportamento de processos, uma vez que diversos eventos de comunicação podem ocorrer no intervalo de monitoração, os quais são vistos como um único evento. Em função disso, novos estudos foram realizados, utilizando a abordagem de extração de eventos de comunicação baseada no projeto *GridBox* (Dodonov *et al.*, 2004), obtendo a seqüência de eventos de comunicação (delimitados pelas chamadas *MPI*), e considerando o tipo de operação (leitura ou escrita de dados), tempo de processamento decorrido entre operações consecutivas, localização e quantidade de dados trafegados. Essa abordagem resultou em conjuntos de dados passíveis de análises detalhadas, conforme observado na figura 5.5, onde os resultados obtidos pela abordagem de monitoração são apresentados em conjunto com aqueles obtidos pela interceptação de eventos.

Como pode ser observado na figura 5.5, enquanto a extração periódica resultou em uma média de uso da rede, a baseada em eventos possibilitou delimitar a seqüência de diferentes operações de entrada e saída de dados de forma mais precisa³. Nota-se também que, enquanto o comportamento extraído por meio de monitoramento difere para todos os casos, o baseado em eventos apresentou características bastante similares. Isso se deve ao fato de que a extração baseada em monitoração considera médias de observações ao longo de um período de tempo, e a baseada em eventos considera dados exatos do momento de comunicação.

Os resultados obtidos por extração baseada em eventos possibilitaram representação mais precisa das características comportamentais de processos. Esses resultados foram confirmados ao realizar o cálculo de expoentes de Lyapunov e Hurst, apresentados na tabela 5.3. Conforme observado, todos os *datasets* apresentaram características caóticas (demonstrado pelo valor maior que zero de expoente de Lyapunov (Kantz, 1994)). Entretanto, enquanto extração baseada em monitoração apresenta baixa previsibilidade de valores futuros em função do passado (caracterizado pelo valor do expoente de Hurst próximo a 0,5), a baseada em eventos resulta em avaliação mais precisa, a qual evidencia maior previsibilidade da série⁴.

Tabela 5.3: Avaliação das propriedades caóticas de aplicações do NAS

<i>Dataset</i>	Expoente de Lyapunov		Expoente de Hurst	
	Kernel	GridBox	Kernel	GridBox
<i>IC.A</i>	2,15184	3,35258	-	0,228951
<i>IC.B</i>	2,60094	3,96393	-	0,228844
<i>CG.A</i>	1,66817	4,27719	0,481394	0,0756644
<i>CG.B</i>	2,47148	5,11646	0,484655	0,082846

Após isso, os experimentos de predição foram repetidos, utilizando dados extraídos por meio da abordagem baseada em eventos, tendo os resultados apresentados na figura 5.6. Como pode ser observado, essa abordagem resultou em predição mais adequada e estável. Além disso,

³Para o *workload CG.A*, apenas os 100 primeiros eventos de comunicação estão apresentados na figura 5.5, a fim de simplificar a visualização

⁴É importante notar que, devido à rápida execução das aplicações *IS.A* e *IS.B*, a abordagem baseada em monitoramento periódico resultou em pequeno número de observações, o que impossibilitou o cálculo de expoente de Hurst da série.

Figura 5.5: Padrão de acesso de *NAS Parallel Benchmark*

Figura 5.6: Predição com rede *TDNN* em função das características comportamentais de processos, extraídos por técnica baseada em eventos

pode-se observar que a predição com base no comportamento extraído por meio de interceptação de eventos apresentou resultados idênticos para *IS.A* e *IS.B*, devido a similaridades entre as seqüências de operações. Dessa forma, os resultados de predição para um dos *workloads* podem ser utilizados para estimar eventos de um outro, caracterizado por comportamento similar, e servir como base para os mecanismos de auto-configuração e auto-otimização do ambiente.

Para exemplificar a predição realizada por rede *TDNN*, considere a linha 1 da tabela 5.4, que apresenta o resultado de predição em função de 5 eventos históricos ($lag = 5$). Quando a aplicação *IS.A* realiza uma série de transições de estado 0 para estados 1, 0, 0 e 1029, o próximo estado esperado é o estado 0 (neste caso, os números dos estados correspondem à estimativa do tráfego, de acordo com agrupamento realizado pela rede *ART-2A*). O estado previsto é retro-alimentado na rede, realizando predições sucessivas. Com base nisso, pode-se tomar as decisões de auto-gerenciamento, tais como migração de aplicação para um *EP* com capacidade de processamento superior, ou antecipação de acessos a dados.

Tabela 5.4: Padrões de acesso do *NAS*, com $lag = 5$, predizendo 1 evento futuro

Número de padrões	IS.A 48	CG.A 3356
Padrão observado	$0 \Rightarrow 1 \Rightarrow 0 \Rightarrow 0 \Rightarrow 1029$	$0 \Rightarrow 0 \Rightarrow 0 \Rightarrow 1 \Rightarrow 0$
Próximo evento esperado	0	0
Padrão observado	$1 \Rightarrow 0 \Rightarrow 0 \Rightarrow 1029 \Rightarrow 0$	$0 \Rightarrow 0 \Rightarrow 1 \Rightarrow 0 \Rightarrow 0$
Próximo evento esperado	0	0
Padrão observado	$0 \Rightarrow 0 \Rightarrow 1029 \Rightarrow 0 \Rightarrow 0$	$0 \Rightarrow 1 \Rightarrow 0 \Rightarrow 0 \Rightarrow 0$
Próximo evento esperado	2106432	1029
Padrão observado	$0 \Rightarrow 1029 \Rightarrow 0 \Rightarrow 0 \Rightarrow 2106432$	$1 \Rightarrow 0 \Rightarrow 0 \Rightarrow 0 \Rightarrow 1029$
Próximo evento esperado	0	0
Padrão observado	$1029 \Rightarrow 0 \Rightarrow 0 \Rightarrow 2106432 \Rightarrow 0$	$0 \Rightarrow 0 \Rightarrow 0 \Rightarrow 1029 \Rightarrow 0$
Próximo evento esperado	0	0

Esse estudo inicial apresentou resultados promissores (Dodonov *et al.*, 2006; Mello *et al.*, 2009) e demonstrou a viabilidade em se aplicar técnicas de predição para prover suporte a

mecanismos de auto-gerenciamento para ambientes distribuídos. Isso motivou estudos mais detalhados, visando avaliar mecanismos de predição em um estudo de caso prático, conforme apresentado a seguir.

5.6 Provendo auto-otimização a ambientes distribuídos por meio do emprego de dinâmicas comportamentais em escalonamento de processos

Após a realização de experimentos iniciais, que comprovaram a viabilidade de uso de técnicas de predição para prover autonomia a ambientes distribuídos, um novo estudo foi conduzido a fim de avaliar de forma prática os benefícios obtidos pela abordagem proposta. Neste estudo, selecionou-se o aspecto de auto-otimização, particularmente, aquele voltado para a redução do tempo de resposta (ou execução) de aplicações que executam sobre grades computacionais. Para avaliar essas otimizações, uma nova política de escalonamento foi proposta, denominada *PredRoute*, que considera o comportamento histórico de utilização de *CPU* e utiliza, também, informações sobre eventos futuros de comunicação entre processos. O problema de escalonamento de processos em grades e a abordagem da política *PredRoute* são descritos nas próximas seções.

5.6.1 Problema de Escalonamento de Processos

Seguindo a formalização de Garey & Johnson (1979), caracteriza-se o problema de otimização de escalonamento de processos como descrito a seguir. Seja o conjunto A de aplicações paralelas $A = \{a_0, a_1, \dots, a_{k-1}\}$ e $\text{tam}(\cdot)$ a função que define o número de processos que compõe cada aplicação. Assim, cada uma das k aplicações é constituída por uma diferente quantidade de processos, por exemplo, $\text{tam}(a_0) = 5$, $\text{tam}(a_1) = 8$ etc. Considere, então, que o conjunto P contém todos os processos de todas as k aplicações paralelas. Sendo assim, a quantidade de elementos em P é igual a $|P| = \sum_{i=0}^{k-1} \text{tam}(a_i)$.

Cada processo $p_j \in P$, com $j = 0, 1, \dots, |P| - 1$, contém características particulares, aqui denominadas comportamento, de utilização dos recursos de processamento, memória, e entrada e saída. Cada processo, portanto, requer quantidades distintas de recursos oferecidos por um conjunto V de computadores do ambiente distribuído (onde $|V|$ define o total de computadores). Além disso, cada computador $v_w \in V$, onde $w = 0, 1, \dots, |V| - 1$, tem capacidades distintas no que se refere ao poder de processamento, latência de acesso e capacidade de armazenamento da memória primária, vazão para leitura e escrita da memória secundária, largura de banda, latência e *overhead* para empacotamento de mensagens. Os computadores em V estão conectados por diferentes redes de comunicação, sendo que todo esse ambiente pode ser modelado por um grafo não dirigido $G = (V, E)$, onde cada vértice representa um computador

$v_w \in V$ e os canais de comunicação entre pares de vértices são arcos desse grafo $\{v_w, v_m\} \in E$. Tais canais de comunicação têm propriedades de largura de banda e latência associadas.

O problema de otimização consiste, portanto, em escalonar o conjunto P de processos sobre os vértices do grafo G a fim de minimizar o tempo total de execução de cada aplicação em A . Esse tempo total é caracterizado pelo somatório de custos de operações de processamento, acesso a memória, disco rígido e rede.

Para simplificar, considere uma instância do problema com duas aplicações paralelas compostas pelos processos apresentados na tabela 5.5, executados no ambiente composto por um conjunto V de computadores, descritos na tabela 5.6 e interconectados conforme a figura 5.7 (MI representa o número de instruções executadas pelos processos em milhões; MIPS representa a capacidade de processamento em milhões de instruções por segundo; ML e ME são, respectivamente, a quantidade de Kbytes por segundo lidos e escritos na memória principal; HDL e HDE são Kbytes por segundo lida e escrita em memória secundária; NETR e NETE são, respectivamente, a quantidade em Kbytes por segundo recebida e enviada na rede de comunicação – nesse caso, também é apresentado o processo origem dos dados para NETR e o destino para NETE).

Tabela 5.5: Comportamento dos processos das aplicações paralelas

Aplicação 1							
Processo	CPU (MI)	ML Kb/s	ME Kb/s	HDL Kb/s	HDE Kb/s	NETR Kb/s	NETE Kb/s
p_0	1.234	123,78	0,00	78,21	0,00	$12,50 - p_1$	$532,12 - p_1$
p_1	1.537	23,54	89,45	0,00	12,30	$532,12 - p_0$	$12,50 - p_0$
Aplicação 2							
Processo	CPU (MI)	ML Kb/s	ME Kb/s	HDL Kb/s	HDE Kb/s	NETR Kb/s	NETE Kb/s
p_2	1.221	823,78	70,00	78,21	543,00	$10,92 - p_3$	$321,12 - p_4$
p_3	1.137	223,54	179,45	324,00	212,31	$423,12 - p_4$	$10,92 - p_2$
p_4	2.237	23,54	17,45	12,00	0,00	$321,12 - p_2$	$423,12 - p_3$

Tabela 5.6: Características dos computadores

Computador	CPU (MIPS)	ML Kb/s	ME Kb/s	HDL Kb/s	HDE Kb/s
v_0	1.200	100.000	40.000	32.000	17.00
v_1	2.100	120.000	50.000	42.000	19.00
v_2	1.800	100.000	30.000	22.000	9.00
v_3	1.700	95.000	20.000	25.000	11.00
v_4	2.500	110.000	60.000	62.000	30.00

Considerando que o operador de alocação, ou escalonamento, é definido por $\propto$, o problema de escalonamento consiste em definir sobre qual computador v_w cada processo p_j será alocado considerando capacidades e ocupações de recursos. Um exemplo de solução para essa instância é dada por $p_0 \propto v_0$, $p_1 \propto v_1$, $p_2 \propto v_2$, $p_3 \propto v_3$ e $p_4 \propto v_4$. Dessa maneira, para cada instância do problema deve-se escalonar $|P|$ processos sobre um ambiente composto por $|V|$ computadores, conseqüentemente, o universo de possíveis soluções de escalonamento é igual

Figura 5.7: Exemplo de rede de interconexão e alocação de processos

a $|V|^{P|}$. Para a instância anteriormente apresentada, o problema teria um universo de soluções igual a $5^5 = 3.125$. Instâncias reais para o problema podem considerar, por exemplo, 1.024 computadores e 64 aplicações, contendo 512 processos cada. Nesse caso o espaço de soluções seria igual a $1.024^{64 \times 512} = 1.024^{32.768}$.

Nesse contexto, observa-se que o espaço de soluções para o problema de escalonamento é exponencial e deve-se, portanto, propor abordagens capazes de solucionar o problema em tempo computacional aceitável. Não se conhece um algoritmo de tempo polinomial que seja capaz de solucionar o problema de forma ótima (ou seja, encontrar a melhor solução), o que permite caracterizá-lo como intratável (Garey & Johnson, 1979). Para compreender a complexidade de uma solução para o problema, considere o problema NP-completo equivalente estudado por Lageweg & Lenstra (1977), também abordado por Garey & Johnson (1979). Esse problema define o escalonamento de processos sobre m processadores conforme:

Instância: Seja o conjunto de tarefas T , um número $m \in \mathbb{Z}^+$ de processadores, para cada tarefa $t \in T$ um comprimento $l(t) \in \mathbb{Z}^+$ e um peso $w(t) \in \mathbb{Z}^+$, e um inteiro positivo K .

Questão: Há um escalonamento σ sobre m processadores para T , tal que o somatório, para todo $t \in T$, de $(\sigma(t) + l(t)) \cdot w(t)$ não é maior que K .

No contexto deste trabalho, ao invés de considerar m processadores, considera-se m computadores com suas capacidades específicas e T , ao invés de representar tarefas, representa processos. O comprimento $l(t)$ define a duração do processo, nesse caso, o consumo de recursos. Além disso, deve-se ressaltar que, segundo Garey & Johnson (1979), o problema permanece NP-completo para qualquer instância com $m \geq 2$. O problema pode, somente, ser resolvido em tempo polinomial caso as durações de processos sejam idênticas, o que não é esperado em ambiente de grade computacional, onde há grande variedade de aplicações para execução.

Dado esse fato, é necessário adotar algoritmos que explorem parte do espaço de soluções e encontrem, por meio de suposições e suas avaliações, uma boa candidata em tempo polinomial não determinístico (Garey & Johnson, 1979). A melhor solução para o problema

de escalonamento seria encontrada por um método exato, que avaliaria todas as possibilidades. Contudo, esse consumiria muitos recursos. Como alternativa a esses métodos, algoritmos de aproximação têm sido propostos (Blum & Roli, 2003), os quais buscam por soluções satisfatórias em tempo polinomial. Nos últimos anos, alguns algoritmos de aproximação têm sido mais explorados e comumente adotados para solucionar tais problemas, dentre eles estão os algoritmos genéticos (Goldberg, 1989; Semenov & Terkel, 2003), algoritmos baseados em colônias (Bonabeau *et al.*, 2000; Dorigo & Di Caro, 1999) e *Simulated Annealing* (Kirkpatrick *et al.*, 1983).

Nesse contexto, esta tese propõe uma política de escalonamento, denominada *PredRoute*, que é baseada na técnica de otimização de algoritmos genéticos para encontrar soluções para o problema anteriormente descrito, avaliando as tendências comportamentais de processos de forma a compreender e estimar eventos de comunicação e a utilização de processamento. As técnicas adotadas neste trabalho são descritas a seguir.

5.6.2 Técnicas adotadas

Ao avaliar as técnicas de predição abordadas no estudo inicial, averiguou-se a viabilidade em aplicar o conhecimento sobre processos para prover capacidades de auto-gerenciamento a ambientes distribuídos. Isso motivou o emprego de abordagens estudadas de predição e análise de dinâmicas comportamentais de processos para prover auto-otimização ao ambiente por meio de escalonamento eficiente.

Em função disso, foram realizados novos estudos com o objetivo de escolher técnicas adequadas para realizar a predição do comportamento de processos em sistemas distribuídos (Mello *et al.*, 2006a; Dodonov *et al.*, 2006; Dodonov & Mello, 2007; Mello *et al.*, 2007b,a). Durante esses estudos observou-se que o comportamento de processos pode apresentar alterações durante a sua execução, o que tende a degradar decisões de escalonamento convencionais. Além disso, observou-se que as abordagens de predição que consideram o comportamento médio nem sempre resultam em estimativas adequadas: por exemplo, considerando a média do comportamento de uma aplicação com utilização variável de rede e *CPU*, essa pode não representar os eventos observados de maneira adequada, prejudicando a tomada de decisões. Nesse caso, um escalonamento dinâmico de aplicações pode apresentar ganhos significativos de desempenho, migrando processos somente quando necessário.

Com base nos estudos realizados, foram estabelecidos princípios básicos para a escolha das técnicas mais adequadas para predição de comportamento de aplicações:

Princípio 1: Predição eficiente é preferida à predição exata. O mecanismo de escalonamento geralmente tem tempo limitado para avaliar o comportamento de processos e tomar decisões. Predição rápida, mesmo que menos precisa é, portanto, mais adequada.

Em outras palavras, *pode tornar-se mais rápido escalonar processos do que aguardar por resultados exatos de predição.*

Princípio 2: Predição eficiente de curto prazo é preferida à predição de longo prazo.

Esta afirmação é baseada no fato de que o mecanismo de escalonamento deve tomar decisões sobre a migração de processos levando em consideração limitações do ambiente. A predição de longo prazo tende a ser pouco eficaz, uma vez que outras tarefas podem entrar em execução, alterando comportamentos esperados. A predição eficiente de curto prazo, por sua vez, possibilita realizar o escalonamento de uma mesma tarefa diversas vezes, mesmo que seu comportamento sofra alterações.

Em outras palavras, *é mais importante conhecer as próximas tendências de execução do que eventos a longo prazo.*

Princípio 3: A predição de padrões comportamentais é preferida à predição de eventos independentes.

O comportamento de processos pode sofrer alterações durante a execução devido a mudanças no ambiente e interferência de outros processos. Entretanto, enquanto isso pode prejudicar a predição de eventos específicos, o padrão de execução é mantido (Feitelson *et al.*, 1997; Dodonov & Mello, 2007, 2009b). A predição de padrões de execução de processos pode, portanto, ser mais eficiente para escalonamento.

Princípio 4: Predição adaptativa é preferida em relação a abordagens estáticas.

Complementando o princípio anterior, este tem por objetivo focar em abordagens que realizam predições sobre seqüências de eventos potencialmente variáveis. As abordagens baseadas em regressões simples e reconstruções de comportamento não oferecem precisão adequada, por não considerarem a evolução comportamental ao longo do tempo (Mello & Yang, 2008).

Princípio 5: As predições devem ser baseadas no comportamento histórico e não em estimativas momentâneas.

Um mesmo estado de execução pode levar a diversas transições comportamentais (Dodonov *et al.*, 2006; Dodonov & Mello, 2007). Com isso, predições baseadas somente no estado de execução atual podem gerar resultados imprecisos.

Com base nessas afirmações, as técnicas discutidas no capítulo 3 foram analisadas (Dodonov & Mello, 2008). *A priori* todas podem ser utilizadas para predizer comportamentos de processos. Contudo, elas apresentam limitações quando avaliadas em função dos princípios propostos. O estudo dessas limitações e a escolha das técnicas mais adequadas são discutidos a seguir.

Inicialmente, as abordagens baseadas em classificação estática e regressão foram eliminadas, uma vez que não oferecem resultados adequados para seqüências variáveis (*Princípios*

4 e 5). As abordagens baseadas em cadeias de Markov e filtros de Kalman também foram desconsideradas, por não cumprirem o *Princípio 5*.

As abordagens baseadas em atrasos temporais, mapas auto-organizáveis e redes auto-expansíveis também foram eliminadas, uma vez que necessitam de treinamento prolongado para realizar previsões eficientes, não satisfazendo o *Princípio 1* (Dodonov & Mello, 2008). A rede neural de *Hopfield* e máquinas de *Boltzmann*, por sua vez, são mais eficientes para problemas de otimização (Nakano-Miyatake & Perez-Meana, 1997a).

Com isso, a escolha foi direcionada para as redes recorrentes (*RRBF* e *LSTM*), técnicas de *Reservoir Computing* e rede neural *SRNN*. Essas abordagens foram avaliadas com maior profundidade, conforme apresentado a seguir.

Inicialmente, foi avaliada a rede neural *LSTM*. Essa rede é composta por diversos blocos de memória, sendo que cada um pode conter uma ou mais células de memória, e as unidades de proteção de entrada, saída e esquecimento. A execução da rede é eficiente, apresentando complexidade $O(n)$. A fase de adaptação, entretanto, tem complexidade $O(n^4)$. Devido à natureza da rede, não há uma distinção clara entre as duas fases, uma vez que a transição de uma para outra é controlada pelas unidades de proteção de memória. A rede é caracterizada por um grande número de elementos funcionais, o que dificulta sua parametrização. Além disso, a capacidade de memória dessa rede é limitada à quantidade de blocos utilizados, o que dificulta sua adoção para seqüências complexas (Gers *et al.*, 2002).

As abordagens utilizadas pelas técnicas de *Reservoir Computing* e pela rede neural *SRNN* são similares, no sentido que ambas consideram um grande conjunto de neurônios interconectados (conhecido como *reservoir*). Durante a etapa de treinamento, esses neurônios são treinados por meio de uma função de supervisão, e a dinâmica interna do *reservoir* é utilizada para realizar previsões de eventos. Entretanto, a natureza desses sistemas tem grande complexidade de espaço e, conseqüentemente, influencia no tempo de execução (Jaeger, 2007).

Finalmente, foi avaliada a abordagem introduzida pela rede *RRBF*. Essa rede é baseada na rede *RBF*, e inclui um mecanismo de memória dinâmica. Com isso, a rede tem custo de execução e de treinamento baixos, ao mesmo tempo que oferece mecanismos precisos para agrupamento de padrões e previsão (Zemouri *et al.*, 2003). Resultados apresentados por Zemouri *et al.* (2003) motivaram a adoção da rede neural *RRBF*, pois comprovam sua capacidade de adaptar saídas em função da dinâmica de séries e demonstram que a rede tem boa eficiência e baixo custo computacional. A arquitetura e funcionamento dessa rede são apresentados a seguir.

Rede neural *RRBF*

A rede neural *RRBF* estende a *RBF* tradicional, introduzindo uma primeira camada recorrente (figura 5.8) para proporcionar funcionalidades de memória dinâmica. Essa camada auxilia na representação de reações causadas por perturbações em séries temporais e, portanto,

Figura 5.8: Arquitetura da rede neural artificial RRBF

na modificação de tendências para predição de comportamento futuro. A segunda camada dessa arquitetura é composta por neurônios com funções radiais de ativação.

$$x(t) = \frac{1 - \exp(-kw_{ii}x(t-1))}{1 + \exp(-kw_{ii}x(t-1))} \quad (5.4)$$

A primeira camada calcula a relevância de padrões de entrada históricos sobre um novo padrão, através da função sigmóide apresentada na equação 5.4, onde $x(t-1)$ representa o valor de uma entrada anterior, k parametriza a função de ativação, e w_{ii} é o peso dado para conexões recorrentes entre neurônios (Zemouri *et al.*, 2003). Valores resultantes da camada de recorrência são processados pela próxima camada, constituída por neurônios de uma rede neural *RBF* tradicional, a qual emprega funções radiais para ativação (geralmente baseadas em Gaussianas e Multiquadráticas). Essas funções são tipicamente projetadas para aproximar funções em relação a dados de entrada. Resultados da *RBF* são gerados pela equação 5.5, que realiza o produto de saídas das N funções radiais $\phi(\cdot)$, centradas em c_i , pelos coeficientes de peso w_i (Powell, 1987).

$$y(x) = \sum_{i=1}^N w_i \cdot \phi(\|x - c_i\|) \quad (5.5)$$

O algoritmo de uma rede *RBF* consiste de duas fases: treinamento e execução. A etapa de treinamento determina o número de neurônios necessários para representar uma série de entrada, os valores para os centros c_i das funções radiais $\phi(\cdot)$ e os coeficientes de peso w_i . Os centros podem ser definidos por valores fixos ou por uma heurística. Coeficientes de peso são calculados de acordo com a equação 5.6, onde w representa os coeficientes, $b = [c_0, c_1, \dots, c_n]$ os centros de funções, e G a matriz de pesos na forma $g_{ij} = \phi(\|c_j - c_i\|)$. Durante a fase de execução, essa rede neural avalia padrões de entrada e computa suas similaridades em relação aos centros (determinadas pelos coeficientes de peso w), utilizando a equação 5.5.

$$w = G^{-1} \times b \quad (5.6)$$

Figura 5.9: Arquitetura da rede neural artificial SONDE

Diversas técnicas podem ser empregadas para definir centros para as funções radiais $\phi(\cdot)$ (Chiu *et al.*, 2003; C. Gruber, 2006; Titsias & Likas, 2000; Powell, 1987). No contexto deste trabalho, adotou-se a rede neural artificial auto-organizável e não supervisionada *SONDE* (*Self-Organizing Novelty Detection*) (Albertini & Mello, 2007), apresentada a seguir.

Rede neural artificial SONDE

A rede neural artificial *SONDE* foi adotada neste trabalho a fim de encontrar agrupamentos, os quais são utilizados para definir centros para neurônios da segunda camada da *RRBF*. A *SONDE* classifica e agrupa padrões de entrada de forma não-supervisionada e *online* e foi inicialmente proposta para detectar novidades em séries temporais (Albertini & Mello, 2007).

A rede *SONDE* foi projetada com o objetivo de integrar características presentes nas técnicas *SOM* (Kohonen *et al.*, 2000), *GWR* (Marsland *et al.*, 2002) e *ART* (Carpenter & Grossberg, 1987). A arquitetura dessa rede é dividida em três camadas (figura 5.9): camada de entrada e pré-processamento – na qual padrões podem ser, opcionalmente, normalizados; camada de neurônios competitivos – na qual ocorre a ativação de neurônios segundo padrões de entrada; e última camada para escolher a unidade mais representativa *BMU* (*best matching unit*) – na qual o melhor neurônio, com ativação mais alta, é estimulado para melhor representar o padrão de entrada recebido.

A *SONDE* representa dados de entrada em neurônios adaptativos, sendo que novos neurônios são criados conforme novidades são detectadas. Cada neurônio c é definido pelo centróide médio $\vec{w}_c$ de seus padrões, raio médio, rad_c , em relação à dispersão desses padrões, e um grau mínimo de similaridade α_c para reconhecer novos padrões. Os padrões de entrada são agrupados de forma não supervisionada, e, após um padrão ser classificado em um neurônio, essa unidade é estimulada para representar um histórico de dados de entrada. Todos os neurônios têm um raio máximo de ação para representar os padrões detectados.

O raio de ativação e o centróide de um neurônio adaptam-se conforme um novo padrão é classificado. Quando nenhum neurônio é capaz de classificar um padrão, um novo é criado, indicando novidade. De acordo com alterações nas entradas, neurônios adaptam-se, esquecendo informações passadas. Os neurônios são descritos por equações que relacionam a entrada atual e o armazenamento dos resultados passados. Para cada padrão $\vec{I}_t$ recebido (opcionalmente normalizado na camada de entrada pela equação 5.7) no instante t , o valor de ativação a_c de cada neurônio da camada competitiva é calculado segundo a equação 5.8. Quando nenhum neurônio é capaz de representar $\vec{I}_t$, ou seja, quando a equação 5.9 é satisfeita, um novo neurônio é criado, e o seu centróide definido segundo os valores dos padrões responsáveis por sua criação (resultando em ativação máxima com $a_{new} = 1$). O nível de similaridade mínimo, α_{new} , é definido segundo a constante α_0 , e o raio médio inicial rad_{new} recebe o valor $-\ln(\alpha_0)$.

$$\vec{I}_t = \frac{\vec{I}_t}{\|\vec{I}_t\|} \quad (5.7)$$

$$a_c = \exp(-\|\vec{I}_t - \vec{w}_c\|) \quad (5.8)$$

$$a_c < \alpha_c, \forall c \quad (5.9)$$

Quando a equação 5.10 é satisfeita, o neurônio vencedor (*BMU*) é obtido por $BMU = \operatorname{argmax}_c(\exp(-\|\vec{I} - \vec{w}_c\|))$. Esse neurônio é aquele que melhor representa o padrão de entrada.

$$\|\vec{I} - \vec{w}_c\| \leq -\ln \alpha_c \quad (5.10)$$

O neurônio vencedor é adaptado a fim de especializar-se (equação 5.11, com t sendo o instante de tempo). Essa especialização é realizada de tal maneira que o raio e centróide do neurônio representem os valores de entrada, segundo as médias móveis exponenciais ponderadas (*EWMA*, *exponential weighted moving average*) apresentadas, respectivamente, nas equações 5.11 e 5.12. Os parâmetros γ e Ω dessas equações definem a influência de padrões passados na situação atual. Quanto maior é o valor desses parâmetros, maior é a taxa de esquecimento da *SONDE*.

Após atualizar o centróide do neurônio vencedor, seu grau de similaridade mínimo (α_c) é modificado para melhor representar padrões de entrada em próximas ativações. Após serem especializados, qualquer padrão divergente pode ser detectado como novidade e ser utilizado na criação de um novo neurônio. Essa atualização é proporcional à taxa de modificação p (equação 5.13) do novo raio médio em relação ao anterior (seguindo a equação 5.14), que assegura dois comportamentos de adaptação diferentes para α_c , os quais seguem a distância entre o raio (que define um limite interno para a especialização do neurônio) e o limite de similaridade $-\ln(\alpha_c)$.

Quanto maior é essa distância, mais acelerada é a especialização a qual converge para representar padrões de entrada.

Quando padrões são uniformemente classificados pela área de cobertura, o limite de similaridade torna-se próximo ao interno, o que implica que o neurônio tem boa adaptação aos padrões de entrada. Esse comportamento é similar ao treinamento da *SVM* (Schölkopf & Smola, 2001), embora sua fase de aprendizado não seja *online*, nem sua execução adaptativa.

$$\vec{w}_{BMU_t} = \vec{w}_{BMU_{t-1}} \cdot (1 - \gamma) + \vec{I}_t \cdot \gamma \quad (5.11)$$

$$rad_{BMU_t} = rad_{BMU_{t-1}} \cdot (1 - \Omega) + \|\vec{I}_t - \vec{w}_{BMU_{t-1}}\| \cdot \Omega \quad (5.12)$$

$$p = \left\| \frac{rad_{BMU_t} - rad_{BMU_{t-1}}}{\max(rad_{BMU_t}, rad_{BMU_{t-1}})} \right\| \quad (5.13)$$

$$\alpha_{BMU_t} = \min((1 + p) \cdot \alpha_{BMU_{t-1}}, \exp^{(-rad_{BMU_t} \cdot (1+p))}) \quad (5.14)$$

Além de redes *RRBF* e *SONDE*, adotadas para realizar previsões sobre comportamento de processos, o modelo de escalonamento proposto neste estudo de caso emprega algoritmos genéticos, a fim de avaliar decisões de alocação de processos e encontrar uma boa solução para o problema. Conceitos sobre algoritmos genéticos, utilizados neste trabalho, são apresentados a seguir.

Algoritmos Genéticos

Algoritmos genéticos são aplicados como técnicas de busca e otimização em diversas áreas. Eles são baseados no mecanismo de seleção natural, focando na sobrevivência do indivíduo mais apto. Esses algoritmos nem sempre encontram a melhor solução possível, porém, fornecem boas soluções locais para problemas NP-completos (Papadimitriou, 1994).

A solução de problemas utilizando algoritmos genéticos envolve dois aspectos: codificação da solução na forma de cromossomos, onde cada cromossomo representa uma possível solução, e uma função de aptidão (*fitness*) que é aplicada para encontrar a melhor solução. Várias técnicas de codificação podem ser utilizadas para diferentes tipos de problemas, como *strings*, binárias, *bitmaps*, números reais dentre outras. A função de aptidão é responsável por avaliar as possíveis soluções. Essa função recebe um cromossomo como parâmetro e retorna um número real que representa a qualidade da solução obtida, por exemplo, o quão adequada é a solução para o problema em estudo. No contexto deste trabalho, os cromossomos podem representar os elementos de processamento disponíveis no ambiente, e o objetivo da função *fitness* é determinar a melhor combinação dos *EPs* para a alocação das tarefas.

Cromossomos mais aptos são identificados e armazenados durante o processo de evolução. Os menos aptos, por outro lado, são eliminados. Diferentes técnicas podem ser aplicadas para a identificação dos melhores cromossomos, como a seleção proporcional, seleção por *ranking* e a seleção baseada em torneio (Back *et al.*, 1999).

Na seleção proporcional, indivíduos são transferidos para a próxima geração de acordo com o valor proporcional ao resultado de sua função de aptidão. Uma das possíveis implementações dessa técnica consiste no uso de uma roleta, dividida em N partes, sendo N o número de indivíduos (cromossomos) da população atual. O tamanho de cada parte é proporcional ao valor da função de aptidão de cada indivíduo. A roleta é girada N vezes e, a cada turno, o indivíduo apontado é selecionado e inserido na próxima geração.

A seleção baseada em *ranking* pode ser dividida em duas etapas. Durante a primeira, as soluções são ordenadas de acordo com seu valor da função de aptidão. Com a lista ordenada, cada indivíduo recebe um novo valor de aptidão de acordo com sua posição no *ranking*. Após isso, um procedimento que seleciona os indivíduos conforme sua posição no *ranking* é aplicado. Assim, os indivíduos de melhores posições têm maiores chances de serem selecionados.

Na seleção baseada em torneio não se atribui automaticamente probabilidades a indivíduos. Um torneio de tamanho k é definido, com $k \geq 2$ indivíduos. Então, k indivíduos são escolhidos aleatoriamente na população atual, e seus valores de aptidão são comparados e o indivíduo com maior valor de aptidão é selecionado para reprodução. O valor de k é definido pelo usuário, representando a pressão de seleção, isto é, a velocidade com a qual os indivíduos mais aptos vão dominar a população, eliminando os menos adaptados.

Uma vez selecionados os indivíduos para reprodução, é necessário modificar suas características usando técnicas de reprodução conhecidas como operadores genéticos. Os operadores mais comuns são o cruzamento e a mutação. O operador de cruzamento permite a troca de material genético entre dois indivíduos, conhecidos como pais, combinando suas informações de modo a aumentar a possibilidade de gerar um novo indivíduo com melhores características que os originais (Hinterding, 2000). O operador de mutação é utilizado para alterar um único gene por um valor aleatório. Quando um indivíduo é representado por um *bitmap*, esse operador escolhe de forma aleatória um gene do cromossomo e troca seu valor de 1 para 0 e vice-versa. O objetivo do operador de mutação é manter a diversidade da população, sempre permitindo que um cromossomo cubra um amplo espaço de busca (Hinterding, 2000). Esse operador é geralmente aplicado com baixa probabilidade, pois, com alta, o resultado tende a ser aleatório.

A possibilidade de utilização de algoritmos genéticos para estimar a melhor distribuição das tarefas pelo ambiente motivou a sua adoção na proposta de uma nova política de escalonamento. Essa política é apresentada a seguir.

5.6.3 ***PredRoute*: política de escalonamento distribuído e dinâmico de processos**

A política *PredRoute*, desenvolvida com base nos estudos conduzidos neste projeto de pesquisa, visa encontrar boas soluções de escalonamento utilizando uma metaheurística baseada em algoritmos genéticos e informações previstas sobre o comportamento de comunicação entre processos. Esta política é composta pelas seguintes etapas:

1. Ao iniciar o sistema, calcula-se a latência de comunicação entre todos computadores do ambiente, determinando uma matriz de atrasos global. Essa matriz é, periodicamente, recalculada e utilizada para avaliar o custo envolvido em transferências (migração) e comunicação entre processos;
2. Processos são inicialmente distribuídos sobre os computadores do ambiente utilizando o algoritmo *RouteGA* (Mello *et al.*, 2007b). Essa abordagem utiliza conhecimento adquirido por meio de técnica *IBL* (Aha *et al.*, 1991) para determinar os elementos de processamento mais adequados para receber as tarefas. Após início de execução, extrai-se, contínua e transparentemente, operações de comunicação empregando a abordagem de interceptação de eventos (Dodonov *et al.*, 2006; Dodonov & Mello, 2007; Andrade Filho *et al.*, 2008);
3. Assim que um número mínimo de eventos é capturado (definido manualmente ou por meio de uma heurística), a série temporal de comportamentos de comunicação de determinado processo é avaliada, e suas dimensões de separação e embutida são determinadas;
4. Inicia-se a montagem da arquitetura de rede neural *RRBF* empregada na regressão de séries temporais de comportamento de processos. O primeiro passo consiste em definir o número de neurônios na primeira camada em função da dimensão de separação. Os proponentes dessa rede sugerem a avaliação de diferentes números de neurônios para essa camada (Zemouri *et al.*, 2003), até obter bons resultados. Contudo, observou-se que adotando a dimensão de separação, encontra-se, automaticamente, o número ideal de neurônios. Esse fato, resultante de conclusões sobre os experimentos conduzidos no decorrer deste trabalho, deve-se à necessidade da primeira camada da *RRBF* ter um número de neurônios capaz de representar a influência de padrões históricos sobre eventos futuros, tal como expresso pela dimensão de separação.

Em seguida, classifica-se os dados utilizando a rede neural artificial *SONDE* (Albertini & Mello, 2007), proposta no contexto do projeto *MidHPC*. Essa rede adota aprendizado contínuo, adaptativo, auto-organizável e não supervisionado, a fim de gerar valores de protótipos, ou centróides, para os agrupamentos de dados, além da dispersão dos padrões classificados em cada um desses agrupamentos. Esses valores são utilizados para definir os parâmetros (centros e abrangência) dos neurônios da segunda camada da arquitetura

Figura 5.10: Estrutura da rede *SRRBF* resultante

Figura 5.11: Arquitetura da política de escalonamento

RRBF, os quais são compostos por funções radiais (tal como a rede neural artificial *RBF* tradicional).

Dessa maneira, os centróides $\vec{w}_c$, de cada neurônio c criado pela rede neural *SONDE*, são utilizados para definir os centros das funções radiais $\phi(\cdot)$, presentes na segunda camada da rede *RRBF*. A arquitetura resultante da combinação das redes *SONDE* e *RRBF*, denominada de *SRRBF* neste trabalho, é apresentada na figura 5.10, onde a memória dinâmica da *RRBF* é composta por neurônios recorrentemente conectados $T_0, T_1, \dots, T_M$, sendo M a dimensão embutida da série. Os neurônios radiais $RBF_1, \dots, RBF_N$ são definidos pela rede neural *SONDE*, onde N é o número de agrupamentos resultantes da etapa de agrupamento;

5. A rede *SRRBF* é treinada com a série que representa o comportamento de processos. Próximos eventos de comunicação são utilizados para quantificar o erro médio quadrático (*MSE*, *Mean Squared Error*) (Lehmann & Casella, 2003). Assim que o erro de treinamento estiver abaixo de um nível pré-estabelecido (por exemplo, 5%), o treinamento cessa, e a rede prediz os próximos N eventos de comunicação, estimando a quantidade de tráfego e os computadores origem e destino das mensagens. Dessa forma, a figura 5.11 apresenta o fluxo de execução da política proposta, sendo que os resultados estimados são submetidos ao mecanismo de escalonamento;
6. Ao receber dados estimados, o mecanismo de escalonamento avalia possíveis decisões a serem tomadas empregando um algoritmo genético. Esse mecanismo calcula o custo de processamento e a latência de comunicação entre computadores para cada solução proposta pelo algoritmo genético. O desempenho esperado de processos é avaliado, resultando na otimização de alocação de aplicações de usuário. O algoritmo genético empregado, baseado no *RouteGA* (Mello *et al.*, 2007b), define soluções segundo cromossomos na forma $cr = \{g_0, g_1, \dots, g_{n-1}\}$, onde índices representam processos a serem escalonados e cada termo, ou gene, g_k propõe um computador destino para o processo k . Esse algoritmo utiliza a função de aptidão definida na equação 5.15, onde ic é o índice do cromossomo a ser avaliado, LV_{ic} é a variação de carga dos computadores do ambiente (definida na equação 5.16), MRT_{ic} representa o tempo de resposta do processo com maior custo de execução (equação 5.17), e η é uma constante utilizada para evitar divisões por zero, caso alguma das variáveis seja nula.

$$F_{ic} = \frac{1}{MRT_{ic} + \eta} + \frac{1}{LV_{ic} + \eta} \quad (5.15)$$

$$LV_{ic} = \sum_{j=1}^i \left(C_j - \frac{\sum_{j=1}^{|ic|} C_j}{j} \right)^2 j \quad (5.16)$$

$$MRT_{ic} = \max(C_j), j = 0, \dots, |j| - 1 \quad (5.17)$$

Nas equações 5.16, 5.17 e 5.18, C_j representa o tempo de execução total da tarefa j , $|ic|$ é o número de termos, ou genes, do cromossomo a ser avaliado, PM_j é o custo total de processamento dessa tarefa, CAP_d representa a capacidade de processamento do computador d , onde o processo j está alocado, CC_j é o custo de comunicação da tarefa j em relação a outras localizadas em demais computadores, e $CPS_j(i_j, i_k)$ é o custo de comunicação entre as tarefas j e k executadas, respectivamente, pelos computadores i_j e i_k .

$$C_j = \frac{PM_j}{CAP_d} + CC_j \quad (5.18)$$

$$CC_j = \sum_{j=1}^{|ic|} \sum_{k=1, j \neq k}^{|ic|} CPS(i_j, i_k) \quad (5.19)$$

As principais diferenças entre o mecanismo proposto e o algoritmo utilizado no *RouteGA* (Mello *et al.*, 2007b) consistem na utilização de operações estimadas de comunicação segundo a equação 5.19, as quais consideram a quantidade de dados transferidos e os computadores destino, e na execução periódica de reescalonamento de acordo com eventos de comunicação previstos ⁵.

Dessa forma, a função de aptidão F_{ic} (equação 5.15) visa a minimização do tempo de resposta de processos com alto custo de execução (MRT_{ic}) e, ao mesmo tempo, busca a menor variação da carga dos computadores do ambiente. Além disso, o algoritmo utiliza informações comportamentais para alocar tarefas em computadores que apresentem latências de comunicação capazes de atender trocas de mensagens. Com isso, atrasos de sincronização, provenientes de comunicação, são reduzidos, o que diminui o tempo total de execução de processos. Após executar o algoritmo genético por diversas gerações, sendo que em cada uma n indivíduos são criados, cruzados e expostos a mutações, ocorre a convergência para uma solução de escalonamento de processos, onde altos valores para a função F_{ic} representam melhores resultados;

7. Processos são transferidos (migrados) conforme a solução proposta pelo algoritmo genético. A migração é realizada de acordo com a política *Route* (Mello & Senger, 2004; Mello *et al.*, 2006a), considerando o tempo necessário para interromper a execução e transferir os processos, de forma a evitar migrações que possam prejudicar o tempo de execução. Em seguida, retoma-se, transparentemente, a execução desses contextos.

Dessa forma, o algoritmo de escalonamento proposto visa reduzir o tempo de comunicação entre os processos, migrando-os para regiões próximas (em termos de atraso de comunicação). Tal abordagem tem o potencial de melhorar, de forma significativa, o desempenho de ambientes de larga escala, ou heterogêneos, onde latências de comunicação apresentam variações consideráveis.

5.6.4 Experimentos e discussões

Neste estudo de caso, foram considerados os *workloads* de aplicações executadas pelo *NERSC* (*National Energy Research Scientific Computing Center*) ⁶. O *NERSC* provê diferentes *workloads* de aplicações paralelas, dentre os quais se selecionou os *datasets* de duas aplicações (*GTC* e *PMEMD*), em duas execuções diferentes: *medium*, que cria 64 processos sobre 4

⁵Neste estudo de caso, o reescalonamento foi executado a cada 100 eventos de comunicação.

⁶As aplicações do *NERSC* são disponíveis em <http://pdsi.nersc.gov/benchmarks.htm>

elementos de processamento e *large*, que distribui 256 processos sobre 16 *EPs*. A comunicação e sincronização entre os processos ocorrem utilizando *MPI*, sendo que um trecho para ilustrar a seqüência de chamadas *MPI* é apresentado na tabela 5.7.

Tabela 5.7: Trecho do conjunto de dados GTC

# org	# dst	chamada <i>MPI</i>	tamanho do buffer (Bytes)	# chamadas	tempo total (s)	tempo mínimo (s)	tempo máximo (s)
197	196	<i>MPI_Sendrecv</i>	2595216	1	$1,6917e-02$	$1,6917e-02$	$1,6917e-02$
228	229	<i>MPI_Sendrecv</i>	2595216	1	$5,8828e-02$	$5,8828e-02$	$5,8828e-02$
248	247	<i>MPI_Sendrecv</i>	2595216	1	$7,4673e-03$	$7,4673e-03$	$7,4673e-03$
160	161	<i>MPI_Sendrecv</i>	2595216	1	$1,9284e-02$	$1,9284e-02$	$1,9284e-02$
211	212	<i>MPI_Sendrecv</i>	2595216	1	$1,5988e-02$	$1,5988e-02$	$1,5988e-02$
14	15	<i>MPI_Sendrecv</i>	2595216	1	$1,0592e-02$	$1,0592e-02$	$1,0592e-02$
109	110	<i>MPI_Sendrecv</i>	2595216	1	$1,2633e-02$	$1,2633e-02$	$1,2633e-02$
215	216	<i>MPI_Sendrecv</i>	2595216	1	$1,5624e-02$	$1,5624e-02$	$1,5624e-02$
60	59	<i>MPI_Sendrecv</i>	2595216	1	$6,4287e-03$	$6,4287e-03$	$6,4287e-03$

As aplicações escolhidas são caracterizadas por grande volume de dados transmitidos, além de alta variação comportamental, o que dificulta a predição de eventos por técnicas convencionais (Mello & Yang, 2008). Essa situação é comprovada pelo expoente de Hurst das seqüências de transmissões de dados (tabela 5.8). O expoente de Hurst caracteriza o conjunto de dados *GTC* como caminhada aleatória (definido pelo valor do expoente próximo a 0,5), o que evidencia falta de correlação com eventos históricos, dificultando estimativas (Alligood *et al.*, 1997). O expoente de Lyapunov indica alguma, mesmo que pouca, correlação de dados históricos, o que tende a apresentar melhores resultados em predições.

Tabela 5.8: Avaliação de expoente de Lyapunov e Hurst para aplicações do *NERSC*

<i>Dataset</i>	Expoente de Lyapunov	Expoente de Hurst
<i>GTC (Large)</i>	1,20287	0,526389
<i>GTC (Medium)</i>	1,56422	0,512138
<i>PMEMD (Large)</i>	0,857796	0,663141
<i>PMEMD (Medium)</i>	0,671032	0,787279

As aplicações do *NERSC* apresentam características diferentes. O *GTC* (*Gyrokinetic Toroidal Code*) é utilizado para estudar micro-turbulências ocorridas em fusão de plasma, e é caracterizado por comportamento *CPU-bound*. *PMEMD* (*Particle Mesh Ewald Molecular Dynamics*), por sua vez, é utilizado para estudos sobre dinâmica molecular, e apresenta comportamento *I/O-Bound*, alternando pequenas e grandes transferências de dados.

Ao estudar a seqüência de eventos de comunicação da aplicação *GTC* (figura 5.12.a), observa-se a dificuldade em prever o seu comportamento, o que é comprovado pelo valor do expoente do Hurst correspondente. Entretanto, ao reconstruir a mesma série utilizando dimensão de separação 2 e dimensão embutida 3, obtém-se a figura 5.12.b, cujo comportamento é mais simples de compreender. Da mesma forma, ao reconstruir o fluxo de execução do *PMEMD* (Figura 5.12.c) com dimensão de separação de 11 e dimensão embutida 4, obtém-se Figura

(a) GTC: Série temporal original

(a) PMEMD: Série temporal original

(b) GTC: Série temporal reconstruída

(b) PMEMD: Série temporal reconstruída

Figura 5.12: Análise temporal caótica do *workload* de NERSC

5.12.d⁷.

Para avaliar a eficiência da política de escalonamento proposto, foram utilizados os *datasets medium* e *large* das aplicações *GTC* e *PMEMD*, descritos previamente. Os experimentos foram conduzidos usando o ambiente de simulação *SchedSim* (Mello & Senger, 2006), que permite realizar experimentos de escalonamento de processos em ambientes distribuídos, simulando elementos com capacidades variável de processamento, memória principal e virtual, leitura e escrita em disco, e latência de comunicação. O *SchedSim* permite, portanto, avaliar diversos tipos de ambientes, tais como *clusters* e grades computacionais.

Neste estudo de caso, a capacidade de processamento de computadores e aplicações foram expressos utilizando *MIPS* (*Millions of Instructions Per Second*), o que permite representar características de processadores reais. Os *traces* de execução foram processados, determinando a quantidade de *MIPS* entre comunicações sucessivas, taxa de comunicação em relação ao tempo total de execução, e a quantidade total de *MIPS* utilizadas durante execução. Com isso, avaliou-se os parâmetros funcionais dos *datasets*, os quais são apresentados na tabela 5.9. Essa tabela

⁷Para simplificar a visualização, a figura 5.12.d foi traçada em 2 dimensões

resume a quantidade total de comunicações entre processos e o tempo de processamento médio, em *MIPS*, decorrido entre comunicações.

Tabela 5.9: Parametrização das aplicações de *NERSC*

<i>Dataset</i>	Execução (s)	<i>GFlops/s</i>	Comunicação (%)	# Pro-cessos	# Mensagens por processo	<i>MIPS</i> médio entre duas mensagens
<i>GTC Large</i>	$9,70225 \cdot 10^3$	32,002	14,68	256	2507	433,324250
<i>GTC Medium</i>	$8,05097 \cdot 10^3$	9,9525	3,314	64	1214	980,080722
<i>PMEMD Large</i>	$2,48490 \cdot 10^3$	15,625	53,1499	256	22526	5,356581
<i>PMEMD Medium</i>	$2,33876 \cdot 10^3$	7,4716	29,178	64	12317	12,718410

Para avaliar a política de escalonamento proposto, foram considerados dois tipos de ambientes distribuídos:

1. Ambiente de *cluster*, caracterizado por máquinas com configuração homogênea e latência de comunicação uniforme. Neste estudo de caso, foram considerados *clusters* de 32 e 128 elementos, com latência de intercomunicação de $40\mu s$ (Dodonov *et al.*, 2005);
2. Ambiente de grade, caracterizado por máquinas com configuração heterogênea e latência de comunicação variável, determinado de acordo com estudos realizados em (Dodonov *et al.*, 2005). Para representar o ambiente de grade, foram avaliadas latências de comunicação em redes *LAN*, *MAN* e *WAN*. Para calcular a latência de comunicação, foi avaliado o tempo necessário para enviar e receber uma mensagem (*RTT*, *Round-Trip Time*) entre dois elementos de processamento. No caso da rede *LAN*, a latência foi avaliada utilizando *clusters* com máquinas interconectadas por rede *Fast Ethernet*. A latência da rede *MAN* foi determinada estimando o *RTT* entre o *ICMC-USP* e o departamento de computação de *UFSCar*. A latência de *WAN* foi obtida avaliando o tempo de comunicação entre *ICMC-USP* e *St. Francis Xavier University*, Canadá.

Os cálculos de latência foram, inicialmente, realizados em 2005, e repetidos em 2008. A distribuição das latências das redes *LAN*, *MAN* e *WAN* é apresentada na figura 5.13. Ao comparar os resultados experimentais de 2005 e 2008, observou-se que valores de latência de redes *MAN* e *WAN* diminuíram, aproximadamente, 10 vezes.

Os dados resultantes foram processados, visando encontrar funções de regressão para representá-los da maneira mais adequada. Os dados de latência de 2005 podem ser representados por uma função de distribuição de probabilidades exponencial com média 11,374, e os de 2008 por função exponencial com média 0,99879. As funções resultantes foram utilizadas para estimar uma matriz de latência de comunicação entre computadores. Essa matriz auxilia a parametrização do simulador com base em dados reais (ambientes de produção).

Com isso, foram gerados os ambientes de grade compostos por 256, 512 e 1024 elementos

Figura 5.13: Avaliações de latências de rede em 2005 e 2008

de processamento, interconectados de maneira aleatória em grupos de até 5 *EPs* para simular ambiente de grade oportunista.

Durante os experimentos, a política de escalonamento proposta, denominada *PredRoute*, foi comparada com outras. Neste estudo de caso, foram consideradas as seguintes políticas: *Random*, *Lowest*, *Round-Robin* (Shivaratri *et al.*, 1992; Zhou & Ferrari, 1987), *Route* (Mello *et al.*, 2006a) e *RouteGA* (Mello *et al.*, 2007b). As três primeiras são mais conhecidas e estudadas na literatura: *Random* determina os computadores onde tarefas são escalonadas aleatoriamente, *Round-Robin* mantém uma lista circular que representa os computadores do ambiente, escolhendo-os de forma sucessiva, e *Lowest* aloca os processos em nós com menor carga.

As políticas *Route* e *RouteGA*, por sua vez, utilizam conhecimento sobre o comportamento de processos para distribuir as tarefas. A política *Route*, proposta por Mello *et al.* (2006a), utiliza conceitos de roteamento em redes de computadores para definir uma vizinhança e localizar recursos mais adequados para receber cargas de trabalho. Experimentos comprovam que essa política supera outras propostas na literatura quando avaliadas em ambientes heterogêneos de larga escala. Essa política é estendida em (Mello *et al.*, 2007b), que propõe uma nova versão da técnica, denominada *RouteGA*, a qual emprega um algoritmo genético para otimizar a alocação inicial de processos. Essa alocação reduz o tempo necessário para estabilizar a distribuição equitativa de cargas no sistema.

Nos experimentos realizados, a política *PredRoute* foi parametrizada da seguinte forma:

- Tamanho da população: 50 cromossomos⁸;

⁸Conforme demonstrado por Mello *et al.* (2007b), a adoção desse tamanho da população resulta em uma rápida convergência da política, ao mesmo tempo apresentando resultados satisfatórios

- Gerações: 100;
- Método de seleção: *Ranking*;
- Probabilidade de mutação: 1%;
- Probabilidade de recombinação (cruzamento): 70%;
- Janela histórica para predição: 100 pontos;
- Horizonte de predição: 100 pontos.

Durante a simulação, foi avaliado o tempo total necessário para executar processos. Os resultados dos experimentos são apresentados nas figuras 5.14, 5.15 e 5.16.

Em ambiente de *cluster* (figura 5.14), a política *PredRoute* apresentou resultados equivalentes às políticas *Route* e *RouteGA*. Esses resultados são bastante previsíveis, uma vez que as latências de comunicação entre os elementos de processamento e suas configurações são homogêneas. Nessa situação, a política proposta não gera ganhos significativos de desempenho.

Em ambientes de grade, entretanto, a política *PredRoute* apresentou o melhor desempenho em todos os casos, ultrapassando outras políticas em até 4 ordens de grandeza, e oferecendo tempo de execução até 10.000 vezes menor (figuras 5.15 e 5.16). Ao analisar os resultados, observou-se que o algoritmo foi capaz de determinar processos que freqüentemente comunicam-se, predizendo futuras operações. Com isso, eles foram alocados em elementos de processamento cuja latência é baixa, reduzindo a latência de eventos de comunicação. Dessa forma, o tempo de execução em ambiente de grade tornou-se compatível com o tempo observado em ambientes de *cluster*, como pode ser observado na tabela 5.10, onde todos os resultados obtidos foram sumarizados.

Na próxima etapa, estudou-se a estabilidade dos resultados de escalonamento apresentados por diversas políticas, avaliando a relação entre o tempo médio de execução e os desvios padrão obtidos. Como pode-se observar, a política *Random* apresentou piores resultados na maioria dos casos devido à imprevisibilidade de escalonamento aleatório. A política *Round Robin* também apresentou resultados inferiores às demais, contudo, a variação do tempo médio de execução foi menor em alguns casos devido à natureza da alocação de processos realizada pela política. As políticas *Lowest*, *Route* e *RouteGA* resultaram em tempo de execução médio compatível e apresentaram grande variação nos resultados (caracterizada por alto valor de desvio padrão), causada pela composição dinâmica do ambiente, o que influenciou na tomada de decisões pelos escalonadores. Uma vez que o tempo de execução foi altamente dependente da alocação correta dos processos sobre *EPs* adequados, variando conforme o aumento da latência de comunicação entre os elementos de processamento, a alocação de processos em *EPs* inaptos resultou em tempo de execução superior. Finalmente, observou-se que a política *PredRoute*

(a) *GTC Large - cluster-32*

(b) *GTC Medium - cluster-32*

(c) *PMEMD Large - cluster-32*

(d) *PMEMD Medium - cluster-32*

(e) *GTC Large - cluster-128*

(f) *GTC Medium - cluster-128*

(g) *PMEMD Large - cluster-128*

(h) *PMEMD Medium - cluster-128*

Figura 5.14: Tempo de execução médio: ambientes de *cluster*

Figura 5.15: Tempo de execução médio: ambientes de grade com 256 e 512 elementos

Figura 5.16: Tempo de execução médio: ambientes de grade com 1024 elementos

Tabela 5.10: Tempo médio de execução apresentado por diversas políticas de escalonamento (segundos)

<i>Política</i>	<i>GTC Medium</i>	<i>GTC Large</i>	<i>PMEMD Medium</i>	<i>PMEMD Large</i>
cluster-32				
<i>Random</i>	$3.4 \cdot 10^2 \pm 1.1 \cdot 10^4$	$8.4 \cdot 10^2 \pm 1.7 \cdot 10^4$	$9.6 \cdot 10^2 \pm 8.3 \cdot 10^3$	$3.5 \cdot 10^4 \pm 6.9 \cdot 10^4$
<i>RoundRobin</i>	$2.9 \cdot 10^2 \pm 1.0 \cdot 10^2$	$7.9 \cdot 10^2 \pm 1.4 \cdot 10^3$	$9.0 \cdot 10^2 \pm 2.0 \cdot 10^2$	$3.5 \cdot 10^4 \pm 1.1 \cdot 10^4$
<i>Lowest</i>	$2.2 \cdot 10^2 \pm 3.9 \cdot 10^1$	$5.6 \cdot 10^2 \pm 5.2 \cdot 10^2$	$8.5 \cdot 10^2 \pm 1.0 \cdot 10^2$	$3.4 \cdot 10^4 \pm 5.7 \cdot 10^3$
<i>Route</i>	$1.7 \cdot 10^2 \pm 1.5 \cdot 10^1$	$4.9 \cdot 10^2 \pm 2.9 \cdot 10^2$	$8.2 \cdot 10^2 \pm 3.6 \cdot 10^1$	$3.4 \cdot 10^4 \pm 3.0 \cdot 10^3$
<i>RouteGA</i>	$1.7 \cdot 10^2 \pm 1.4 \cdot 10^1$	$4.9 \cdot 10^2 \pm 2.9 \cdot 10^2$	$8.2 \cdot 10^2 \pm 3.6 \cdot 10^1$	$3.4 \cdot 10^4 \pm 2.7 \cdot 10^3$
<i>PredRoute</i>	$1.7 \cdot 10^2 \pm 1.3 \cdot 10^1$	$5.0 \cdot 10^2 \pm 3.0 \cdot 10^2$	$8.2 \cdot 10^2 \pm 1.0 \cdot 10^1$	$3.4 \cdot 10^4 \pm 4.1 \cdot 10^3$
cluster-128				
<i>Random</i>	$5.1 \cdot 10^2 \pm 4.5 \cdot 10^4$	$1.1 \cdot 10^3 \pm 1.2 \cdot 10^5$	$1.1 \cdot 10^3 \pm 6.8 \cdot 10^4$	$3.5 \cdot 10^4 \pm 2.7 \cdot 10^5$
<i>RoundRobin</i>	$5.1 \cdot 10^2 \pm 4.1 \cdot 10^4$	$1.0 \cdot 10^3 \pm 6.3 \cdot 10^2$	$1.0 \cdot 10^3 \pm 5.4 \cdot 10^4$	$3.5 \cdot 10^4 \pm 4.6 \cdot 10^3$
<i>Lowest</i>	$5.2 \cdot 10^2 \pm 3.0 \cdot 10^4$	$6.3 \cdot 10^2 \pm 1.6 \cdot 10^2$	$1.1 \cdot 10^3 \pm 2.6 \cdot 10^4$	$3.5 \cdot 10^4 \pm 8.9 \cdot 10^2$
<i>Route</i>	$1.6 \cdot 10^2 \pm 1.3 \cdot 10^1$	$4.2 \cdot 10^2 \pm 3.5 \cdot 10^2$	$8.1 \cdot 10^2 \pm 1.3 \cdot 10^1$	$3.5 \cdot 10^4 \pm 2.2 \cdot 10^2$
<i>RouteGA</i>	$1.4 \cdot 10^2 \pm 1.0 \cdot 10^1$	$4.0 \cdot 10^2 \pm 5.7 \cdot 10^1$	$8.1 \cdot 10^2 \pm 1.6 \cdot 10^1$	$3.5 \cdot 10^4 \pm 6.0 \cdot 10^2$
<i>PredRoute</i>	$1.6 \cdot 10^2 \pm 8.6 \cdot 10^0$	$4.3 \cdot 10^2 \pm 2.3 \cdot 10^2$	$8.0 \cdot 10^2 \pm 3.3 \cdot 10^1$	$3.5 \cdot 10^4 \pm 2.1 \cdot 10^2$
grid-256				
<i>Random</i>	$8.1 \cdot 10^6 \pm 1.5 \cdot 10^{13}$	$1.6 \cdot 10^7 \pm 1.9 \cdot 10^{13}$	$2.1 \cdot 10^7 \pm 9.4 \cdot 10^{12}$	$8.6 \cdot 10^8 \pm 3.3 \cdot 10^{14}$
<i>RoundRobin</i>	$9.0 \cdot 10^6 \pm 2.0 \cdot 10^{13}$	$1.3 \cdot 10^7 \pm 1.0 \cdot 10^2$	$2.1 \cdot 10^7 \pm 8.9 \cdot 10^{12}$	$8.6 \cdot 10^8 \pm 1.0 \cdot 10^3$
<i>Lowest</i>	$9.1 \cdot 10^6 \pm 2.5 \cdot 10^{13}$	$1.5 \cdot 10^7 \pm 5.9 \cdot 10^{12}$	$2.1 \cdot 10^7 \pm 6.9 \cdot 10^{12}$	$8.6 \cdot 10^8 \pm 7.1 \cdot 10^{13}$
<i>Route</i>	$3.2 \cdot 10^6 \pm 1.5 \cdot 10^{11}$	$8.0 \cdot 10^6 \pm 2.3 \cdot 10^{11}$	$1.9 \cdot 10^7 \pm 9.5 \cdot 10^{11}$	$8.5 \cdot 10^8 \pm 1.0 \cdot 10^{14}$
<i>RouteGA</i>	$2.7 \cdot 10^6 \pm 2.3 \cdot 10^{10}$	$7.8 \cdot 10^6 \pm 1.1 \cdot 10^{11}$	$1.6 \cdot 10^7 \pm 4.0 \cdot 10^{10}$	$8.5 \cdot 10^8 \pm 1.2 \cdot 10^{14}$
<i>PredRoute</i>	$2.9 \cdot 10^2 \pm 2.6 \cdot 10^2$	$1.0 \cdot 10^3 \pm 4.4 \cdot 10^3$	$9.1 \cdot 10^2 \pm 3.0 \cdot 10^2$	$3.3 \cdot 10^4 \pm 3.1 \cdot 10^4$
grid-512				
<i>Random</i>	$7.6 \cdot 10^6 \pm 1.5 \cdot 10^{13}$	$1.5 \cdot 10^7 \pm 2.6 \cdot 10^{13}$	$2.1 \cdot 10^7 \pm 5.5 \cdot 10^{12}$	$8.6 \cdot 10^8 \pm 9.1 \cdot 10^{13}$
<i>RoundRobin</i>	$1.0 \cdot 10^7 \pm 6.4 \cdot 10^{13}$	$1.2 \cdot 10^7 \pm 1.3 \cdot 10^9$	$2.1 \cdot 10^7 \pm 8.1 \cdot 10^{12}$	$8.7 \cdot 10^8 \pm 1.1 \cdot 10^{14}$
<i>Lowest</i>	$7.1 \cdot 10^6 \pm 8.5 \cdot 10^{12}$	$1.4 \cdot 10^7 \pm 6.5 \cdot 10^{12}$	$2.0 \cdot 10^7 \pm 3.8 \cdot 10^{12}$	$8.7 \cdot 10^8 \pm 6.1 \cdot 10^{13}$
<i>Route</i>	$3.1 \cdot 10^6 \pm 1.2 \cdot 10^{11}$	$7.9 \cdot 10^6 \pm 2.3 \cdot 10^{11}$	$1.9 \cdot 10^7 \pm 8.2 \cdot 10^{11}$	$8.7 \cdot 10^8 \pm 1.0 \cdot 10^{14}$
<i>RouteGA</i>	$2.6 \cdot 10^6 \pm 1.1 \cdot 10^{10}$	$8.1 \cdot 10^6 \pm 1.8 \cdot 10^{10}$	$1.7 \cdot 10^7 \pm 2.3 \cdot 10^{11}$	$8.5 \cdot 10^8 \pm 1.7 \cdot 10^{13}$
<i>PredRoute</i>	$4.2 \cdot 10^5 \pm 2.5 \cdot 10^8$	$5.2 \cdot 10^5 \pm 4.3 \cdot 10^{10}$	$4.7 \cdot 10^6 \pm 4.9 \cdot 10^{10}$	$7.8 \cdot 10^7 \pm 6.2 \cdot 10^{14}$
grid-1024				
<i>Random</i>	$5.3 \cdot 10^6 \pm 5.7 \cdot 10^{12}$	$6.6 \cdot 10^6 \pm 6.7 \cdot 10^{12}$	$1.2 \cdot 10^7 \pm 8.0 \cdot 10^{12}$	$4.7 \cdot 10^8 \pm 3.7 \cdot 10^{13}$
<i>RoundRobin</i>	$4.0 \cdot 10^6 \pm 5.4 \cdot 10^{12}$	$7.9 \cdot 10^6 \pm 7.9 \cdot 10^{12}$	$1.1 \cdot 10^7 \pm 3.6 \cdot 10^{12}$	$4.7 \cdot 10^8 \pm 3.4 \cdot 10^{13}$
<i>Lowest</i>	$5.4 \cdot 10^6 \pm 1.9 \cdot 10^{13}$	$6.9 \cdot 10^6 \pm 7.0 \cdot 10^{12}$	$1.1 \cdot 10^7 \pm 2.6 \cdot 10^{12}$	$4.7 \cdot 10^8 \pm 4.1 \cdot 10^{13}$
<i>Route</i>	$1.8 \cdot 10^6 \pm 9.8 \cdot 10^{10}$	$4.5 \cdot 10^6 \pm 1.1 \cdot 10^{11}$	$1.0 \cdot 10^7 \pm 4.4 \cdot 10^{11}$	$4.7 \cdot 10^8 \pm 3.9 \cdot 10^{13}$
<i>RouteGA</i>	$1.2 \cdot 10^6 \pm 1.6 \cdot 10^8$	$4.0 \cdot 10^6 \pm 3.5 \cdot 10^9$	$9.1 \cdot 10^6 \pm 6.8 \cdot 10^{10}$	$4.6 \cdot 10^8 \pm 6.4 \cdot 10^{12}$
<i>PredRoute</i>	$4.9 \cdot 10^2 \pm 1.0 \cdot 10^3$	$1.7 \cdot 10^3 \pm 2.0 \cdot 10^4$	$1.1 \cdot 10^3 \pm 1.9 \cdot 10^3$	$3.3 \cdot 10^4 \pm 9.8 \cdot 10^4$

apresentou resultados superiores em relação às demais políticas na maioria dos casos, devido ao emprego de reescalonamento periódico com base na predição de tendências comportamentais de processos. Com isso, evita-se a permanência de processos em elementos de processamento que deterioram o desempenho, migrando-os para outros *EPs* dinamicamente.

5.6.5 Avaliação da complexidade da abordagem proposta

Na próxima etapa, a abordagem proposta foi avaliada com o objetivo de determinar sua eficiência de predição *online* e contínua. Para isso, foram analisadas as diversas etapas que constituem a abordagem, estimando sua complexidade e o tempo de execução.

A complexidade da função de aptidão considerada pela política de escalonamento *PredRoute*, definida pela equação 5.15, depende das equações 5.17 e 5.16 que, por sua vez,

depende das equações 5.18 e 5.19 (seção 5.6.3). A equação 5.19 tem ordem de complexidade dada por $O(n^2)$. A equação 5.18 é composta por uma fração e pelo resultado da equação 5.19, e a sua complexidade é, portanto, dominada pela equação 5.19, sendo, conseqüentemente, igual a $O(n^2)$. A equação 5.17 deve percorrer todos os termos de C_j para encontrar o valor máximo, o que requer um laço de repetição externo ao cálculo de C_j , e requisita, conseqüentemente, uma ordem de complexidade $O(n^3)$. A equação 5.16 realiza um somatório sobre C_j o que resulta em ordem de complexidade $O(n^3)$, contudo, há outro somatório mais externo que acrescenta complexidade e faz com que a equação 5.16 seja da ordem de $O(n^4)$. A equação 5.15 é dominada pela equação 5.16, dessa forma, sua ordem de complexidade é de $O(n^4)$.

Essas equações são calculadas para cada indivíduo proposto pelo algoritmo genético adotado pela política *PredRoute*. Esse algoritmo propõe n gerações, onde para cada uma z indivíduos são gerados. Dessa maneira, a complexidade do algoritmo genético é de $n \cdot z \cdot O(n^4)$ o que tende, assintoticamente, a um algoritmo de complexidade polinomial da ordem de $O(n^6)$.

O tempo de execução da política de *PredRoute* é apresentado na figura 5.17, considerando o número de segundos consumidos para a execução de cada uma das políticas de escalonamento. Esses custos são reduzidos para ambientes com 32 computadores. *Clusters* com 128 computadores passam a apresentar maior custo, em torno de 21 segundos para a política *RouteGA* e 27 segundos para *Route* e *PredRoute*.

Em ambiente de grade com 256 computadores, os tempos consumidos pelas políticas *Route*, *RouteGA* e *PredRoute* ficam em torno de 2 segundos. Essa queda em relação a *clusters* com 128 máquinas deve-se ao fato que tais políticas consideram uma vizinhança de computadores, em função da latência de rede, na busca por soluções. Dessa forma, reduz-se, significativamente, o espaço de busca por alternativas de escalonamento. A mesma situação é observada em ambientes de grade com 1024 computadores, onde os tempos consumidos pelas políticas *Route*, *RouteGA* e *PredRoute* foram próximos a 7 segundos.

A política *Lowest* apresentou maior consumo de recursos para realizar escalonamentos em ambientes de grade, no entanto seu custo não é significativo em ambientes de *cluster*. O custo das políticas *Random* e *Round-Robin* é zero, pois nenhum parâmetro do ambiente é avaliado na tomada de decisões. Durante a realização desses experimentos, observou-se que, apesar de apresentar custo superior a algumas abordagens, tais como *Round-Robin* e *Random*, a política *PredRoute* reduz, em algumas ordens de grandeza, o tempo de execução de aplicações em ambientes heterogêneos de larga escala, tais como grades computacionais. Em ambientes de grade com 256 e 1024 computadores, a política *PredRoute* reduz, significativamente, o tempo de execução de processos. Isso evidencia que a predição de eventos de comunicação, e quaisquer outras operações executadas por aplicações, auxilia na tomada de decisões de escalonamento.

Figura 5.17: Custo de execução das políticas de escalonamento considerando conjunto de dados de *NERSC*

Custos Envolvidos na Abordagem de Predição

A abordagem de predição é composta pelas etapas de cálculo da dimensão de separação pela técnica de auto-informação mútua, cálculo da dimensão embutida pela técnica dos falsos vizinhos, reconstrução da série temporal segundo o teorema de imersão de Takens (1980), e modelagem da série reconstruída utilizando a rede neural *RRBF*.

Na primeira etapa, aplica-se a equação 4.3 (Cap. 4) para computar a auto-informação mútua. O cálculo de probabilidades dessa equação considera transições entre estados de uma cadeia de Markov (Kennel, 2002). Nesse caso, deve-se computar as probabilidades de transição de cada estado X para Y , o que apresenta ordem de complexidade igual a $O(n^2)$. Além disso, a equação calcula a integral desses valores o que aumenta a complexidade total da equação para $O(n^3)$.

A segunda etapa consiste no cálculo da dimensão embutida por meio da técnica de falsos vizinhos. Essa técnica encontra, primeiramente, os vizinhos mais próximos para cada ponto. Dessa forma, deve-se avaliar a distância de cada ponto p em relação aos demais, o que apresenta ordem de complexidade $O(n^2)$. Essas distâncias são calculadas para cada dimensão d , o que aumenta a complexidade total da técnica para $O(n^3)$.

Na terceira etapa, a série é reconstruída segundo o teorema de imersão de Takens (1980), que apresenta ordem de complexidade de $O(n^2)$, dado que para cada valor de entrada deve-se encontrar m outros segundo um deslocamento τ no tempo.

Em seguida, a série reconstruída é modelada pela rede neural *RRBF*, responsável por predições de eventos. Essa rede tem uma etapa de treinamento onde são definidos parâmetros para as funções $\phi(\cdot)$ de sua segunda camada. Esses parâmetros são encontrados após a execução da rede neural *SONDE*, que apresenta ordem de complexidade $O(n^2)$. Em seguida, define-se as funções $\phi(\cdot)$ e inicia-se o treinamento da segunda camada da rede neural *RRBF*, a qual calcula a equação 5.6 como forma de configurar pesos de conexões entre neurônios. Essa etapa realiza uma multiplicação de matrizes para cada termo modelado da série de entrada, o que resulta em complexidade $O(n^3)$.

Por meio dessa análise, observa-se que a complexidade total da etapa de predição é da ordem de $O(n^3)$.

Custos de Execução

A abordagem proposta foi avaliada com o objetivo de determinar os custos envolvidos na predição de eventos. Para isso, contabilizou-se os tempos necessários para determinar as dimensões de separação e embutida, realizar a classificação de dados utilizando a rede neural *SONDE*, treinar o modelo e predizer eventos. Eventos históricos foram enviados para a abordagem na forma de vetores no formato *{Bytes trafegados, ID do computador de*

Figura 5.18: Custo de execução das operações de pré-processamento e treinamento da abordagem proposta

destino}, em seguida, submeteu-se um número inteiro representando a quantidade de eventos a serem previstos. Após predição, dados resultantes foram retornados para o mecanismo de escalonamento.

O modelo de predição foi implementado utilizando a linguagem *Python 2.5* e as aplicações *false_nearest* e *mutual* do pacote de análise de séries não lineares *TISEAN 3.0.1* (Hegger & Kantz, 1999). Essas duas aplicações foram utilizadas para calcular, respectivamente, as dimensões embutida e de separação das séries em estudo. Os experimentos foram executados 30 vezes em um computador Intel Core2Duo T7250 com 3GB de memória RAM (embora menos de 10MB de RAM foram utilizados durante os experimentos) sobre o *kernel* do sistema operacional Linux versão 2.6.25. Essas repetidas execuções permitem estatísticas mais representativas (teorema do limite central) (Shefler, 1988).

A figura 5.18 apresenta os custos médios envolvidos nas etapas de cálculo das dimensões de separação e embutida, agrupamento pela rede neural *SONDE* (essas etapas são aqui denominadas pré-processamento) e treinamento do modelo de predição para um conjunto de dados com 5.000 pontos. Esse conjunto representa o tamanho total médio do *GTC* e do *PMEMD*. Quanto maior o tamanho desse conjunto, maior o custo para pré-processar e treinar o modelo.

Outros experimentos foram realizados para avaliar o custo envolvido na predição de operações de comunicação. Nessa situação, treinou-se o modelo com 100 pontos históricos, assim como nos experimentos que comparam a nova abordagem a outras existentes⁹, para predizer os próximos 100 a 1,000 pontos (adotando intervalos de 100 pontos). Os resultados para os conjuntos de dados *GTC* e *PMEMD* são apresentados na figura 5.19. Observou-se que a abordagem proposta apresenta custo máximo igual a 2 segundos para a predição de eventos,

⁹Esse fato comprova que 100 pontos históricos foram suficientes para que a abordagem proposta obtivesse bons resultados em relação a outras existentes.

Figura 5.19: Custo de execução das operações de predição da abordagem proposta

além de tendência linear.

Tanto os custos envolvimento no pré-processamento e treinamento do modelo, quanto aqueles referentes à predição de eventos, são baixos, principalmente, para ambientes de grade computacional, onde a melhor alocação de processos pode reduzir, em várias ordens de grandeza (tabela 5.10), o tempo de execução de processos. Conseqüentemente, esses custos não tornam proibitiva a adoção dessa abordagem em ambientes reais.

5.7 Considerações finais

Os estudos realizados no decorrer desta tese demonstraram que a predição do comportamento de processos auxilia a prover aspectos autônomos a ambientes distribuídos. Nesse contexto, a avaliação da dinâmica comportamental tem o fator fundamental na compreensão e predição do comportamento de processos, especialmente em casos onde esse não é detectado por técnicas convencionais.

Com base nisso, foi realizado um estudo de caso, propondo uma política de escalonamento de processos para sistemas distribuídos heterogêneos, que usa a predição do comportamento de processos como base para tomar as decisões de escalonamento. A política foi avaliada e comparada com outras propostas na literatura, em ambientes de *cluster* e grade computacional, por meio de *datasets* diferentes, baseados em aplicações reais. Os experimentos realizados apresentaram resultados promissores, sendo que a política proposta apresentou resultados satisfatórios em ambientes de *cluster*. Em ambientes de grade, por sua vez, a política proposta superou as demais avaliadas em algumas ordens de grandeza, ao mesmo tempo requisitando custo computacional suficientemente baixo, o que possibilita a sua adoção para escalonamento *online* em ambientes de grade.

Conclusões

A evolução de sistemas distribuídos resultou em aumento de complexidade de manutenção e gerenciamento, tornando pouco eficientes técnicas convencionais baseadas em intervenções manuais (Horn, 2001). Esse fato motivou pesquisas a fim de tornar tais ambientes auto-gerenciáveis, o que deu origem ao paradigma de computação autônoma (*Autonomic Computing*). Esse paradigma contempla aspectos de auto-identificação, auto-configuração, auto-recuperação, auto-otimização e auto-proteção do sistema (Parashar, 2007).

Motivado pelo potencial proveniente do paradigma de computação autônoma, esta tese teve como objetivo prover aspectos de autonomia a ambientes existentes, sem necessidade de alterar o paradigma de programação e aplicações dos usuários. Essa pesquisa foi desenvolvida no contexto do projeto *MidHPC (Middleware for High-Performance Computing)* (Andrade Filho *et al.*, 2008; Andrade Filho, 2008), que permite a execução transparente de aplicações legadas sobre grades, convertendo *threads* para processos distribuídos, e tratando aspectos de comunicação por meio de um mecanismo de memória compartilhada distribuída.

Nesse contexto, inicialmente, buscou-se prover aspectos de auto-identificação ao ambiente, extraindo, de forma transparente, o comportamento de processos, e convertendo-o para uma representação passível de análise. A evolução desses estudos demonstrou que a predição correta de eventos tem o potencial de aumentar, de forma significativa, o desempenho do sistema (Dodonov *et al.*, 2006; Mello *et al.*, 2009). O conhecimento sobre o comportamento, portanto, é essencial para compreender tendências de processos, o que motivou estudos detalhados sobre diversas técnicas de extração de comportamento.

Experimentos conduzidos no decorrer desses estudos demonstraram que tanto as abordagens de extração baseadas em monitoração quanto as de interceptação de eventos permitem realizar predição eficiente. Contudo, notou-se que as abordagens baseadas em monitoração

tendem a simplificar as tendências comportamentais, traçando uma média das observações, o que nem sempre representa o comportamento exato. Isso motivou a adoção de técnicas de extração baseadas em eventos, as quais permitem compreender e modelar, de maneira precisa, o comportamento de processos.

No decorrer dos estudos sobre o comportamento de processos, observou-se que o volume de informações extraído dificulta a realização de previsões. Isso motivou a adoção de técnicas de agrupamento, com o objetivo de reduzir a dimensionalidade de dados e determinar os padrões de execução relevantes. Além disso, observou-se que o comportamento de processos pode ser compreendido de forma mais adequada, aplicando conceitos de sistemas dinâmicos e ferramentas de teoria do caos, que permitiriam compreender dinâmicas internas do comportamento, e estimar acontecimentos futuros mesmo em casos onde abordagens comuns apresentam baixa eficiência.

Os estudos anteriormente mencionados motivaram a proposta de uma abordagem para extração, representação e previsão de comportamento de processos, que provê aspectos de auto-identificação ao ambiente (Dodonov & Mello, 2006, 2007).

Para validar a abordagem proposta nesta tese, foi elaborado um estudo de caso, que utiliza conhecimentos sobre o comportamento de processos, extraído na etapa de auto-identificação, para prover capacidade de auto-otimização ao ambiente. A partir disso, foi desenvolvida uma política de escalonamento distribuído, *PredRoute* (Dodonov & Mello, 2009b), a qual utiliza o conhecimento sobre dinâmicas comportamentais de processos para otimizar, transparentemente, a alocação de recursos. Essa política foi avaliada por meio de diversos experimentos, os quais demonstraram seu potencial de aumentar o desempenho em até 4 ordens de grandeza. Além disso, essa política tem um custo computacional suficientemente baixo, o que possibilita sua adoção para escalonamento *online* em ambientes de grade.

6.1 Contribuições do trabalho

A contribuição efetiva desta tese de Doutorado está na proposta de uma abordagem para modelagem *online* da dinâmica comportamental de processos, a qual pode ser empregada para compreender e estimar eventos futuros. Além disso, no contexto do projeto *MidHPC*, a elaboração desta tese gerou as seguintes contribuições indiretas:

1. Avaliação de diversas abordagens de memória compartilhada distribuída (*DSM*) e desenvolvimento de um *DSM* específico para o projeto *MidHPC* (Dodonov & Mello, 2006);
2. Desenvolvimento de duas abordagens de extração do comportamento de processos, sendo uma baseada em monitoramento contínuo e outra em interceptação de eventos (Dodonov *et al.*, 2006; Goularte *et al.*, 2007; Andrade Filho *et al.*, 2008; Mello *et al.*, 2009; De Mello *et al.*, 2009);

3. Avaliação de diversas técnicas de classificação e predição, as quais comprovaram a viabilidade de adoção de técnicas estocásticas e as baseadas em inteligência artificial para compreender e modelar o comportamento de processos distribuídos (Dodonov *et al.*, 2006; Dodonov & Mello, 2007, 2008; Mello *et al.*, 2009);
4. Proposta de uma abordagem para compreender e modelar comportamento de processos com base em ferramentas de teoria do caos e sistemas dinâmicos (Dodonov & Mello, 2009b);
5. Avaliação e comparação de diversas políticas de escalonamento, a qual resultou na proposta da política de escalonamento distribuído *PredRoute* (Dodonov & Mello, 2009a,b);

6.2 Publicações

O desenvolvimento desta tese de doutorado gerou as seguintes publicações:

Periódicos Internacionais

1. MELLO, R. F.; GOULARTE, R.; DODONOV, E.; YANG, L. T.; PARK, J. H., KIM, T.-H. *On modeling and evaluating multicomputer transcoding architectures for live-video streams*. Multimedia Tools and Applications, v. 43(2), 2009.
2. MELLO, R. F.; DODONOV, E.; BERTAGNA, R.; SENGHER, L. J. *Extracting and predicting the communication behaviour of parallel applications*. International Journal of Parallel, Emergent and Distributed Systems, v. 24(3), 2009.
3. DODONOV, E.; MELLO, R. F. *A Novel Approach for Distributed Application Scheduling Based on Prediction of Communication Events*. Future Generation Computing Systems, 2009. doi:10.1016/j.future.2009.05.004

Conferências Internacionais

1. DODONOV, E.; MELLO, R. F.; YANG, L. T. *A Network Evaluation for LAN, MAN and WAN Grid Environments*. In: The 2005 IFIP International Conference on Embedded And Ubiquitous Computing (EUC'2005), 2005, Nagasaki. The 2005 IFIP International Conference on Embedded And Ubiquitous Computing (EUC'2005), 2005. v. 1. p. 1133-1146.
2. DODONOV, E.; MELLO, R. F.; YANG, L. T. *Adaptive Technique for Automatic Communication Access Pattern Discovery Applied to Data Prefetching in Distributed Applications Using Neural Networks and Stochastic Models*. In: 4th International Symposium Parallel

- and Distributed Processing and Applications, 2006, Sorrento. Parallel and Distributed Processing and Applications. Berlin : Springer Berlin / Heidelberg, 2006. v. 4330. p. 292-303.
3. GOULARTE, R.; MELLO, R. F.; DODONOV, E.; YANG, L. T. *A Model to Estimate Transcoding Costs Applied in Live-Video Transmissions*. In: The 2007 International Conference on Intelligent Pervasive Computing, 2007, Jeju Island. p. 1-8.
 4. DODONOV, E.; MELLO, R. F. *A Model for Automatic online Process Behavior Extraction, Classification and Prediction in Heterogeneous Distributed Systems*. In: The First IEEE TCSC Doctoral Symposium, 2007, Rio de Janeiro. Proceedings on the 7th IEEE International Symposium on Cluster Computing and the Grid. Rio de Janeiro : IEEE Computer Society, 2007. v. 1. p. 1-7.
 5. MELLO, R. F.; ANDRADE FILHO, J. A.; DODONOV, E.; LI, K.-C.; YANG, L. T. *Supporting the Transparent Execution of High Performance Applications on Grids*. In: IEEE TENCON, 2007, Taipei. IEEE TENCON, 2007. p. 1-4.
 6. MELLO, R. F.; ANDRADE FILHO, J. A.; DODONOV, E.; ISHII, R. P.; YANG, L. T. *Optimizing distributed data access in Grid environments by using artificial intelligence techniques*. In: The Fifth International Symposium on Parallel and Distributed Processing and Applications, 2007, Niagara Falls. Proceedings on The Fifth International Symposium on Parallel and Distributed Processing and Applications, 2007. v. 1. p. 1-12.
 7. ANDRADE FILHO, J. A.; MELLO, R. F.; DODONOV, E.; SENGHER, L. J.; YANG, L. T.; LI, K.-C. *Toward an Efficient Middleware for Multithreaded Applications in Computational Grid*. In: IEEE 11th International Conference on Computational Science and Engineering, 2008, São Paulo. IEEE 11th International Conference on Computational Science and Engineering. San Jose, California : IEEE Computer Press, 2008. v. 1. p. 1-8. (*outstanding paper award*)

Capítulos de livros

1. DODONOV, E.; YANG, L. T.; MELLO, R. F. *On Application Behavior Extraction and Prediction to Support and Improve Process Scheduling Decisions* (Aceito para publicação). In: Kuan-Ching Li; Ching-Hsien Hsu; Laurence Tianruo Yang; Jack Dongarra; Hans Zima. (Org.). Handbook of Research on Scalable Computing Technologies. 1 ed. Hershey, EUA: IGI Global, 2008, v. 1, p. 1-20.

Relatórios técnicos

1. DODONOV, E.; MELLO, R.F. *Estudo sobre abordagens de extração, classificação e predição de comportamento de processos*. Technical Report, 100 p., ICMC/USP, São Carlos, Brazil, 2008.

6.3 Trabalhos futuros

Os resultados obtidos no decorrer do desenvolvimento da pesquisa motivam diversos trabalhos futuros. Nesses trabalhos, pretende-se aplicar as abordagens de compreensão e predição de padrões comportamentais de processos a outras áreas de computação autônoma, abordando aspectos de auto-configuração, auto-proteção e auto-recuperação. Além disso, pretende-se estudar e avaliar outras técnicas de predição, comparando-as com as abordadas neste trabalho, e realizar estudos para compreender tendências comportamentais de processos e evolução de suas dinâmicas internas ao longo da execução. Finalmente, visa-se realizar estudos adicionais sobre a previsibilidade de escalonamento em ambientes distribuídos de grande porte, e avaliar outras abordagens de escalonamento de processos.

Referências

- Aarts, E.; Korst, J. (1987). Boltzmann machines and their applications. *PARLE Parallel Architectures and Languages Europe*, v.1, p.34–50.
- Abarbanel, H. D. I.; Brown, R.; Sidorowich, J. J.; Tsimring, L. S. (1993). The analysis of observed chaotic data in physical systems. *Reviews of Modern Physics*, v.65, p.1331–1392.
- Abramson, D.; Buyya, R.; Giddy, J. (2002). A computational economy for grid computing and its implmentation in the nimrod-g resource broker. cite-seer.ist.psu.edu/abramson02computational.html.
- Adibi, J.; Shen, W. M. (2001). Self-similar layered hidden markov models. *PKDD '01: Proceedings of the 5th European Conference on Principles of Data Mining and Knowledge Discovery*, p. 1–15, London, UK. Springer-Verlag.
- Agarwal, M.; Bhat, V.; Liu, H.; Matossian, V.; Putty, V.; Schmidt, C.; Zhang, G.; Zhen, L.; Parashar, M. (2003). Automate: Enabling autonomic grid applications. *In the Autonomic Computing Workshop, 5th Annual International Active Middleware Services Workshop (AMS2003)*.
- Aguiar, C. S.; Cruz, D. I.; Ulson, R. S.; Cavenaghi, M. A. (2008). The application of distributed virtual machines for enterprise computer management: A two-tier network file system for image provisioning and management. *Computer Software and Applications Conference, Annual International*, v.0, p.428–431.
- Aha, D. W.; Kibler, D.; Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, v.6, p.37–66.
- Alahakoon, D.; Halgamuge, S. K.; Srinivasan, B. (1998). Self growing cluster development approach to data mining. *IEEE International Conference on Systems, Man, and Cybernetics*, v.3, p.2901–2906.
- Albertini, M. K.; Mello, R. F. (2007). A self-organizing neural network for detecting novelties. *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, p. 462–466, New York, NY, USA. ACM Press.

- Allcock, W.; Bester, J.; Bresnahan, J.; Chervenak, A.; Foster, I.; Kesselman, C.; Meder, S.; Nefedova, V.; Quesnel, D.; Tuecke, S. (2001). Data management and transfer in highperformance computational grid environments.
- Alligood, K. T.; Sauer, T. D.; Yorke, J. A. (1997). *Chaos: An Introduction to Dynamical Systems*. Springer.
- Andrade, N.; Cirne, W.; Brasileiro, F.; Roisenberg, P. (2003). OurGrid: An approach to easily assemble grids with equitable resource sharing. *Proceedings of the 9th Workshop on Job Scheduling Strategies for Parallel Processing*.
- Andrade Filho, J. A. (2008). Midhpc: Um suporte para a execução transparente de aplicações em grids computacionais. Dissertação (Mestrado), Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo.
- Andrade Filho, J. A.; Mello, R. F.; Dodonov, E.; Senger, L. J.; Yang, L. T.; Li, K. C. (2008). Toward an efficient middleware for multithreaded applications in computational grid. *Proceedings the 11th IEEE International Conference on Computational Science and Engineering (CSE-08)*, São Paulo, Brazil.
- Apple Computer, I. (2006). X-grid project. <http://www.apple.com/acg/xgrid/>.
- Arnautovic, E.; Kaindl, H.; Falb, J.; Popp, R.; Szép, A. (2007). Gradual transition towards autonomic software systems based on high-level communication specification. *SAC '07: Proceedings of the 2007 ACM symposium on Applied computing*, p. 84–89, New York, NY, USA. ACM.
- Arpaci-Dusseau, A. C.; Culler, D. E.; Mainwaring, M. (1998). Scheduling with Implicit Information in Distributed Systems. *Proceedings of ACM SIGMETRICS'98*, p. 233–248.
- Aumage, O.; Mercier, G. (2003). Mpi-ch/madiii: a cluster of clusters enabled mpi implementation. *Proceedings of 3rd International Symposium on Cluster Computing and the Grid*, p. 26–33.
- Back, T.; Fogel, D. B.; Michalewicz, Z., editores (1999). *Basic Algorithms and Operators*. IOP Publishing Ltd., Bristol, Reino Unido.
- Bailey, D. H.; Barszcz, E.; Barton, J. T.; Browning, D. S.; Carter, R. L.; Dagum, D.; Fatoohi, R. A.; Frederickson, P. O.; Lasinski, T. A.; Schreiber, R. S.; Simon, H. D.; Venkatakrisnan, V.; Weeratunga, S. K. (1991). The nas parallel benchmarks. *The International Journal of Supercomputer Applications*, v.5, n.3, p.63–73.
- Bailey, D. H.; Barzcz, E.; Dagum, L.; Simon, H. D. (1993). Nas parallel benchmark results. *IEEE Parallel Distrib. Technol.*, v.1, n.1, p.43–51.

- Baker, M. (2000). Cluster computing white paper. <http://www.dcs.port.ac.uk/~mab/tfcc/WhitePaper/WhitePaper.htm>.
- Balakrishnan, S.; Rajwar, R.; Upton, M.; Lai, K. K. (2005). The impact of performance asymmetry in emerging multicore architectures. *ISCA*, p. 506–517. IEEE Computer Society.
- Balram, N.; Belo, C.; Moura, J. M. F. (1988). Parallel processing on supercomputers: a set of computational experiments. *International Supercomputer Applications*, v. 1, p. 247–257.
- Baluja, S.; Fahlman, S. (1994). Reducing network depth in the cascade-correlation learning architecture. Relatório Técnico CMU-CS-94-209, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA.
- Barak, A.; La’adan, O. (1998). The MOSIX multicomputer operating system for high performance cluster computing. *Future Generation Computer Systems*, v.13, n.4–5, p.361–372.
- Barreto, G. A.; Araujo, A. F. R. (2004). Identification and control of dynamical systems using the self-organizing map. *Special issue of IEEE TNN on temporal coding*, p. 1244–1259.
- Barrett, B.; Squyres, J.; Lumsdaine, A. (2003). Integration of the LAM/MPI environment and the PBS scheduling system. *Proceedings, 17th Annual International Symposium on High Performance Computing Systems and Applications*, Quebec, Canada.
- Barve, R.; Kallahalla, M.; Varman, P. J.; Vitter, J. S. (1997). Competitive parallel disk prefetching and buffer management. *Proceedings of the Fifth Workshop on Input/Output in Parallel and Distributed Systems*, p. 47–56, San Jose, CA. ACM Press.
- Bauer, M. (2006). Paranoid penguin: an introduction to novell apparmor. *Linux J.*, v.2006, n.148, p.13.
- Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, www.stat.ucla.edu/history/essay.pdf.
- Berthold, M. R. (1994). A time delay radial basis function network for phoneme recognition. *Neural Networks, 1994. IEEE World Congress on Computational Intelligence., 1994 IEEE International Conference on*, v.7.
- Bianchini, R.; Kontothanassis, L. I.; Pinto, R.; De Maria, M.; Abud, M.; Amorim, C. L. (1996). Hiding communication latency and coherence overhead in software DSMs. *Proc. of the 7th Symp. on Architectural Support for Programming Languages and Operating Systems (ASPLOS VII)*, p. 198–209.
- Billings, S. A.; Zheng, G. L. (1995). Radial basis function network configuration using genetic algorithms. *Neural Networks.*, v.8, n.6, p.877–890.

- Blum, C.; Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, v.35, n.3, p.268–308.
- Bolch, G.; Greiner, S.; de Meer, H.; Trivedi, K. S. (1998). *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. Wiley-Interscience, New York, NY, USA.
- Bonabeau, E.; Dorigo, M.; Theraulaz, G. (2000). Inspiration for optimization from social insect behaviour. *Nature*, v.406, n.6791, p.39–42.
- Bors, A. G.; Pitas, I. (1996). Median radial basis function neural network. *IEEE Transactions on Neural Networks*, v.7, n.6, p.1351–64.
- Bouckaert, R. (1994). Properties of measures for bayesian belief network learning. *10th Conf. Uncertainty in Artificial Intelligence, Seattle, WA, 1994*.
- Box, G. E. P.; Jenkins, G. (1990). *Time Series Analysis, Forecasting and Control*. Holden-Day, Incorporated.
- Bradley, P. S.; Fayyad, U. M.; Reina, C. (1998). Scaling clustering algorithms to large databases. *Knowledge Discovery and Data Mining*, p. 9–15.
- Brecht, T.; Guha, K. (1996). Using Parallel Program Characteristics in Dynamic Processor Allocation Policies. *Performance Evaluation*, v.27/28, n.4, p.519–539.
- Burns, G.; Daoud, R.; Vaigl, J. (1994). LAM: An Open Cluster Environment for MPI. *Proceedings of Supercomputing Symposium*, p. 379–386.
- Burzevski, V.; Mohan, C. (1996). Hierarchical growing cell structures. *ICNN96: Proc. Int'l. Conf. Neural Networks*.
- Buyya, R. (1999). *High Performance Cluster Computing: Architectures and Systems*. Prentice Hall PTR, Upper Saddle River, NJ, USA.
- Byna, S.; Sun, X.-H.; Gropp, W.; Thakur, R. (2004). Predicting memory-access cost based on data-access patterns. *CLUSTER '04: Proceedings of the 2004 IEEE International Conference on Cluster Computing*, p. 327–336, Washington, DC, USA. IEEE Computer Society.
- C. Gruber, M. Coduro, B. S. (2006). Signature verification with dynamic rbf networks and time series motifs. *Proceedings of Tenth International Workshop on Frontiers in Handwriting Recognition*.
- Cao, P.; Felten, E. W.; Karlin, A. R.; Li, K. (1995). A study of integrated prefetching and caching strategies. *Proceedings of the 1995 ACM SIGMETRICS jointinternational conference on Measurement and modeling ofcomputer systems*, p. 188–197. ACM Press.

- Caparrós, G. J.; Ruiz, M. A. A.; Hernández, F. S. (1997). Hopfield neural network applied to optimization problems: Some theoretical and simulation results. *IWANN '97: Proceedings of the International Work-Conference on Artificial and Natural Neural Networks*, p. 556–565, London, UK. Springer-Verlag.
- Cappello, F.; Etiemble, D. (2000). MPI versus MPI+OpenMP on the IBM SP for the NAS Benchmarks. p. 51–51.
- Carpenter, G. A. (1989). Neural network models for pattern recognition and associative memory. *Neural Networks*, v.2, p.243–257.
- Carpenter, G. A.; Grossberg, S. (1987). Art 2: Selforganisation of stable category recognition codes for analog input patterns. *Applied Optics*, v.26, p.4919–4930.
- Carpenter, G. A.; Grossberg, S. (1990). Art3: Hierarchical search using chemical transmitters in self-organizing pattern recognition architectures. *Neural Networks*, v.3, p.129–152.
- Carpenter, G. A.; Grossberg, S.; Markuzon, N.; Reynolds, J.; Rosen, D. (1992). Fuzzy artmap: A neural network architecture for incremental supervised learning of analog multidimensional maps. *IEEE Transactions on Neural Networks*, v.3, n.5, p.698–713.
- Carpenter, G. A.; Grossberg, S.; Reynolds, J. (1991a). Artmap: Supervised real-time learning and classification of nonstationary data by a self-organizing neural network. *Neural Networks*, v.4, p.565–588.
- Carpenter, G. A.; Grossberg, S.; Rosen, D. (1991b). Fuzzy art: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural Networks*, v.4, p.759–771.
- Champrasert, P.; Farms, S. (2006). Symbioticsphere: A biologically-inspired autonomic architecture for self-adaptive and self-healing server farms. *WOWMOM '06: Proceedings of the 2006 International Symposium on on World of Wireless, Mobile and Multimedia Networks*, p. 469–474, Washington, DC, USA. IEEE Computer Society.
- Chan, A.; Gropp, W.; Lusk, E. (1998). User's guide for mpe extensions for mpi programs. Technical Report ANL-98/xx, Argonne National Laboratory. <ftp://ftp.mcs.anl.gov/pub/mpi/mpeman.ps>.
- Chappell, G. J.; Taylor, J. G. (1993). The temporal kohonen map. *Neural Networks Journal*, v.6, n.3, p.441–445.
- Chen, B.-J.; Chang, M.-W.; Lin, C.-J. (Nov. 2004). Load forecasting using support vector machines: a study on eunite competition 2001. *Power Systems, IEEE Transactions on*, v.19, n.4, p.1821–1830.

- Cheng, G.; Zell, A. (2000). Externally growing cell structures for pattern classification. Bothe, H.; Rojas, R., editores, *Proceeding of the ICSC Symposia on Neural Computation (NC'2000) May 23-26, 2000 in Berlin, Germany*. ICSC Academic Press.
- Cheng, J.; Greiner, R. (2001). Learning Bayesian belief network classifiers: Algorithms and system. *Lecture Notes in Computer Science*, v.2056, p.141.
- Chiu, B.; Keogh, E.; Lonardi, S. (2003). Probabilistic discovery of time series motifs. *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, p. 493–498, New York, NY, USA. ACM.
- Chung, P. (1997). Checkpointing in cosmic: a user-level process migration environment.
- Clark, J. D. (1990). Modeling and simulating complex spatial dynamic systems: a framework for application in environmental analysis. *SIGSIM Simul. Dig.*, v.21, n.2, p.9–19.
- Codish, M.; Taboch, C. (1997). A semantic basis for termination analysis of logic programs and its realization using symbolic norm constraints. *ALP/HOA*, p. 31–45.
- Cohn, S.; Sivakumaran, N.; Todling, R. (1994). A fixed-lag kalman smoother for retrospective data assimilation. *Mon. Wea. Rev.*, 122, p. 2838–2867.
- Cooper, E. C. (1987). Distributed systems technologies survey. <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.56.2600>.
- Corbalan, J.; Martorell, X.; Labarta, J. (2001). Improving Gang Scheduling through Job Performance Analysis and Malleability. *International Conference on Supercomputing*, p. 303–311, Sorrento, Italy.
- Cortes, T.; Girona, S.; Labarta, J. (1996). PACA: A cooperative file system cache for parallel machines. Relatório Técnico 96-07, UPC-CEPBA.
- Cortes, T.; Girona, S.; Labarta, J. (1997a). Avoiding the cache-coherence problem in a parallel/distributed file system. *Proceedings of High-Performance Computing and Networking*, p. 860–869.
- Cortes, T.; Girona, S.; Labarta, J. (1997b). Design issues of a cooperative cache with no coherence problems. *Proceedings of the Fifth Workshop on Input/Output in Parallel and Distributed Systems*, p. 37–46, San Jose, CA. ACM Press.
- Cortes, T.; Labarta, J. (1999). Linear aggressive prefetching: A way to increase the performance of cooperative caches. *Proceedings of the Joint International Parallel Processing Symposium and IEEE Symposium on Parallel and Distributed Processing*, p. 45–54, San Juan, Puerto Rico.

- Costa, P.; Guardia, H. C.; Zorzo, S. (2003). Progrid: A proxy-based architecture for grid operation and management. *Proceedings of SBAC-PAD 2003*.
- Coughlin, J. P.; Baran, R. H. (1991). Optimization in cascaded boltzmann machines with a temperature gradient: an alternative to simulated annealing. *ANNA '91: Proceedings of the conference on Analysis of neural network applications*, p. 176–184, New York, NY, USA. ACM.
- Creeger, M. (2005). Curmudgeon: Multicore CPUs for the masses. *ACM Queue*, v.3, n.7, p.64, 63.
- Damminda, A.; Saman, H. K.; Bala, S. (2000). Dynamic self-organizing maps with controlled growth for knowledge discovery. *IEEE Transactions on Neural Networks*, v.11, n.3, p.601–614.
- Day, S. P.; Davenport, M. R. (1993). Continuous-time temporal back-propagation with adaptable time delays. *IEEE Transactions on Neural Networks*, v.4, n.2, p.348–354.
- De Mello, R. F.; Goularte, R.; Dodonov, E.; Yang, L. T.; Park, J. H.; Kim, T.-H. (2009). On modeling and evaluating multicomputer transcoding architectures for live-video streams. *Multimedia Tools Appl.*, v.43, n.2, p.109–129.
- del Rosario, J. M.; Choudhary, A. (1994). High performance I/O for parallel computers: Problems and prospects. *IEEE Computer*, v.27, n.3, p.59–68.
- Deris, M. M.; Abawajy, J. H.; Suzuri, H. M. (2004). An efficient replicated data access approach for large-scale distributed systems. *CCGRID '04: Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid*, p. 588–594, Washington, DC, USA. IEEE Computer Society.
- Devarakonda, M. V.; Iyer, R. K. (1989). Predictability of Process Resource Usage: A Measurement-based Study on UNIX. *IEEE Transactions on Software Engineering*, v.15, n.12, p.1579–1586.
- Ding, M.; Zhou, Q.; Wang, Q. (2006). The application of self-adaptive kalman filter in ngimu/gps integrated navigation system. *ISDA '06: Proceedings of the Sixth International Conference on Intelligent Systems Design and Applications (ISDA'06)*, p. 61–65, Washington, DC, USA. IEEE Computer Society.
- Distributed.net (2006). The distributed.net project. <http://www.distributed.net>.
- Doblinger, G. (1998). An adaptive kalman filter for the enhancement of noisy ar signals.
- Dodonov, E. (2004). Um mecanismo integrado de *Cache e Prefetching* para sistemas de entrada e saída de alto desempenho. Dissertação (Mestrado), DC/UFSCar.

- Dodonov, E.; Guardia, H. C. (2002). An architecture for integrated caching and prefetching mechanisms for distributed parallel file systems. *Proceedings of the 2002 CLEI*.
- Dodonov, E.; Mello, R. (2009a). On application behavior extraction and prediction to support and improve process scheduling decisions. *Handbook of Research on Scalable Computing Technologies*, p. 1–20. IGI Global.
- Dodonov, E.; Mello, R.; Yang, L. T. (2006). Adaptive technique for automatic communication access pattern discovery applied to data prefetching in distributed applications using neural networks and stochastic models. *Proceedings of the ISPA-06*.
- Dodonov, E.; Mello, R. F. (2006). Modelo para avaliação e predição do comportamento de processos aplicado a sistemas distribuídos heterogêneos. Qualificação de Doutorado.
- Dodonov, E.; Mello, R. F. (2007). A model for automatic on-line process behavior extraction, classification and prediction in heterogeneous distributed systems. *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, p. 899–904, Washington, DC, USA. IEEE Computer Society.
- Dodonov, E.; Mello, R. F. (2008). Estudo sobre abordagens de extração, classificação e predição de comportamento de processos. Relatório Técnico 322, Instituto de Ciências Matemáticas e de Computação, USP, São Carlos, SP, Brazil.
- Dodonov, E.; Mello, R. F. (2009b). A novel approach for distributed application scheduling based on prediction of communication events. *Future Generation Computer Systems*, v.1.
- Dodonov, E.; Mello, R. F.; Yang, L. T. (2005). A network evaluation for lan, man and wan grid environments. Yang, L. T.; Amamiya, M.; Liu, Z.; Guo, M.; Rammig, F. J., editores, *EUC*, v. 3824 de *Lecture Notes in Computer Science*, p. 1133–1146. Springer.
- Dodonov, E.; Sousa, J. Q.; Guardia, H. C. (2004). Gridbox: securing hosts from malicious and greedy applications. *Proceedings of the 2nd workshop on Middleware for grid computing*, p. 17–22, New York, NY, USA. ACM Press.
- Dokur, Z.; Ölmez, T. (2002). Segmentation of ultrasound images by using a hybrid neural network. *Pattern Recogn. Lett.*, v.23, n.14, p.1825–1836.
- Dorigo, M.; Di Caro, G. (1999). The ant colony optimization meta-heuristic. Corne, D.; Dorigo, M.; Glover, F., editores, *New Ideas in Optimization*, p. 11–32. McGraw-Hill, Londres.
- Drosinos, N.; Koziris, N. (2006). The Effect of Process Topology and Load Balancing on Parallel Programming Models for SMP Clusters and Iterative Algorithms. *Journal of Supercomputing*, v.35, n.1, p.65–91.

- Déharbe, D. (2003). A tutorial introduction to symbolic model checking. *Logic for concurrency and synchronisation*, v.1, p.215–237.
- Eckmann; Ruelle, D. (1985). Ergodic theory of chaos and strange attractors. *Reviews of Modern Physics*, v.57, n.3, p.617+.
- Edmonds, A. N. (1996). *Time Series Prediction Using Supervised Learning and Tools from Chaos Theory*. Tese (Doutorado), University of Luton.
- Elmer, F.-J. (1998). The lyapunov exponent. <http://monet.unibas.ch/~elmer/pendulum/lyapexp.htm>.
- Erñanský, M.; Makula, M.; Beňušková, U. (2007). Organization of the state space of a simple recurrent network before and after training on recursive linguistic structures. *Neural Networks*, v.20, n.2, p.236–244.
- Evensen, G. (2000). An ensemble kalman smoother for nonlinear dynamics. *Mon. Weather Rev.*, 128, p. 1852–1867.
- Faassen, J. (1997). Using full parallel Boltzmann Machines for Optimization. Relatório Técnico AIB-02-1997, RWTH Aachen.
- Fahlman, S. E.; Lebiere, C. (1990). The cascade-correlation learning architecture. Touretzky, D. S., editor, *Advances in Neural Information Processing Systems*, v. 2, p. 524–532, Denver 1989. Morgan Kaufmann, San Mateo.
- Fang, W.; Wang, C. L.; Zhu, W.; Lau, F. C. M. (2004). Pat: a postmortem object access pattern analysis and visualization tool. *CCGRID 2004, 4th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2004), April 19-22, 2004, Chicago, Illinois, USA*, p. 379–386. IEEE Computer Society.
- Farnstrom, F.; Lewis, J. (2008). Fast, single-pass k-means algorithms. <http://citeseer.ist.psu.edu/327212.html>.
- Fayyad, U. M.; Reina, C.; Bradley, P. S. (1998). Initialization of iterative refinement clustering algorithms. *Knowledge Discovery and Data Mining*, p. 194–198.
- Feamster, N.; Rexford, J. (2007). Network-wide prediction of bgp routes. *IEEE/ACM Trans. Netw.*, v.15, n.2, p.253–266.
- Feitelson, D. G.; Rudolph, L.; Schwiegelshohn, U.; Sevcik, K. C.; Wong, P. (1997). Theory and Practice in Parallel Job Scheduling. *Job Scheduling Strategies for Parallel Processing*, v. 1291, p. 1–34. Springer Verlag. Lect. Notes Comput. Sci. vol. 1291.
- Ferreira, E. A.; Mello, R. F. (2004). Avaliação de desempenho de protocolos e bibliotecas de comunicação entre processos. *Proceedings of 3rd International Information and Telecommunication Technologies Symposium*, p. 1–8, Sao Carlos, Sao Paulo, Brazil.

- Fet, Y. I. (1995). Vertical processing systems: A survey. *IEEE Micro*, v.15, n.1, p.65–75.
- Fine, S.; Singer, Y.; Tishby, N. (1998). The hierarchical hidden markov model: Analysis and applications. *Machine Learning*, v.32, n.1, p.41–62.
- Fischer, P. C. (1965). On formalisms for turing machines. *J. ACM*, v.12, n.4, p.570–580.
- Fitzgerald, S.; Foster, I.; Kesselman, C.; von Laszewski, G.; Smith, W.; Tuecke, S. (1997). A directory service for configuring high-performance distributed computations. *Proc. 6th IEEE Symp. on High Performance Distributed Computing*, p. 365–375. IEEE Computer Society Press.
- Flynn, M. J. (1999). Basic issues in microprocessor architecture. *Journal of Systems Architecture*, v.45, p.12–13.
- Fortz, B. (2000). Internet traffic engineering by optimizing ospf weights. *In Proceedings of IEEE INFOCOM 2000, Tel Aviv*.
- Foster, I.; Kesselman, C. (1997). Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, v.11, n.2, p.115–128.
- Foster, I.; Kesselman, C.; Tuecke, S. (2001). The anatomy of the Grid: Enabling scalable virtual organizations. *Lecture Notes in Computer Science*, v.2150, p.1.
- Fraser, A. M.; Swinney, H. L. (1986). Independent coordinates for strange attractors from mutual information. *Phys. Rev. A*, v.33, n.2, p.1134–1140.
- Freeman, J. A.; Skapura, D. M. (1991). *Neural Networks: Algorithms, Applications and Programming Techniques*. Addison-Wesley.
- Fritzke, B. (1995). A growing neural gas network learns topologies. Tesauro, G.; Touretzky, D. S.; Leen, T. K., editores, *Advances in Neural Information Processing Systems 7*, p. 625–632. MIT Press, Cambridge MA.
- Ganger, G. R.; Worthington, B. L.; Hou, R. Y.; Patt, Y. N. (1993). Disk subsystem load balancing: Disk striping vs. conventional data placement. *Proceedings of the Twenty-Sixth Annual Hawaii International Conference on System Sciences*, v. I, p. 40–49.
- Gao, H.; Sollacher, R.; Kriegel, H. P. (2007). Spiral recurrent neural network for online learning. *Proceedings of 15th European Symposium on Artificial Neural Networks*, p. 483–488.
- Garey, M. R.; Johnson, D. S. (1979). *Computers and Intractability : A Guide to the Theory of NP-Completeness*. Series of Books in the Mathematical Sciences. W. H. Freeman.

- Gers, A. F.; Schmidhuber, J. (2000). Long short-term memory learns context free and context sensitive languages. *IDSIA-03-00*.
- Gers, F.; Gers, F.; Schmidhuber, J. (2000). Recurrent nets that time and count. Schmidhuber, J., editor, *Proc. IEEE-INNS-ENNS International Joint Conference on Neural Networks IJCNN 2000*, v. 3, p. 189–194 vol.3.
- Gers, F.; Schraudolph, N.; Schmidhuber, J. (2002). Learning precise timing with lstm recurrent networks. *Journal of Machine Learning Research*, v. 3, p. 115–143.
- Gibbons, R. (1997). A Historical Application Profiler for Use by Parallel Schedulers. *Job Scheduling Strategies for Parallel Processing*, p. 58–77. Springer Verlag.
- Giunchiglia, F.; Traverso, P. (1999). Planning as model checking. *ECP*, p. 1–20.
- Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, EUA.
- Goldchleger, A.; Kon, F.; Goldman, A.; Finger, M.; Bezerra, G. C. (2004). InteGrade: Object-Oriented Grid Middleware Leveraging Idle Computing Power of Desktop Machines. *Concurrency and Computation: Practice and Experience*, v.16, n.5, p.449–459.
- Goularte, R.; Mello, R. F.; Dodonov, E.; Yang, L. T. (2007). A model to estimate transcoding costs applied in live-video transmissions. *IPC*, v.0, p.341–348.
- Guo, B.; Wang, D. H.; Shen, Y.; Li, Z. S. (2007). A hopfield neural network approach for power optimization of real-time operating systems. *Neural Comput. Appl.*, v.17, n.1, p.11–17.
- Guo, H.; Hsu, W. (2002). A survey on algorithms for real-time bayesian network inference. *joint AAAI-02/KDD-02/UAI-02 workshop on Real-Time Decision Support and Diagnosis Systems*.
- Hamker, F.; Heinke, D. (1997). Implementation and comparison of growing neural gas. *Technical Report 1/97, Technical University of Ilmenau*.
- Hammerton, J. (2003). Named entity recognition with long short-term memory. *Proceedings of the seventh conference on Natural language learning at HLT-NAACL 2003*, p. 172–175, Morristown, NJ, USA. Association for Computational Linguistics.
- Harchol-Balter, M.; Downey, A. B. (1997). Exploiting Process Lifetimes Distributions for Dynamic Load Balancing. *ACM Transactions on Computer Systems*, v.15, n.3, p.253–285.
- He, J.; Tan, A.-H.; Tan, C.-L. (2004). Modified art 2a growing network capable of generating a fixed number of nodes. *IEEE Transactions on Neural Networks*, v.15, n.3, p.728–737.
- Hegger, R.; Kantz, H. (1999). Practical implementation of nonlinear time series methods: the TISEAN package. *Chaos*, v.9, n.2, p.413–435.

- Hemker, P. W. (1984). Multigrid algorithms run on supercomputers. *International Supercomputer Applications*, v.4, n.???, p.44–51.
- Hennessy, J. L.; Patterson, D. A. (2002). *Computer Architecture: A Quantitative Approach (The Morgan Kaufmann Series in Computer Architecture and Design)*. Morgan Kaufmann.
- Henzinger, T.; Jhala, R.; Majumdar, R.; Sutre, G. (2003). Software verification with blast. *Proceedings of the Tenth International Workshop on Model Checking of Software (SPIN)*, p. 235–239. Lecture Notes in Computer Science 2648, Springer-Verlag.
- Heyer, L. J.; Kruglyak, S.; Yooseph, S. (1999). Exploring expression data: Identification and analysis of coexpressed genes. *Genome Res.*, v.9, n.11, p.1106–1115.
- Hinterding, R. (2000). Representation, mutation and crossover issues in evolutionary computation. *Proc. of the 2000 Congress on Evolutionary Computation*, p. 916–923, Piscataway, NJ. IEEE Service Center.
- Hirose, S.; Shimizu, K.; Kanai, S.; Kuroda, Y.; Noguchi, T. (2007). Poodle-l: a two-level svm prediction system for reliably predicting long disordered regions. *Bioinformatics*, v.23, n.16, p.2046–2053.
- Hochreiter, S.; Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, v.9, n.8, p.1735–1780.
- Hopfield, J. J. (1988). Neural networks and physical systems with emergent collective computational abilities. *Neurocomputing: foundations of research*, v.1, p.457–464.
- Horn, P. (2001). Autonomic computing: Ibm’s perspective on the state of information technology.
- Hu, X.; Prokhorov, D. V.; Wunsch II, D. C. (2007). Time series prediction with a weighted bidirectional multi-stream extended kalman filter. *Neurocomput.*, v.70, n.13-15, p.2392–2399.
- Huang, Z.; Purvis, M.; Werstein, P. (2005). View oriented update protocol with integrated diff for view-based consistency. *The 2005 International Workshop on DSM (DSM2005)*, Cardiff, May 2005. IEEE Computer Society.
- Huband, S.; McDonald, C. (2001). A Preliminary Topological Debugger for MPI Programs. *Proc. of the First IEEE/ACM International Symposium on Cluster Computing and the Grid*, v.1, p.422–429.
- Hwang, K. (2008). Massively distributed systems : From grids and p2p to clouds. *Advances in Grid and Pervasive Computing*, p. 1. Springer Berlin / Heidelberg.

- IBM (2001). Autonomic computing manifesto: Ibm's perspective on the state of information technology. <http://www.research.ibm.com/autonomic/manifesto>.
- Ishii, R. P.; Mello, R. F.; Yang, L. T. (2007). A complex network-based approach for job scheduling in grid environments. Perrott, R. H.; Chapman, B. M.; Subhlok, J.; de Mello, R. F.; Yang, L. T., editores, *HPCC*, v. 4782 de *Lecture Notes in Computer Science*, p. 204–215. Springer.
- Jacob, E.; Sasikumar, R.; Nair, K. N. R. (2005). A fuzzy guided genetic algorithm for operon prediction. *Bioinformatics*, v.21, n.8, p.1403–1407.
- Jaditz, T. (1995). Time series prediction: Forecasting the future and understanding the past : Andreas s. weigend and neil a. gershenfeld, eds., (reading, ma: Addison-we. *Journal of Economic Behavior & Organization*, v.26, n.2, p.302–305. available at <http://ideas.repec.org/a/eee/jeborg/v26y1995i2p302-305.html>.
- Jaeger, H. (2007). Echo state network. *Scholarpedia*, v.2, n.9, p.2330.
- Jain, R. (1991). *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. John Wiley and Sons.
- Jiang, D.; Tang, C.; Zhang, A. (2004). Cluster analysis for gene expression data: A survey. *IEEE Transactions on Knowledge and Data Engineering*, v.16, n.11, p.1370–1386.
- Johnson, S. (1977). Lint, a c program checker. Relatório técnico, Bell Laboratories, Computer Science Techinal Report 65.
- Jones, C. L.; Lonergan, G. T.; Mainwaring, D. E. (1996). Wavelet packet computation of the hurst exponent. *Journal of Physics A: Mathematical and General*, v.29.
- Kallahalla, M.; Varman, P. J. (2001). Optimal prefetching and caching for parallel I/O systems. *Proceedings of the Thirteenth Symposium on Parallel Algorithms and Architectures*, p. 219–228. ACM Press.
- Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, v.82, n.Series D, p.35–45.
- Kantz, H. (1994). A robust method to estimate the maximal lyapunov exponent of a time series. *Physics Letters A*, v.185, p.77–87.
- Kaplan, I. (2003). Estimating the hurst expoent. http://www.bearcave.com/misl/misl_tech/wavelets/hurst/.
- Kaski, S.; Oja, E. (1999). *Kohonen Maps*. Elsevier Science Inc., New York, NY, USA.

- Kennel, M. (2002). The multiple-dimensions mutual information program – disponível em: <http://www-ncsl.postech.ac.kr/en/software/archives/mmi.tar.z>. acesso em: 10 de setembro de 2007.
- Kennel, M. B.; Brown, R.; Abarbanel, H. D. I. (1992). Determining embedding dimension for phase-space reconstruction using a geometrical construction. *Phys. Rev. A*, v.45, n.6, p.3403–3411.
- Kephart, J. O.; Chess, D. M. (2003). The vision of autonomic computing. *Computer*, v.36, n.1, p.41–50.
- Keskar, D.; Leibowitz, M. (2005). Speeding up openoffice . profiling, tools, approaches. *OpenOffice.org Conference*.
- Kim, S.-J.; Iltis, R. (2008). Performance comparison of particle and extended kalman filter algorithms for gps c/a code tracking and interference rejection. cite-seer.ist.psu.edu/article/kim02performance.html.
- Kimbrel, T.; Karlin, A. R. (1996). Near-optimal parallel prefetching and caching. *IEEE Symposium on Foundations of Computer Science*, p. 540–549.
- Kimbrel, T.; Tomkins, A.; Patterson, R. H.; Bershad, B.; Cao, P.; Felten, E.; Gibson, G.; Karlin, A. R.; Li, K. (1996). A trace-driven comparison of algorithms for parallel prefetching and caching. *Proceedings of the 1996 Symposium on Operating Systems Design and Implementation*, p. 19–34. USENIX Association.
- Kirkpatrick, S.; Gelatt, C. D.; Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, v.220, 4598, p.671–680.
- Kirshner, S.; Smyth, P.; Robertson, A. W. (2004). Conditional chow-liu tree structures for modeling discrete-valued vector time series. *AUAI '04: Proceedings of the 20th conference on Uncertainty in artificial intelligence*, p. 317–324, Arlington, Virginia, United States. AUAI Press.
- Kocjan, W. (2002). Dynamic scheduling: State of the art report.
- Kohonen, T.; Kaski, S.; Lagus, K.; Salojrvi, J.; Honkela, J.; Paatero, V.; Saarela, A. (2000). Self organization of a massive document collection. *IEEE Transactions on Neural Networks*.
- Koskela, T.; Varsta, M.; Heikkonen, J.; Kaski, K. (1997). Time series prediction using RSOM with local linear modesl. Relatório Técnico B15, Helsinki University of Technology, Espoo, Finland.
- Kotz, D. (1991). *Prefetching and Caching Techniques in File Systems for MIMD Multiprocessors*. Tese (Doutorado), Duke University. Available as technical report CS-1991-016.

- Kotz, D. (1994–2000). Bibliography about Parallel I/O. Available on the WWW at <http://www.cs.dartmouth.edu/pario/bib/>.
- Kotz, D.; Ellis, C. S. (1991). Practical prefetching techniques for parallel file systems. *Proceedings of the First International Conference on Parallel and Distributed Information Systems*, p. 182–189. IEEE Computer Society Press.
- Kotz, D.; Ellis, C. S. (1993). Caching and writeback policies in parallel file systems. *Journal of Parallel and Distributed Computing*, v.17, n.1–2, p.140–145.
- Kotz, D.; Nieuwejaar, N. (1994). Dynamic file-access characteristics of a production parallel scientific workload. *Proceedings of Supercomputing '94*, p. 640–649, Washington, DC. IEEE Computer Society Press.
- Kotz, D.; Nieuwejaar, N. (1996). Flexibility and performance of parallel file systems. *ACM Operating Systems Review*, v.30, n.2, p.63–73.
- Kremer, S. C. (1995). On the computational power of elman-style recurrent networks. *Neural Networks, IEEE Transactions on*, v.6, n.4, p.1000–1004.
- Kroeger, T. M.; Long, D. D. E. (1999). The case for efficient file access pattern modeling. *Workshop on Hot Topics in Operating Systems*, p. 14–19.
- Krueger, P.; Babbar, D. (1993). Stealth: a liberal approach to distributed scheduling for networks of workstations.
- Kuehn, J. T.; Smith, B. J. (1988). The Horizon supercomputing system: architecture and software. *International Supercomputer Applications*, v. 1, p. 28–34.
- Kunze, M.; Steffens, J. (1995). Growing cell structure and neural gas; incremental neural networks.
- Lageweg, B. J.; Lenstra, J. K. (1977). Private communication.
- Lampinen, J.; Oja, E. (1989). Self-organizing maps for spatial and temporal AR models. Pietikäinen, M.; Röning, J., editores, *Proc. 6 SCIA, Scand. Conf. on Image Analysis*, p. 120–127, Helsinki, Finland. Suomen Hahmontunnistustutkimuksen seura r. y.
- Lee, L.-W.; Wang, L.-H.; Chen, S.-M. (2008). Temperature prediction and taifex forecasting based on high-order fuzzy logical relationships and genetic simulated annealing techniques. *Expert Syst. Appl.*, v.34, n.1, p.328–336.
- Lehmann, E.; Casella, G. (2003). *Theory of Point Estimation*. Springer.
- Lei, H.; Duchamp, D. (1997). An analytical approach to file prefetching. *1997 USENIX Annual Technical Conference*, Anaheim, California, USA.

- Lendasse, A.; Oja, E.; Simula, O. (2004). Time series prediction competition: The cats benchmark. *Proceedings of International Joint Conference on Neural Networks*, p. 1615–1620, Budapest, Hungary. IEEE.
- Leski, J. M. (2008). An insensitive fuzzy c-means clustering. citeseer.ist.psu.edu/550525.html.
- Lesyng, B.; Bala, P.; Erwin, D. (2003). Eurogrid: European computational grid testbed. *J. Parallel Distrib. Comput.*, v.63, n.5, p.590–596.
- Li, Z.; Parashar, M. (2006). Enabling dynamic composition and coordination for autonomic grid applications using the rudder agent framework. *Knowl. Eng. Rev.*, v.21, n.3, p.221–230.
- Liao, T. W.; Celmins, A. K.; Robert J. Hammell, I. (2003). A fuzzy c-means variant for the generation of fuzzy term sets. *Fuzzy Sets Syst.*, v.135, n.2, p.241–257.
- Lin, M. C.; Yang, C. B.; Huang, K. S. (2002). Prediction of rna secondary structures by genetic algorithms. *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics, SCI 2002*, p. 439–444.
- Liu, H.; Parashar, M.; Member, S. (2005). Accord: A programming framework for autonomic applications. *IEEE Transactions on Systems, Man and Cybernetics, Part C*, v.36, p.341–352.
- Liu, Y. (1997). High-order polytomous Boltzmann machine and forecasting time series I. Rogers, S. K., editor, *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, v. 3077 de Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference, p. 396–408.
- Loiseaux, C.; Graf, S.; Sifakis, J.; Bouajjani, A.; Bensalem, S. (1995). Property preserving abstractions for the verification of concurrent systems. *Formal Methods in System Design*, v.6, n.1, p.11–44.
- Lorenz, E. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Science*, v.20, p.130–141.
- Luecke, G.; Coyle, J.; Hoekstra, J.; Jespersen, H. (1992). Performance of workstations and mainframes for scientific computing. *International Supercomputer Applications*, v.9, n.6, p.10–19.
- LWN (2007). The rotating staircase deadline scheduler. <http://lwn.net/Articles/224865/>.
- Madhyastha, T. M.; Reed, D. A. (1997). Input/output access pattern classification using hidden Markov models. *Proceedings of the Fifth Workshop on Input/Output in Parallel and Distributed Systems*, p. 57–67, San Jose, CA. ACM Press.

- Marchese, M.; Mongelli, M.; Garibbo, A.; Raviola, A. (2006). Mpls versus ip for interworking of wide area subsystems with qos guarantees. Dini, P.; Åhlund, C.; Dini, C.; Borcoci, E., editores, *ICWMC*, p. 26. IEEE Computer Society.
- Marsland, S.; Shapiro, J.; Nehmzow, U. (2002). A self-organising network that grows when required. *Neural Networks*, v.15, n.8-9, p.1041–1058.
- Martin, M.; Harper, P.; Sorin, D.; Hill, M.; Wood, D. (2003). Using destination-set prediction to improve the latency /bandwidth tradeoff in shared memory multiprocessors. *Proceedings of the 30th Annual International Symposium on Computer Architecture*.
- Matzke, D. (1997). Will physical scalability sabotage performance gains? *Computer*, v.30, n.9, p.37–39.
- Mañé, R. (1980). *On the dimension of the compact invariant sets of certain nonlinear maps*. Springer.
- Mello, R.; Dodonov, E.; Bertagna, R.; Senger, L. (2009). Extracting and predicting the communication behavior of parallel applications. *International Journal of Parallel, Emergent and Distributed Systems (to appear)*, v.24, n.3.
- Mello, R. F. (2003). Proposta e avaliação de desempenho de um algoritmo de balanceamento de carga para ambientes distribuídos heterogêneos escaláveis. Tese (Doutorado) - Escola de Engenharia de São Carlos, Universidade de São Paulo, São Paulo.
- Mello, R. F.; Andrade Filho, J. A.; Dodonov, E.; Ishii, R. P.; Yang, L. T. (2007a). Optimizing distributed data access in grid environments by using artificial intelligence techniques. Stojmenovic, I.; Thulasiram, R. K.; Yang, L. T.; Jia, W.; Guo, M.; de Mello, R. F., editores, *ISPA*, v. 4742 de *Lecture Notes in Computer Science*, p. 125–136. Springer.
- Mello, R. F.; Andrade Filho, J. A.; Senger, L. J.; Yang, L. T. (2007b). Routega: A grid load balancing algorithm with genetic support. *AINA*, p. 885–892. IEEE Computer Society.
- Mello, R. F.; Senger, L. J. (2004). A new migration model based on the evaluation of processes load and lifetime on heterogeneous computing environments. *16th Symposium on Computer Architecture and High Performance Computing (SBAC'2004)*, p. 222–227, Foz do Iguaçu, PR, Brazil.
- Mello, R. F.; Senger, L. J. (2006). Model for simulation of heterogeneous high-performance computing environments. *7th International Conference on High Performance Computing in Computational Sciences – VECPAR 2006*, p. 11.
- Mello, R. F.; Senger, L. J.; Yang, L. T. (2005). Automatic text classification using an artificial neural network. *High Performance Computational Science and Engineering*, v.1, p.1–21.

- Mello, R. F.; Senger, L. J.; Yang, L. T. (2006a). Performance evaluation of route: A load balancing algorithm for grid computing. *RITA*, v.13, n.1, p.87–108.
- Mello, R. F.; Senger, L. J.; Yang, L. T. (2006b). A routing load balancing policy for grid computing environments. *AINA (1)*, p. 153–158. IEEE Computer Society.
- Mello, R. F.; Yang, L. T. (2008). Prediction of dynamical, non-linear and unstable process behavior. *Journal of Supercomputing (to appear)*. Springer Netherlands.
- Meng, A. (2003). An introduction to markov and hidden markov models. <http://www2.imm.dtu.dk/pubdb/p.php?3313>.
- Mesbahi, L.; Benyettou, A.; Hendel, F.; Berrached, N. (2004). Reactive navigation of mobile robot by temporal radial basis function approach. *Proceedings of TENCON 2004*, v. 4, p. 483 – 486. IEEE.
- Michael, M.; Moreira, J. E.; Shiloach, D.; Wisniewski, R. W. (2007). Scale-up x scale-out: A case study using nutch/lucene. *Parallel and Distributed Processing Symposium, 2007. IPDPS 2007. IEEE International*, p. 1–8.
- Mielniczuk, J.; Wojdylo, P. (2007). Estimation of hurst exponent revisited. *Computational Statistics & Data Analysis*, v.51, n.9, p.4510–4525.
- Monnerat, L. R.; Bianchini, R. (1998). Efficiently adapting to sharing patterns in software DSMs. *Proc. of the 4th IEEE Symp. on High-Performance Computer Architecture (HPCA-4)*.
- Moy, J. T. (1998). *OSPF: Anatomy of an Internet Routing Protocol*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.
- Müller-Olm, M.; Schmidt, D. A.; Steffen, B. (1999). Model-checking: A tutorial introduction. *SAS '99: Proceedings of the 6th International Symposium on Static Analysis*, p. 330–354, London, UK. Springer-Verlag.
- Nagel, W. E.; Arnold, A.; Weber, M.; Hoppe, H. C.; Solchenbach, K. (1996). VAMPIR: Visualization and analysis of MPI resources. *Supercomputer*, v.12, n.1, p.69–80.
- Naik, V. K.; Setia, S. K.; Squillante, M. S. (1997). Processor Allocation in Multiprogrammed Distributed-memory Parallel Computer Systems. *Journal of Parallel and Distributed Computing*, v.47, n.1, p.28–47.
- Nakano-Miyatake, M.; Perez-Meana, H. (1997a). Adaptive filtering and prediction based on hopfield neural networks. *Proceedings of the IEEE International Conference on Neural Networks*, v. 2, p. 680 – 684.

- Nakano-Miyatake, M.; Perez-Meana, H. (1997b). A continuous time structure for filtering and prediction using hopfield neural networks. *Proceedings of the International Work-Conference on Artificial and Natural Neural Networks: Biological and Artificial Computation: From Neuroscience to Technology*, p. 1241 – 1250.
- Nethercote, N.; Seward, J. (2007). Valgrind: a framework for heavyweight dynamic binary instrumentation. *SIGPLAN Not.*, v.42, n.6, p.89–100.
- Nieuwejaar, N.; Kotz, D.; Purakayastha, A.; Ellis, C. S.; Best, M. (1995). File-access characteristics of parallel scientific workloads. Relatório Técnico PCS-TR95-263, Dept. of Computer Science, Dartmouth College.
- Oliker, L.; Canning, A.; Carter, J.; Shalf, J.; Ethier, S. (2008). Scientific application performance on leading scalar and vector supercomputing platforms. *Int. J. High Perform. Comput. Appl.*, v.22, n.1, p.5–20.
- Omar, W. M.; Taleb-Bendiab, A.; Karam, Y. (2006). Autonomic middleware services for just-in-time grid services provisioning. *Journal of Computer Science*, v.6.
- Ooi, C. H.; Tan, P. (2003). Genetic algorithms applied to multi-class prediction for the analysis of gene expression data. *Bioinformatics*, v.19, n.1, p.37–44.
- Ousterhout, J. K.; Cherenson, A. R.; Douglass, F.; Nelson, M. N.; Welch, B. B. (1988). The sprite network operating system. *Computer*, v.21, n.2, p.23–36.
- Palacios-González, F. (2007). Tvcars models for forecasting. *Neurocomput.*, v.70, n.13-15, p.2379–2391.
- Papadimitriou, C. (1994). *Computational Complexity*. Addison-Wesley, Reading, Massachusetts.
- Parashar, M. (2007). *Autonomic Computing: Concepts, Infrastructure, and Applications / Editor(s): Manish Parashar and Salim Hariri*. Taylor & Francis, Inc., Bristol, PA, USA.
- Pavlovic, V.; Frey, B.; Huang, T. S. (1999). Time series classification using mixed-state dynamic bayesian networks. *IEEE Conf. Computer Vision and Pattern Recognition*, Ft. Collins, CO.
- Pérez-Ortiz, J. A.; Gers, F. A.; E., D.; Schmidhuber, J. (2003). Kalman filters improve LSTM network performance in problems unsolvable by traditional recurrent nets. *Neural Networks*, v. 16(2).
- Pham, D.; Prince, J. (1998). An adaptive fuzzy c-means algorithm for image segmentation in the presence of intensity inhomogeneities. *SPIE Medical Imaging 1998: Image Processing*, v. 3338, p. 555–563.

- Powell, M. (1987). Radial basis functions for multivariable interpolation: a review. *Clarendon Press Institute Of Mathematics And Its Applications Conference Series*, v.1, p.143–167.
- Qiao, L.; Agrawal, D.; Abbadi, A. E. (2003). Supporting sliding window queries for continuous data streams. *Scientific and Statistical Database Management, International Conference on*, v.0, p.85.
- Ribeiro, S.; Alquézar, R. (2002). Incremental construction of lstm recurrent neural network. *Proceedings of Iberoamerican Conference on Pattern Recognition*.
- Rilling, L.; Morin, C. (2005). A practical transparent data sharing service for the grid. *The 2005 International Workshop on DSM (DSM2005)*, Cardiff, May 2005. IEEE Computer Society.
- Roberson, J. (2003). Ule: a modern scheduler for freebsd. *BSDC'03: Proceedings of the BSD Conference 2003 on BSD Conference*, p. 3–3, Berkeley, CA, USA. USENIX Association.
- Rosen, E. C. (1982). Exterior gateway protocol (egp).
- Rosenstein, M. T.; Collins, J. J.; De Luca, C. J. (1993). A practical method for calculating largest lyapunov exponents from small data sets. *Physica D*, v.65, p.117–134.
- Sakr, M.; Giles, C.; Levitan, S.; Horne, B.; Maggini, M.; Chiarulli, D. (1996). On-line prediction of multiprocessor memory access patterns. *Proceedings of the IEEE International Conference on Neural Networks*, p. 1564–1569.
- Sakr, M. F.; Levitan, S. P.; Chiarulli, D. M.; Horne, B. G.; Giles, C. L. (1997). Predicting multiprocessor memory access patterns with learning models. *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*, p. 305–312, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.
- Salehie, M.; Tahvildari, L. (2005). Autonomic computing: emerging trends and open problems. *DEAS '05: Proceedings of the 2005 workshop on Design and evolution of autonomic application software*, n. 4 in 30, p. 1–7, New York, NY, USA. ACM Press.
- Santos, M. L.; Mello, R. F.; Yang, L. T. (2007). Extraction and classification of user behavior. Kuo, T.-W.; Sha, E. H.-M.; Guo, M.; Yang, L. T.; Shao, Z., editores, *EUC*, v. 4808 de *Lecture Notes in Computer Science*, p. 493–506. Springer.
- Särkkä, S.; Vehtari, A.; Lampinen, J. (2007). Cats benchmark time series prediction by kalman smoother with cross-validated noise density. *Neurocomput.*, v.70, n.13-15, p.2331–2341.
- Scheinerman, E. R. (1996). *Invitation to Dynamical Systems*. Prentice-Hall, Upper Saddle River, NJ 07458, USA.

- Schuster, A. (2003). Scalable distributed model checking: Experiences, lessons, and expectations. *Electr. Notes Theor. Comput. Sci.*, v.89, n.1.
- Schölkopf, B.; Smola, A. J. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning)*. The MIT Press.
- Scott, L.; Chahine, J.; Ruggiero, J. (2000). Prediction of protein structures using a hopfield network. *SBRN '00: Proceedings of the VI Brazilian Symposium on Neural Networks (SBRN'00)*, p. 284, Washington, DC, USA. IEEE Computer Society.
- Seiler, L.; Carmean, D.; Sprangle, E.; Forsyth, T.; Abrash, M.; Dubey, P.; Junkins, S.; Lake, A.; Sugerman, J.; Cavin, R.; Espasa, R.; Grochowski, E.; Juan, T.; Hanrahan, P. (2008). Larrabee: a many-core x86 architecture for visual computing. *ACM Trans. Graph.*, v.27, n.3, p.1–15.
- Semenov, M. A.; Terkel, D. A. (2003). Analysis of convergence of an evolutionary algorithm with self-adaptation using a stochastic lyapunov function. *Evol. Comput.*, v.11, n.4, p.363–379.
- Senger, L. J. (2004). An instance-based approach for predicting parallel application execution times. *Proceedings of I2TS*, Sao Carlos, Sao Paulo, Brazil.
- Senger, L. J.; Mello, R. F.; Santana, M. J.; Santana, R. C. (2005). An on-line approach for classifying and extracting application behavior on linux. Yang, L. T.; Guo, M., editores, *High Performance Computing: Paradigm and Infrastructure*, chapter 20. John Wiley and Sons.
- Senger, L. J.; Santana, M. J.; Santana, R. H. C. (2003). A new approach for acquiring knowledge of resource usage in parallel applications. *Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'2003)*, p. 607–614.
- Sevcik, K. C. (1989). Characterizations of Parallelism in Applications and their use in Scheduling. *Performance Evaluation Review*, v.17, n.1, p.171–180.
- Shah, P.; Pâris, J. F.; Amer, A. (2004). Identifying stable file access patterns. *Proceedings of the 21st IEEE/12th NASA Goddard Conference on Mass Storage Systems and Technologies (MSST 2004)*, p. 230–240.
- Shefler, W. C. (1988). *Statistics: Concepts and Applications*. The Benjamin/Cummings.
- Shivaratri, N. G.; Krueger, P.; Singhal, M. (1992). Load distributing for locally distributed systems. *Computer*, v.25, n.12, p.33–44.
- Silva, F. A. B. D.; Scherson, I. D. (2000). Improving Parallel Job Scheduling Using Runtime Measurements. Feitelson, D. G.; Rudolph, L., editores, *Job Scheduling Strategies for Parallel Processing*, p. 18–38. Springer Verlag. Lect. Notes Comput. Sci. vol. 1911.

- Smith, J. E.; Nair, R. (2005). The architecture of virtual machines. *Computer*, v.38, n.5, p.32–38.
- Smith, W.; Foster, I. T.; Taylor, V. E. (1998). Predicting Application Run Times Using Historical Information. *JSSPP*, p. 122–142.
- Spessoto, E. A. (2003). Um mecanismo de ajuste dinâmico para diminuição de latência e do tempo de resposta em sistemas de arquivos paralelo. Dissertação (Mestrado), DC/UFSCar.
- Spinellis, D. (2008). A tale of four kernels. Schäfer, W.; Dwyer, M. B.; Gruhn, V., editores, *ICSE '08: Proceedings of the 30th International Conference on Software Engineering*, p. 381–390, New York. Association for Computing Machinery.
- Squires, C.; Shavlik, J. (1991). Experimental analysis of aspects of the cascade-correlation learning architecture. *Machine learning research group working paper 91-1, Computer Sciences Department, University of Wisconsin-Madison*.
- Steijvers, M.; Grunwald, P. (1996). A recurrent network that performs a contextsensitive prediction task. *Proceedings of the 18th Annual Conference of the Cognitive Science Society*.
- Stockinger, H.; Samar, A.; Allcock, B.; Foster, I.; Holtman, K.; Tierney, B. (2001). File and object replication in data grids.
- Sudarsanam, A.; Srinivasan, M.; Panchanathan, S. (2004). Resource estimation and task scheduling for multithreaded reconfigurable architectures. *icpads*, v.00, p.323.
- Sum, J.; Leung, C.; Chan, L. (1996). Extended kalman filter in recurrent neural network training and pruning. *Technical report CS-TR-96-05, Department of Computer Science and Engineering, The Chinese University of Hong Kong*.
- Suykens, J. A. K.; Vandewalle, J. (2000). The k.u.leuven competition data: a challenge for advanced neural network techniques. *ESANN*, p. 299–304.
- Takens, F. (1980). Detecting strange attractors in turbulence. *Dynamical Systems and Turbulence*, p. 366–381. Springer.
- Tanenbaum, A. S.; Steen, M. V. (2001). *Distributed Systems: Principles and Paradigms*. Prentice Hall PTR, Upper Saddle River, NJ, USA.
- Tannenbaum, A. S. (2001). *Modern Operating Systems, 2nd edition*. Prentice Hall.
- Tannenbaum, T.; Wright, D.; Miller, K.; Livny, M. (2001). Condor – a distributed job scheduler. Sterling, T., editor, *Beowulf Cluster Computing with Linux*. MIT Press.
- Thacker, N. A.; Abraham, I.; Courtney, P. (1997). Supervised learning extensions to the clam network. *Neural Networks*, v.10, n.2, p.315–326.

- Thacker, N. A.; Mayhew, J. E. W. (1990). Designing a layered network for context sensitive pattern classification. *Neural Networks.*, v.3, n.3, p.291–299.
- Thelen, E. (2000). Intel paragon. <http://ed-thelen.org/comp-hist/intel-paragon.html>.
- Thomas, J.; Sycara, K. (1999). Integrating genetic algorithms and text learning for financial prediction. *Proceedings of the GECCO-2000 Workshop on Data Mining with Evolutionary Algorithms*.
- Tian, X.; Chen, Y.-K.; Girkar, M.; Ge, S.; Lienhart, R.; Shah, S. (2003). Exploring the use of hyper-threading technology for multimedia applications with intel openmp compiler. citeseer.ist.psu.edu/tian03exploring.html.
- Tianfield, H.; Unland, R. (2004). Towards autonomic computing systems. *Engineering Applications of Artificial Intelligence*, v.17, n.7, p.689–699.
- Titsias, M.; Likas, A. (2000). A probabilistic rbf network for classification. *IJCNN '00: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00)-Volume 4*, p. 4238, Washington, DC, USA. IEEE Computer Society.
- Venayagamoorthy, G. K. (2007). 2007 special issue: Online design of an echo state network based wide area monitor for a multimachine power system. *Neural Networks*, v.20, n.3, p.404–413.
- Verstraten, D.; Schrauwen, B.; Stroobandt, D.; Campenhout, J. V. (2005). Isolated word recognition with the liquid state machine: a case study. *Inf. Process. Lett.*, v.95, n.6, p.521–528.
- Vetter, J.; Chambreau, C. (2008). mpip: Lightweight, scalable mpi profiling. <http://mpip.sourceforge.net/>.
- Vlassis, N. A.; Dimopoulos, A.; Papakonstantinou, G. K. (1997). The probabilistic growing cell structures algorithm. *ICANN '97: Proceedings of the 7th International Conference on Artificial Neural Networks*, p. 649–654, London, UK. Springer-Verlag.
- Voelker, G. M.; Anderson, E. J.; Kimbrel, T.; Feeley, M. J.; Chase, J. S.; Karlin, A. R.; Levy, H. M. (1998). Implementing cooperative prefetching and caching in a globally-managed memory system. *Proceedings of the Joint International Conference on Measurement and Modeling of Computer Systems*, p. 33–43. ACM Press.
- Waibel, A.; Hanazawa, T.; Hinton, G.; Shikano, K.; Lang, K. (1989). Phoneme recognition using time delay neural networks. *IEEE Transactions on Acoustics, Speech and Signal Processing*, v.37, p.328–339.

- Wang, D.; Li, G. (2008). Efficient distributed bandwidth management for mpls fast reroute. *IEEE/ACM Trans. Netw.*, v.16, n.2, p.486–495.
- Weigend, A. S.; Gershenfeld, N. A. (1994). Time series prediction: Forecasting the future and understanding the past. Weigend, A. S.; Gershenfeld, N. A., editores, *Santa Fe Institute Studies in the Sciences of Complexity, Proceedings of the NATO Advanced Research Workshop on Comparative Time Series Analysis, held in Santa Fe, New Mexico, May 14-17, 1992, Reading, MA: Addison-Wesley, 1994, edited by Weigend, Andreas S.; Gershenfeld, Neil A.*
- Wende, M.; Schoettner, M.; Goeckelmann, R.; Bindhammer, T.; Schulthess, P. (2002). Performance evaluation of a transactional dsm system.
- Werthimer, D.; Cobb, J.; Lebofsky, M.; Anderson, D.; Korpela, E. (2001). Seti@home—massively distributed computing for seti. *Comput. Sci. Eng.*, v.3, n.1, p.78–83.
- Whitney, H. (1936). Differentiable manifolds. *The Annals of Mathematics*, v.37, n.3, p.645–680.
- Wilding, M.; Behman, D. (2005). *Self-Service Linux: Determining Problems and Finding Solutions*. Prentice Hall PTR, Upper Saddle River, NJ, USA.
- Wong, C. S.; Tan, I.; Kumari, R. D.; Wey, F. (2008). Towards achieving fairness in the linux scheduler. *SIGOPS Oper. Syst. Rev.*, v.42, n.5, p.34–43.
- Wypychowski, J.; Pytliński, J.; Skorwider, L.; Nazaruk, M.; Benedyczak, K.; Wroński, M.; Bala, P. (2004). Life sciences grid in eurogrid and grip projects. *New Gen. Comput.*, v.22, n.2, p.147–156.
- Xi, Y. (2001). An analytical model for buffer hit rate prediction. *MSc Thesis, Dept. of Computing and Information Science, Queen's University*.
- Yaffee, R. A.; McGee, M. (2000). *Introduction to Time Series Analysis and Forecasting: With Applications of SAS and SPSS*. Academic Press, Inc., Orlando, FL, USA.
- Yee, P.; Haykin, S. (1998). A dynamic regularized radial basis function network for nonlinear, nonstationary time series prediction. *IEEE Transaction on Signal Processing*, v. 47, p. 2503–2521.
- Yeo, C.; Buyya, R.; Pourreza, H.; Eskicioglu, R.; Graham, P.; Sommers, F. (2006). Cluster computing: High-performance, high-availability, and high-throughput processing on a network of computers. *Handbook of Nature-Inspired and Innovative Computing*, p. 521–551. Springer USA.

- Yuan, M.; Xie, M. (2002). An incremental representation of conceptual symbols using rce neural network. *ICDL '02: Proceedings of the 2nd International Conference on Development and Learning*, p. 102, Washington, DC, USA. IEEE Computer Society.
- Yule, G. U. (1927). On a method of investigating periodicities in disturbed series, with special reference to wolfer's sunspot numbers. *Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character*, v.226, p.267–298.
- Zemouri, R.; Racocanu, D.; Zerhouni, N. (2003). Recurrent radial basis function network for time-series prediction. *Engineering Applications of Artificial Intelligence*, v.16, n.5-6, p.453–463.
- Zhang, N.; Poole, D. (1994). A simple approach to bayesian network computations. *Proceedings of the Tenth Canadian Conference on Artificial Intelligence*, p. 171–178.
- Zhao, J.; Shawe-Taylor, J. (1994). A recurrent network with stochastic weights. *Internal Report, Department of Computer Science, Royal Holloway University of London*.
- Zhou, S.; Ferrari, D. (1987). An experimental study of load balancing performance. Relatório técnico, University of California at Berkeley, Berkeley, CA, USA.
- Zorzatto, J. A. (2005). Um sistema web para monitoramento e controle da grade progrid. Dissertação (Mestrado), DC/UFSCar.