UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RODRIGO ARAUJO REAL

TiPS, uma proposta de escalonamento direcionada à computação pervasiva

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, novembro de 2004

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Rodrigo Araujo Real,

TiPS, uma proposta de escalonamento direcionada à computação pervasiva /

Rodrigo Araujo Real. – Porto Alegre: PPGC da UFRGS, 2004.

115 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2004. Orientador: Cláudio Fernando Resin Geyer.

1. Escalonamento. 2. Computação pervasiva. 3. Tratamento de incertezas. I. Geyer, Cláudio Fernando Resin. II. Título.

"O que sabemos é uma gota, o que ignoramos é um oceano."
– Isaac Newton

AGRADECIMENTOS

Gostaria de agradecer aos professores, funcionários e colegas do Instituto de Informática que de forma direta ou indireta contribuíram para a realização este trabalho.

Agradeço também aos colegas e amigos que habitaram e tornaram a sala 205 um ótimo ambiente de trabalho. Dentre eles, agradeço especialmente aos amigos: Adenauer, sempre trazendo novidades e mostrando caminhos a serem seguidos; Iara que nos trazia na linha, tentando amansar a dispersão que embora nos fizesse crescer, muitas vezes nos tirava do foco principal do trabalho; Luciano, pela eterna disposição em ajudar e por compartilhar seu qualificado conhecimento; Mario, pelas conversas sobre redes bayesianas e por ajudar significativamente na manutenção do bom humor no grupo; Gustavo pela participação ativa como colega, demonstrando sempre boa vontade em participar das discussões.

Agradeço ao Cláudio Geyer, meu orientador, pela amizade, pelas idéias, pela boa vontade em atender minhas solicitações e pela preocupação em manter e consolidar um excelente grupo de pesquisa e trabalho.

Agradeço também aos meus familiares que me apoiaram na decisão de fazer o curso, em especial ao tio Duca, a minha mãe e minha írmã.

Agradeço ao apoio financeiro, concedido na forma de bolsa pela CAPES.

Por fim, mas considerando que esta é uma posição de destaque, gostaria de agradecer a Rita, que é uma grande companheira, por ter aceitado a idéia de me acompanhar durante a realização deste trabalho, por ter compreendendido as minhas ausências.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS 8

LISTA DE FIGURAS 10

LISTA DE TABELAS 11

RESUMO 12

ABSTRACT 13

1 INTRODUÇÃO 14
1.1 Motivação e objetivos 15
1.2 Contribuição do autor 16
1.3 Estrutura da dissertação 16

2 ISAM, O ESCOPO DE PESQUISA 17
2.1 Contribuições da pesquisa em áreas relacionadas 17
2.1.1 Contribuições das pesquisas em sistemas distribuídos 18
2.1.2 Contribuições das pesquisas em computação móvel 18
2.2 Questões de pesquisa na computação pervasiva 19
2.2.1 Caracterizando a computação pervasiva 19
2.2.2 Estratégia de adaptação 21
2.2.3 Consciência de contexto 22
2.3 Computação pervasiva na visão do projeto ISAM 22
2.4 Arquitetura ISAM 22

3 ESCALONAMENTO NA PERSPECTIVA DA COMPUTAÇÃO PERVASIVA 27
3.1 O gerenciamento de carga 27
3.2 Políticas de escalonamento estáticas e dinâmicas 28
3.3 Nível de complexidade 30
3.4 Estratégias de escalonamento determinísticas e estocásticas . . 31
3.5 Políticas de escalonamento centralizadas e distribuídas 32
3.6 Componentes da política de escalonamento 35
3.6.1 Política de informação 35
3.6.2 Política de transferência 38
3.6.3 Política de seleção 39
3.6.4 Política de localização 40
3.7 Classificação de algoritmos de escalonamento 41

3.7.1 O escopo da base de decisão e do espaço de migração 41
3.7.2 A taxonomia de Casavant 42
3.7.3 O esquema de classificação ESR 46

4 EXEHDA: O AMBIENTE DE EXECUÇÃO DO ISAM 50
4.1 O ambiente de execução EXEHDA 50
4.2 O ISAM*pe* 51
4.3 Características do EXEHDA 51
4.4 A organização do escalonamento no EXEHDA 53
4.4.1 A adaptação multinível colaborativa 53
4.5 O disparo de um `OX` sob a ótica do escalonamento 54
4.5.1 A abstração `OX` 54
4.6 O disparo de um `OX` na visão do EXEHDAnodo 56

5 TIPS: CONCEPÇÃO E MODELAGEM 58
5.1 Integração do TiPS com o EXEHDA 58
5.2 O problema do escalonamento e o TiPS 59
5.3 A política de escalonamento utilizada 59
5.3.1 Política de informação 60
5.3.2 Política de transferência 60
5.3.3 Política de seleção 60
5.3.4 Política de localização 61
5.4 O tratamento das incertezas no TiPS 61
5.4.1 As redes bayesianas e o sensoriamento 61
5.4.2 O modelo de sensoriamento do TiPS 62
5.5 Efetivando a decisão de escalonamento no TiPS 63
5.6 O aprendizado no TiPS 64
5.7 Caracterizando a operação do TiPS 65
5.8 Os componentes do TiPS 69
5.8.1 Gerenciador de componentes 69
5.8.2 Coletor 70
5.8.3 Distribuidor 70
5.8.4 Gerenciador de dados 70
5.8.5 Tomador de decisões 70
5.8.6 Rede bayesiana 71
5.8.7 Gerador de utilidades 71
5.8.8 Aprendizado 72

6 TIPS: PROTÓTIPO E RESULTADOS 74
6.1 Prototipação dos componentes do TiPS 74
6.2 Integração com o sistema de monitoração 75
6.3 Integração do NWS com o EXEHDA 75
6.4 Escalonadores de referência: NWSSched e InstSched 76
6.5 A plataforma de equipamentos empregada 76
6.6 A aplicação utilizada 77
6.6.1 Oportunidades de paralelização da aplicação 78
6.7 O emprego de carga sintética 79
6.8 Resultados obtidos 80

7 TRABALHOS RELACIONADOS 83
7.1 *Network Weather Service* 83
7.1.1 Monitores de desempenho 84
7.1.2 Previsão 85
7.1.3 A relação do NWS com o TiPS 86
7.2 Condor 86
7.2.1 A estrutura de escalonamento 87
7.2.2 Tolerância a falhas no Condor 87
7.2.3 Atuais frentes de trabalho 88
7.2.4 Relacionando o Condor e o TiPS 88
7.3 APST: *AppLeS Parameter Sweep Template* 88
7.3.1 Modelo de aplicação e do ambiente 89
7.3.2 O algoritmo de escalonamento 89
7.3.3 Relacionando o APST com o TiPS 91
7.4 PaRT 91
7.4.1 Distribuição uniforme de carga 91
7.4.2 Alocação de trabalho por demanda 92
7.4.3 Estratégia determinística baseada em sensores 92
7.4.4 Estratégia baseada em redes de decisão 92
7.4.5 Estratégia baseada em redes de decisão com adaptação 92
7.4.6 Avaliação das estratégias empregadas 92
7.4.7 Avaliando o TiPS em relação ao PaRT 93
7.5 Comparativo dos trabalhos em relação ao TiPS 93

8 CONCLUSÃO E TRABALHOS FUTUROS 94
8.1 Conclusão 94
8.1.1 Contribuições da pesquisa 94
8.1.2 Publicações durante o mestrado 95
8.2 Trabalhos futuros 97

REFERÊNCIAS 99

APÊNDICE A UMA BREVE REVISÃO DA ÁREA DE TEORIA DA DECISÃO 108
A.1 Teoria das probabilidades 108
A.2 Redes causais 109
A.2.1 Conexões seriais 110
A.2.2 Conexões divergentes 110
A.2.3 Conexões convergentes 110
A.3 Redes bayesianas 111
A.3.1 A regra de Bayes 112
A.3.2 A regra da cadeia para redes bayesianas 112
A.3.3 Exemplo de aplicação de redes bayesianas 112
A.4 Teoria da utilidade 114
A.5 Utilidade esperada 114

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
APST	*AppLeS Parameter Sweep Template*
AR	Aprendizado por reforço
AVA	Ambiente Virtual da Aplicação
AVU	Ambiente Virtual do Usuário
BDA	Base de Dados pervasiva das Aplicações
CIB	*Cell Information Base*
ESR	*Events, Surrounding, Requirements*
EXEHDA	*Execution Environment for High Distributed Applications*
FCFS	*First Come First Served*
GAD	Grafo Acíclico Dirigido
GASS	*Global Access to Secondary Storage*
ISAM	Infra-estrutura de Suporte às Aplicações Móveis Distribuídas
ISAMpe	ISAM *pervasive enviroment*
J2SE	*Java 2 Standard Edition*
J2ME	*Java 2 Micro Edition*
JVM	*Java Virtual Machine*
LDAP	*Lightweight Directory Access Protocol*
NWS	*Network Weather Service*
OX	*Objeto eXehda*
PBS	*Portable Batch System*
PDA	*Personal Digital Assistant*
RB	*Resource Broker*
RMI	*Remote Method Invocation*
TCP	*Transmission Control Protocol*
TiPS	*TiPS is a Probabilistic Scheduler*

TMC	Tempos Mínimos de Conclusão
TPC	Tabela de Probabilidades Condicionais
URL	*Uniform Resource Locator*

LISTA DE FIGURAS

Figura 2.1: O relacionamento da computação pervasiva com a computação móvel e os sistemas distribuídos 19
Figura 2.2: Linha evolutiva da mobilidade na visão do ISAM 23
Figura 2.3: Visão geral do Projeto ISAM 24
Figura 2.4: Visão geral da arquitetura ISAM 24

Figura 3.1: A base de decisão e o espaço de migração 42
Figura 3.2: Taxonomia de Casavant 44

Figura 4.1: Elementos básicos do ambiente de execução 52
Figura 4.2: Adaptação multinível 53
Figura 4.3: Alocação inter-celular de recursos de processamento 55
Figura 4.4: Disparo de um `OX` em um EXEHDAnodo 56

Figura 5.1: Modelo de sensor 61
Figura 5.2: Modelo de rede bayesiana adotado 63
Figura 5.3: O diagrama de influência de um recurso 64
Figura 5.4: Diagrama de Influência 64
Figura 5.5: Exemplo de instanciação do diagrama de influência 65
Figura 5.6: Algoritmo de aprendizado 66
Figura 5.7: Exemplo de comparação entre dois recursos 68
Figura 5.8: O *framework* TiPS 69
Figura 5.9: Fluxograma geral de aprendizado 72

Figura 6.1: EXEHDA-AMI: administração do TiPS 75
Figura 6.2: Área de $\frac{1}{4}$ círculo 77
Figura 6.3: Algoritmo de cálculo do π 78
Figura 6.4: Carga sintética de 20% de ocupação média 79
Figura 6.5: Carga sintética de 50% de ocupação média 79
Figura 6.6: Carga sintética de 80% de ocupação média 79
Figura 6.7: *Speedup* obtido no pior caso dos experimentos 82
Figura 6.8: *Speedup* médio dos experimentos 82
Figura 6.9: *Speedup* obtido no melhor caso dos experimentos 82

Figura 7.1: A arquitetura lógica do NWS 84
Figura 7.2: Modelo de aplicação e de ambiente 89
Figura 7.3: Esqueleto do algoritmo de escalonamento 90

LISTA DE TABELAS

Tabela 3.1: O esquema de classificação ESR 47
Tabela 3.2: Atributos da classificação ESR de estratégias de escalonamento . 48

Tabela 5.1: Discretização da informação do sensor de carga de CPU 66
Tabela 5.2: Probabilidades condicionais do sCPU dado CPU 67
Tabela 5.3: Probabilidades condicionais de sConstância dado Constância . . . 67
Tabela 5.4: Utilidades atribuídas a cada classe 69

Tabela 6.1: Cluster corisco 76
Tabela 6.2: Percentual médio de ocupação do processador de cada classe de nodo 77
Tabela 6.3: Caracterização dos experimentos 80
Tabela 6.4: Médias dos valores de utilidade conforme a carga sintética 80
Tabela 6.5: Resumo do comportamento das execuções 81
Tabela 6.6: Número de vezes que cada nodo foi escolhido 81
Tabela 6.7: Resumos dos índices de *speedup* obtidos 81

Tabela 7.1: Comparativo de características entre o TiPS e os demais sistemas analisados 93

RESUMO

A evolução das tecnologias de rede estão impulsionando o avanço da área da computação pervasiva. O projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas), em desenvolvimento no II/UFRGS, tem como foco atender as demandas de pesquisa desta área e tem como premissa uma abordagem integrada na concepção do ambiente de desenvolvimento e do ambiente de execução. O EXEHDA (*Execution Environment for High Distributed Applications*) constitui o ambiente de execução do ISAM, sendo responsável pela gerência do processamento das aplicações pervasivas. Esta dissertação propõe um *framework* de escalonamento denominado TiPS concebido como um módulo do EXEHDA. O escopo de pesquisa do TiPS tem como tônica o escalonamento na computação pervasiva e a sua concepção na forma de um *framework* permite a utilização de diferentes estratégias de escalonamento, através da troca de seus componentes mesmo durante a execução. A modelagem do TiPS considera o uso de inteligência artificial baseada em redes bayesianas na proposição da heurística de escalonamento a ser empregada no seu *framework*. A utilização de redes bayesianas tem por objetivo o tratamento das incertezas relacionadas à elevada dinamicidade típica do ambiente de computação pervasiva. O TiPS foi implementado em Java, com suas funcionalidades integradas aos outros serviços do EXEHDA. Neste sentido foi desenvolvido para gerenciar o *framework* do TiPS, um módulo para a ferramenta EXEHDA-AMI, utilizada para gerenciar o EXEHDA como um todo. O TiPS foi comparado com outros dois escalonadores, para tanto foi desenvolvida uma aplicação de teste e um módulo de geração sintética de carga para promover a dinamicidade do contexto de execução. Os resultados obtidos com o TiPS foram promissores e apontam para a viabilidade do emprego de heurísticas de escalonamento que envolvem técnicas de inteligência artificial na computação pervasiva.

Palavras-chave: Escalonamento, computação pervasiva, tratamento de incertezas.

TiPS, a scheduling propose directed to the pervasive computing

ABSTRACT

The evolution of the network technologies are strengthening the pervasive computing development. The ISAM (*Infra-estrutura de Suporte às Aplicações Pervasivas*) is under development in the II/UFRGS and has as it's main focus on assisting the research demands related to this theme, and its approach is to integrate the development environment and the execution environment. The EXEHDA (Execution Environment for High Distributed Applications) constitutes the execution environment of ISAM, being responsible for the management of the pervasive applications execution. This dissertation proposes a framework for scheduling called TiPS, which was conceived as an EXEHDA module. The research scope of TiPS has as its tonic the scheduling in the pervasive computing environment, and its conception as a framework permits the use of different scheduling strategies, by the exchange of its components even during the execution. The TiPS modeling considers the integration of an artificial intelligence strategy based on bayesian networks, within the scheduling framework. The use of bayesian networks has the objective to handle the uncertainties related to the highly dynamic behavior, which is typical in the pervasive computing. TiPS was implemented in Java and its functionalities were integrated to other EXEHDA services, in this sense it was also developed a management module to the EXEHDA-AMI tool, which is used to manage EXEHDA. TiPS was compared to two other schedulers, for this comparison it was developed a test application and a synthetic load generator to create dynamics of the execution environment. The results obtained by TiPS points to the viability of the use of scheduling heuristics based on artificial intelligence tools in the pervasive computing.

Keywords: scheduling, pervasive computing, uncertainty handling.

1 INTRODUÇÃO

A disseminação crescente da Internet, somada ao aumento da velocidade operacional das redes de computadores e das suas conexões, levam a uma perspectiva de uso unificado dos recursos distribuídos, o qual pode ser realizado a partir de qualquer equipamento das redes interligadas. Esta nova forma de processamento em rede assume diferentes perspectivas: *metacomputing* (FOSTER; KESSELMAN, 1997), computação em grade (FOSTER; KESSELMAN, 1999), *internet computing* (REAL; DUARTE FILHO; YAMIN, ????) e mais recentemente *peer-to-peer computing* (ORAM, 2001; BAKER; BUYYA; LAFORENZA, 2001).

Por sua vez, a computação móvel é uma nova perspectiva que amplia o conceito de rede-sem-fio. Nesta perspectiva, o usuário portando dispositivos móveis como computadores portáteis e de mão, independentemente da sua localização física, terá acesso a uma infra-estrutura de serviços (NOBLE, 2000).

Integrando estas duas visões, observa-se um movimento em direção à computação pervasiva (SATYANARAYANAN, 2001), criando aplicações com novas funcionalidades, as quais tentam aumentar sua usabilidade e atender às necessidades dos usuários que se deslocam. Computação pervasiva é a proposta de um novo paradigma computacional, que permite ao usuário o acesso ao seu ambiente computacional a partir de qualquer lugar, a todo e qualquer tempo, usando vários tipos de dispositivos, sejam eles móveis ou não. Em contraste com a premissa dos sistemas distribuídos de fornecer transparência da distribuição para os usuários, esta nova classe de aplicações é *context-aware* e tenta tirar vantagem desta informação (AUGUSTIN et al., 2002; AUGUSTIN, 2004. 194p). Para alcançar esta consciência, a aplicação ou o ambiente de execução pró-ativamente monitoram e controlam as condições do ambiente. A aplicação ou o sistema reagem às alterações no ambiente através do processo de adaptação. Este processo requer a existência de múltiplos caminhos de execução para uma aplicação, ou configurações alternativas, as quais exibem diferentes perfis de utilização (históricos) dos elementos computacionais.

Esta visão apresenta uma série de novos e renovados desafios, que começam a ser abordados, oriundos do dinamismo e heterogeneidade do ambiente, além dos novos requisitos das aplicações no estilo siga-me (*follow-me*) da computação pervasiva. O dinamismo está tanto na aplicação quanto no sistema de execução. Ambos operam em um ambiente cujas condições na disponibilidade e no acesso aos recursos são variáveis no tempo e no espaço. Como conseqüência, novos tipos de aplicações estão aparecendo, as quais têm um comportamento determinado pela sua sensibilidade à variação nas condições de alguns elementos do ambiente. O ambiente é definido por elementos computacionais que podem ser medidos, como largura de banda, latência da rede, consumo de energia, localização do usuário, preferências do usuário, entre

outros (AUGUSTIN et al., 2002).

É no cenário da computação pervasiva que a necessidade da adaptação se potencializa. Somente através de um mecanismo eficiente de adaptação as aplicações para este tipo de ambiente podem sobrepor os desafios, tais como variação imprevisível na qualidade da rede, grande disparidade na disponibilidade de serviços remotos e variação da capacidade disponível de processamento. Sistemas distribuídos tradicionais são construídos com suposições sobre a infra-estrutura física de execução, como conectividade permanente e disponibilidade dos recursos necessários. Porém, essas suposições não são válidas na computação pervasiva (SATYANARAYANAN, 2001).

O projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas) (ISAM, 2002), em desenvolvimento no Instituto de Informática da UFRGS, mais especificamente no GPPD (Grupo de Processamento Paralelo e Distribuído), objetiva apontar respostas a questões que surgem no ambiente de computação pervasiva. O ISAM contempla uma arquitetura de software que aborda de forma integrada os problemas da adaptação considerando cenários altamente dinâmicos, envolvendo tanto mobilidade física, na qual o equipamento e o usuário poderão se deslocar, como lógica, em que o software poderá transitar entre os equipamentos. Esta dissertação está inserida no projeto ISAM, tratando algumas questões relativas ao escalonamento no ambiente da computação pervasiva.

1.1 Motivação e objetivos

Uma das formas de **adaptação** no âmbito do projeto ISAM é o **escalonamento** de recursos. O escalonamento ótimo em sistemas distribuídos é conhecido no caso geral como um problema NP-completo (KWOK; AHMAD, 1999), este problema se potencializa no cenário da computação pervasiva. Um dos motivos que fazem com que o problema não tenha uma solução ótima é a limitada capacidade de manutenção de um estado suficientemente preciso do ambiente, além disto, as necessidades da aplicação também, em geral, não são conhecidas de forma precisa.

A complexidade do escalonamento em um ambiente tão dinâmico e heterogêneo introduz custos, tanto no gerenciamento, como na execução da aplicação propriamente dita. Estes custos podem se tornar elevados devido à necessidade de efetuar predições e estimativas de comportamentos futuros da aplicação e do próprio sistema. Por outro lado, os ganhos potenciais podem ser significativos. Nigel Davies *et al.* (1997) acredita que é somente através de um processo de **adaptação**, fornecendo às aplicações informações gerenciadas sobre trocas na sua infra-estrutura de suporte, que se pode operar eficientemente em um ambiente distribuído altamente dinâmico.

É usual, nas soluções atuais, que o escalonador empregue um modelo para tomada de decisão que não consiga contemplar a complexidade inerente a dinamicidade dos elementos do meio de execução.

Considerando a dificuldade e o alto custo em obter informações precisas e atualizadas com relação ao estado global do ambiente, e a dificuldade em realizar predições do que irá acontecer no futuro, usando métodos determinísticos, o escalonador precisa tomar decisões em condições de incerteza com relação ao passado, ao presente e ao futuro próximo (SANTOS; PROENÇA, 2001, 2002).

Os resultados apresentados por Luís Paulo Peixoto dos Santos & Alberto Proença (2002) indicam que estratégias de escalonamento dinâmico baseadas em sensores com

amostragem dinâmica e que incluem um modelo do ambiente a ser gerenciado na forma de uma rede de Bayes, podem obter significativas melhorias de desempenho em relação a estratégias mais tradicionais, como distribuição uniforme do trabalho, alocação de tarefa sob demanda e estratégias determinísticas baseadas em sensores. Luís Paulo Peixoto dos Santos & Alberto Proença (2002) mostra também que os custos adicionais impostos pelo mecanismo mais sofisticado de tomada de decisão são compensados pela melhor qualidade do escalonamento de recursos que pode ser obtido.

Este trabalho tem como objetivo geral propor um módulo de escalonamento para o ambiente de execução da arquitetura ISAM (EXEHDA) que faça tratamento adequado à dinamicidade do ambiente.

Dentre os objetivos específicos, destacam-se:

- utilizar redes bayesianas como um mecanismo probabilístico para tratar a incerteza com relação ao estado do ambiente de execução;
- avaliar alternativas de ajuste continuado de probabilidades ou parâmetros da rede bayesiana;
- considerar na heurística as chances de fracasso das alternativas de decisão a serem tomadas.

1.2 Contribuição do autor

As principais contribuições deste trabalho são: o modelo de escalonamento de recursos utilizando rede bayesianas para o tratamento de incertezas; a modelagem de um *framework* para que outras heurísticas possam ser conectadas ao escalonador; a implementação de um módulo de configurações acoplado à interface gráfica de administrador de sistema; e o desenvolvimento de um protótipo que implementa o modelo de forma integrada à arquitetura ISAM.

Para avaliação da estratégia de escalonamento proposta, o autor implementou dois outros escalonadores que foram utilizados como referência para as avaliações. Durante este trabalho foi desenvolvida uma interface de comunicações entre o módulo de monitoramento do EXEHDA e o NWS.

Por fim, uma outra contribuição aconteceu no momento de avaliar o comportamento do escalonador em um cenário semelhante ao da computação pervasiva. O autor projetou e prototipou um gerador de cargas artificiais, baseado em uma distribuição de probabilidades exponencial.

1.3 Estrutura da dissertação

Este trabalho é composto por 8 capítulos. No capítulo 2 é apresentada a arquitetura do projeto ISAM e suas áreas de atuação em pesquisa. Uma revisão sobre temas de escalonamento relacionados com a computação pervasiva é apresentada no capítulo 3. O capítulo 4 é dedicado ao EXEHDA, o ambiente de execução do ISAM. A modelagem e a arquitetura do TiPS são apresentadas no capítulo 5. O protótipo implementado e os resultados obtidos são discutidos no capítulo 6. O capítulo 7 aborda os principais trabalhos relacionados ao TiPS. Por fim, no capítulo 8 é apresentada a conclusão do trabalho, bem como os trabalhos futuros.

2 ISAM, O ESCOPO DE PESQUISA

Em 1991, Weiser (1991) propôs uma nova visão da computação a qual chamou de computação ubíqua (*ubiquitous computing*). A essência desta visão era criação de ambientes dotados de capacidade de computação e de comunicação perfeitamente integrados com usuários. Na época em que esta idéia foi mencionada a tecnologia não havia avançado o suficiente para permitir a sua efetiva materialização.

Com o avanço da tecnologia, muitos dos elementos de hardware centrais para a computação ubíqua que não eram disponíveis em 1991 são agora oferecidos comercialmente no mercado: computadores de mão, computadores "de vestir", redes sem fio e dispositivos para sensoreamento e controle de equipamentos domésticos.

Recentemente, a IBM propôs o termo computação pervasiva (*pervasive computing*) com uma visão mais factível com a tecnologia atual. Autores como Satyanarayanan (2001) consideram a computação pervasiva como sinônimo da computação ubíqua. A visão do projeto ISAM (AUGUSTIN, 2004. 194p; YAMIN, 2004. 194p; ISAM, 2002) é diferente, e considera a computação pervasiva uma etapa a ser vencida para que a computação ubíqua (onipresente, invisível ao usuário) se torne uma realidade.

Neste início de década, diversos projetos surgiram para tentar solucionar as questões que apareceram com a computação pervasiva, alguns exemplos são: Aura [1], Endeavor [2], Oxygen [3] e Portalano [4]. Cada um destes projetos atacam um diferente conjunto de questões da computação pervasiva e uma mistura de objetivos de curto e longo prazo.

Neste capítulo serão apresentadas algumas considerações sobre a computação pervasiva, buscando caracterizar as questões e peculiaridades deste novo tipo de ambiente de computação. Nas seções seguintes são identificadas as contribuições da pesquisa nas áreas de sistemas distribuídos e de computação móvel, bem como as questões em aberto a serem resolvidas pela computação pervasiva.

2.1 Contribuições da pesquisa em áreas relacionadas

A computação pervasiva representa um grande passo na evolução de uma linha de trabalho que se inicia na década de 70. Dois diferentes passos anteriores nesta linha de evolução são os sistemas distribuídos e a computação móvel. Alguns dos problemas da computação pervasiva correspondem a problemas já identificados e

[1] http://www-2.cs.cmu.edu/~aura/
[2] http://www.cs.berkeley.edu/~randy/Endeavour/
[3] http://oxygen.lcs.mit.edu/
[4] http://portolano.cs.washington.edu/

estudados anteriormente. Em alguns destes casos, as soluções existentes se aplicam diretamente, em outros casos, os requisitos da computação pervasiva são suficientemente diferentes para que novas soluções sejam desenvolvidas. Existem também novos problemas introduzidos pela computação pervasiva que não apresentam mapeamento direto para os problemas estudados anteriormente. A seguir, detalha-se melhor esta questão.

2.1.1 Contribuições das pesquisas em sistemas distribuídos

A área de sistemas distribuídos surgiu da intersecção de computadores pessoais e redes locais. A pesquisa desenvolvida nesta linha criou uma série de áreas de pesquisa, bem como uma base de conhecimento importante para trabalhos envolvendo dois ou mais computadores conectados por uma rede, seja ela cabeada ou sem fio, móvel ou estática. Destas áreas de pesquisa, se sobressaem (COULORIS; DOLLIMORE; KINDBERG, 2001):

- comunicação remota, incluindo protocolos em camadas, chamada remota de procedimento, o uso de *timeouts*;
- tolerância a falhas, integrando transações atômicas, transações distribuídas e aninhadas, fechamento (*commit*) em duas fases;
- alta disponibilidade, incluindo controle de réplicas otimista e pessimista, execução de forma espelhada (BORG et al., 1989), e recuperação otimista (STROM; YEMINI, 1985);
- acesso a informações remotas, incluindo gerência de *cache*, sistemas de arquivos distribuídos, bancos de dados distribuídos (SATYANARAYANAN, 1990);
- segurança, considerando autenticação mútua e controle de privacidade com criptografia (NEEDHAM; SCHROEDER, 1978).

2.1.2 Contribuições das pesquisas em computação móvel

O surgimento das redes sem fio e o crescimento do número de equipamentos portáteis levaram para o surgimento de problemas que aparecem na construção de sistemas distribuídos com clientes móveis. Apesar de muitos dos princípios básicos da computação distribuída continuarem a ser aplicados, quatro questões chave forçaram o desenvolvimento de técnicas especializadas: variação imprevisível na qualidade da rede, baixa confiabilidade dos elementos móveis, limitações de recursos locais impostos pela necessidade de baixo peso e tamanho dos dispositivos, e a preocupação com o consumo de energia (SATYANARAYANAN, 1996).

A computação móvel é ainda um campo ativo de pesquisa. Os resultados atingidos até o momento podem ser agrupados nas áreas:

- redes móveis, considerando *Mobile IP* (BHAGWAT; TRIPATHI; PERKINS, 1995), protocolos ad-hoc (ROYER; TOH, 1999), e técnicas para melhoramento do desempenho do protocolo TCP para redes sem fio (BAKRE; BADRINATH, 1995; BREWER, 1998);

- acesso a informação móvel, incluindo operação desconectada (KISTLER; SATYANARAYANAN, 1992), acesso com largura de banda adaptativa à arquivos (MUMMERT; EBLING; SATYANARAYANAN, 1995), e controle seletivo de consistência de dados (TAIT; DUCHAMP, 1992);

- suporte para aplicações adaptativas, incluindo codificação por parte de *proxies* (FOX et al., 1996), e gerenciamento adaptativo de recursos (NOBLE et al., 1997);

- técnicas de redução do consumo de energia, adaptação para *energy-aware* (FLINN; SATYANARAYANAN, 1999), variação na velocidade de escalonamento de processos (WEISER et al., 1994), e gerenciamento de memória sensível a energia (LEBECK et al., 2000);

- sensibilidade de localização, incluindo sensoreamento (WANT et al., 1992; WARD; JONES; HOPPER, 1997) e comportamento *location-aware* (SCHILIT; THEIMER; WELCH, 1993; VOELKER, 1995).

2.2 Questões de pesquisa na computação pervasiva

Já que a locomoção é uma atividade comum no dia-a-dia, e que a computação pervasiva apresenta-se como uma forma de computação perfeitamente integrada com o usuário, é necessário que esta tecnologia suporte mobilidade física. Além das questões de pesquisa incorporadas da computação móvel, a computação pervasiva adiciona mais quatro pontos de pesquisa (SATYANARAYANAN, 2001), conforme é apresentado na figura 2.1.

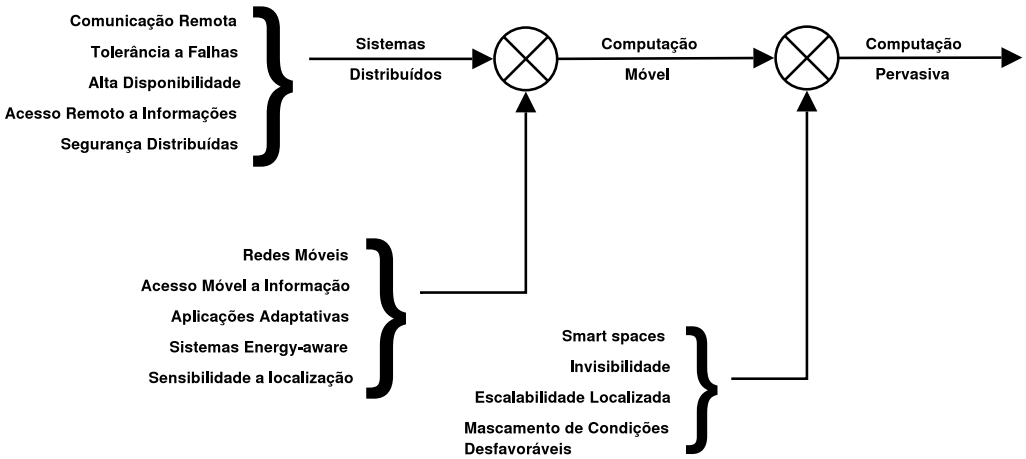

Figura 2.1: O relacionamento da computação pervasiva com a computação móvel e os sistemas distribuídos

2.2.1 Caracterizando a computação pervasiva

- *Smart spaces*

Um espaço pode ser uma área fechada, como uma sala ou corredor, ou uma área aberta bem definida. Pela inclusão de recursos de computação na infra-estrutura dos prédios, um *smart space* faz a união de dois mundos até então separados. Esta união permite que um dos mundos faça controle e sensoreamento do outro. Um exemplo simples é o controle de temperatura automático e nível de luminosidade baseado no perfil dos usuários de uma sala. A influência na outra direção também é possível, um software em um computador pode ter comportamento diferente conforme o local onde o equipamento se encontra no momento.

- **Invisibilidade**

  Na proposta da computação pervasiva está também o desaparecimento completo da consciência do usuário a respeito da computação presente. Se um ambiente de computação pervasiva atinge, continuamente, as expectativas dos usuários e raramente apresenta surpresas, ele permite interações em um nível quase que subconsciente (SATYANARAYANAN, 2001).

- **Escalabilidade localizada**

  Conforme os *smart spaces* crescem em sofisticação, a quantidade de interações entre o espaço do computador pessoal do usuário e seu entorno aumenta. Isto leva a implicações severas na largura de banda necessária e no consumo de energia para o usuário de rede sem fio. A presença de muitos usuários complica mais ainda este quadro. A escalabilidade portanto é um problema crítico na computação pervasiva. Trabalhos anteriores em escalabilidade ignoraram a distância física (SATYANARAYANAN, 2001), um servidor *web* pode atender o maior número possível de clientes, independente da localização física dos mesmos. A situação na computação pervasiva é bastante diferente, é necessário reduzir a densidade de interações conforme o usuário se afasta, caso contrário o sistema receberá muitas informações de distâncias longas, que são de pouca relevância. Apesar de usuários móveis distantes de seu ambiente de trabalho realizarem interações distantes com *sites* de interesse, a preponderância de suas interações deve ser local.

- **Mascaramento de condições desfavoráveis**

  A taxa de penetração da tecnologia da computação pervasiva na infra-estrutura computacional do dia-a-dia deverá variar consideravelmente dependendo de muitos fatores não técnicos, tais como organização estrutural, economia e modelos econômicos. A penetração uniforme ainda está longe de ser obtida. Neste interim, haverão grandes diferenças quanto a qualidade do serviço de diferentes ambientes, aquilo que estará disponível em uma sala de conferência bem equipada, em um escritório, ou uma sala de aula pode ser mais ou menos sofisticado do que em outros locais. Estas diferenças reduzem a perspectiva de invisibilidade da computação pervasiva.

  Uma forma de minimizar a quantidade de variações percebidas pelo usuário, é fazer com que seu computador pessoal faça compensações quando operando em ambientes menos favorecidos. Um exemplo simples é um sistema capaz de operação desconectada mascarar a falta de acesso a rede sem fio, devido a falta de cobertura em um ambiente.

2.2.2 Estratégia de adaptação

A adaptação se faz necessária quando existe uma discrepância significativa entre a capacidade e a demanda por um recurso. O recurso em questão pode ser a largura de banda de uma rede sem fio, energia, ciclos de computação, memória, e assim por diante. Existem três estratégias alternativas para adaptação na computação pervasiva (SATYANARAYANAN, 2001):

- um cliente pode dirigir suas aplicação para alterações de comportamento para que ela reduza a demanda por um recurso escasso. Esta alteração geralmente reduz a qualidade que o usuário percebe em uma aplicação. O Odyssey (FLINN; SATYANARAYANAN, 1999; NOBLE et al., 1997) é um exemplo de sistema que utiliza esta estratégia;

- um cliente pode solicitar que seja garantido um certo nível de um determinado recurso. Esta abordagem é tipicamente utilizada em sistemas QoS baseados em reserva (NAHRSTEDT; CHU; NARAYAN, 1999). Do ponto de vista do cliente, esta técnica pode aumentar a capacidade de demanda de um recurso escasso para atender a necessidade do cliente;

- um cliente pode solicitar uma ação corretiva para o usuário. Se o usuário seguir esta orientação, é possível que a capacidade do recurso fique adequada para atender a demanda. Um exemplo seria o sistema sugerir que um usuário de rede sem fio se desloque para outra região, menos congestionada.

Todas estas três estratégias são importantes na computação pervasiva. A existência de *smart spaces* sugere que alguns ambientes encontrados por um usuário seja capazes de aceitar reserva de recursos. Ao mesmo tempo, ambientes com condições desfavoráveis sugerem que o usuário não será atendido somente com uma estratégia de reservas, caso o ambiente esteja muito desfavorecido, o cliente pode optar pela redução na qualidade das aplicações. Ações corretivas aumentam as possibilidades de adaptação por envolver o usuário, e podem ser bastante úteis quando a redução de qualidade não é uma solução aceitável. Diversas questões ainda estão sem resposta (SATYANARAYANAN, 2001):

- a escolha da estratégia de adaptação mais adequada precisa levar em conta diversos fatores, incluindo a participação do usuário;

- a qualidade da execução da aplicação é de grande importância, estratégias baseadas em reserva podem ser utilizadas, porém os custos para garantia na qualidade dos serviços podem inviabilizar a execução;

- caso se assuma um sistema com reservas, estas terão de ser honradas e os conflitos deverão ser tratados. Precisa ser avaliado sob quais tipos de recursos o sistema de reserva pode ser empregado;

- as ações corretivas talvez tenham que ser evitadas, pois podem causar distúrbios na execução das aplicações, estes distúrbios poderão ser percebidos pelos usuários.

2.2.3 Consciência de contexto

Um sistema de computação pervasiva que pretende ser minimamente intrusivo precisa ter consciência de contexto (*context-aware*). Em outras palavras, ele precisa reconhecer o estado do usuário e do ambiente, bem como modificar seu comportamento baseando-se neste conhecimento. O contexto do usuário pode ser muito rico de informações, tais como localização física, estado fisiológico (temperatura do corpo, freqüência de batimentos cardíacos), estado emocional, história pessoal, padrões de comportamento diário, e etc. Se um assistente humano recebesse informações de um contexto deste tipo, ele tomaria decisões de forma pró-ativa, antecipando as necessidades dos usuários. Ao tomar estas decisões, o assistente não causaria distúrbios ao usuário em momentos inoportunos, a não ser em caso de emergências.

Um desafio chave é a obtenção de informações para que o sistema funcione de forma adequada com consciência de contexto. Em alguns casos, a informação desejada já pode ser parte do espaço computacional do usuário. Por exemplo, este espaço já pode possuir a agenda do usuário, listas de a fazeres, lista de contatos, etc. Informações mais dinâmicas precisam ser sensoreadas em tempo real a partir do ambiente do usuário. Exemplos deste tipo de informações incluem posição, orientação, identidades das pessoas próximas, objetos e ações locais observáveis, estado emocional e fisiológico.

A implementação de um sistema dotado de consciência de contexto necessita que muitas questões sejam resolvidas, como por exemplo:

- as formas de representação e armazenamento do contexto e como estas informações serão utilizadas junto ao sistemas e às aplicações;
- a periodicidade de atualização das informações de contexto deve ser definida e os custos de fazer estas atualizações devem ser considerados;
- os serviços mínimos necessários para que o sistema seja dotado de consciência de contexto precisam ser definidos, bem como estratégias de utilização dos históricos de comportamento;

2.3 Computação pervasiva na visão do projeto ISAM

Na visão do projeto ISAM (AUGUSTIN, 2004. 194p), a computação pervasiva será uma realidade anterior (futuro próximo) à computação ubíqua (futuro distante). Esta não contém a propriedade de invisibilidade do sistemas ubíquos, mas o acesso pervasivo ao ambiente computacional do usuário em escala global. As aplicações pervasivas são distribuídas, móveis, adaptativas ao contexto por natureza. Argumenta-se que a infra-estrutura de suporte à computação pervasiva pode ser construída a partir da integração das áreas de computação em grade, computação móvel, *internet* e *context-aware computing* (Figura 2.2).

Com esta visão foi construída a arquitetura ISAM, que fornece uma infra-estrutura para suporte ao desenvolvimento e execução de aplicações no ambiente pervasivo.

2.4 Arquitetura ISAM

O Projeto ISAM integra as contribuições das frentes de pesquisa em andamento. Uma resumida caracterização desta integração seria (vide figura 2.3):

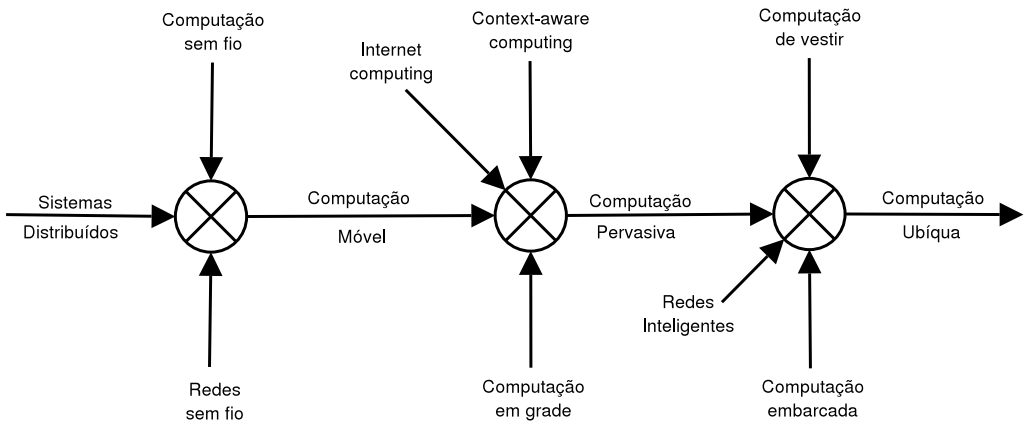

Figura 2.2: Linha evolutiva da mobilidade na visão do ISAM

- Holoparadigma: define o modelo e a linguagem de programação utilizados (BARBOSA, 2002. 213p);

- ISAMadapt: especifica as abstrações para expressar, em tempo de desenvolvimento, a adaptação ao contexto em aplicações móveis distribuídas direcionadas para a Computação Pervasiva. Dentre outras, ISAMadapt especifica abstrações para definir: adaptadores para os códigos das entidades de modelagem da aplicação; políticas de adaptação que orientam a tomada de decisão do *middleware*; e elementos de contextos para nortear os mecanismos adaptativos do EXEHDA (AUGUSTIN, 2004. 194p);

- EXEHDA: propõe a arquitetura dos mecanismos para coordenação, comunicação e adaptação na execução das aplicações, na perspectiva da semântica siga-me da Computação Pervasiva (YAMIN, 2004. 194p).

Desta forma, o Holoparadigma contempla, de forma intrínseca, questões de mobilidade lógica e distribuição. O ISAMadapt estende o Holoparadigma para prover suporte à mobilidade física e adaptação ao contexto no tocante à linguagem de programação. O código ISAMadapt é compilado para Java potencializando aspectos de portabilidade. A aplicação ISAMadapt é gerenciada pelo EXEHDA, que viabiliza um comportamento ativo e reativo na gerência das entidades de modelagem da aplicação. O *middleware* contempla o requisito de elevada escalabilidade, e suporta cooperação com as definições feitas em tempo de desenvolvimento, através do ISAMadapt, no momento de executar os procedimentos adaptativos. Estes procedimentos adaptativos têm como núcleo um mecanismo de gerência totalmente integrado ao controle da execução distribuída.

A Figura 2.4 apresenta uma visão geral da arquitetura de software ISAM. A representação da consciência do contexto nesta figura como um módulo virtual, tem por objetivo caracterizar sua importância na arquitetura, ressaltando sua presença na concepção de todos os outros componentes.

As aplicações desenvolvidas com o ISAMadapt são caracterizadas pelos aspectos de mobilidade lógica e física, de distribuição, de adaptação e de regência pela semântica siga-me. Com o intuito de minimizar o custo de especificar estes aspectos quando do desenvolvimento da aplicação, os mecanismos associados com a sua

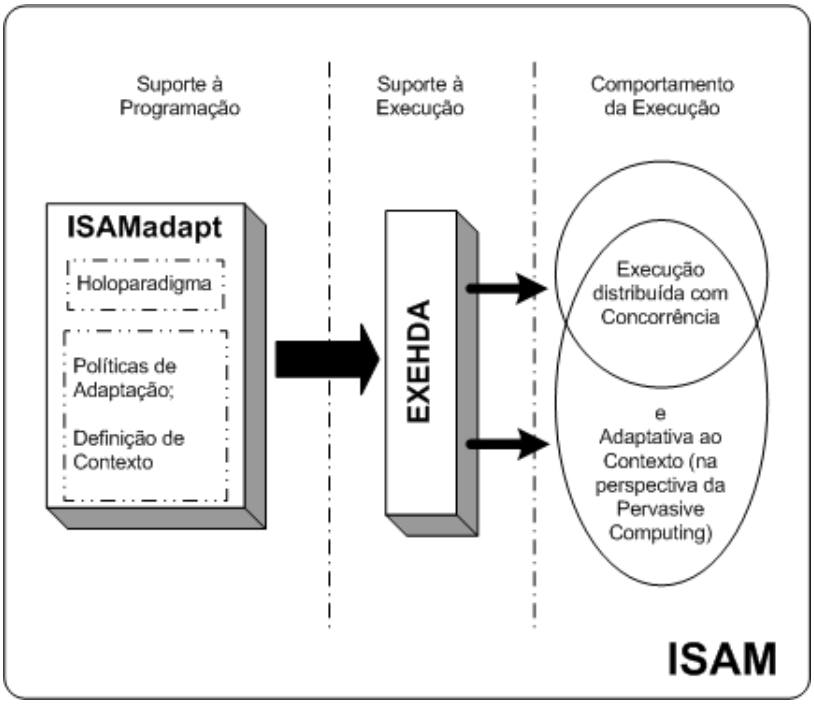

Figura 2.3: Visão geral do Projeto ISAM

gerência podem ser integrados ao ambiente de execução. Desta forma, a arquitetura ISAM, modelada com este objetivo, apresenta uma organização lógica em três camadas: (sup) camada de aplicação; (interm) camada de suporte e ambiente de execução; (inf) camada de sistemas básicos (vide Figura 2.4).

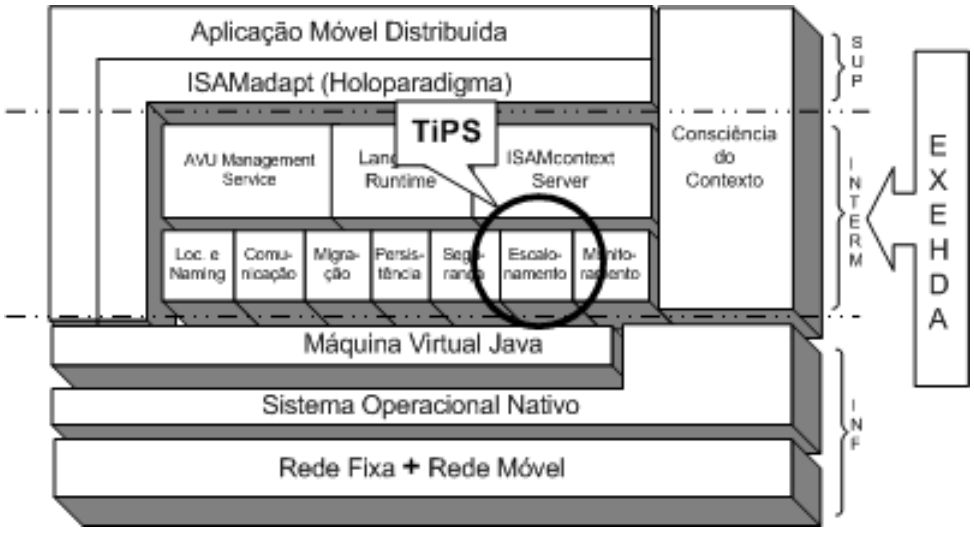

Figura 2.4: Visão geral da arquitetura ISAM

A **camada superior** (aplicação) oferece o paradigma (HoloParadigma) e a linguagem de programação (ISAMadapt), bem como a forma explícita de programar aplicações pervasivas (AUGUSTIN, 2004. 194p; AUGUSTIN et al., 2003, 2002).

A **camada *middleware*** (EXEHDA) oferece os mecanismos de suporte à execução da aplicação pervasiva e às estratégias de adaptação(YAMIN, 2004. 194p; YAMIN et al., 2003, 2002; SILVA, 2003; REAL et al., 2003). Esta camada é for-

mada por dois níveis:

O primeiro nível é composto por três módulos de serviço à aplicação: Acesso Pervasivo a Código e Dados, Reconhecimento de Contexto e o Ambiente de Execução da Aplicação. Suas funções são:

- o acesso pervasivo a dados e código compreende os componentes que disponibilizam o Ambiente Pervasivo (ISAMpe), incluindo Ambiente Virtual do Usuário (AVU), Ambiente Virtual da Aplicação (AVA), Base de Dados pervasiva das Aplicações (BDA). O Ambiente Virtual do Usuário compõe-se dos elementos que integram a interface de interação do usuário com o sistema. Este módulo é o responsável por implementar o suporte para que a aplicação que o usuário está executando em uma localização possa ser instanciada e continuada em outra localização sem descontinuidade, permitindo o uso de aplicações siga-me (*follow-me*) num ambiente pervasivo. O desafio da adaptabilidade para implementar o AVU é suportar os usuários em diferentes localizações, com diferentes sistemas de interação, que demandem diferentes formas de apresentação, dentro dos limites da mobilidade.

- o Serviço de Reconhecimento de Contexto é responsável por informar o estado dos elementos de contexto de interesse da aplicação e do próprio ambiente de execução.

- o Ambiente de Execução da Linguagem, por sua vez, é o componente-chave do gerenciamento da execução adaptativa na arquitetura ISAM.

No segundo nível da **camada intermediária** estão os serviços básicos do ambiente de execução ISAM que provêem a funcionalidade necessária para o primeiro nível e cobrem vários aspectos, tais como migração, mecanismos para deslocar um componente de software de uma localização física (equipamento) para outra; persistência, mecanismo para aumentar a disponibilidade e o desempenho do acesso aos dados; descoberta de recursos para dar suporte ao movimento dos dispositivos móveis e dos componentes entre diferentes células, mantendo a execução durante o deslocamento; comunicação, implementada através de multi-espaços de objetos distribuídos com suporte a tupla de natureza reativa; **escalonamento**, permite decidir o melhor nodo para criar os componentes da aplicação; monitoramento, sensores que fornecem informações sobre o ambiente de execução e a aplicação.

A **camada inferior** da arquitetura é composta pelos sistemas e linguagens nativas que integram o meio físico de execução. Por questões de portabilidade, nesta camada a plataforma base de implementação é a Máquina Virtual Java em suas diferentes versões: J2SE e J2ME. A arquitetura supõe a existência de uma rede pervasiva, que é móvel, infra-estruturada e que dê suporte de acesso global aos usuários.

Como visto, a arquitetura ISAM está organizada em camadas lógicas, com níveis diferenciados de abstração, e está direcionada para a busca da manutenibilidade da qualidade de serviços oferecida ao usuário móvel através do conceito de adaptação. Nesta, o sistema se adapta para fornecer qualidade dos serviços prestados, enquanto que a aplicação se adapta para atender a expectativa do usuário móvel, mantendo a funcionalidade da aplicação. Como o escopo de atuação do Projeto ISAM é amplo, a arquitetura proposta é abrangente e está sendo refinada à medida que seus subprojetos avançam.

Um dos sub-projetos inseridos no ISAM é o TiPS, que atua no segundo nível da camada intermediária tratando questões relativas ao escalonamento.

3 ESCALONAMENTO NA PERSPECTIVA DA COMPUTAÇÃO PERVASIVA

Sistemas paralelos e distribuídos são compostos por um conjunto de recursos heterogêneos que cooperam para resolver um determinado problema. Nesta cooperação muitas vezes observa-se que alguns recursos estão trabalhando e outros estão ociosos (BAUMGARTNER; WAH, 1991). Isto indica que um melhor uso poderia ser feito destes recursos, a área de escalonamento lida com as alternativas possíveis, que permitam um melhor aproveitamento da infra-estrutura computacional existente. Este capítulo irá tratar do tema escalonamento, serão apresentadas algumas técnicas e estudos relacionados, particularmente ao escalonamento na computação pervasiva.

3.1 O gerenciamento de carga

O problema de escalonamento possui cinco componentes: as tarefas (incluindo os eventos significativos relacionados a elas), o sistema distribuído, os requisitos de desempenho, o escalonamento e o escalonador. Os eventos das tarefas, as características do sistema e os requisitos de desempenho servem como entrada para o escalonador, ou o gerenciador de carga, e o mapeamento, ou escalonamento, é a saída.

Os dados de entrada relacionados ao sistema fazem referência a todas características do sistema que causam impacto no escalonamento, por exemplo a capacidade de processamento disponível em um determinado momento. Os eventos são mensagens relacionadas a carga de trabalho, estes eventos informam a chegada de novas tarefas, o encerramento do processamento de outras, ou apenas o estado atual de alguma delas.

O sistema distribuído e a carga de trabalho em conjunto, constituem o ambiente sobre o qual o escalonador deve atuar para atingir os requisitos de desempenho. Com objetivo de aumentar a eficiência, muitos escalonadores mantêm uma representação do estado do ambiente, esta representação é alterada sempre que alguma nova informação a respeito do ambiente, ou da carga de trabalho é recebida. Este tipo de informação é chamada de métricas do ambiente, e são adquiridas por sensores especializados.

Os requisitos de desempenho especificam quais objetivos devem ser atingidos pelo escalonador. Isso inclui a maximização do *throughput*, minimização do tempo de execução, etc.

O escalonamento é o mapeamento das tarefas aos recursos, da forma como foi

gerado pelo escalonador. Pode ser uma lista ordenada de pares (tarefas, recursos) gerada previamente a execução, desde que as características do sistema e das tarefas sejam previamente conhecidas, ou pode ser gerada dinamicamente em tempo de execução por um conjunto de regras. Estas regras especificam ações de correção para uma melhor distribuição da carga de trabalho e para que os requisitos de desempenho sejam atingidos.

O escalonador é o responsável por preparar o escalonamento, baseado no que é conhecido do estado corrente do ambiente e dos requisitos de desempenho. Para conseguir atender estes requisitos, o escalonador necessita de um modelo interno de execução, que reflita o comportamento do ambiente e as relações entre a carga de trabalho e as entidades do ambiente distribuído. Este modelo de execução é usado para gerar estimativas de estados futuros, e é usado como entrada para o mecanismo de tomada de decisão do escalonador. Sendo este modelo uma simplificação do problema real, ele está sujeito a estimativas imprecisas, estas imprecisões devem ser consideradas e tratadas para que a eficiência não seja comprometida.

3.2 Políticas de escalonamento estáticas e dinâmicas

O grau de flexibilidade do escalonador depende da demanda de requisitos de desempenho e da complexidade do ambiente que deve ser gerenciado. Weiss (1999), Russell & Norvig (1995) sugerem que o ambiente seja classificado de acordo com as seguintes propriedades:

- **acessível / inacessível** – um ambiente é acessível se os sensores do escalonador oferecem informações completas, precisas e atualizadas a respeito do estado do ambiente. Caso os sensores não ofereçam informações suficientes para que se tenha o completo conhecimento do estado do ambiente, então o ambiente é classificado como inacessível, ou parcialmente observável;

- **determinístico / não determinístico** – um ambiente é determinístico se o próximo estado pode ser determinado a partir do estado atual e da ação selecionada pelo escalonador. Em um ambiente determinístico cada ação possui um efeito garantido, conhecido pelo escalonador. Em ambientes não determinísticos a mesma ação, se aplicada duas vezes, em circunstâncias aparentemente idênticas podem apresentar efeitos completamente diferentes, e mais importante, o efeito desejado pode não ocorrer;

- **estático / dinâmico** – um ambiente estático se mantém inalterado até que seja disparada alguma ação pelo escalonador. Caso o ambiente se altere de forma independente e sem o controle do escalonador, então ele é dito dinâmico.

Os ambientes em que escalonadores mais complexos são necessários, em geral são: inacessíveis, não determinísticos e dinâmicos.

- **inacessível** – o ambiente pode ser tão complexo que o escalonador não pode considerar todos os aspectos relevantes no modelo de execução interno. Algumas das entidades e das relações precisam ser negligenciadas ou resumidas para que se obtenha um modelo de execução gerenciável. Esta simplificação impede que o escalonador tenha uma imagem completa do estado do ambiente.

Além disto, a informação disponível para o escalonador, pode muitas vezes ser imprecisa, pois pode ser muito custoso, ou mesmo inviável, obter informações precisas do estado do ambiente. Outro fator importante, está relacionado a idade da informação, como o ambiente está sempre se alterando, em geral as medições disponíveis já são de um período anterior ao corrente;

- **não determinístico** – o ambiente se altera enquanto o escalonador está processando e enquanto as ações estão sendo processadas. O escalonador não pode assumir uma certeza absoluta com relação as conseqüências das ações tomadas. Observe-se que se o ambiente é inacessível, ele pode assumir um comportamento não determinístico, já que o agente não pode manter um conhecimento de aspectos inacessíveis;

- **dinâmico** – em um sistema distribuído compartilhado, muitas aplicações podem ser disparadas por vários usuários, estas aplicações exibem padrões de carga e de comunicação altamente variáveis e irregulares. Além disto, estes usuários podem, independentemente do escalonador, disparar e encerrar aplicações a qualquer instante.

Políticas de escalonamento estáticas geram o escalonamento antes do tempo de execução, baseando-se nas propriedades do sistema e nos requisitos das tarefas. As políticas dinâmicas, por outro lado, geram o escalonamento em tempo de execução, usando um conjunto de regras para especificar quais as ações de correção possíveis. As políticas dinâmicas tem condições de oferecer melhores resultados que as estáticas em ambientes dinâmicos, pois podem explorar melhor as flutuações do estado do sistema (LU; LAU, 1998). Dentre as políticas dinâmicas, três diferentes abordagens podem ser consideradas:

- aquelas que não consideram o estado do ambiente, decidindo de forma cega;

- aquelas que utilizam as informações do estado do ambiente como entrada para um conjunto de regras, esperando tomar decisões acertadas;

- aquelas que usam os dados de estado do ambiente para modificar suas próprias regras, ou seja, o modelo de execução é alterado em tempo de execução com o objetivo de melhor representar o ambiente. Estas são em geral classificadas como políticas adaptativas.

Estratégias estáticas e dinâmicas cegas em geral são inadequadas para sistemas compartilhados, já que as variações no comportamento do ambiente são ignoradas. Estratégias que regularmente fazem medições do estado do ambiente podem ser mais adequadas, pois permitem que o escalonador reaja a flutuações no estado do ambiente. Estas são referenciadas como estratégias dinâmicas de escalonamento baseadas em sensores, pois o escalonador coleta dados do estado do ambiente através de seus sensores. Foi mostrado que as políticas adaptativas oferecem bom desempenho quando o estado do sistema se altera em tempo de execução (ZHOU, 1988; SHIVARATRI; KRÜGER; SINGHAL, 1992), e que diferentes estratégias de gerenciamento de carga se encaixam melhor com diferentes padrões de carga (BECKER; WALDMANN, 1995). Estes resultados sugerem que para atingir os requisitos de

desempenho e para tratar de forma efetiva as propriedades do ambiente, o escalonador deve utilizar uma mecanismo de tomada de decisão altamente dinâmico, e até mesmo adaptativo.

Wolfgang Becker & Gerlinde Waldmann (1995) identificam três oportunidades para escalonamento adaptativo:

- **correção de perfil e da predição de carga** – abordagens sofisticadas possibilitam a exploração de predição de carga de trabalho, baseando-se no perfil da aplicação. O desafio para o escalonador é gerar ou melhorar as predições sobre o futuro próximo a partir do perfil da aplicação. A partir da comparação do resultado esperado, com o comportamento real da aplicação, a exatidão e a importância destas predições podem ser avaliadas e corrigidas;

- **determinação de fatores difusos no modelo de execução** – mecanismos de tomada de decisão e o gerenciamento de informações em tempo de execução incorrem em sobrecarga, portanto devem ser baseados em modelos simples de execução. Efeitos colaterais importantes são muitas vezes negligenciados, e correlações entre os itens considerados e os resultados desejados não são fortes o suficiente. Com o uso de realimentação, o escalonador pode aumentar o conhecimento sobre o comportamento do sistema e pode corrigir o modelo de execução inicial;

- **controle da relação entre sobrecarga e ganho** – o escalonamento dinâmico incorre em sobrecarga de gerenciamento do paralelismo, de comunicação e de computação. Abordagens adaptativas devem minimizar os esforços para o escalonamento e automaticamente otimizar as relações de custo/benefício.

Uma abordagem comum para mapeamento sub-ótimo incorre no uso de heurísticas. Escalonadores heurísticos fazem uso de parâmetros especiais que afetam o desempenho do sistema de forma indireta. Estes parâmetros geralmente introduzem pouco custo computacional ao escalonador, mas é difícil garantir, para diferentes condições, uma relação direta entre a heurística empregada e o resultado obtido. Uma abordagem possível é apresentada por Casavant & Kuhl (1988). O objetivo é de minimizar o congestionamento (número de canais lógicos mapeados em cada ligação física) e a dilatação (*dilation*) (número de ligações físicas que um canal lógico precisa percorrer, ou seja, a distância entre dois vizinhos lógicos) e equalizar a carga entre os processadores. Duas estratégias são possíveis:

- instanciar processos prontos para executar de forma concorrente em processadores diferentes, objetivando a maximização do grau de paralelismo;

- instanciar tarefas que se comunicam freqüentemente em um mesmo processador, para explorar a localidade.

3.3 Nível de complexidade

Uma questão importante é de determinar qual o nível apropriado de complexidade para a política de gerenciamento da carga. A literatura registra casos em que políticas relativamente simples podem oferecem ganhos substanciais, enquanto que políticas mais complexas não conseguem oferecer melhorias (CASAVANT; KUHL,

1988b; EAGER; LAZOWSKA; ZAHORJAN, 1986; ZHOU, 1988). Políticas complexas, muitas vezes, dependem de informações detalhadas sobre o estado do sistema e o comportamento da aplicação. Além do custo para obtenção destas informações ser, muitas vezes, alto, alguns valores não podem ser precisamente conhecidos. Políticas mais complexas estão atreladas em geral a uma negociação mais complexa entre os nodos, o que pode elevar os custos de comunicação. Outra questão é a instabilidade, políticas mais complexas podem reagir a variações muito pequenas, levando o sistema a um estado de instabilidade, no qual a maior parte do tempo é consumido em gerenciamento. Segundo Derek Eager *et al.* (1986):

- distribuições extremamente simplificadas de carga, levam a bons ganhos de desempenho em relação aos casos em que não há troca de carga;
- políticas complexas, que tentam encontrar a melhor alternativa, não oferecem melhorias significativas.

Os experimentos realizados por Derek Eager *et al.* (1986) não consideram interdependência entre as tarefas e foram realizados em sistemas distribuídos pequenos, homogêneos e dedicados. Tarefas paralelas freqüentemente exibem dependências de dados, caso esta característica não seja levada em consideração, o bom desempenho do escalonador fica dificultado.

Wolfgang Becker & Gerlinde Waldmann (1995) propõe uma abordagem hierárquica para o gerenciamento de carga com uma política um tanto sofisticada. Esta política considera não apenas demanda por processador e carga por processador, mas também os custos de comunicação. Becker argumenta que um gerenciamento de carga sofisticado apresenta melhores resultados do que simplesmente a entrega de carga para qualquer nodo, de forma indiscriminada. No entanto, ele conclui que o gerenciamento de carga complexo só oferece ganhos se for possível manter a coleta de dados e o mecanismo de tomada de decisão dentro de um tempo aceitável. Uma estratégia de adaptação de complexidade é proposta, a qual reduz o nível de complexidade se o escalonador ficar muito sobrecarregado, ou incapaz de tomar as decisões requisitadas dentro do tempo necessário.

A complexidade de uma política de escalonamento é determinada pela riqueza do modelo de execução considerado. Um ambiente distribuído e compartilhado é, em geral, um ambiente complexo, com a carga de trabalho alocada em cada recurso variando significativamente devido as ações dos usuários. Se a aplicação a ser escalonada exibe um padrão de requisitos de recursos muito irregular e imprevisível, então provavelmente seja necessário um modelo de execução relativamente complexo, que permita que o escalonado atinja os índices de desempenho desejados.

3.4 Estratégias de escalonamento determinísticas e estocásticas

As estratégias de escalonamento podem ser classificadas como determinísticas ou estocásticas, conforme as regras que governam suas decisões (XU; LAU, 1997). Métodos determinísticos tomam decisões de acordo com um conjunto de regras que se utilizam de valores determinísticos, ou diretos, para parametrizar o modelo de execução do escalonador. Os métodos estocásticos tomam as decisões utilizando

mecanismos probabilísticos, como uma tentativa de maximizar, com alta probabilidade, as chances de atingir as metas de desempenho. Estes métodos em geral usam modelos de execução estocásticos que são parametrizados com valores probabilísticos ou estocásticos, ao invés de utilizar valores determinísticos. Os valores estocásticos são especialmente adequados para quantificar as características do ambiente que sofre variações com o tempo e características cujo valor exato não é conhecido. Ao invés de quantificar estas variáveis com valores exatos, e pontuais, uma probabilidade é aplicada a cada um dos possíveis valores (BERMAN et al., 1996; SCHOPF; BERMAN, 1999).

Ryou & Juang (1994) defendem que os algoritmos determinísticos sempre conseguem um melhor desempenho que os algoritmos estocásticos, mas eles consideram apenas abordagens probabilísticas que não utilizam as informações de estado do ambiente. Peter Kok Keong Loh *et al.* (1996) argumenta que os modelos estocásticos são mais realistas, já que eles capturam as características variáveis no tempo da aplicação.

Duas estratégias estocásticas bem conhecidas são a alocação aleatória e o Automata de Aprendizado Estocástico (*Stochastic Learning Automata*). Stankovic (1985) propõe um escalonador baseado na teoria de decisão bayesiana, o qual aprende os parâmetros estocásticos durante a execução e consegue, com sucesso gerenciar a carga de sistemas distribuídos pequenos. Jennifer Schopf & Francine Berman (1999) propõem um escalonador em nível de aplicação baseado em um modelo de execução estocástico. Eles consideram que a modelagem estocástica é mais adequada que a modelagem determinística, principalmente nos casos em que as alterações no ambiente não estão completamente sobre o controle do escalonador.

O método *simulated annealing* é uma otimização estocástica física que pode ser usada para tratar do problema de gerenciamento de carga. Este método simula os movimentos aleatórios de um conjunto de átomos vibrando em um processo de resfriamento. O objetivo é atingir uma configuração ótima dos átomos em baixa temperatura. Quando aplicada ao gerenciamento de carga, a configuração corresponde ao estado do sistema, e a configuração final, ao resultado final após a ação do escalonador, é esperado que este estado seja balanceado em relação a distribuição de carga (XU; LAU, 1997).

Um modelo estocástico do ambiente deve ser mais adequado que um modelo determinístico quando o estado do sistema distribuído está sujeito a variações no tempo, e os estados futuros não podem ser precisamente previstos. Nos casos em que as conseqüências das ações disponíveis para o escalonador não são precisamente conhecidas, estas conseqüências também podem ser modeladas de forma estocástica.

3.5 Políticas de escalonamento centralizadas e distribuídas

As políticas de escalonamento podem ser centralizadas, distribuídas ou alguma forma híbrida entre os dois. Abordagens centralizadas empregam um único elemento para coleta de informações de estado e para a tomada de decisão. Wolfgang Becker & Jörg Zedelmayr (1994) , de um ponto de vista lógico, consideram que abordagens centralizadas são ótimas se:

- as informações de estado são coerentes no elemento central e não precisam estar replicadas em nodos do sistema, o que causa tráfego de mensagens adicional;

sistemas em que cada nodo mantém uma imagem local do estado do sistema, em geral, tendem a possuir diferenças entre as imagens dos nodos;

- o conhecimento global de todo estado do sistema, assim como o conhecimento do progresso do processamento das tarefas e das relações entre elas podem ser explorados, evitando decisões que levam o sistema a obter um desempenho pior que o esperado. O conhecimento de informações parciais do sistema pode levar a decisões que diminuam o desempenho das aplicações (JACQMOT; MILGROM, 1993).

Abordagens distribuídas dividem o gerenciamento de informações e a tomada de decisão entre os nodos do sistema, em geral empregando um elemento do escalonador por nodo. Cada elemento mantém informações sobre o estado de um conjunto de vizinhos. Esta abordagem é escalável, pois o aumento no número de nodos não aumenta a quantidade de informação que cada nodo deve gerenciar. As principais desvantagens das abordagens distribuídas estão relacionadas com o fato de que cada elemento do escalonador possui apenas uma visão parcial de todo o ambiente, e que elementos ligados a outros nodos podem ter uma visão diferente do estado do ambiente em função da idade das informações. Isto pode levar a decisões contraditórias que podem levar a uma redução no desempenho global.

Apesar de alguns autores afirmarem que as políticas centralizadas atingem melhores resultados, são mais simples e mais efetivas que as distribuídas (ZHOU, 1988), e que são escaláveis (THEIMER; LANTZ, 1989), a grande maioria concorda que a política de gerenciamento de carga deve possuir algum grau de distribuição para evitar gargalos, e portanto ser escalável (WILLEBEEK-LEMAIR; REEVES, 1993; KREMIEN; KRAMER, 1992; FABERO et al., 1996; SHIVARATRI; KRÜGER; SINGHAL, 1992; OZDEN; SILBERSCHATZ, 1993; LULING; MONIEN, 1993). O gerenciamento de carga centralizado não possui desvantagens lógicas, mas não é escalável, pois tende a causar sobrecarga de processamento e comunicação conforme a atividade e o tamanho do sistema crescem. Acima de algum limite, as abordagens centralizadas não conseguem aumentar o *throughput* geral do sistema.

O escalonamento implica em pelo menos duas atividades que requerem comunicação entre os nodos do sistema: aquisição de informações do ambiente e transferência de tarefas. A necessidade de cada nodo interagir com todos outros nodos poderá introduzir custos elevados de comunicação. Tal custo se potencializa na perspectiva de sistema distribuídos com elevada escalabilidade.

Para reduzir o gargalho de comunicação e os atrasos por tempo de processamento no elemento escalonador central, muitas abordagens particionam o sistema em conjuntos de nodos, chamados de domínios. Um nodo troca informações e tarefas apenas com membros do mesmo domínio. Em alguns casos, o domínio de um nodo é restringido apenas aos seus vizinhos (lógicos ou físicos) e este domínio é estático ao longo de toda execução. Os algoritmos que se utilizam desta estratégia são chamados de *nearest-neighbour*. Os algoritmos baseados neste princípio são iterativos por natureza, portanto ficam sucessivamente fazendo distribuições de carga locais ao domínio, o progresso ocorre na direção de uma distribuição global uniforme de carga, já que os domínios se sobrepõem (LULING; MONIEN, 1993; XU; LAU, 1997). Uma desvantagem destes esquemas, é que em geral eles são lentos para a distribuição de carga entre os domínios quando uma sub-região do sistema se torna

abruptamente sobrecarregada (LULING; MONIEN; RAMME, 1991; SCHEURER; SCHEURER; KROPF, 1995).

Ryou & Juang (1994) e Suen & Wong (1992) propõem uma abordagem que não considera vizinhança física, mas reduz o número de nodos para os quais um nodo deve enviar atualizações sobre a informação de carga. Este conjunto de nodos é referenciado como o conjunto de envio (*sending set*) do processador. Com o uso de conjuntos de envio balanceados minimiza-se a tráfego de mensagens com informação de carga, e a distribuição pode ocorrer entre qualquer par de nodos, através de negociação direta, ou por um nodo árbitro que possui conhecimento das cargas dos nodos daquele conjunto.

Yi-Chieh Chang & Kang Shin (1995) propõe um método baseado em domínios estáticos (*buddy sets*) e listas de preferências. A lista de preferências de cada processador indica com quais nodos ele deve trocar tarefas e informações de carga. Cada nodo deve selecionar nodos com pouca carga de processamento (*underloaded*) para trocar tarefas, respeitando a ordem de sua lista de preferências. Os nodos aparecem em várias listas de preferências, em diferentes ordens (tentando manter uma relação com a distância física), o que reduz a possibilidade de dois ou mais nodos sobrecarregados selecionarem um mesmo nodo pouco carregado para fazer transferência de carga.

Uma abordagem alternativa é de aplicar um gerenciador de carga centralizado dentro de cada domínio. Dentro do domínio podem ser empregados esforços bastante rígidos em manter a distribuição de carga, enquanto que entre os domínios pode-se fazer menos exigência com relação ao gerenciamento da carga, o que leva a menos interação e transferência de carga entre os domínios. Esta abordagem é chamada de *distributed clustering* (BECKER, 1995; BECKER; WALDMANN, 1994, 1995; OZDEN; SILBERSCHATZ, 1993; BECKER; ZEDELMAYR, 1994). Esta abordagem pode ser re-aplicada de forma hierárquica, formando agregados de agregados, ou o escalonamento entre os agregados pode ser realizado diretamente apenas entre os vizinhos. A hierarquização tende a provocar sobrecarga nos gerenciadores de mais alto nível, enquanto que a interação entre vizinhos necessita de dois mecanismos de gerenciamento de carga (entre os vizinhos, e dentro de cada domínio), e pode mostrar decisões contra-produtivas de forma similar a abordagem completamente distribuída, embora em uma escala menor. Já foi mostrado que a abordagem utilizada em agregados pode obter bons resultados, pois são capazes de explorar algumas vantagens de estratégias centralizadas e também são escaláveis.

O tamanho dos domínios deve ser cuidadosamente definido. Se for muito grande, vai estar sujeito aos mesmos problemas dos esquemas centralizados. Se for muito pequeno os problemas relacionados às abordagens totalmente distribuídas deverão se manifestar. Uma solução é de adaptar dinamicamente o tamanho do domínio, caso o escalonador de um dado domínio fique sobrecarregado e incapaz de tomar decisões em um tempo aceitável. Neste caso, este domínio deve ser subdividido em um ou mais domínios, cada um com um número menor de elementos. Por outro lado, se o escalonador ficar com uma carga muito baixa, ele pode procurar em seus vizinhos por um que deva ser subdividido, incorporando novos nodos ao seu domínio. A subdivisão do domínio quando o sistema está em sobrecarga, leva a *overheads* de comunicação e processamento, consumindo recursos exatamente no momento em que eles são necessários. Portanto, devem ser consideradas algumas alternativas. Uma alternativa é o escalonador trocar para uma estratégia mais simples e rápida, caso

o aumento de carga no sistema for supostamente de curta duração, voltando para a estratégia original assim que a carga voltar ao normal, ou caso a sobrecarga seja de longa duração adotar a estratégia de subdivisão do domínio. A incorporação de nodos a domínios pouco carregados não é crítica, pois emprega os recursos quando eles estão pouco carregados.

Tim Roughgarden (2001) propõe alternativas para o uso de um sistema compartilhado, no qual o escalonador não possui total controle sobre as tarefas que estão sendo processadas nos nodos. Nesta proposta são utilizadas estratégias utilizadas em jogos, os usuários que possuem controle sobre os equipamentos são considerados "egoístas". É apresentada uma abordagem híbrida de controle, incluindo uma unidade central gerenciadora de tarefas, que opera de forma independente do "controle egoísta" imposto pelos usuários do sistema.

A escolha entre uma arquitetura centralizada, distribuída ou híbrida para o escalonador, depende dos objetivos do mesmo, do tamanho e da atividade geral do sistema. Se a confiabilidade do sistema é de relevância central, as arquiteturas centralizadas devem ser evitadas, já que a falha do escalonador central pode comprometer todo o desempenho do sistema. Se a abordagem a ser adotada deve ser escalável em termos de tamanho e atividade do sistema, então deve ser adotada uma arquitetura distribuída, já que as abordagens centralizadas ficam comprometidas com o aumento do tamanho do sistema.

3.6 Componentes da política de escalonamento

As políticas de escalonamento freqüentemente são descritas em termos de quatro componentes e do mecanismo de transferência utilizado para movimentação das tarefas no sistema distribuído (SHIVARATRI; KRÜGER; SINGHAL, 1992).

- política de informação – um conjunto de regras para decidir quando, de onde (de qual recurso ou tarefa) e qual informação deve ser coletada sobre o estado do ambiente;
- política de transferência – um conjunto de regras para determinar se um recurso está em um estado propício para participar de uma transferência de carga;
- política de localização – um conjunto de regras utilizado para localizar os diversos elementos envolvidos em uma dada ação, por exemplo, transferência de carga;
- política de seleção – um conjunto de regras para decidir qual trabalho deve ser transferido entre os elementos envolvidos na ação, identificados pela política de localização.

3.6.1 Política de informação

A utilidade da política de escalonamento é altamente dependente da qualidade da medição e predição de carga (WATTS; TAYLOR, 1998). A política de informação determina quando, onde e quais informações devem ser coletadas através dos sensores que o escalonador dispõe. As políticas de informação podem ser classificadas em três categorias, considerando-se também alternativas híbridas entre elas:

- políticas dirigidas pela demanda – as informações sobre o estado do ambiente somente são coletadas no momento em que são necessárias para uma tomada de decisão. Podem ser utilizados em conjunto com esta política, dois mecanismos: o de busca (*probing*) e o de oferta (*bidding*). No mecanismo de busca, sempre que um nodo tem interesse em participar de uma transferência de carga, ele busca por parceiros, fazendo uma seleção seguida de um teste para verificar se este pode participar da transferência, esta operação é repetida até que seja encontrado um parceiro adequado, ou um limite no número de testes seja excedido. No método por oferta uma requisição por ofertas é feita para um grupo de nodos e as ofertas são recebidas por aqueles interessados em participar da transferência de carga. As ofertas são avaliadas e um parceiro adequado é escolhido (CASAVANT; KUHL, 1988b; CHANG; SHIN, 1995). Estes métodos não se utilizam de nenhuma informação histórica sobre as decisões anteriores de escalonamento, ou o estado de outros nodos, portanto não são inseridos *overheads* neste sentido. Por outro lado, a informação precisa ser coletada sempre que uma tarefa precisa ser transferida, o que introduz atrasos adicionais para a conclusão da operação;

- políticas periódicas – nesta política as informações são coletadas periodicamente. Um *overhead* fixo é imposto ao sistema em função da coleta e da manutenção das informações do sistema, independentemente delas estarem sendo ou não utilizadas. Por outro lado, não existe o atraso de coleta imposto pela política controlada pela demanda, no momento da tomada de decisão. A imagem do sistema em um dado momento pode não ser fiel a realidade, em função da própria periodicidade das medições, e também dos atrasos impostos pela rede de comunicações. Este fenômeno é chamado de *aging*, e está intimamente relacionado ao período de coleta das informações. O aumento da freqüência de coleta causa um aumento no *overhead* imposto pelo escalonador ao sistema, e não resolve o problema dos atrasos provocados pela rede de comunicações. Esta abordagem, se aplicada a políticas distribuídas, pode fazer com que as imagens do sistema se tornem diferentes em cada um dos nodos. Este efeito pode levar a complicações para a política de localização.

- políticas dirigidas por mudanças de estado – dados são coletados sempre que ocorrem alterações significativas. A maioria dos algoritmos controlados por mudança de estado classificam os estados dos recursos em n possibilidades. Considerando L a carga de trabalho e T_x um limiar, pode-se especificar uma política de três estados (KREMIEN; KRAMER; MAGEE, 1993; WILLEBEEK-LEMAIR; REEVES, 1993; NI; XU; GENDREAU, 1985; LU; LAU, 1998):

$$\begin{array}{cl} L < T_1 & \text{Nodo com baixa carga} \\ T_1 < L < T_2 & \text{Nodo com carga média} \\ L > T_2 & \text{Nodo sobrecarregado} \end{array}$$

O escalonador deve dispor de sensores instalados nos nodos, para fazer a coleta de informações. Estes sensores precisam ser projetados e implementados de forma a introduzir o mínimo de *overhead* nos nodos, minimizando o nível de intrusão, e ainda assim mantendo um bom nível de qualidade na informação fornecida. Os sensores podem ser classificados e dirigidos por eventos e por tempo. Os sensores dirigidos

por eventos são ativados pela ocorrência de eventos em particular, enquanto que os dirigidos por tempo fazem amostragens periódicas. As políticas dirigidas pela demanda e por alterações de estado fazem uso natural de sensores dirigidos por eventos, enquanto que as políticas periódicas se adequam mais aos sensores dirigidos por tempo.

Uma questão chave é encontrar as métricas adequadas para descrever a carga corrente do recurso. Uma boa métrica de carga deve (SHIVARATRI; KRÜGER; SINGHAL, 1992; NI; XU; GENDREAU, 1985; FERRARI; ZHOU, 1987):

- estar bem relacionada com os tempos de resposta das tarefas, já que é utilizada para predizer o desempenho de uma tarefa caso ela seja executada em um recurso;
- ser útil para a predição de carga em um futuro próximo, já que o tempo de resposta da tarefa deverá ser afetado mais pela carga futura do que pela presente;
- ser relativamente estável, pequenas variações na carga devem ser descartadas;
- não oferecer custos computacionais significativos.

Um conjunto de métricas de carga tem sido apresentado na literatura: tamanho da fila de CPU, utilização da CPU, tempo de resposta normalizado, tamanho da fila de I/O, utilização de memória, taxa de troca de contexto, taxa de chamadas de sistema, etc (KUNZ, 1991; FERRARI; ZHOU, 1987). Foi observado que uma tarefa em um nodo está propensa a solicitar demanda de um conjunto recursos (por exemplo: CPU, memória, disco), portanto pode ser importante definir a carga não apenas baseada em um único recurso do nodo, mas como uma coleção de recursos. Domenico Ferrari & Songnian Zhou (1988) propõe uma combinação linear de tamanho de fila de recursos como uma métrica de carga. Se uma tarefa solicita s_j segundos de um recurso r_j, e o tamanho da fila daquele recurso é q_j, então a métrica de carga deve ser calculada conforme a equação (3.1). Sendo N o número de diferentes recursos.

$$l = \sum_{j=1}^{N} (S_j * q_j) \tag{3.1}$$

Este índice de carga é dependente da tarefa já que ele considera a demanda de cada uma delas. Por outro lado, a necessidade de conhecer previamente a demanda da tarefa pode ser uma pré-condição difícil de ser atingida em muitos casos. Ferrari estudou também uma outra métrica baseada na combinação linear do tamanho das filas de diversos recursos. Os resultados apresentados por ele mostram que as diferenças de desempenho entre os casos que somente utilizaram métricas baseadas no tamanho da fila da CPU, e os casos em que I/O e contenção de memória foram também considerados, não foram significativas, sugerindo que a CPU é o recurso predominante no nodos. Kunz (1991) obteve resultados semelhantes fazendo experiências com métricas de carga de uma só dimensão (tamanho da fila de CPU, memória disponível, taxa de troca de contexto, etc) e combinações lineares destas métricas. Estes resultados e os apresentados em (NI; XU; GENDREAU, 1985; HARCHOL-BALTER; DOWNEY, 1997; ZOMAYA; CLEMENTS; OLARIU, 1998)

sugerem que o tamanho da fila de CPU é umas das métricas mais adequadas e também que não necessário um valor muito preciso desta métrica. Domenico Ferrari & Songnian Zhou (1988) utilizou uma média de tamanho de fila de CPU amortecida exponencialmente sobre uma janela de tempo, ao invés de utilizar o valor instantâneo, para eliminar as trocas de alta freqüência na medição.

3.6.2 Política de transferência

A política de transferência determina se um recurso está em um estado adequado para participar de uma transferência de carga, seja como fornecedor ou como recebedor. A maioria das políticas de transferência são baseadas em limiares (*threshold based*) ou são baseadas em valores relativos de carga. As políticas baseadas em limiar classificam um recurso como fornecedor caso a métrica de carga exceda um limiar T_s, ou como recebedor caso a métrica indique um valor abaixo do limiar T_r (LU; LAU, 1998; SHIVARATRI; KRÜGER; SINGHAL, 1992). A escolha destes limiares é essencial para o desempenho do sistema, o melhor valor de limiar depende da carga do ambiente e do custo de transferência de carga. Em situações de baixa carga e baixo custo de transferência, os limiares devem favorecer as transferências. Em situações de alta carga e alto custo de transferência, a execução remota deve ser evitada. Apesar de Derek Eager *et al.* (1986) considerarem que um limiar ótimo não afeta muito a carga do ambiente, diversas técnicas foram estudadas, as quais adaptam de forma eficiente e em tempo de execução os limiares de carga do sistema (BECKER; WALDMANN, 1995).

Políticas relativas de transferência recebem como entrada a diferença entre a carga de um recurso e seus vizinhos. Os recursos são considerados aptos a participar de uma transferência caso a diferença de carga ultrapasse um determinado valor. Estes recursos podem então transferir um determinado conjunto de tarefas, ou uma fração da diferença de carga (CASAVANT; KUHL, 1988b; LULING; MONIEN; RAMME, 1991; SCHEURER; SCHEURER; KROPF, 1995; SHIVARATRI; KRÜGER; SINGHAL, 1992; XU et al., 1995).

Qualquer política de transferência de carga deve procurar a minimização das comunicações. Quando o sistema está altamente carregado, os atrasos pelas transferências tendem a ser maiores que a média, o que pode degradar o desempenho do sistema. Em muitos casos, apenas uma pequena parcela das tarefas precisa ser transferida para atingir um balanceamento de carga efetivo (KREMIEN; KRAMER, 1992).

A política de transferência pode ser periódica ou disparada por eventos. Um algoritmo pode verificar periodicamente o estado dos recursos, analisando a necessidade de realizar transferências de carga. Contudo, a grande maioria das políticas propostas é disparada por eventos, tais como alterações de estado, recebimento de solicitações por parte de outros nodos, para realizar transferências.

A transferência de tarefas pode ser classificada como *sender-initiated*, *receiver-initiated* ou *symmetrically-initiated*. Nas políticas *sender-initiated* recursos sobrecarregados procuram por potenciais recebedores de carga; nas políticas *receiver-initiated* recursos com baixa carga procuram por nodos sobrecarregados; nas políticas *symmetrically-initiated* ambos podem iniciar transferências de carga.

Os algoritmos do tipo *sender-initiated* podem apresentar resultados ruins em sistemas sobrecarregados, já que a maioria dos recursos são fornecedores de carga, seria difícil que muitos deles encontrassem parceiros para realizar trocas de carga.

Além disto, eles podem sobrecarregar nodos recebedores, enviando muito trabalho. Isto pode ser evitado desde que exista uma política de limite na quantidade de trabalho recebido, porém isto implicaria em um maior *overhead* de controle. Neste tipo de política, o peso de iniciar uma atividade de redistribuição fica a cargo de nodos que já estão sobrecarregados.

Em políticas *receiver-initiated* o *overhead* é colocado nos nodos mais livres, portanto o fenômeno de colocar mais *overhead* em quem está sobrecarregado não acontece. Por outro lado, se o sistema estiver pouco carregado, estas políticas podem não conseguir encontrar um parceiro adequado. O recebedor pode suspender as atividades, mas neste caso ele não poderá detectar nodos sobrecarregados no futuro, a não ser que sua atividade seja periodicamente reiniciada. Uma desvantagem desta abordagem é que o recebedor não sabe quando outros nodos se tornarão potenciais fornecedores. Uma abordagem alternativa é permitir que os recebedores deixem reservas nos outros recursos, os quais saberão com quem fazer transferência em caso de sobrecarga. Uma vantagem importante das políticas *receiver-initiated* está diretamente relacionada com o fato de elas ficarem desativadas automaticamente em caso de sobrecarga no sistema como um todo, reduzindo-se o número de mensagens de controle (SHIVARATRI; KRÜGER; SINGHAL, 1992; EAGER; LAZOWSKA; ZAHORJAN, 1986; KREMIEN; KRAMER, 1992; NI; XU; GENDREAU, 1985; THEIMER; LANTZ, 1989; ZHOU, 1988).

Os algoritmos *symmetrically-initiated* possuem vantagens e desvantagens de ambos algoritmos: *sender-initiated* e *receiver-initiated*. Um algoritmo *symmetrically-initiated* foi proposto por Niranjan Shivaratri *et al.* (1992), este algoritmo alterna seu comportamento entre *sender-initiated* e *receiver-initiated* conforme o estado do ambiente.

3.6.3 Política de seleção

Uma vez que uma decisão é tomada no sentido de envolver um recurso como um fornecedor, alguma política deve oferecer as regras para seleção de quais tarefas vão participar da transferência.

Uma transferência de carga pode ser preemptiva ou não-preemptiva. Transferências preemptivas envolvem a troca de tarefas parcialmente executadas. Esta operação é bastante custosa, pois precisa coletar o estado do processo em execução, alterar seus canais de comunicação, etc. Embora muitos autores não considerem transferências preemptivas (ZHOU, 1988; THEIMER; LANTZ, 1989), outros alegam fazer isto com sucesso. Scheurer *et al.* (1995) apresentam uma ferramenta que faz migração de processos de forma preemptiva em uma rede de Transputers, apresentado bons resultados; Mor Harchol-Balter & Allen Downey (1997) usam uma estratégia de migração preemptiva a qual usa como critério o tempo de vida do processo.

Transferências de tarefas de forma não-preemptiva envolvem apenas tarefas que ainda não tiveram sua execução iniciada, portanto não há estado a ser preservado. Um recurso pode estar sobrecarregado, e ainda assim não possuir nenhuma tarefa a ser transferida de forma não-preemptiva. Quando realizando transferências com tarefas preemptivas, o escalonador pode se valer de informações coletadas em tempo de execução para explorar melhor as alternativas, isto não é possível quando se utiliza apenas transferências de tarefas não-preemptivas.

Quando uma aplicação paralela utiliza o mesmo algoritmo em um grande con-

junto de pontos, este conjunto de dados pode ser particionado e alocado em muitos processadores, com o objetivo de ser processado no menor tempo possível. Este tipo de carga computacional pode ser classificada segundo a propriedade de divisibilidade dos dados. Caso o conjunto de dados possa ser dividido em um número qualquer de segmentos de qualquer tamanho fracionado, então a carga é classificada como arbitrariamente divisível. Caso exista um limite no grau de divisibilidade da carga, então ela pode ser classificada como modularmente divisível. Um escalonador trabalhando com cargas divisíveis pode decidir pela divisão de uma tarefa que já está em execução.

Uma política de seleção deve utilizar diversos fatores em consideração (HARCHOL-BALTER; DOWNEY, 1997; WATTS; TAYLOR, 1998; SHIVARATRI; KRÜGER; SINGHAL, 1992):

- os *overheads* de transferência de tarefas devem ser minimizados; transferências não-preemptivas e de pequenas tarefas em termos de tamanho, carregam pouco *overhead*;

- o tempo de execução da tarefa a ser transferida deve ser suficiente para justificar o custo de transferência; mesmo que o tempo de execução das tarefas seja desconhecido, elas devem ser classificadas como curtas ou longas, e o escalonador deve considerar apenas as longas para migração; Songnian Zhou (1988) mostrou que alguns erros de classificação pode ser tolerada, já que os escalonadores são bastante robustos com relação a estes parâmetros;

- a dependência das tarefas com relação aos recursos deve ser mínima; estes recursos podem incluir dados e dispositivos específicos, etc.

3.6.4 Política de localização

A política de localização seleciona um parceiro adequado para transferência de carga, geralmente fazendo uso de informações do estado dos recursos. Ela deve evitar sobrecarregar um recurso com baixa carga, isto pode acontecer no caso de um recurso ser selecionado simultaneamente como destino de tarefas por diversos escalonadores. Algumas políticas tentam encontrar o melhor parceiro dentro do domínio, enquanto outras apenas procuram por um parceiro adequado. A política de localização aleatória, seleciona um parceiro sem utilizar nenhuma informação do estado dos equipamentos. Esta política já demonstrou bons resultados (EAGER; LAZOWSKA; ZAHORJAN, 1986). Algumas políticas de localização utilizam esquemas probabilísticos ao invés de determinísticos, estes esquemas distribuem tarefas de acordo com um conjunto de distribuição de probabilidades.

Pankaj Mehra (1992) propõe uma abordagem que utiliza uma rede neural como comparador para cada recurso. Esta rede aprende a predizer o *speedup* para cada tarefa recebida, utilizando apenas a observação dos padrões de utilização dos recursos antes da chegada das tarefas. A falta de informações específicas da tarefa é superada por uma comparação de *speedups* relativos entre recursos diferentes com respeito a mesma tarefa, esta abordagem é utilizada ao invés de tentar obter um *speedup* absoluto. Os parâmetros desta abordagem dinâmica são ajustados através de um algoritmo genético. As redes neurais são treinadas utilizando um sistema de aprendizado *off-line*.

A política de localização pode lidar com algumas restrições quando procurando por um recurso destino. Estas restrições podem incluir requisitos de recursos, precedências entre tarefas (BECKER, 1995), inter-relações entre as tarefas e localidade de dados. A localidade de dados está relacionada ao fato das tarefas necessitarem de um conjunto de dados, que devem ser buscados de uma localização remota (BECKER; WALDMANN, 1995).

3.7 Classificação de algoritmos de escalonamento

O principal objetivo de se desenvolver ou adotar uma classificação é de aumentar e organizar o conhecimento global sobre uma classe de problemas. Este objetivo pode ser atingido através de dois passos: *(i)* especificação de um problema e *(ii)* demonstração de relações entre os problemas.

Segundo Baumgartner & Wah (1991) existem pelo menos quatro atributos desejáveis em uma classificação. O primeiro é identificar as características significativas do problema, já que isto deverá contribuir para a obtenção de uma solução eficiente. O segundo é demonstrar claramente as relações entre os problemas, pois muitas vezes encontrando soluções que se aplicam a um deles, pode-se encontrar também soluções para outros fortemente relacionados. Por outro lado, se um problema não possui solução, possivelmente um outro fortemente relacionado a este também não possua. A expansibilidade e contractibilidade de uma classificação são também importantes, pois permitem que os aspectos mais importantes sejam focalizados e que aspectos pouco significativos sejam negligenciados, reduzindo assim a complexidade de representação. Por fim, é desejável que a classificação separe a especificação do problema de sua solução. Esta separação permite uma comparação clara entre as estratégias de escalonamento, evitando também possíveis confusões entre estratégia e problema.

3.7.1 O escopo da base de decisão e do espaço de migração

Luling *et al.* (1991) defendem que uma descrição completa da política de gerenciamento de carga pode se tornar muito complexa, invibializando uma discussão. Eles propõem uma abordagem simples para classificação de características gerais do algoritmo.

Um escalonador dinâmico pode ser separado em um componente de decisão e outro de migração. O componente de decisão pode utilizar uma informação local de carga, relativa aos seus próprios recursos e a seus vizinhos próximos, ou por outro lado, utilizar informações sobre o sistema como um todo. O primeiro é chamado de base de decisão local, e o segundo de base de decisão global. O componente de migração pode permitir a migração de tarefas apenas para vizinhos diretos ou para qualquer um dos nodos do sistema. A primeira alternativa é chamada de espaço de migração local, enquanto que a segunda é chamada de espaço de migração global.

De acordo com esta distinção entre espaços locais (L) e globais (G), novas alternativas aparecem com respeito a base de migração e a base de decisão. O diagrama apresentado na figura 3.1 apresenta as alternativas disponíveis, relacionando a base de decisão com o espaço de migração. Deve ser considerada ainda a informação de política de transferência adotada: *sender-initiated* (s), *receiver-initiated* (r) ou *symmetrically-initiated* (sr).

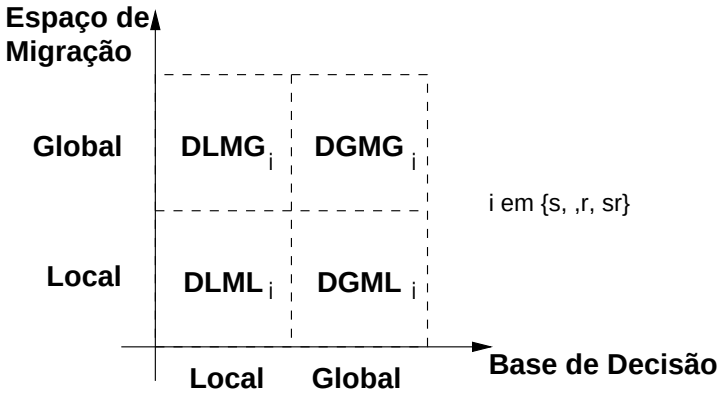

Figura 3.1: A base de decisão e o espaço de migração

3.7.2 A taxonomia de Casavant

Na sua taxonomia, originalmente publicada em (CASAVANT; KUHL, 1988a), Casavant utilizou uma proposta híbrida: uma abordagem hierárquica e outra horizontal. Na visão hierárquica os níveis foram organizados objetivando manter a descrição o menor possível, sendo que a ordem dos mesmos não traduz grau de importância. A classificação horizontal foi empregada para as situações em que os descritores do sistema classificado não implicam em uma ordem específica.

Classificação hierárquica

A estrutura da classificação hierárquica pode ser vista na figura 3.2, e uma discussão da mesma está a seguir.

Local e Global

Na taxonomia de Casavant este é o nível mais alto da hierarquia. Escalonamento local diz respeito às metodologias de compartilhamento de tempo (*time-sharing*), utilizadas na situação de vários processos estarem concorrendo a um único processador. Estas metodologias não são avaliadas na taxonomia de Casavant.

O escalonamento global envolve decidir o lugar (processador) no qual será executado determinado processo, ficando a tarefa de escalonamento local entregue ao sistema operacional do processador alocado. Esta clara separação objetiva minimizar a responsabilidade e o conseqüente overhead do mecanismo de escalonamento global. O desdobramento da taxonomia de Casavant, feito a seguir, diz respeito ao escalonamento global.

Escalonamento Estático e Dinâmico

Estas opções dizem respeito ao momento no qual as decisões sobre o escalonamento são tomadas.

Escalonamento Estático

Neste caso, as informações que definem quais são os processos da aplicação, bem

como as que caracterizam as possíveis paralelizações de tarefas, estão disponíveis no momento do início da execução. Os módulos objeto da aplicação são combinados com os módulos de carga, de tal forma que cada imagem executável gerada fica pré-destinada a um processador específico. Assim, as decisões são tomadas antes da execução ser iniciada e são fixas ao longo da mesma. Tipicamente, as informações que precisam estar disponíveis são o tempo estimado de execução dos processos a serem paralelizados, o volume de comunicação de cada um, as características dos processadores (sobretudo no caso de uma arquitetura heterogênea) e os custos de comunicação em função da topologia da rede de interconexões da arquitetura paralela.

Na taxonomia de Casavant, escalonamento estático é sinônimo de escalonamento determinístico.

Escalonamento Estático - Ótimo e Sub-ótimo

No caso onde todas as informações correspondentes ao estado do sistema e às necessidades de recurso dos processos são conhecidas, um escalonamento ótimo pode ser feito. Como exemplo de otimizações que podem ser atingidas, temos a minimização do tempo de execução, a maximização da utilização de recursos e a maximização do *throughput* do sistema. Muitas vezes a complexidade que atinge o escalonamento ótimo o torna computacionalmente intratável, bem como é pouco usual a disponibilidade de todas as informações necessárias antes da execução iniciar; daí surgem as soluções sub-ótimas.

Escalonamento Estático - Sub-ótimo - Aproximado e Heurístico

O escalonamento sub-ótimo aproximado utiliza os mesmos algoritmos do escalonamento ótimo, mas ao invés de explorar todo o espaço das possíveis soluções ideais, ele se satisfaz quando encontra uma considerada "boa". Para as situações em que possam ser definidas métricas para reconhecer uma "boa" solução, esta alternativa pode reduzir consideravelmente o tempo necessário para obter um escalonamento para a aplicação em questão. Por sua vez, a principal característica do escalonamento heurístico é o uso de parâmetros genéricos que sabidamente afetam o comportamento do sistema paralelo. Por princípio, estes parâmetros devem ser simples de obter ou calcular. Por exemplo, agrupar processos com elevada taxa de comunicação em um mesmo processador. A expectativa é que tal procedimento melhore o desempenho do sistema como um todo, porém não existe, neste caso, uma preocupação em garantir uma relação direta entre o mesmo e o resultado desejado. Em outras palavras, em função do conhecimento da dinâmica da arquitetura paralela, provavelmente a utilização de determinadas heurísticas e seus respectivos parâmetros conduzirá a um melhor escalonamento, mas isto não pode ser provado.

Escalonamento Dinâmico

O escalonamento dinâmico tem por base que poucas informações a respeito das necessidades e do comportamento da aplicação estarão disponíveis de antemão. Deste modo, nenhuma decisão é tomada até que a mesma inicie sua execução. A decisão do lugar (em quais nodos processadores) os processos da aplicação serão

computados é responsabilidade do ambiente de execução da arquitetura.

Escalonamento Dinâmico - Fisicamente distribuído e não distribuído

Neste nível, a taxonomia trata se o trabalho pertinente às tomadas de decisão de escalonamento estará distribuído entre alguns processadores da arquitetura, ou se ficará a cargo de uma tarefa, localizada em um único processador.

Escalonamento Dinâmico - Fisicamente distribuído - Cooperativo e Não cooperativo

O foco, neste nível do escalonamento distribuído, é o grau de autonomia que cada processador tem na determinação de como seus recursos podem ser utilizados. No caso cooperativo, cada processador tem a responsabilidade de contribuir com sua parte na tarefa de escalonamento, e todos estão trabalhando segundo uma meta global da arquitetura. No modelo não-cooperativo, os processadores atuam como entidades autônomas e decidem o uso dos seus recursos sem considerar o efeito disto no resto da arquitetura. Para o escalonamento dinâmico distribuído e cooperativo se aplicam as mesmas subdivisões existentes para o escalonamento estático, quais sejam: ótimo, sub-ótimo aproximado e sub-ótimo heurístico.

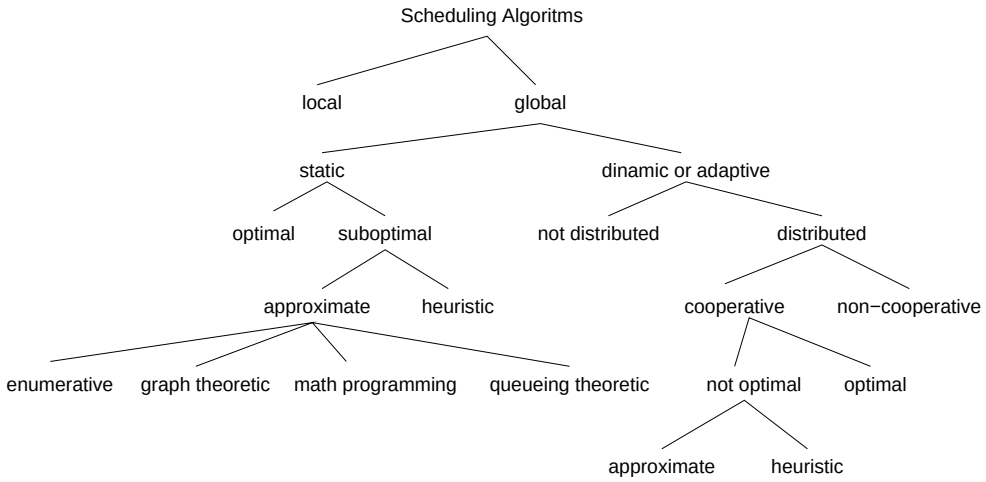

Figura 3.2: Taxonomia de Casavant

Classificação horizontal

As características descritas na classificação horizontal de Casavant podem ser encaixadas em todos os ramos da sua classificação hierárquica. O objetivo de separar as classificações é tornar a taxonomia mais clara e objetiva. Não existe por parte do autor distinção de importância entre as mesmas.

Escalonamento Adaptativo e Não adaptativo

Uma solução adaptativa para o problema do escalonamento em sistemas paralelos e distribuídos, é aquela na qual os parâmetros, bem como os algoritmos utilizados para implementar a política de escalonamento, podem ser alterados, durante a execução, em função das respostas do sistema ao escalonador. Um caso típico, neste

sentido, seria o escalonador em função do andamento da execução desconsiderar algum parâmetro (ou reduzir a sua importância), se entender que o mesmo traduz uma informação inconsistente (ou insignificante) com relação às outras do sistema.

Em contraste, uma política não adaptativa não modifica seu mecanismo de controle de escalonamento em função do comportamento da execução paralela.

Balanceamento de carga

A idéia básica desta política é fazer com que os processos progridam em todos os nodos a uma mesma razão. Para isto, a informação sobre a carga nos diferentes processadores é compartilhada através da rede de interconexão de forma periódica ou sob demanda, de forma que todos os nodos tenham uma visão do estado global do sistema. Assim, todos os nodos cooperam com o objetivo de remover trabalho dos processadores com elevada carga para os mais liberados. Esta estratégia usualmente é mais efetiva (menos complexa de implementar) quando os nodos do sistema são homogêneos. Cuidados precisam ser tomados na escolha da unidade de medida da carga nos processadores, bem como para prevenir migrações de carga que resultem em baixa produtividade global.

Escalonamento por Licitação (Ofertas ou Leilão)

Nesta proposta de escalonamento, os nodos da arquitetura tanto podem assumir o papel de gerente, como de contratante. O nodo gerente é aquele que tem tarefas para serem compartilhadas, e o contratante é o que tem disponibilidade computacional para executar novos processos. Os nodos gerentes anunciam a existência de tarefas a serem processadas e recebem propostas (ofertas de compra de tarefas) dos nodos contratantes. O tipo e a quantidade de informações trocadas são determinantes para o desempenho desta proposta. Uma característica importante desta classe de escalonadores é o fato de que os gerentes têm autonomia para decidir, entre os nodos que responderam ao seu anuncio, para qual irá enviar a tarefa a ser executada (os critérios para decisão podem ser ótimos, sub-ótimos ou heurísticos). Some-se a isto que em função da evolução da computação na arquitetura como um todo, os contratantes não são obrigados a aceitar uma tarefa pela qual já manifestaram interesse.

Escalonamento Probabilístico

A motivação desta estratégia é a constatação que muitas vezes se mostra proibitivo o tempo necessário para tratar analiticamente todo espaço de soluções para o mapeamento de processos aos nodos da arquitetura. A idéia é gerar segundo alguma distribuição estatística um mapeamento. Após terem sido gerados diversos possíveis mapeamentos, o conjunto obtido é então analisado, e escolhido o melhor mapeamento. Os critérios de escolha adotados são definidos em função do sistema paralelo como um todo (características de hardware e software).

Escalonamento de Atribuição Única e Reatribuição Dinâmica

Nesta classificação é considerado o momento em que o escalonador recebe as

informações sobre a aplicação paralela a ser executada. Na proposta de Atribuição Única, as informações são disponibilizadas no momento que a aplicação é submetida à execução. Por sua vez, na reatribuição dinâmica, após uma execução parcial, são reavaliados os parâmetros passados inicialmente ao escalonador, utilizando informações dinamicamente criadas durante a própria execução. Este ciclo de execuções parciais e avaliação pode ocorrer diversas vezes durante todo processamento da aplicação.

A Reatribuição Dinâmica é um recurso que os administradores de sistemas paralelos têm para evitar que os usuários burlem a estrutura de prioridades anteriormente estabelecida, informando parâmetros que não correspondem à aplicação que será submetida à execução.

3.7.3 O esquema de classificação ESR

Baumgartner & Wah (1991) apresentam um esquema de classificação que se detém ao problema, separando completamente a especificação do problema da especificação da solução.

A classificação estabelece três grupos de atributos correspondentes aos componentes de entrada do escalonador: os eventos, referentes as características da carga de trabalho; o ambiente (*surroundings*), referente ao sistema computacional; e os requisitos (*requirements*) de desempenho do escalonador. A sigla ESR significa *events*, *surroundings* e *requirements*. Neste estágio não são feitas tentativas de classificar as soluções para o problema de escalonamento. Apenas as propriedades dos problemas são classificadas.

Os reais atributos usados em cada um desses grupos não estão previamente especificados. Atributos podem ser adicionados na tentativa de melhor descrever cada problema em particular. A capacidade de expansão e contração deste esquema de classificação advém desta característica, já que o conjunto de atributos a serem usados e seus possíveis valores podem receber acréscimos ou serem restringidos para descrever o problema no nível de abstração mais adequado.

A tabela 3.1 apresenta uma possível caracterização dos atributos a serem adotados. Neste exemplo, os eventos são classificados conforme o grau de dependência entre as tarefas, os padrões de recebimento de tarefas, seus requisitos de recursos e o tipo de decomposição utilizada para implementar a aplicação paralela. O sistema distribuído é classificado conforme a heterogeneidade e quantidade de recursos, as características físicas e a sua disponibilidade. O modelo de comunicações também foi incluído. Por fim são especificados os requisitos de desempenho, incluindo a meta do escalonador e o nível de desempenho a ser atingido.

A notação proposta pelos autores sugerem que a especificação do problema deva ser escrita conforme o exemplo:

$$E : \left\{ \begin{array}{l} \text{dependências de precedência} \\ \text{recebimento estocástico de tarefas} \\ \text{decomposição funcional} \end{array} \right\}$$

$$-S : \left\{ \begin{array}{l} \text{30 nodos} \\ \text{recursos heterogêneos} \\ \text{disponibilidade estocástica} \\ \text{sobrecargas estocásticas de comunicação} \end{array} \right\} -R : \left\{ \begin{array}{l} \text{tempo real} \\ \text{desempenho sub-ótimo} \end{array} \right\}$$

Tabela 3.1: O esquema de classificação ESR

Categoria	**Atributo**	**Valores**
Eventos	Dependência entre tarefas	independente precedência comunicação
	Recebimento de tarefas	estático periódico estocástico
	Requisitos de recursos	determinístico estocástico
	Decomposição	funcional por domínio – tarefas indivisíveis por domínio – tarefas divisíveis
Surroundings (ambiente)	Classes de recursos	homogêneos heterogêneos
	Número	1,n
	Características físicas	velocidade tamanho de memória
	Disponibilidade	determinística estocástica
	Overhead de comunicação	nenhum determinístico estocástico
Requisitos	Meta	minimização do tempo de execução maximização de *throughput* tempo real tolerância a falhas
	Qualidade	qualquer solução sub-ótimo ótimo

Tabela 3.2: Atributos da classificação ESR de estratégias de escalonamento

Atributo	**Valores**
Espaço de informações	local global
Espaço de migração	local global
Adaptação	estática dinâmica adaptativa
Localização do controle	distribuído hierárquico centralizado
Tipo de transferências	uma só vez não preemptiva preemptiva tarefas divisíveis
Mecanismo de decisão	determinístico estocástico
Nível de informações do estado do ambiente	nenhuma simples detalhada
Modelo de execução da aplicação	nenhum determinístico estocástico

Deve ser definida também uma classificação para as estratégias de escalonamento, a qual deve apresentar uma completa separação entre a especificação do problema e soluções particulares. Esta classificação inclui um conjunto de atributos, os quais são também selecionados pelo projetista da execução paralela, mantendo portanto as características de expansibilidade e contractibilidade da classificação. A tabela 3.2 apresenta um conjunto de atributos que poderiam ser usados para classificação de estratégias de escalonamento.

O próximo capítulo resume o ambiente de execução no qual a heurística de escalonamento que está sendo proposta será utilizada.

4 EXEHDA: O AMBIENTE DE EXECUÇÃO DO ISAM

O objetivo central deste capítulo é caracterizar o ambiente de execução onde o TiPS atua. Será contemplada uma descrição do ambiente de execução da arquitetura ISAM, o EXEHDA, e serão particularmente destacados os aspectos de escalonamento.

4.1 O ambiente de execução EXEHDA

Os esforços de pesquisa do EXEHDA (*Execution Environment for High Distributed Applications*) têm por foco definir os componentes da arquitetura para um ambiente de execução destinado a aplicações na computação pervasiva (REAL et al., 2003; YAMIN et al., 2003; SILVA, 2003; YAMIN, 2004. 194p). No EXEHDA, as condições de contexto são pró-ativamente monitoradas e o suporte à execução deve permitir que tanto a aplicação como ele próprio utilizem estas informações na gerência de seus aspectos funcionais e não funcionais. Também, a premissa siga-me da computação pervasiva deverá ser suportada, garantindo a execução da aplicação do usuário em qualquer tempo e lugar.

As aplicações alvo são distribuídas e compreendem mobilidade de hardware e software, sendo baseadas no modelo de programação ISAMadapt (AUGUSTIN, 2004. 194p) empregado pelo projeto ISAM. O ISAMadapt é um ambiente de desenvolvimento de aplicações distribuídas, móveis e conscientes do contexto da computação pervasiva. A aplicação torna-se consciente de seu contexto e adapta-se a ele, através de um processo que envolve duas fases: concepção e execução. Comandos relativos às abstrações foram adicionados à linguagem-base Holo (BARBOSA, 2002. 213p),e o ambiente de execução EXEHDA implementa a funcionalidade dinâmica dessas abstrações e gerencia o ambiente pervasivo.

O mecanismo de adaptação do EXEHDA propõe uma estratégia colaborativa entre aplicação e ambiente de execução, através da qual é facultado ao programador individualizar políticas de adaptação para reger o comportamento de qualquer dos componentes da aplicação (YAMIN et al., 2002). As políticas que irão reger os mecanismos de adaptação, funcionais ou não, são especificadas pelo ambiente de desenvolvimento provido pelo ISAMadapt, algumas destas políticas são relativas ao escalonamento (AUGUSTIN, 2004. 194p).

4.2 O ISAM*pe*

O meio de execução "pervasivo" do projeto ISAM, chamado ISAM*pe* - ISAM *pervasive environment* pode ser caracterizado como um sistema distribuído de grande abrangência (YAMIN et al., 2003).

No tocante à organização física do meio de execução, a premissa do ISAM de integrar os cenários, especialmente o (i) da computação em grade, (ii) da computação móvel e (iii) da computação sensível ao contexto, é mapeada em uma organização composta pela agregação de células de execução, chamadas de EXEHDAcel. Esta organização está representada Figura 4.1. Considera-se que os nodos móveis também devam usufruir da infra-estrutura de rede cabeada existente, beneficiando-se de ambientes como o oferecido pela Internet. Os elementos básicos do meio de execução ISAM*pe* são:

- **EXEHDAcel:** denota a área de atuação de uma EXEHDAbase, sendo composta por esta e por EXEHDAnodos;
- **EXEHDAbase:** é o ponto de contato para os EXEHDAnodos. É responsável por todos os serviços básicos do ISAM*pe* e embora constitua uma referência lógica única, seus serviços, sobretudo por aspectos de escalabilidade, poderão estar distribuídos entre vários equipamentos. Isto é representado na Figura 4.1 pelo sombreado associado à figura da EXEHDAbase;
- **EXEHDAnodo:** são os processadores disponíveis no ISAM*pe*, sendo responsáveis pela execução das aplicações. Em um EXEHDAnodo é ativado apenas o núcleo mínimo do EXEHDA, sendo carregados sob demanda os outros serviços que possam ser necessários;
- **EXEHDAnodo móvel:** são os nodos do sistema com possibilidade de operação móvel. São funcionalmente análogos aos EXEHDAnodos, porém contam com interface de rede para operação sem fio.

4.3 Características do EXEHDA

Esta seção objetiva destacar os aspectos que caracterizam o EXEHDA. Uma discussão destas características pode ser encontrada em (YAMIN, 2004. 194p):

- sua operação ocorre sobre o sistema operacional, e sem exigir alteração do mesmo. Isto potencializa a portabilidade;
- pode suportar tanto execuções paralelas como distribuídas. Para tal, interfaces de programação para comunicação interprocessos, tanto síncronas quanto assíncronas, são disponibilizadas;
- não está comprometido com uma heurística de escalonamento em particular. Ao contrário, disponibiliza facilidades para que novas heurísticas sejam implementadas. Esta política também se aplica no que diz respeito aos procedimentos de sensoriamento;
- a heurística de escalonamento/sensoriamento a ser utilizada é selecionada e/ou contextualizada por usuário e aplicação;

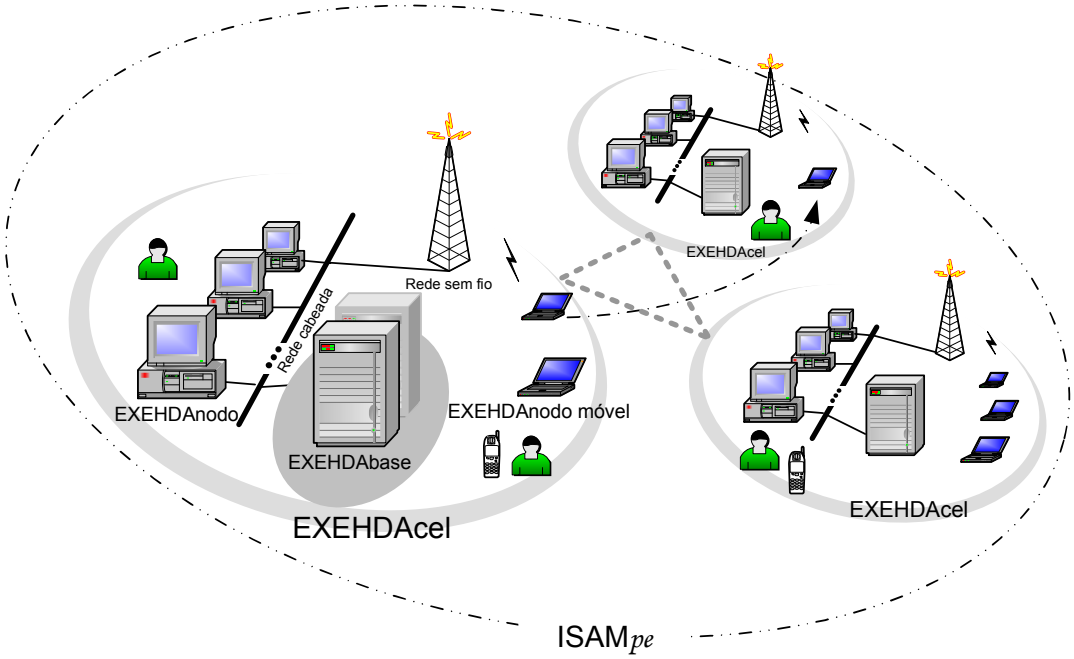

Figura 4.1: Elementos básicos do ambiente de execução

- os componentes que tomam decisão são replicados, e são capazes de atividades autônomas e assíncronas. Este aspecto é indispensável para uma operação com elevada escalabilidade.

No que diz respeito às estratégias para maximização do desempenho da execução de aplicações ISAMadapt, o escalonamento no EXEHDA oferece suporte as seguintes alternativas:

- balanceamento de carga nos nodos responsáveis pelo processamento;
- localização dos recursos (software e hardware) mais próximos (reduzir custo de comunicação);
- emprego de replicação de serviços e de dados;
- disponibilização antecipada, por usuário, da demanda de componentes das aplicações e dos dados;
- otimização no volume de comunicações, utilizando transferências de contextos e componentes de aplicação personalizadas por usuário;
- monitoração da comunicação praticada pelos componentes das aplicações em execução, com intuito de otimizar aspectos de mapeamento;
- uso de uma estratégia colaborativa entre o ambiente de execução e a aplicação na tomada de decisões de escalonamento (adaptação).

O emprego destes procedimentos fica potencializado pela possível alternância do ponto de conexão dos EXEHDAnodos móveis no contexto da rede, comportamento este inerente à computação móvel.

4.4 A organização do escalonamento no EXEHDA

No ISAM, as aplicações solicitam direta ou indiretamente recursos do escalonador. Algumas podem especificar uma determinada necessidade de qualidade de serviço (QoS), outras podem aceitar o "melhor-possível" nos níveis de serviço. Assim, o módulo de escalonamento do *middleware*, tem a estratégia de trabalhar com diferentes políticas de gerenciamento para diferentes aplicações, usuários e/ou domínios de execução. Neste caso, as estratégias de adaptação exigem do mecanismo de escalonamento o tratamento de problemas de otimização utilizando critérios múltiplos.

4.4.1 A adaptação multinível colaborativa

A proposta ISAM contempla um comportamento adaptativo em dois segmentos: (1) na aplicação, a qual define o comportamento da adaptação (alternativas) e o contexto de seu interesse; (2) no ambiente de execução (EXEHDA), o qual tem um comportamento inerentemente adaptativo no momento em que processa a aplicação. Uma visão estrutural da Adaptação Multinível Colaborativa proposta pode ser vista na Figura 4.2.

Na codificação da aplicação, o programador especifica os elementos computacionais que afetam o comportamento da aplicação, os respectivos níveis de variação suportados e codifica comportamentos alternativos para atender à variação nas condições ambientais (variação nos elementos computacionais no contexto da aplicação).

Estas informações influenciam o comportamento adaptativo a ser adotado pelo ambiente de execução. Por sua vez, o ambiente de execução fornece meios (a) para que sejam monitorados elementos computacionais do ambiente, (b) para que a aplicação possa registrar seu interesse em determinados elementos, (c) para notificar à aplicação das alterações ocorridas, e (d) para selecionar o comportamento alternativo mais adequado ao ajuste das novas condições ambientais. Como o sistema que gerencia as aplicações, este também pode ter um comportamento pró-ativo e executar adaptações relativas à administração e desempenho do sistema de forma global. A proposta de adaptação multinível colaborativa está detalhada em (YAMIN et al., 2002).

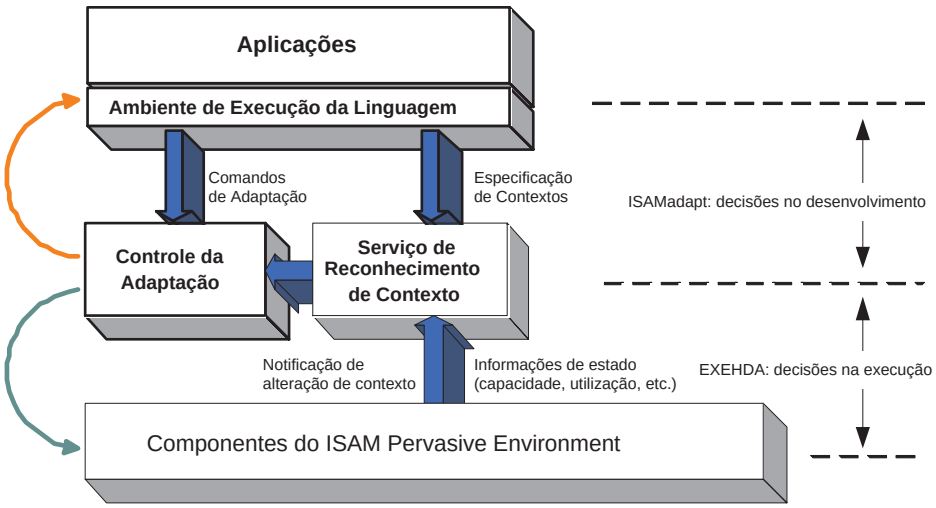

Figura 4.2: Adaptação multinível

4.5 O disparo de um OX sob a ótica do escalonamento

Para um melhor entendimento de como um OX (Objeto eXehda) é disparado no EXEHDA, é necessário que se apresente o OX, bem como dois serviços do EXEHDA que têm papel bastante importante nos procedimentos necessários para esta operação. São eles o CIB e o ResourceBroker (RB).

4.5.1 A abstração OX

As decisões de escalonamento no EXEHDA envolvem sempre um OX que é uma abstração para suporte de execuções pervasivas. O OX consiste de um objeto ao qual podem ser vinculados atributos de execução, este objeto é instanciado através de um serviço chamado Executor. Estes atributos são disponibilizados de forma pervasiva pelo serviço OXManager do EXEHDA, sendo utilizados por outros serviços do EXEHDA para coordenação da execução de operações distribuídas.

A abstração OX também é importante quando da definição dos mapeamentos entre os elementos modelados na linguagem de programação e as funcionalidades oferecidas pelos serviços do *middleware*. Desta forma, o OX atua como um elemento de ligação entre as funcionalidades providas pelos diversos serviços, permitindo a implementação de abstrações de mais alto nível definidas nas linguagens de programação para o cenário da computação pervasiva.

Adicionalmente os atributos do OX são também utilizados para armazenar as políticas que devem ser utilizadas pelo serviço de escalonamento denominado Scheduler.

O Scheduler é o serviço central na gerência das adaptações de cunho não-funcional no EXEHDA, isto é, que não implicam alteração de código. Nesse sentido, o Scheduler emprega a informação de monitoração, obtida junto ao serviço Collector, para orientar operações de mapeamento. A operação do Scheduler acontece quando de instanciações remotas ou migrações realizadas pelo serviço Executor, ou quando de chamadas de re-escalonamento, originadas do estado atual de um recurso não satisfazer mais as necessidades de um objeto anteriormente a ele alocado.

O serviço CIB (*Cell Information Base*) mantém os atributos relacionados ao gerenciamento da ISAM*pe*, descrevendo os recursos que constituem a célula e sua vizinhança, assim como os atributos relacionados às aplicações em execução e aos recursos alocados para elas. Portanto, quando um novo EXEHDAnodo é incorporado à execução distribuída de uma aplicação, as instâncias locais de serviços do EXEHDA executando naquele nodo podem recuperar a partir do CIB todos atributos da aplicação necessários para sua execução. Além disto, o serviço CIB mantém as informações com relação aos usuários registrados na célula.

O serviço ResourceBroker do EXEHDA implementa o controle de acesso aos recursos da célula. Ele oferece uma interface interna à célula para os serviços locais, bem como uma interface externa para serviços de outras células. As visões, interna e externa, apresentadas por ele podem ser diferentes, fazendo um controle de visibilidade externa. O ResourceBroker interage principalmente com o serviço *Scheduler* da célula local.

Quando uma tarefa não for destinada a um EXEHDAnodo específico, o disparo da execução da mesma irá envolver o escalonador.

A figura 4.3 resume os procedimentos realizados pelo EXEHDA quando da instanciação do OX correspondente a esta tarefa. Nesta figura, como a célula em que

foi originada a demanda não tinha recursos para atendê-la, o OX foi instanciado e colocado em execução em uma célula vizinha.

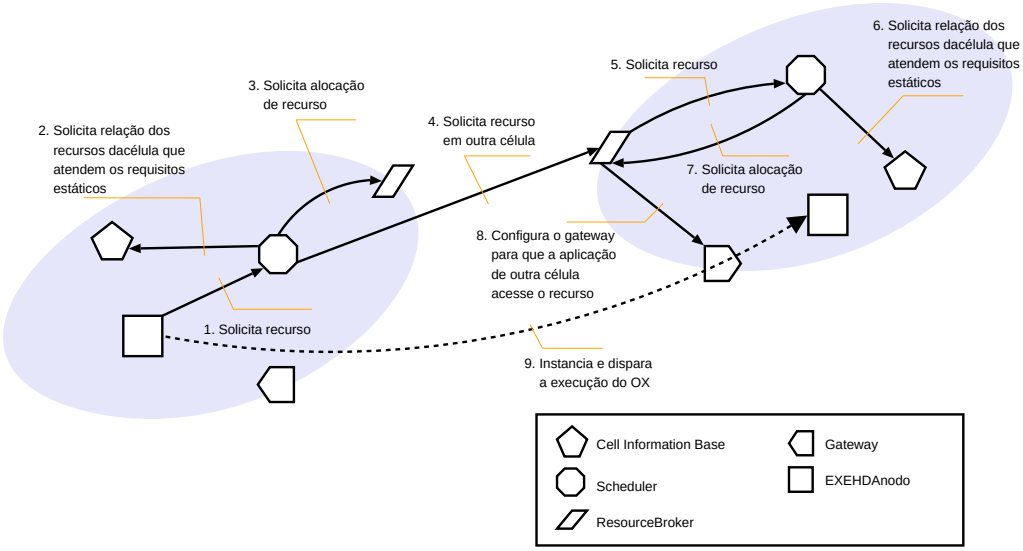

Figura 4.3: Alocação inter-celular de recursos de processamento

Uma caracterização de como ocorre o disparo de um OX compreende os seguintes procedimentos:

1. **solicitação de recursos ao Scheduler**: o EXEHDAnodo com tarefas a serem colocadas em execução solicita recursos ao Scheduler da sua célula;

2. **avaliação da disponibilidade do tipo de recurso na célula**: o Scheduler verifica ante a CIB se a natureza dos recursos existentes na EXEHDAcel atende a especificação feita. Neste procedimento são verificados os requisitos pertinentes às características estáticas dos recursos, por exemplo: a quantidade de memória, o tipo de display e a capacidade do disco.

3. **seleção entre os recursos disponíveis**: existindo recursos na célula que potencialmente podem ser utilizados para instanciar o OX, o Scheduler organiza uma lista destes recursos e a repassa ao RB em uma determinada ordem. Também é informado ao RB se o OX a ser instanciado exige um EXEHDAnodo em regime exclusivo.

4. **alocação de recursos na EXEHDAcel local**: o RB mantém o registro de quais nodos estão em uso, tanto no modo exclusivo como compartilhado e ao receber a lista de recursos inicia verificando, pela ordem de prioridade construída pelo escalonador, a disponibilidade dos nodos indicados. Caso algum dos nodos indicados como candidatos for factível de uso, uma confirmação da existência do mesmo será retornada ao Scheduler, e este EXEHDAnodo será alocado pelo RB para a aplicação. No caso de nenhum nodo da lista estar disponível, o Scheduler irá gerar uma nova seleção de nodos, este procedimento de submeter de uma lista ao RB irá se repetir até que estejam esgotados os recursos da EXEHDAcel.

5. **busca de recurso em EXEHDAcel vizinha**: confirmada a não existência de recursos na célula, o RB mantém contato com o RB de células vizinhas, e repassa a solicitação de recursos. Internamente nesta célula remota, acontece de forma análoga ao já retratado nos procedimentos 1, 2, 3 e 4. No caso de falhar a alocação na célula vizinha selecionada, o Scheduler é informado e outra célula entre as vizinhas será avaliada.

6. **configuração de acesso por células remotas**: na célula aonde for encontrado o recurso solicitado, os respectivos serviços de RB e Gateway interagem, de forma que o RB configure o Gateway ficando este recurso visível aos equipamentos de células remotas.

4.6 O disparo de um OX na visão do EXEHDAnodo

A criação de um OX é uma operação para atender a demanda da aplicação em execução, sendo disparada a partir de um nodo específico. Um EXEHDAnodo, entretanto, pode estar operando desconectado do ISAMpe, deste modo se mostra oportuno que seja dado suporte no EXEHDA para a criação de OXes sem a obrigatoriedade de que sejam acessadas as instâncias celulares dos serviços, particularmente do Scheduler.

A aplicação, ao executar uma operação de criação de OX, gera uma requisição a instância local do serviço Executor. Este serviço faz uma chamada à instância local do Scheduler, passando por um filtro que indica qual foi à heurística de escalonamento selecionada pela aplicação. O Scheduler avalia a possibilidade de atender a requisição de criação de forma desconectada, caso isto seja possível a criação do OX será feita utilizando somente a instância nodal dos serviços. Quando o nodo voltar a se conectar as instâncias celulares dos serviços serão atualizadas com as informações do OX criado. Uma visão do disparo de um OX no contexto de um EXEHDAnodo pode ser visto na Figura 4.4.

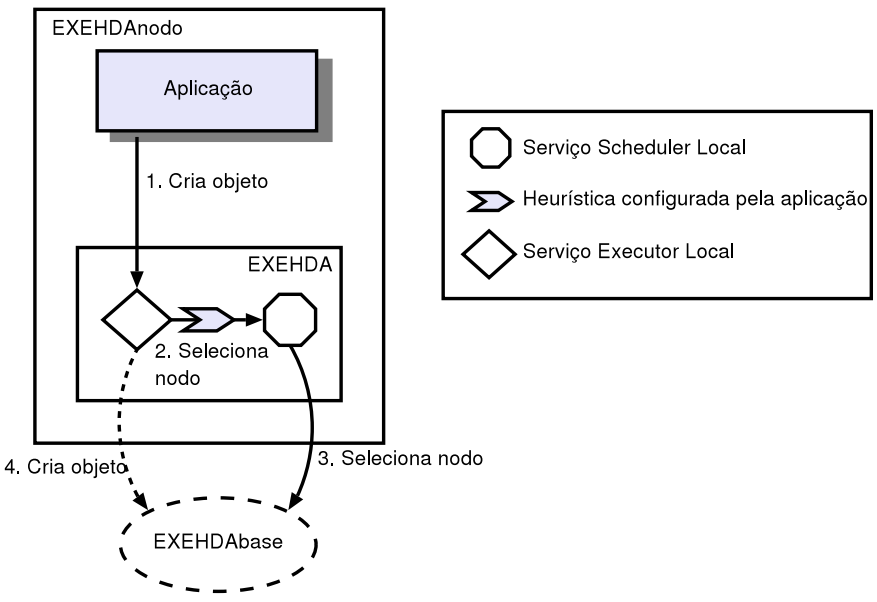

Figura 4.4: Disparo de um OX em um EXEHDAnodo

Nem sempre a criação sem envolver a instância celular dos serviços será possível,

por exemplo, uma heurística de escalonamento que busque o melhor desempenho possível poderá exigir uma consulta à instância celular do escalonador na busca do nodo mais adequado para atender esta demanda, considerando o contexto global da EXEHDAcel.

As características comportamentais e de construção do ambiente de execução identificadas neste capítulo foram utilizadas na concepção do TiPS, a qual é apresentada a seguir.

5 TIPS: CONCEPÇÃO E MODELAGEM

O TiPS (*TiPS is a Probabilistic Scheduler*) foi concebido na forma de um *framework* para construção de escalonadores que utilizem estratégias de inteligência artificial com ajuste dinâmico de parâmetros. Estas características possibilitam a construção de heurísticas que tratem as incertezas relacionadas à elevada dinamicidade do meio de execução da computação pervasiva. Diferentes estratégias de escalonamento podem ser exploradas sobre este *framework*, a partir da re-implementação de determinados componentes.

Neste capítulo são apresentados aspectos de modelagem do TiPS, a aplicação do TiPS para o escalonamento baseado no poder de processamento disponível em cada nodo do sistema e, por fim, a arquitetura do TiPS.

5.1 Integração do TiPS com o EXEHDA

Quando é necessário instanciar um `OX`, o TiPS recebe uma requisição de seleção de nodo de processamento. O nodo selecionado pelo TiPS é então utilizado para criação do `OX`. Para possibilitar a utilização de heurísticas que empreguem conhecimento sobre o uso dos recursos na célula, o TiPS se registra para receber os dados de sensores instalados pelo ambiente de execução nos nodos de cada célula. Os sensores são configurados para publicarem os dados apenas quando variações significativas são percebidas, desta forma é reduzido o custo de comunicação necessário para que o escalonador mantenha suas informações atualizadas.

As incertezas com relação ao nível de disponibilidade de cada um dos recursos que constituem uma célula, motivam a criação de estratégias que tratem de forma adequada estas características. A carga de processamento de cada nodo é uma das variáveis que, em geral, é de interesse do escalonamento de aplicações distribuídas, a partir destas informações é possível buscar nodos menos carregados.

Por outro lado, as aplicações podem, colaborativamente, fornecer informações do comportamento de cada um de seus `OX`es. Esta colaboração, expressa na forma de políticas (AUGUSTIN, 2004. 194p), parametriza as tomadas de decisão do escalonador.

O re-escalonamento de `OX`es também é disparado, quando necessário, a partir de parâmetros configurados pela aplicação junto ao servidor de contexto. As aplicações podem estabelecer critérios para o disparo de um re-escalonamento. A operação é recebida pelo TiPS como uma solicitação de recurso, portanto é encarada como uma operação regular de escalonamento.

5.2 O problema do escalonamento e o TiPS

O problema do escalonamento a ser tratado no TiPS foi modelado utilizando como guia os parâmetros da classificação ESR (BAUMGARTNER; WAH, 1991), apresentada na seção 3.7.3. A partir desta proposta são construídas as possíveis alternativas para tratar o problema.

O EXEHDA apresenta de forma independente os `OX`es a serem escalonados, portanto não é necessário que o escalonador gerencie a ordem de execução dos `OX`es. Por outro lado, o recebimento de solicitações de escalonamento de `OX`es é de difícil previsão, pois cada aplicação tem um comportamento específico. O ISAMadapt (AUGUSTIN, 2004. 194p) pré-determina os requisitos de cada `OX`, estes, podem ser utilizados pelo escalonador para delimitar os recursos que podem ser entregues para um `OX`. Os `OX`es não podem ser redimensionados, ou repartidos pelo escalonador, portanto são vistos como processos indivisíveis.

O entorno do problema é composto por um grande número de recursos heterogêneos, cuja disponibilidade é estocástica. Dependendo do recurso as variações podem estar relacionadas ao seu aparecimento ou desaparecimento, às flutuações no nível de utilização, ou a uma combinação destas duas. O custo de comunicação é considerado homogêneo, pois o TiPS trata de recursos que estão presentes em uma mesma célula, servidos por uma rede local.

O requisito de desempenho para solução do problema é a minimização do tempo de processamento da aplicação.

As principais fontes de incertezas encontradas são: (i) recebimento estocástico de tarefas e (ii) disponibilidade estocástica dos recursos. Para aumentar as chances de atender o requisito de otimização no tempo de processamento, as questões originárias de incertezas devem ser tratadas. A manutenção de conhecimento histórico integrada a técnicas probabilísticas de tomada de decisão constitui uma alternativa.

O recebimento estocástico de tarefas é um problema de difícil tratamento, especialmente se o mecanismo de escalonamento for genérico e não possuir informações específicas das aplicações. A previsão do momento em que ocorrerão operações de criação de `OX`es é dependente de características específicas de cada aplicação e podem variar dinamicamente em tempo de execução. Por outro lado, a disponibilidade estocástica dos recursos pode ser tratada, pois existem no ambiente de execução formas para monitorar estes recursos e assim manter um conhecimento histórico a respeito de cada recurso.

O TiPS trata o problema de disponibilidade estocástica de recursos buscando a manutenção de um perfil histórico dos recursos disponíveis. Na manutenção do perfil, são consideradas as flutuações dos níveis de utilização e as probabilidades de um recurso manter-se em cada um dos possíveis estados.

5.3 A política de escalonamento utilizada

A política de escalonamento no TiPS é apresentada na forma de quatro componentes: política de informação, política de transferência, política de localização e política de seleção.

5.3.1 Política de informação

No TiPS é utilizada uma política de informação orientada à mudança de estado (vide seção 3.6.1). Sensores instalados nos recursos monitoram o estado dos mesmos, fazendo a publicação das informações sempre que a variação do dado sensorado ultrapassar um determinado limiar. Os dados a serem monitorados são discretizados em n possíveis estados e sempre que ocorre uma troca entre um estado e outro, a informação do sensor é propagada.

Para receber dados de um determinado recurso, o TiPS deve se registrar junto ao CIB (vide seção 4.5) da célula em que ele está operando. O CIB mantém o registro de quais componentes devem receber quais dados sensorados. Um outro componente, chamado defletor, tem então as funções de receber os dados sensorados e publicá-los para os componentes que farão uso dos mesmos. Na operação de registro são estabelecidos os limiares que determinam as trocas de estado, bem como a periodicidade da monitoração local realizada pelo sensor.

Os sensores do ISAM também oferecem a possibilidade da realização de consultas para verificação do valor corrente de medição. Desta forma, o TiPS pode também operar segundo uma política de informação dirigida pela demanda. Esta característica é importante em casos em que não é necessário monitorar continuamente uma determinada informação dos recursos, ficando a cargo do TiPS a decisão do momento em que uma determinada variável do sistema será inspecionada.

A utilização da política de propagação das informações por mudança de estado permite que sejam otimizados os custos de processamento relacionados com a operação de escalonamento. As comunicações são minimizadas e a freqüência de amostragem local dos sensores pode ser dinamicamente alterada, reduzindo ou aumentando a carga nos recursos conforme estes se mostram mais estáveis ou mais instáveis.

5.3.2 Política de transferência

O tomador de decisões do TiPS opera como um controlador centralizado para cada uma das células do sistema. O TiPS determina quais nodos estão mais aptos a receberem cargas de processamento

As aplicações executando de forma distribuída criam `OX`es e alguns deles são configurados para execução remota, sendo necessário que o TiPS selecione recursos para estas execuções.

5.3.3 Política de seleção

A política de seleção envolve a tomada de decisão de qual `OX` será selecionado para escalonamento. Uma revisão sobre a política de seleção é apresentada na seção 3.6.3.

O TiPS segue uma política em que as requisições são atendidas na ordem de recebimento, desta forma, os objetos que solicitam recursos são atendidos conforme a ordem de chegada de suas requisições.

No ISAM a aplicação define quais `OX`es podem ser executados em recursos remotos auxiliando na manutenção do bom desempenho da execução da aplicação. A aplicação, desta forma, fornece informações que qualificam as decisões do ambiente de execução. Estas definições são feitas pelo programador da aplicação, através das abstrações oferecidas pelo ISAMadapt (AUGUSTIN, 2004. 194p). Esta integração entre o ambiente de execução e a linguagem de programação é chamada no ISAM de colaboração multinível, apresentada na seção 4.4.1.

OXes em execução podem ser movidos, desde que seja solicitada uma operação de re-escalonamento ao EXEHDA. Solicitações de re-escalonamento são colocadas na mesma fila que as de escalonamento de novos OXes.

5.3.4 Política de localização

A política de localização, que seleciona recursos para instalação de carga (vide seção 3.6.4), é uma preocupação central no TiPS. O TiPS tenta encontrar o melhor recurso dentre os disponíveis toda vez que uma solicitação de recurso é recebida.

No TiPS é mantido um histórico dos dados sensorados e dos dados gerados pelos componentes que fazem parte do seu núcleo. Este histórico é mantido por um período controlado por uma janela deslizante de tamanho regulável, e a partir dele são geradas informações que caracterizam o perfil do recurso.

Os dados históricos são utilizados para o cálculo de probabilidades relacionadas à dinamicidade de variáveis monitoradas nos recursos. Estas probabilidades irão auxiliar na definição do perfil do recurso para que, no momento da escolha de um nodo, possa se decidir qual o mais adequado.

5.4 O tratamento das incertezas no TiPS

Para tratar das incertezas relacionadas à disponbilidade de recursos foi proposto o emprego de redes bayesianas. Cada recurso é modelado na forma de uma rede bayesiana que fornece probabilidades com relação ao estado do mesmo. Estas probabilidades são obtidas utilizando o histórico de funcionamento do recurso juntamente com seu estado corrente.

5.4.1 As redes bayesianas e o sensoriamento

Em geral, um tomador de decisões adquire informações do ambiente através de sensores, ajusta suas variáveis de evidência e infere suas crenças com relação ao estado do mundo. Para melhorar o realismo do modelo, é necessário que seja considerada a possibilidade de que os sensores retornem informações incorretas ou com ruído. Para reduzir a influência do ruído nos dados sensorados é necessário trabalhar com um modelo de sensor que estabeleça a possibilidade de ocorrência destas imperfeições. O modelo de sensor é trabalhado na forma de uma tabela de probabilidades condicionais $TPC(E|X)$ associada ao nodo de percepção (SANTOS; PROENÇA, 2002).

O sentido da relação causal é uma propriedade importante na definição estrutural da rede bayesiana. Considera-se que o estado do mundo provoca um determinado valor no sensor, definindo-se desta forma o sentido da relação: a variável "estado do mundo" provoca um efeito na variável "valor do sensor". Como em geral o que é possível obter são valores de sensores, o processo de inferência ocorre no sentido oposto, dado o valor do sensor obtém-se o estado do contexto global.

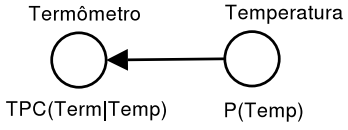

Figura 5.1: Modelo de sensor

A figura 5.1 apresenta um exemplo de rede bayesiana, na qual o termômetro mede a temperatura ambiente. A relação de dependência entre a temperatura indicada pelo termômetro e a temperatura do ambiente é definida através do sentido da flecha que une as duas variáveis. O modelo de sensor é definido por uma tabela de probabilidades condicionais: $TPC(Term|Temp)$.

Desta forma, a tabela de probabilidades condicionais associada a esta rede bayesiana, depende das características de funcionamento do termômetro, tais como valores mínimo e máximo que podem ser medidos. Esta tabela não depende das características particulares do ambiente onde o termômetro está inserido, facilitando assim a sua construção. A tabela define, por exemplo, quais as probabilidades de o termômetro indicar cada uma das possíveis temperaturas, dado que o ambiente está por exemplo a 10^oC. A inferência é feita no sentido oposto, caso o termômetro indique 10^oC, são calculadas quais probabilidades do ambiente estar em cada uma das possíveis temperaturas.

Se o sensor oferece uma informação perfeita, sem ruídos nem erros, então a TPC do sensor é puramente determinística. Em um modelo mais realista, no qual são consideradas as imperfeições do sensor, o ruído e os erros captados são refletidos nas probabilidades de leituras incorretas.

5.4.2 O modelo de sensoriamento do TiPS

O modelo criado para o TiPS utiliza duas variáveis para cada recurso na modelagem do ambiente. Cada uma destas variáveis possui uma instância do modelo de sensor apresentado na seção 5.4.1. A figura 5.2 apresenta o modelo empregado para cada um dos recursos, os nodos *sMétrica* e *sConstância* atuam de forma análoga ao nodo *Termômetro* do exemplo apresentado, recebendo informações de sensoriamento do ambiente. Estes dois nodos de entrada apresentam as seguintes características:

- ***sMétrica*** - o nodo *sMétrica* tem seus possíveis estados discretizados conforme o tipo de informação que será recebida. Um exemplo desta discretização pode ser visto na figura 5.2, a qual possui n estados possíveis. A cada recurso da célula de execução é associada uma instância desta rede bayesiana, o nodo *sMétrica* desta instância recebe probabilidade 1 no estado informado pelo sensor e probabilidade 0 nos outros estados.

- ***sConstância*** - é discretizada em dois estados complementares (*Sim e Não*). A probabilidade de *Sim*, neste texto também chamada de probabilidade de constância, indica a probabilidade de o recurso se manter constante com relação à variável sensorada. A probabilidade de constância é atualizada por um algoritmo de aprendizado que se baseia no comportamento histórico do recurso. A probabilidade de *Não* é sempre atualizada de forma complementar às variações da probabilidade de *Sim*.

Os nodos *Métrica* e *Constância* são representações estimadas do estado e da constância em que o recurso efetivamente se encontra. A relação causal entre o estado real e o sensor geralmente é conhecida, porém a relação inversa é difícil de calcular. Neste caso, a situação é semelhante à apresentada na seção 5.4.1, pois pode-se estabelecer sem muita dificuldade o quanto o estado do recurso influencia na medição do sensor, mas é difícil determinar qual é exatamente o estado do recurso

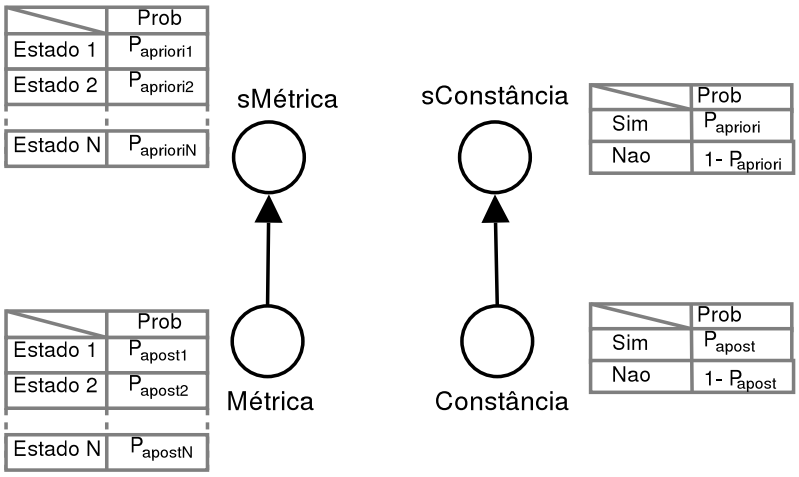

Figura 5.2: Modelo de rede bayesiana adotado

dadas as informações do sensoriamento. Para obter as probabilidades de cada um dos possíveis estados do recurso utiliza-se o teorema de Bayes A.3.1.

As probabilidades *a priori* são estabelecidas na chegada de informação dos sensores, enquanto que as probabilidades *a posteriori* se estabelecem após a aplicação do teorema de Bayes e a conseqüente propagação de probabilidades. Por exemplo, se o dado fornecido pelo sensor indica que o recurso se encontra no "Estado 2" (vide figura 5.2) o valor 1 é atribuído à probabilidade deste estado, os outros estados ficam com probabilidade 0. Após a aplicação do teorema de Bayes, a tabela do nodo "Métrica" é preenchida com os valores de probabilidades para cada um dos estados.

5.5 Efetivando a decisão de escalonamento no TiPS

Para que um tomador de decisões utilize dados diretamente de uma rede bayesiana, seria necessário que ele avaliasse parâmetros distribuídos nos nodos da rede. A utilização de informações advindas de variáveis contidas na rede bayesiana teria que ser integrada ao tomador de decisões, portanto ele teria que conhecer todos os tipos de informações que poderiam vir a ser sensoradas. O diagrama de influência possibilita a integração dos dados da rede bayesiana de forma independente do tomador de decisões, fornecendo um índice que resume o estado do ambiente distribuído. O tomador de decisões passa então a utilizar este índice para selecionar as melhores opções.

Para transformar a rede bayesiana em um diagrama de influência é necessário que se adicione nodos de ganho e nodos de decisão (vide Anexo A). Os nodos de ganho caracterizam os pesos de cada uma das combinações de estados. Estes pesos são multiplicados pela probabilidade dos respectivos estados e depois somados, gerando um índice chamado de utilidade. Os nodos de decisão são a interface propriamente dita para que elementos externos avaliem o estado do ambiente. Os nodos de decisão ficam ligados aos nodos de ganho e informam uma tabela com cada uma das alternativas e suas respectivas utilidades. Quanto maior o valor da utilidade, melhor a alternativa.

A figura 5.3 apresenta o diagrama de influência modelado para a rede bayesiana apresentada na figura 5.2. Neste diagrama, através da introdução das informações

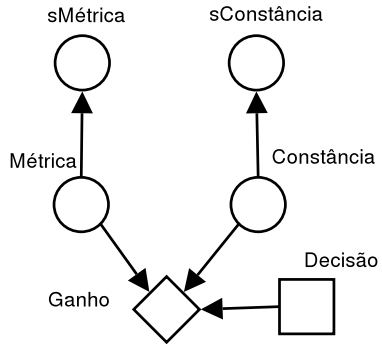

Figura 5.3: O diagrama de influência de um recurso

sensoradas, é gerada uma utilidade que indica a qualidade do recurso.

Para possibilitar a ação do tomador de decisões, é necessário que seja montado um diagrama envolvendo todos os recursos da célula, desta forma possibilitando a geração de utilidades para cada um deles. O diagrama de influência modelado é apresentado na figura 5.4, sendo as utilidades calculadas conforme apresentado na seção A.5. Por este novo diagrama, cada um dos recursos caracteriza uma alternativa para o tomador de decisões. O nodo de decisão passa a interligar os nodos de ganho relacionados a cada um dos recursos.

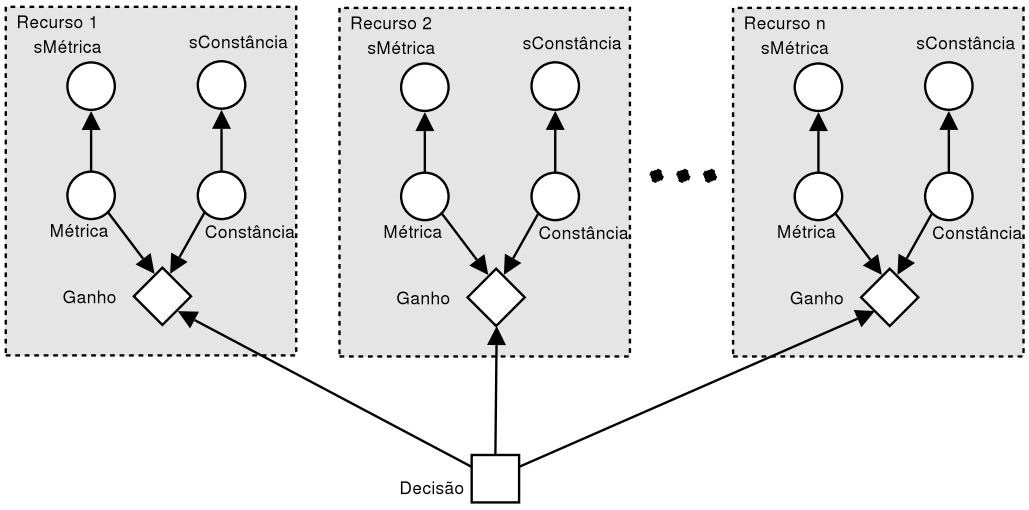

Figura 5.4: Diagrama de Influência

5.6 O aprendizado no TiPS

No TiPS diversas estratégias de aprendizado podem ser empregadas com o intuito de traçar perfis de comportamento do ambiente e dos recursos. As estratégias de aprendizado podem atuar alterando componentes inteiros do TiPS, ou modificando parâmetros de determinados componentes.

No TiPS podem ser utilizadas estratégias de aprendizado *offline* (RUSSELL; NORVIG, 1995), que atuam enquanto o sistema está em repouso. O aprendizado *offline* é realizado a partir de dados coletados ao longo da execução do sistema.

Além disto, podem ser adotadas estratégias aprendizado *online* (RUSSELL;

NORVIG, 1995) que fazem ajustes ao longo da execução baseadas nas informações correntes sensoradas.

As estratégias de aprendizado *online* buscam melhor representar o momento corrente do sistema, sem o ônus de um processamento computacionalmente intensivo, que implicaria a parada total do sistema ou um atraso excessivo nas tomadas de decisão.

Embora o TiPS faculte o aprendizado *offline* tanto de parâmetros das redes bayesianas, como da sua estrutura, devido as características extremamente dinâmicas da computação pervasiva, neste trabalho optou-se pelo uso de uma estratégia de aprendizado *online*. No TiPS o perfil de cada recurso é modelado pela constância, o aprendizado materializado pelo ajuste da probabilidade de constância influencia a utilidade dos recursos. Este ajuste é feito de forma continuada e é alimentado pelos dados sensorados junto aos recursos.

5.7 Caracterizando a operação do TiPS

Para demonstrar o funcionamento do modelo proposto para o TiPS, nesta seção será apresentado um exemplo passo a passo de operação. É considerado que o recurso a ser perseguido pelo escalonador são nodos de processamento com baixa ocupação do processador.

O cenário do exemplo é composto de dois nodos processadores de mesmo poder de processamento. Considera-se que é necessário selecionar um deles para instanciação de um `OX`. É necessário definir a métrica que será empregada para a seleção do recurso, bem como os estados possíveis em que o recurso pode estar, com relação a esta métrica, e suas faixas de valores. O algoritmo de aprendizado, que atualiza as probabilidades do nodo "sConstância", também precisa ser definido.

O nodo de sensoriamento da rede bayesiana é instanciado para operar com o sensor de CPU, ficando o diagrama de influência configurado conforme a figura 5.5. Os nodos "sCPU" e "CPU" foram discretizados em 4 estados de carga de processamento: baixa, leve, média e alta. Nodos com pouca utilização do processador são classificados como carga baixa, enquanto que nodos com bastante ocupação são classificados como carga alta.

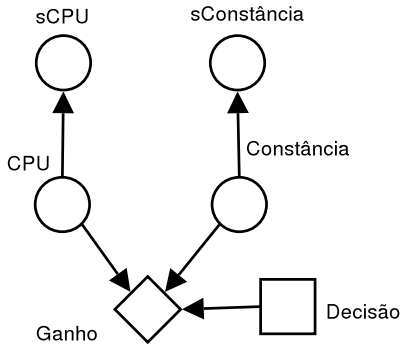

Figura 5.5: Exemplo de instanciação do diagrama de influência

O sensor de ocupação de CPU é instalado nos dois nodos de processamento e a rede bayesiana passa a receber os dados de ocupação do processador. Os valores fornecidos pelo sensor precisam ser classificados segundo a discretização prevista pelo

nodo da rede bayesiana, desta forma, é possível atribuir probabilidade 1 ao estado correspondente ao valor sensorado no nodo "sCPU". Esta discretização é apresentada na tabela 5.1, a qual relaciona os valores de ocupação oferecidos pelo sensor com os possíveis estados da rede bayesiana. O espaço de valores gerado pelo sensor foi dividido uniformemente entre as classes disponíveis no nodo "sCPU".

Tabela 5.1: Discretização da informação do sensor de carga de CPU

Estado	Baixa	Leve	Média	Alta
Carga Sensorada	0 - 24%	25 - 49%	50 - 75%	76 - 100%

Para determinação da constância foi adotado um algoritmo (vide figura 5.6) que compara as duas últimas leituras em uma janela de tempo e ajusta as probabilidades, indicando se o nodo é mais ou menos constante. Sempre que ocorrem variações bruscas no nível de ocupação de CPU a constância é reduzida, conforme o nodo se mantém com o mesmo valor de ocupação de CPU, a constância vai sendo aumentada. O algoritmo é disparado periodicamente para que a avaliação seja feita e para que as probabilidades do nodo "sConstância" sejam atualizadas.

```
read(cpu);
if ( abs(cpu - old_cpu) > threshold)
     decrement_constancy();
else
     increment_constancy();
old_cpu = cpu;
```

Figura 5.6: Algoritmo de aprendizado

Após a determinação das probabilidades nos nodos "sCPU" e "sConstância" da rede bayesiana, é necessário propagar as probabilidades para os nodos "CPU" e "Constância" através do emprego do teorema de Bayes (vide Anexo A). A propagação das probabilidades irá gerar a chamada distribuição de probabilidades *a posteriori*.

Para que o teorema de Bayes possa ser aplicado, é necessário estabelecer as tabelas de probabilidades condicionais que relacionam as variáveis "CPU" e "sCPU", bem como "Constância" e "sConstância". A tabela 5.2 apresenta as probabilidades a serem obtidas no nodo "sCPU" dadas as probabilidades no nodo "CPU", enquanto que a tabela 5.3 apresenta esta relação para os nodos "sContância" e "Constância". Além destas tabelas é necessário definir a distribuição de probabilidades estabelecida para cada um dos nodos de sensoriamento. Neste caso adotou-se uma distribuição uniforme, no caso do nodo "sCPU" foi empregada $P = 0.25$ para cada estado e no caso do nodo "sConstância" foi adotada $P = 0.5$ para cada uma das alternativas. Desta forma considera-se as mesmas chances de ocorrência em cada um dos estados das variáveis.

A distribuição de probabilidades de "CPU" é igual para os dois nodos de processamento em questão. Tanto a propagação das probabilidades entre os nodos "sCPU" e "CPU" e os nodos "sConstância" e "Constância" podem ser calculadas pela aplicação do teorema de Bayes, apresentado na seção A.3.1. A título de ilustração, a situação apresentada neste exemplo para os nodos "CPU" e "sCPU" tem as equações do teorema de Bayes desenvolvidas a seguir:

Tabela 5.2: Probabilidades condicionais do sCPU dado CPU

	CPU			
sCPU	Baixa	Leve	Média	Alta
Baixa	0.80	0.05	0.00	0.00
Leve	0.10	0.80	0.10	0.00
Média	0.10	0.10	0.80	0.05
Alta	0.00	0.05	0.10	0.95

Tabela 5.3: Probabilidades condicionais de sConstância dado Constância

	Constância	
sConstância	Sim	Não
Sim	0.90	0.10
Não	0.10	0.90

$$
\begin{aligned}
P(CPU = Baixa|sCPU = Leve) &= \frac{P(sCPU=Leve|CPU=Baixa)*P(CPU=Baixa)}{P(sCPU=Leve)} \\
&= \frac{0.1*0.25}{P(sCPU=Leve)} \\
P(CPU = Leve|sCPU = Leve) &= \frac{P(sCPU=Leve|CPU=Leve)*P(CPU=Leve)}{P(sCPU=Leve)} \\
&= \frac{0.8*0.25}{P(sCPU=Leve)} \\
P(CPU = Média|sCPU = Leve) &= \frac{P(sCPU=Leve|CPU=Média)*P(CPU=Média)}{P(sCPU=Leve)} \\
&= \frac{0.1*0.25}{P(sCPU=Leve)} \\
P(CPU = Alta|sCPU = Leve) &= \frac{P(sCPU=Leve|CPU=Alta)*P(CPU=Alta)}{P(sCPU=Leve)} \\
&= \frac{0*0.25}{P(sCPU=Leve)}
\end{aligned}
$$

A constante de normalização $P(sCPU = Leve)$ só pode ser calculada após calculados os numeradores da fração, sendo ela neste caso:

$P(sCPU = Leve) = 0.1 * 0.25 + 0.8 * 0.25 + 0.1 * 0.25 + 0 = 0.25$.

Voltando então ao cálculo das probabilidades de "CPU", tem-se:

$$
\begin{aligned}
P(CPU = Baixa|sCPU = Leve) &= \frac{0.1*0.25}{0.25} &= 0.1 \\
P(CPU = Leve|sCPU = Leve) &= \frac{0.8*0.25}{0.25} &= 0.8 \\
P(CPU = Média|sCPU = Leve) &= \frac{0.1*0.25}{0.25} &= 0.1 \\
P(CPU = Alta|sCPU = Leve) &= \frac{0*0.25}{0.25} &= 0
\end{aligned}
$$

Para ilustrar o exemplo, foi considerado que os dois nodos de processamento se encontram com uma carga de ocupação de CPU considerada leve. Porém, no nodo 1 a $P(sConstância = Sim) = 0.3$, indica que este é um nodo com muitas flutuações de carga, já no nodo 2, a carga de processamento apresentou poucas variações, o que levou a $P(sConstância = Sim) = 0.8$, portanto mais elevada (vide figura 5.5). Neste caso será mais vantajoso disparar o objeto ativo no nodo 2, já que este tem

a tendência de se manter com baixas flutuações de carga, possibilitando assim uma melhor condição de processamento.

A figuras 5.7 apresenta o estado em que as redes bayesianas dos nodos de processamento 1 e 2 se encontram no momento em que é requisitada a instanciação do objeto.

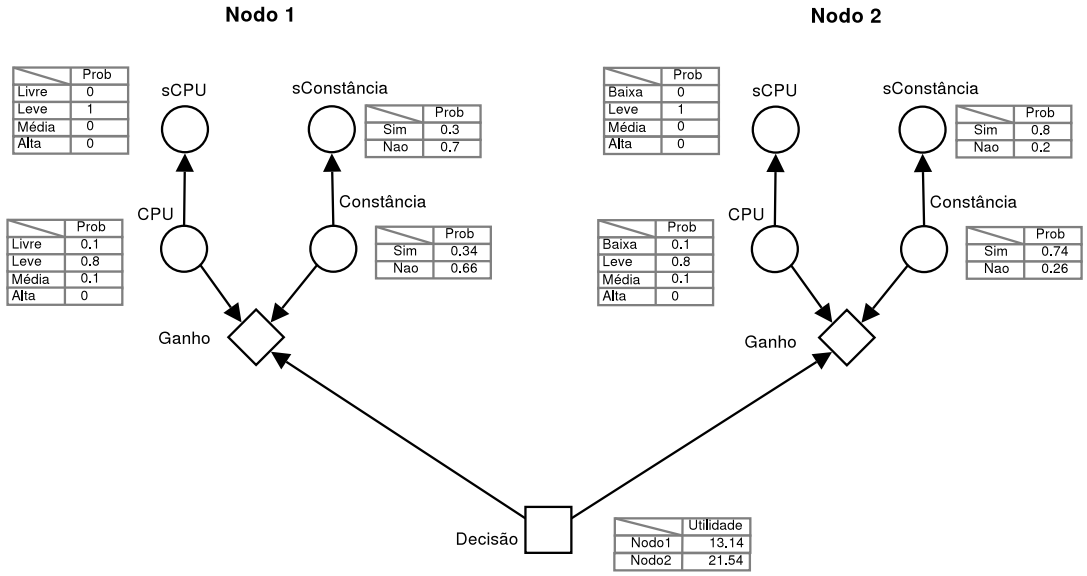

Figura 5.7: Exemplo de comparação entre dois recursos

Para que uma decisão seja tomada é necessário que a utilidade de cada um dos nodos seja calculada e, então, que o nodo com maior utilidade seja escolhido. Para cálculo da utilidade são atribuídos os pesos apresentados na tabela 5.4. Na tabela é possível observar que foram atribuídos pesos maiores aos nodos mais constantes e com cargas baixas e pesos menores aos nodos que se apresentam com carga constante e elevada. Portanto, os nodos considerados piores são os nodos estáveis e ocupados, enquanto que os nodos estáveis e livres são considerados melhores.

As utilidades esperadas para o nodo 1 e nodo 2 são calculadas utilizando a equação A.3, neste caso, da seguinte forma:

$$\begin{aligned} EU(nodo_1) = \; & 0.1 * 0.34 * 100 + 0.1 * 0.66 * 20 + \\ & 0.8 * 0.34 * 30 + 0.8 * 0.66 * 10 + \\ & 0.1 * 0.34 * (-70) + 0.1 * 0.66 * (-40) + \\ & 0 * 0.34 * (-100) + 0 * 0.66 * (-90) = \\ & = 4.72 + 13.44 - 5.02 + 0 = \\ & = 13.14 \end{aligned}$$

$$\begin{aligned} EU(nodo_2) = \; & 0.1 * 0.74 * 100 + 0.1 * 0.26 * 20 + \\ & 0.8 * 0.74 * 30 + 0.8 * 0.26 * 10 + \\ & 0.1 * 0.74 * (-70) + 0.1 * 0.26 * (-40) + \\ & 0 * 0.74 * (-100) + 0 * 0.26 * (-90) = \\ & = 7.92 + 19.84 - 6.22 + 0 = \\ & = 21.54 \end{aligned}$$

Tabela 5.4: Utilidades atribuídas a cada classe

Sim				Não			
Baixa	Leve	Media	Alta	Baixa	Leve	Media	Alta
100	30	-70	-100	20	10	-40	-90

5.8 Os componentes do TiPS

O TiPS foi concebido como uma arquitetura de componentes na forma de um *framework* integrado ao EXEHDA (YAMIN, 2004. 194p). Neste *framework* cada componente possui uma funcionalidade específica e pode ser instanciado de forma independente dos outros. Os componentes possuem interfaces bem definidas, o que permite a substituição de componentes de forma independente do restante da arquitetura. A figura 5.8 apresenta os componentes do *framework* e a forma como eles se organizam. Nas subseções seguintes serão apresentados e detalhados cada um destes componentes, sendo eles: coletor, distribuidor, rede bayesiana, aprendizado, gerador de utilidades, gerenciador de dados, gerenciador de componentes e tomador de decisões.

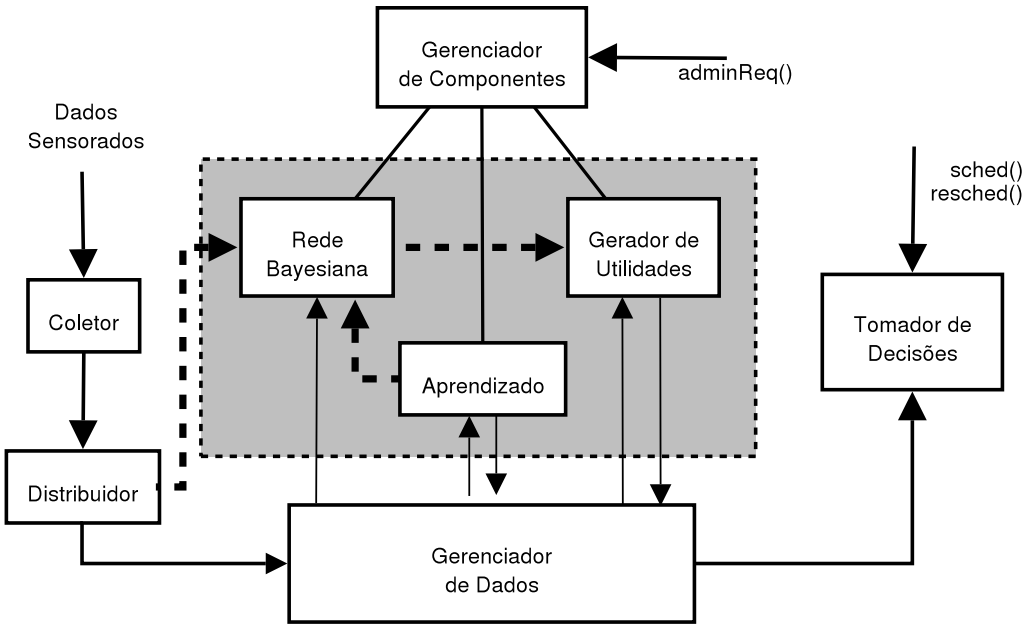

Figura 5.8: O *framework* TiPS

5.8.1 Gerenciador de componentes

O "gerenciador de componentes" é o elemento que controla os principais componentes do núcleo do TiPS. Ele é o primeiro elemento a ser disparado no *boot* da EXEHDAbase e é responsável por ler o arquivo de configurações do sistema e por carregar corretamente os componentes selecionados inicialmente.

Além disso, o gerenciador de componentes também oferece uma interface para requisições administrativas, possibilitando que ao longo da execução sejam alterados parâmetros dos componentes e que sejam trocados componentes do TiPS. A substituição de um componente do TiPS é facilitada em função de os dados ficarem sob a gerência de um componente em separado, desta forma os componentes podem ser

substituídos enquanto o sistema está em execução. Os componentes que fazem parte do núcleo não são responsáveis pelo armazenamento de dados, mas sim por operações sobre estes. Os dados gerados por estes componentes são também enviados ao gerenciador de dados, ficando disponíveis para outros componentes.

5.8.2 Coletor

O "coletor" é responsável por receber os dados monitorados junto aos recursos da arquitetura. Os dados são recebidos pelo coletor na forma de tuplas, contendo informações como identificação do recurso, tipo de sensor e valor atual. Além disso, o coletor é responsável pelo registro dos sensores necessários para o correto funcionamento do escalonador.

A operação de registro dos sensores de interesse consiste em um acesso ao serviço CIB, que além de outras funções, é responsável pela registro de sensores de interesse, pela instalação de sensores nos recursos e publicação das informações observadas.

As tuplas recebidas pelo coletor são traduzidas para um formato interno do escalonador e repassadas para o componente distribuidor.

5.8.3 Distribuidor

O "distribuidor" é o primeiro componente pelo qual os dados sensorados passam e que possui os canais de comunicação com os componentes rede bayesiana e gerenciador de dados. O distribuidor recebe as informações do coletor já em um formato interno, os dados então são repassados ao gerenciador de dados. Após o armazenamento dos dados é necessário sinalizar o componente rede bayesiana da chegada de uma nova informação, o distribuidor também assume este papel.

A partir da sinalização da chegada de novos dados, os outros componentes da arquitetura são ativados. Cada componente então recupera do gerenciador de dados as novas informações e providencia as atualizações necessárias.

5.8.4 Gerenciador de dados

As principais funções do "gerenciador de dados" são armazenar e administrar as informações relevantes para o escalonador. Os outros componentes do escalonador se utilizam dos serviços do gerenciador de dados para leitura e escrita de informações. É também responsabilidade do gerenciador de dados manter um histórico de cada informação recebida, este histórico é mantido seguindo uma estratégia de fila circular. Desta forma é mantida uma janela de tempo das informações, a chegada de dados mais recentes faz com que sejam descartados os dados mais antigos. O tamanho da fila é um parâmetro de configuração deste componente.

O armazenamento em meio permanente também é responsabilidade do gerenciador de dados, em muitos casos isto pode ser importante para que seja mantido um histórico de mais longo prazo do comportamento dos elementos da célula. Os momentos de escrita no disco são selecionados pelo próprio TiPS, que avalia as situações de baixa carga de processamento no computador que hospeda o TiPS, e sinaliza o gerenciador de dados para que este descarregue informações para o disco.

5.8.5 Tomador de decisões

O "tomador de decisões" oferece uma interface para que sejam solicitados recursos para alocação. No momento de uma requisição de escalonamento, o tomador de

decisões acessa o gerenciador de dados, recebendo uma lista ordenada em função dos valores de utilidade atribuídos a cada um dos recursos, com os melhores recursos no momento corrente. A partir desta lista é implementada uma política de escolha do melhor recurso.

São oferecidas três interfaces pelo tomador de decisões:

- direta: na interface direta o tomador de decisões deve estar em operação na mesma JVM que o componente que efetua a solicitação;
- remota: a interface remota é oferecida através de um objeto remoto permitindo que o serviço execute em outra JVM, possivelmente em outro equipamento;
- EXEHDA: o TiPS oferece também uma interface de acesso através do serviço CCManager, o qual gerencia as comunicações entre os componentes do ambiente de execução.

5.8.6 Rede bayesiana

Este componente é responsável pelo processamento da rede bayesiana empregada no TiPS. A rede bayesiana deve ser instanciada para receber informações de um sensor em particular. O componente recebe como informações o estado atual dos recursos e a constância dos mesmos. Após a aplicação do teorema de Bayes é gerada uma distribuição de probabilidades que representa as estimativas da informação sensorada e da constância.

Além das informações recebidas pelos sensores, a rede bayesiana recebe também uma informação relacionada com a constância da informação sensorada. A informação de constância é discretizada nas categorias "Sim" e "Não", com probabilidades complementares. O componente de aprendizado fornece as probabilidades para estas categorias, baseado também nas informações de sensores, utilizando uma estratégia individual por recurso. Um recurso bastante estável, que apresente poucas variações nos dados sensorados, deve possuir uma alta probabilidade de ser constante (categoria "Sim"), um recurso que apresente alto grau de instabilidade possui baixa probabilidade de ser constante.

A rede bayesiana, ao receber os dados dos sensores ou de constância dispara um processo de inferência que propaga os valores de probabilidade das variáveis inter-relacionadas, gerando uma nova distribuição de probabilidades. Esta nova distribuição caracteriza uma estimativa de o recurso estar em cada um dos possíveis estados.

Esta abordagem privilegia a escolha de recursos mais estáveis com relação à informação monitorada. Faz parte da heurística de escalonamento considerar melhores os recursos que apresentam-se mais constantes.

5.8.7 Gerador de utilidades

O "gerador de utilidades" desempenha um papel bastante importante para o processo de tomada de decisão. Ele é responsável pela geração do índice, chamado de utilidade, que indica quais são os melhores recursos. A geração da utilidade é feita a partir das probabilidades obtidas nos nodos "Constância" e "Métrica" da rede bayesiana.

Para geração da utilidade podem ser empregadas diversas técnicas. Algumas fazem com que o escalonador assuma rapidamente as novas características do sistema, enquanto que outras fazem com que as alterações sejam dadas de forma suave.

Para que o TiPS opere com sistemas heterogêneos, faz-se necessário a utilização de dados estáticos que informem as características fixas dos *hosts*. No caso da capacidade de processamento, além do nível de ocupação do processador, é necessário conhecer uma medida de desempenho da máquina quando está em regime de uso dedicado, para o TiPS foi empregado o *benchmark* Linpack (DONGARRA; VORST, 1992). O EXEHDA oferece um sensor que fornece esta informação para cada um dos EXEHDAnodos. A utilidade então é produzida com a multiplicação da utilidade calculada com o diagrama de influência pelo índice de *benchmark* fornecido pelos sensores do EXEHDA.

5.8.8 Aprendizado

O componente de aprendizado é responsável pela atualização das probabilidades de constância do recurso. O componente de aprendizado é ativado periodicamente para que a constância seja avaliada em função dos dados disponíveis.

A figura 5.9 apresenta a idéia geral do algoritmo de aprendizado. O componente opera com um fluxo de execução independente do escalonador, coletando informações sensoradas, avaliando a necessidade de alterar as probabilidades da constância.

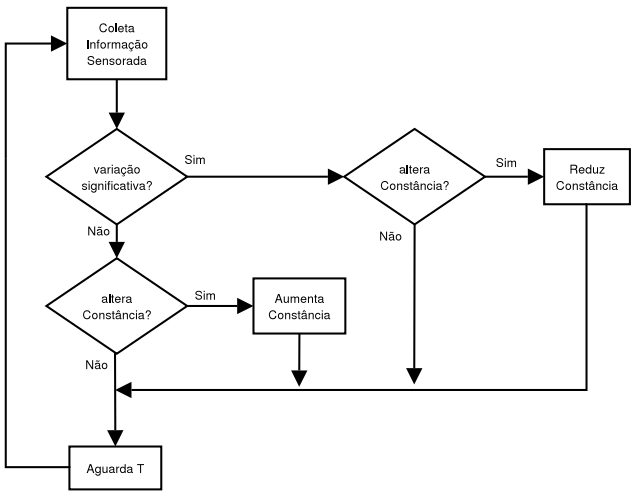

Figura 5.9: Fluxograma geral de aprendizado

No TiPS é empregado um esquema de aprendizado baseado em uma abordagem freqüentista, na qual a contagem de situações específicas é utilizada para construção das probabilidades de constância. Inicialmente é feita a coleta das últimas informações sensoradas, caso tenha ocorrido uma variação significativa com relação aos dados coletados anteriormente e for momento de alterar a constância, esta é reduzida. Caso os dados não apresentem variação significativa a constância pode ser aumentada.

A decisão de alterar o valor da constância não depende somente da variação dos dados sensorados, no caso do TiPS, emprega-se um algoritmo que tende a reduzir mais facilmente a constância em relação ao aumento do valor da mesma. À primeira indicação de variação, a constância é reduzida. Porém a constância só terá seu valor aumentado caso os dados se mantenham estáveis por alguns ciclos de processamento

do algoritmo. Este número de ciclos é um parâmetro do algoritmo, assim como o tempo "T" que é dado como intervalo entre um ciclo e outro.

A concepção deste componente permite que técnicas baseadas em aprendizado por reforço (AR) sejam empregadas (SUTTON; BARTO, 1999), pois algumas características também são encontradas nos ambientes em que o aprendizado por reforço é utilizado. Sistemas baseados em AR são capazes de aprender sem a necessidade de um modelo completo do ambiente, o que também acontece neste caso, em que o escalonador possui uma visão parcial do estado do ambiente. Além disso, ações que resultaram em sucesso podem ser recompensadas nas estratégias que utilizam AR, no caso do TiPS, a recompensa poderia ocorrer na forma de variações nos parâmetros da rede bayesiana de cada recurso, ou nos índices empregados para a geração da utilidade.

Neste capítulo foram tratadas a concepção e a modelagem do TiPS, no próximo capítulo são apresentados o protótipo implementado, bem como os resultados obtidos sob diferentes condições de teste.

6 TIPS: PROTÓTIPO E RESULTADOS

Este capítulo tem como principais objetivos discutir aspectos da implementação do TiPS, bem como avaliar os resultados obtidos. Para tanto serão caracterizados o protótipo construído, os recursos utilizados, bem como a aplicação desenvolvida para os testes e a comparação do TiPS com outros escalonadores. Para as avaliações foi desenvolvido um gerador de carga sintética e outros dois escalonadores utilizados como referência nas comparações.

6.1 Prototipação dos componentes do TiPS

Os componentes do TiPS foram implementados em Java, assim como outros serviços do EXEHDA. O TiPS emprega dois elementos para comunicação com o EXEHDA: o tomador de decisões e o gerenciador de componentes.

O tomador de decisões do TiPS possui uma interface de comunicação com a instância celular do serviço Scheduler do EXEHDA. A principal funcionalidade oferecida por esta interface é a seleção de nodo para processamento. O Scheduler utiliza esta interface para solicitar nodos de processamento ao TiPS toda vez que for necessário instanciar remotamente um novo `OX`.

Por sua vez, o gerenciador de componentes do TiPS consiste de uma ferramenta de administração. Esta ferramenta disponibiliza ao gerente da EXEHDAcel funcionalidade para substituição das implementações dos componentes Rede Bayesiana, Aprendizado e Gerador de Utilidades. Para utilizar esta funcionalidade o gerente da célula pode empregar o utilitário gráfico EXEHDA-AMI (vide figura 6.1), acessando o módulo de administração específico do TiPS. O menu "Services" oferece acesso administrativo a serviços do EXEHDA, tais como o TiPS. Ao selecionar o TiPS, é apresentado ao usuário um sub-menu com as opções "Start", "Stop", para respectivamente, iniciar e parar o serviço e a opção "Configure", que permite que sejam trocados componentes da arquitetura do TiPS.

Os componentes do TiPS que podem ser substituidos em tempo de execução, são carregados no *framework* utilizando a API de reflexão disponível na plataforma Java. A carga de código é feita através do acesso à base de armazenamento pervasiva do ISAM, chamada BDA (YAMIN, 2004. 194p).

O componente Aprendizado (vide figura 5.8) é implementado com `Thread`s Java, possibilitando que este execute de forma concorrente com os componentes que fazem a recepção dos dados sensorados (Coletor, Distribuidor, Gerenciador de Dados) e também com o componente Tomador de Decisões.

Com o intuito de manter a consistência das informações, o Gerenciador de Dados implementa exclusão mútua, utilizando métodos com modificador `synchronized`,

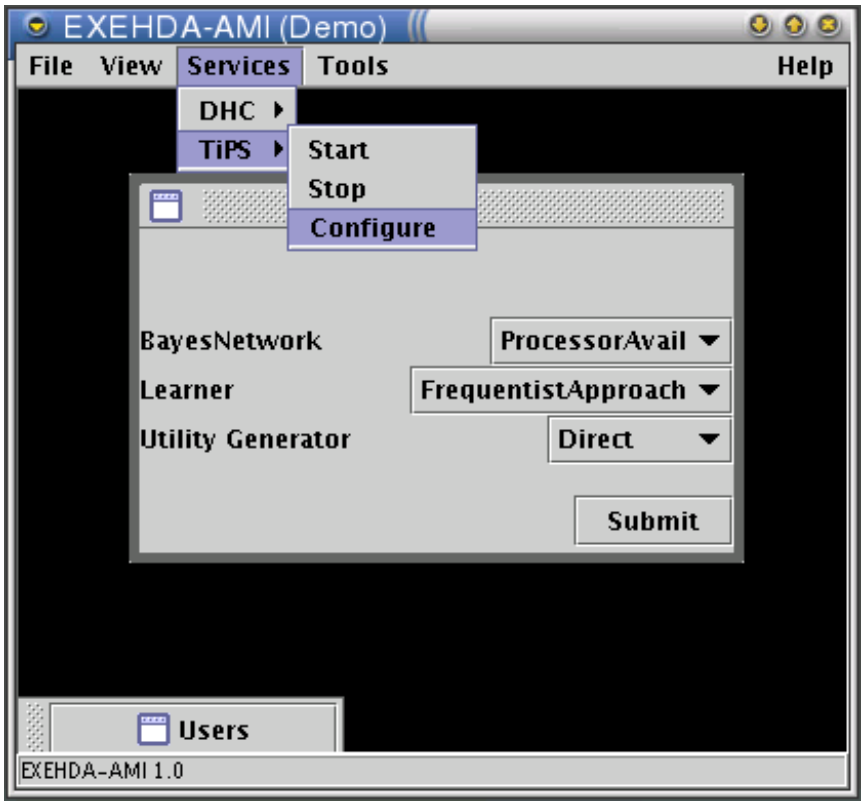

Figura 6.1: EXEHDA-AMI: administração do TiPS

na seção do código que manipula a base de dados. Isto se faz necessário porque o Gerenciador de Dados possui múltiplas *threads*, que atendem concorrentemente a demanda de informações de todos os componentes do *framework* TiPS.

6.2 Integração com o sistema de monitoração

Cada EXEHDAnodo dispõe de um conjunto de sensores de contexto. Estes nodos podem ser configurados de forma particularizada quanto à captação dos dados produzidos pelo sensoreamento. O serviço CIB registra para todos os nodos de uma EXEHDAcel quais sensores estão operacionais.

Com o objetivo de minimizar a sobrecarga nos componentes que capturam e distribuem as informações sensoradas, no EXEHDA os sensores somente irão publicar seus dados, quando estes variarem além de um patamar definido pelo programador.

Quando do desenvolvimento de um aplicativo, o programador de aplicação define a política de adaptação e os respectivos sensores. Os sensores são selecionados por este programador considerando o contexto de interesse da aplicação. Os dados que constituem o contexto são capturados pelo *middleware* e entregues ao TiPS. Caso um sensor com determinado perfil operacional não esteja disponível, a elaboração do mesmo é solicitada a um programador do *middleware*.

6.3 Integração do NWS com o EXEHDA

A proposta do NWS é produzir uma previsão do estado futuro dos recursos monitorados a partir do histórico de dados coletados (vide seção 7.1). Foi feita uma

avaliação comparativa da heurística do TiPS com a heurística de escalonamento baseada nas previsões do NWS. Para isto foram implementados sensores específicos para coleta dos dados gerados pelo NWS. Nesta perspectiva, qualquer sistema que possa executar o NWS, pode fornecer dados de monitoramento para o EXEHDA, sem a necessidade de alterações no código ou na operação do NWS.

O sensor desenvolvido para o EXEHDA que coleta os dados do NWS foi implementado em C++, utilizando a API fonecida pelo NWS e minimizando a intrusão a ser inserida pela coleta dos dados.

6.4 Escalonadores de referência: NWSSched e InstSched

Para avaliar o TiPS foram construídos escalonadores, chamados de NWSSched e InstSched, com propostas diferenciadas. Um deles não faz nenhum tipo de previsão quanto ao possível comportamento futuro dos recursos, e outro reconhecido exatamente pelo uso de uma estratégia que busca prever estados futuros dos recursos para uso nas decisões de escalonamento.

O InstSched é um escalonador que utiliza como parâmetro para a tomada de decisão o percentual de ocupação instantâneo de cada um dos nodos. Esta alternativa é usualmente empregada em propostas de escalonamento em agregados de computadores e, recentemente, na computação em grade.

O NWSSched faz o escalonamento baseado na previsão de ocupação do processador construída pelo NWS para cada um dos nodos do sistema. As previsões entregues ao NWSSched são coletadas por sensores desenvolvidos especificamente para esta finalidade que interagem com o NWS.

6.5 A plataforma de equipamentos empregada

Para avaliação do TiPS foi utilizado o cluster corisco do Instituto de Informática da UFRGS[1], composto de 16 computadores homogêneos, cuja configuração é apresentada na tabela 6.1. Estes computadores são interligados por duas redes: uma rede FastEthernet e outra Myrinet II[2].

Para simular um ambiente dinâmico, no qual cargas de processamento podem surgir independentemente do controle do TiPS, foi elaborado o gerador de cargas "CPUSteal" apresentado na seção 6.7. Com o "CPUSteal" foi possível controlar a carga média imposta a cada nodo de processamento.

Tabela 6.1: Cluster corisco

Processador	Intel Pentium III dual
Freqüência	1.13MHz
Memória	256 MB
Sistema Operacional	Linux (debian) kernel 2.4.22 (otimizado para Pentium III)
Gerenciador de *jobs*	OpenPBS

Os nodos de processamento foram divididos em quatro classes de ocupação (livre, leve, média, alta), conforme a carga de processamento sintética imposta a cada um

[1] http://gppd.inf.ufrgs.br/corisco
[2] http://www.myricom.com/

deles. A tabela 6.2 apresenta os valores das cargas médias de cada uma das classes de nodos.

Tabela 6.2: Percentual médio de ocupação do processador de cada classe de nodo

Livre	Leve	Média	Alta
0%	20%	50%	80%

Neste cenário composto de 16 nodos homogêneos, divididos em 4 classes de ocupação de processador o TiPS foi comparado com as outras duas alternativas de escalonadores.

6.6 A aplicação utilizada

Para avaliação do comportamento do TiPS foi prototipada uma aplicação do tipo mestre/trabalhador para cálculo estimado do número π, chamada "CalcPi".

Na aplicação "CalcPi", o π é estimado através do método de Monte Carlo (PRESS et al., 1992). A precisão do cálculo é dependente do gerador de números aleatórios empregado e do número total de iterações do algoritmo. A seguir são apresentados detalhes do emprego do método de Monte Carlo nesta aplicação.

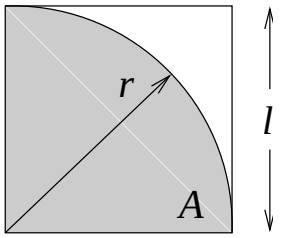

Figura 6.2: Área de $\frac{1}{4}$ círculo

Considerando que A é a área de $\frac{1}{4}$ de círculo representada na figura 6.2, tem-se que:

$$A = \frac{1}{4} * \pi * r^2 \tag{6.1}$$

Pelo Método de Monte Carlo,

$$A = l^2 * \frac{P_{in}}{P_{total}} \tag{6.2}$$

sendo P_{in} os pontos lançados aleatoriamente que ficaram dentro da área em cinza, e P_{total} o número total de pontos gerados.

Unindo as equações 6.1, e 6.2 obtém-se:

$$\frac{1}{4} * \pi * r^2 = l^2 * \frac{P_in}{P_total}$$
$$\pi = \frac{l^2 * \frac{P_{in}}{P_{total}}}{\frac{1}{4} * r^2}$$

Considerando-se $l = r = 1$ obtém-se:

$$\pi = 4 * \frac{P_{in}}{P_{total}} \tag{6.3}$$

O algoritmo de cálculo consiste em gerar pontos $P(x, y)$ sendo $0 \leq x \leq 1$ e $0 \leq y \leq 1$ (vide figura 6.3). Identificar e contar os pontos P_{in} e P_{total}, e aplicar a equação 6.3.

Entrada: número de iterações da execução em n
Resultado: valor estimado do π em PI

1: **para** $P_{total} = 1$ to n **faça**
2: $x \leftarrow$ número aleatório entre 0 e 1
3: $y \leftarrow$ número aleatório entre 0 e 1
4: **se** $(x^2 + y^2 \leq 1)$ **então**
5: $P_{in} \leftarrow P_{in} + 1$
6: **fim se**
7: **fim para**
8: PI $\leftarrow 4 * \frac{P_{in}}{P_{total}}$

Figura 6.3: Algoritmo de cálculo do π

6.6.1 Oportunidades de paralelização da aplicação

Considerando que o processamento de cada uma das iterações pode ocorrer de forma independente, uma das formas de particionar o problema é dividindo o número total de iterações desejadas pelo número de processadores disponíveis. Esta abordagem apresenta problemas de desempenho, pois na barreira de sincronização, ao final da execução, os nodos podem terminar o processamento defasados uns dos outros. Isto ocorre especialmente quando os nodos de processamento tem capacidade de entregar diferentes poderes de processamento em função de diferenças de desempenho de cada processador, ou em função de cargas de processamento que estão sendo impostas concorrentemente à execução da aplicação.

Uma forma de reduzir o problema é particionar o espaço de processamento em um número maior de tarefas do que o número de nodos, desta forma nodos mais poderosos, ou mais livres, podem consumir mais tarefas que outros mais ocupados ou com menor poder de processamento. Por outro lado, a divisão do problema em tarefas muito pequenas acarreta em um maior volume de comunicações, aumentando também o custo de gerência do paralelismo.

Para que a aplicação obtenha um bom desempenho, deve ser encontrado o ponto de equilíbrio entre as duas abordagens.

Na execução paralela desta aplicação foi utilizada uma estratégia de gerar mais tarefas que o número de processadores disponibilizados para o processamento, porém o tamanho das tarefas foi ajustado estaticamente para que se obtivesse uma granulosidade alta. O mestre particiona e gerencia a distribuição das tarefas aos trabalhadores, que as solicitam, processam e retornam os resultados obtidos.

Os nodos do cenário de avaliação são homogêneos e as variações de poder computacional estão diretamente ligadas à carga sintética imposta a cada nodo. Se nodos com carga sintética elevada forem escolhidos para processamento, ocorrerão atrasos no processamento de suas tarefas. Estes atrasos poderão gerar um aumento no tempo total de execução da aplicação, ao passo que a escolha de nodos totalmente livres deve apresentar resultados melhores.

6.7 O emprego de carga sintética

Para as avaliações envolvendo o TiPS, foi concebido e prototipado o "CpuSteal", um gerador de cargas baseado em uma distribuição de probabilidades exponencial.

O "CpuSteal" opera em ciclos de ativação e desativação. Quando ativo, ele gera operações de ponto flutuante ocupando o processador, quando inativo ele entra em repouso. O controle do nível de ocupação média do processador é feito através da determinação de quanto tempo ele permanece ativo e quanto tempo ele fica em repouso em cada um dos ciclos.

Quando o "CpuSteal" é ativado, ele recebe como parâmetro o nome de um arquivo que descreve a carga a ser gerada, indicando os tempos de ativação e os tempos de repouso para cada ciclo de operação. A parametrização através de arquivo de dados permite que os experimentos sejam repetidos várias vezes, mantendo-se as mesmas características de carga. Nos experimentos apresentados neste capítulo, o arquivo descritor de carga é gerado pelo programa "LoadGen", também desenvolvido para realizar a avaliação do TiPS. O "LoadGen" gera os tempos de ativação e desativação baseado em uma distribuição de probabilidades exponencial.

As figuras 6.4, 6.5 e 6.6 apresentam os perfis, de cargas, respectivamente, *leve*, *média* e *alta* utilizados para os experimentos apresentados. É possível observar a média de ocupação pelo nível de preenchimento do gráfico.

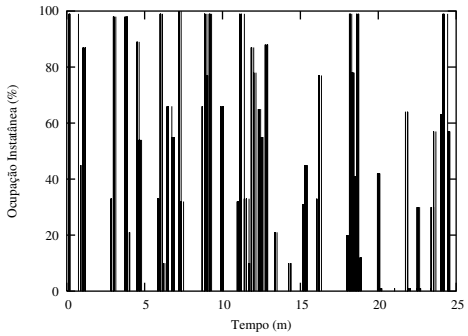

Figura 6.4: Carga sintética de 20% de ocupação média

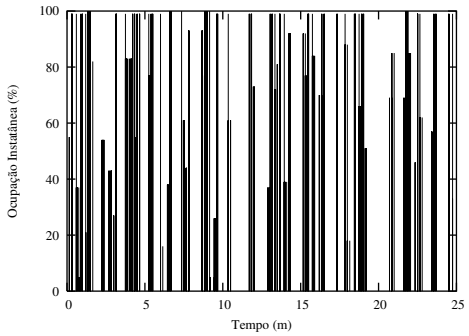

Figura 6.5: Carga sintética de 50% de ocupação média

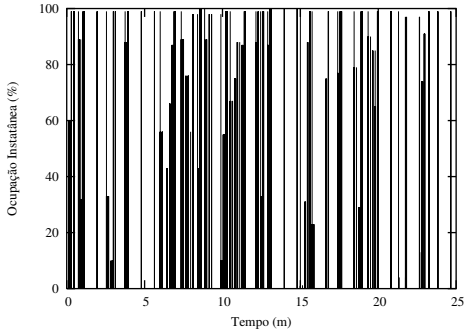

Figura 6.6: Carga sintética de 80% de ocupação média

6.8 Resultados obtidos

Para avaliar o comportamento do TiPS foram feitas execuções da aplicação "CalcPi" nos equipamentos descritos na tabela 6.1. Os nodos do cluster corisco foram divididos em classes e submetidos a diferentes cargas sintéticas. Foram montadas 3 configurações de experimentos, em cada uma delas foi utilizado um número diferente de nodos. A tabela 6.3 resume estas configurações utilizadas em cada um dos experimentos.

Em cada experimento o escalonador tinha por objetivo definir quais nodos iriam participar do processamento da aplicação. Por exemplo, no experimento 3 o escalonador tinha que escolher 4 nodos de um total de 16 nodos disponíveis. Cada experimento foi executado 20 vezes para cada um dos três escalonadores: TiPS, InstSched e NWSSched.

Tabela 6.3: Caracterização dos experimentos

Experimento	Nodos					
	Processamento	Livres	carga de 20%	carga de 50%	carga de 80%	Total
1	2	2	2	2	2	8
2	3	3	3	3	3	12
3	4	4	4	4	4	16

As escolhas do TiPS são guiadas pelos valores de utilidade construídos para cada um dos nodos. A tabela 6.4 apresenta os valores médios das utilidades construídas, observa-se que os valores de utilidade mais altos ocorreram nos nodos livres, enquanto que os nodos com taxa de ocupação média de 80% ficaram com os menores valores de utilidade.

Tabela 6.4: Médias dos valores de utilidade conforme a carga sintética

Média (80%)	Média (50%)	Média (20%)	Média (0%)
-52,78	-32,85	2,54	81,82

Em todos experimentos foram obtidos os tempos de execução da aplicação, o tempo médio de execução, o desvio padrão e os tempos máximo e mínimo de execução. A tabela 6.5 apresenta de forma resumida os resultados dos experimentos 1, 2 e 3.

Observa-se pela tabela 6.5 que o tempo médio de execução da aplicação com o TiPS apresentou-se melhor que os obtidos com as outras as duas alternativas. Embora o NWSSched tenha apresentado na média um bom resultado quanto ao tempo de execução, observa-se que o desvio padrão foi bastante elevado, considerando o tempo total de processamento das execuções. O mesmo ocorreu com o InstSched, que apresentou um resultado global pior que o do TiPS e apresentou um desvio padrão também bastante elevado. Tanto o NWSSched como o InstSched mostram-se pouco estáveis com relação ao tempo de execução da aplicação, parte desta característica pode ser observada nos tempos máximo e mínimo apresentados no experimento.

A tabela 6.6 apresenta o somatório de vezes que cada nodo de processamento foi escolhido por cada um dos escalonadores. A partir deste dado é possível concluir

Tabela 6.5: Resumo do comportamento das execuções

Experimento	**Escalonador**	**Tempo Médio (s)**	**Desvio Padrão**	**Tempo Máximo (s)**	**Tempo Mínimo (s)**
1	TiPS	489,74	1,25	491,98	486,84
	NWSSched	497,13	14,78	522,69	486,95
	InstSched	629,28	50,47	695,43	524,33
2	TiPS	326,492	0,83	327,99	324,56
	NWSSched	331,418	9,85	348,45	324,64
	InstSched	419,519	33,64	463,62	349,55
3	TiPS	244,75	0,64	245,99	243,34
	NWSSched	248,07	7,12	261,69	243,61
	InstSched	324,16	23.66	349,37	266,70

os motivos que levaram o TiPS a obter melhores resultados. Os nodos escolhidos pelo TiPS foram sempre os que estavam sem ocupação do processador, com isto foi possível obter um tempo de execução com melhor regularidade em todos os testes e ao mesmo tempo facultou que fossem praticados os menores tempos de processamento. Por sua vez, tanto o NWSSched como o InstSched tiveram, em alguns casos escolhas ruins durante os testes, contemplando nodos que estavam parcialmente ocupados com a carga sintética.

Tabela 6.6: Número de vezes que cada nodo foi escolhido

Ocupação	20%				50%				80%				Livre			
Nodos	l1	l2	l3	l4	l5	l6	l7	l8	l9	l10	l11	l12	l13	l14	l15	l16
TiPS	0	0	0	0	0	0	0	0	0	0	0	0	20	20	20	20
NWSSched	3	3	1	3	0	0	0	0	0	0	0	0	19	17	16	18
InstSched	1	1	1	1	4	5	4	6	14	14	14	14	1	0	0	0

Os experimentos realizados contemplaram também medições de *speedup*. Foram feitas medições para dois, três e quatro nodos de processamento para cada escalonador. Os *speedups* são apresentados na tabela 6.7, nesta tabela estão caracterizados os casos em que o TiPS se saiu pior, melhor e a média dos resultados. As figuras 6.7, 6.8 e 6.9 apresentam gráficos dos *speedups* obtidos.

Tabela 6.7: Resumos dos índices de *speedup* obtidos

	Nodos	**InstSched**	**NWSSched**	**TiPS**
Pior Caso	2	1.396	1.857	1.973
	3	2.094	2.786	2.959
	4	2.778	3.709	3.946
Caso Médio	2	1.542	1.953	1.982
	3	2.314	2.929	2.973
	4	2.994	3.913	3.966
Melhor Caso	2	1.851	1.993	1.994
	3	2.777	2.990	2.991
	4	3.639	3.984	3.989

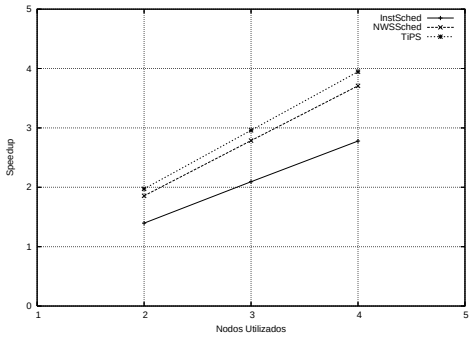

Figura 6.7: *Speedup* obtido no pior caso dos experimentos

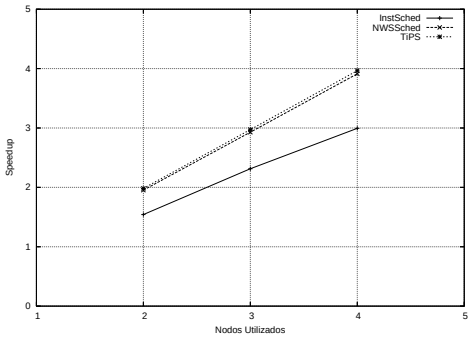

Figura 6.8: *Speedup* médio dos experimentos

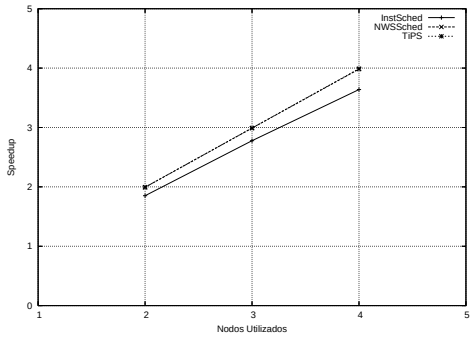

Figura 6.9: *Speedup* obtido no melhor caso dos experimentos

Os resultados, em todos os casos, favoreceram o TiPS e ressaltaram o pior desempenho da estratégia de escalonamento que somente considera valores instantâneos.

No próximo capítulo é feita uma discussão dos principais trabalhos relacionados ao TiPS.

7 TRABALHOS RELACIONADOS

Até o momento do término desta revisão não tinham sido identificados na literatura trabalhos comprometidos especificamente com o tema escalonamento para o ambiente pervasivo. Desta forma, os trabalhos analisados neste capítulo, tiveram como critério de seleção a presença de tópicos de pesquisa que se interseccionam com os temas desenvolvidos no TiPS.

7.1 *Network Weather Service*

O *Network Weather Service* (NWS) é uma ferramenta desenvolvida para prover informações sobre o estado dos recursos em sistemas distribuídos. Esta ferramenta opera com um conjunto distribuído de sensores, os quais fornecem leituras das condições nos diferentes instantes, as quais são empregadas na construçao das previsões. Desta forma, funcionando como um serviço que antecipa o quanto cada recurso poderá disponibilizar para uma aplicação a medida que a execução avança (WOLSKI; SPRING; HAYES, 1999).

O NWS foi desenvolvido com intuito de fornecer informações a escalonadores em um ambiente de computação distribuída, e dentre os principais projetos que o utilizam, destacam-se: o AppLeS (CASANOVA et al., 2000), o Legion (GRIMSHAW et al., 1999) e o Globus (FOSTER; KESSELMAN, 1999).

A Figura 7.1 apresenta a arquitetura lógica adotada no NWS.

Para aumentar a robustez e limitar a intrusão (custo do sensoriamento), o sistema provê um subsistema de controle de sensores que é distribuído e replicado. Os sensores geram pares <*timestamp*, medidas de desempenho>. Os sensores tem sua operação parametrizada pelo NWS, como exemplo de parâmetro configurável temos a freqüência de amostragem.

Objetivando minimizar a utilização de memória, os sensores de desempenho do NWS não mantém leituras anteriores. Os dados sensorados são transferidos para o sub-sistema de armazenamento persistente através de uma interface *socket*. A localização e o número de repositórios de armazenamento persistente são parâmetros a serem definidos na instalação do sistema. Registre-se que novos repositórios podem ser acrescidos com o NWS em operação. Previsões dos níveis operacionais dos recursos são baseadas em dados históricos registrados neste sistema.

O sub-sistema de previsão é extensível, permitindo a inclusão de novos modelos de previsão na biblioteca de previsões. Por questões de flexibilização do uso do NWS, o sistema de previsões pode ser acessado remotamente pelas aplicações, ou carregado diretamente nas aplicações.

Todos componentes de uma instalação do NWS registram-se em um servidor

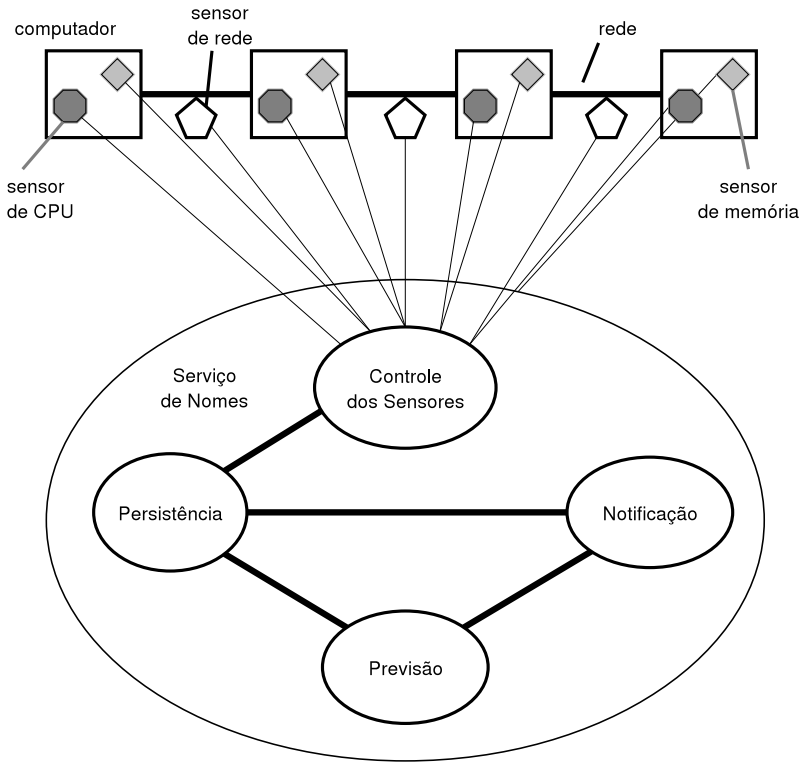

Figura 7.1: A arquitetura lógica do NWS

de nomes. O serviço de nomes mantém informações quanto ao tipo, localização (endereço IP e número de porta) e a configuração de parâmetros associados a cada processo do NWS. O estado geral do sistema é determinado pelos registros ativos contidos em uma dada instância do servidor de nomes.

Cada instância do servidor de nomes faculta uma instalação autocontida do NWS. Múltiplas instalações podem cobrir um mesmo conjunto de recursos através da utilização de espaços de nomes diferenciados para isolar as instâncias do NWS. Esta característica permite fazer depuração e experimentos em um ambiente de produção, mantendo-se assim mais de uma instância do NWS em funcionamento ao mesmo tempo.

7.1.1 Monitores de desempenho

No NWS existem duas principais categorias de monitores de desempenho: passivos e ativos.

Um monitor passivo emprega métricas que já foram coletados por um mecanismo básico do recurso, por exemplo o sistema operacional nativo. Tomando como exemplo o nível de ocupação do processador, quase a totalidade dos sistemas UNIX implementa um mecanismo de monitoramento neste sentido. Assim, o NWS pode, interagindo com o sistema operacional adquirir esta informação. A principal característica do sensoriamento passivo é provocar baixa intrusão.

Por sua vez, um monitor ativo consiste de um processo concebido especificamente para coleta de informações no recurso sensorado. Por exemplo, um monitor de desempenho para grade pode periodicamente executar um processo no recurso a ser monitorado e armazenar a resposta de desempenho observada. Esta abordagem ativa tem a vantagem de representar melhor a relação da medida monitorada

com o desempenho que efetivamente pode ser obtido. O problema com esta abordagem é que o monitor precisa carregar completamente o recurso para fazer uma medida confiável, desta forma deixando por um curto espaço de tempo o sistema sobrecarregado.

O NWS implementa um sensor de CPU que combina as estratégias passiva e ativa, deste modo são integradas a média de carga do Unix e medidas ativas de CPU. O sensor lê o valor da média de 1 minuto periodicamente, conforme definido em parâmetro de inicialização do sensor. Além disto, ele também inicia um processo CPU intensivo (chamado de CPU *probe*) com uma periodicidade bem menor que a do sensor, e armazena as taxas de utilização experimentadas. A duração de cada teste de CPU é um parâmetro de inicialização.

Uma vez obtidos os valores ativos e passivos da monitoração, o NWS converte a média de carga em uma estimativa de utilização. O NWS assume que a fila de execução será servida seguindo um algoritmo *round-robin* e que todos trabalhos são CPU *bound*, portanto é esperado que uma fatia de tempo igual seja dada para cada um deles. O sensor combina ambos, o índice gerado pelo CPU *probe* e a média de carga fornecida pelo sistema operacional, é calculado um índice de ajuste. Se, por exemplo, a previsão baseada na média de carga for 10% menor que o observado com o CPU *probe*, o índice de ajuste resulta em 10. Caso contrário, se a média de carga super-estrimar a utilização, o índice resultante é negativo e proporcional a diferença.

7.1.2 Previsão

O mecanismo de previsão do NWS se vale de diversos métodos de predição. Cada método de previsão é configurado no sistema e parametrizado individualmente. A medida que a nova predição é gerada, baseada no histórico existente de medições e predições, é tido como método de predição mais adequado aquele que atingir melhor precisão.

A previsão é gerada sob demanda, baseada em um histórico existente de medições e predições. Ou seja, para cada método de previsão f, no instante de medição t, tem-se:

$$prediction_f(t) = METHOD_f(history_f(t)) \tag{7.1}$$

onde:

- $prediction_f(t)$: o valor da previsão feita pelo método f para a medida no instante $t + 1$;
- $history_f(t)$: um histórico finito de medições, previsões e erros de previsões geradas anteriormente ao tempo t usando o método f;
- $METHOD_f$: método de previsão f.

Cada método recebe as medições, sendo responsável por manter o histórico das previsões construídas.

$$err_f(t) = value(t) - prediction_f(t - 1) \tag{7.2}$$

Na equação (7.2) é apresentado o cálculo do erro residual associado a medida $value(t)$ obtida no tempo t e uma predição gerada pelo método f para aquela medida no tempo $t-1$.

As técnicas de previsão usadas como primárias no NWS incluem métodos baseados em médias e medianas para produção de previsões. Baseando-se no fato de que dados mais recentes são mais indicativos das condições atuais, os preditores primários variam a quantidade de história utilizada, através de técnicas de janelas deslizantes. Além disto, são também utilizadas técnicas de amortecimento exponencial para produção das previsões.

7.1.3 A relação do NWS com o TiPS

O foco da comparação entre o TiPS e o NWS está relacionado ao monitoramento e construção de perfis de comportamento dos recursos. No TiPS as previsões apresentam-se na forma de um índice cujos valores correspondentes aos diversos recursos, são relativos entre si. Por sua vez, no NWS os índices fornecidos como previsão possuem representação direta, podendo ser avaliados individualmente.

Outro aspecto diz respeito a integração das informações correspondentes aos diversos sensores, no TiPS esta integração acontece de forma orgânica a sua operacionalidade, enquanto que no NWS caso o escalonador tenha interesse em unir mais de uma variável, é necessário que isto seja feito por outro elemento externo ao mesmo.

O NWS foi projetado para fornecer previsões de futuro próximo, que são válidas por um curto período de tempo. O TiPS procura traçar um perfil de comportamento dos recursos, construindo previsões válidas por um período maior.

No NWS as comunicações de sensoriamento são realizadas de forma periódica, consumindo recursos da rede mesmo quando não ocorram alterações nos dados observados localmente. Por sua vez, os sensores utilizados no TiPS publicam as informações que coletam quando ocorrerem mudanças nos valores, otimizando assim a utilização da rede de interconexão.

7.2 Condor

O Condor é um sistema de escalonamento desenvolvido no âmbito de um projeto de pesquisa da Universidade de Wisconsin-Madison. Inicialmente o sistema funcionava apenas no departamento de ciências da computação, atualmente atinge toda a universidade, totalizando cerca de 1000 estações de trabalho. Além disso, tem uma versão distribuída livremente através do endereço `http://www.cs.wisc.edu` e é utilizado em diversos centros de pesquisa e outras universidades no mundo (TANNENBAUM et al., 2001).

A pesquisa realizada por Matt Mutka & Miron Livny (1988), ainda na década de 80, impulsionou significativamente o projeto Condor. Nesta pesquisa foi observado um conjunto de estações de trabalho durante cinco meses. Esta análise concluiu que no máximo 30% da capacidade disponível de processamento era efetivamente utilizada. O estudo revelou também que, além dos finais de semana e das noites, muitos equipamentos também ficavam ociosos por períodos de tempo significativos durante o horário normal de trabalho dos usuários. Estas observações colaboraram com a idéia de que as estações de trabalho de uma instituição eram boas candidatas a fornecedoras de ciclos de processamento.

O Condor é direcionado para trabalhos de longa duração, os quais irão se valer dos processadores ociosos da instituição. Dentre as contribuições científicas obtidas com o Condor, destacam-se: a análise dos padrões de utilização de estações de trabalho, o projeto de algoritmos para gerenciamento da capacidade de processamento das estações e o desenvolvimento de ferramentas para execução remota (LITZKOW; LIVNY; MUTKA, 1988).

7.2.1 A estrutura de escalonamento

O escalonamento de trabalhos no Condor libera o usuário de especificar os nomes dos equipamentos e quais serão efetivamente utilizados, permanecendo necessário estabelecer os requisitos mínimos da aplicação, como por exemplo, quantidade de memória.

As decisões de escalonamento no Condor são feitas de forma centralizada. Um coordenador central, instalado em um dos equipamentos é responsável pela distribuição dos trabalhos para as estações que efetivamente vão realizar o processamento.

O coordenador central consulta periodicamente as estações para verificar quais delas estão disponíveis para servir como fonte de ciclos de processamento, além disto são também verificadas quais estações possuem trabalhos esperando para serem executados.

Cada escalonador local verifica se a estação pode servir como fonte de trabalho para execução remota, dois são os critérios empregados: (i) a presença de usuário interativo e (ii) o tamanho da fila de processos aguardando para serem executados.

Por outro lado, trabalhos podem ser migrados entre as estações caso o coordenador central detecte que existem estações com muitos trabalhos esperando pelo processador, enquanto que outras estações estão livres. O *login* de um usuário também dispara um procedimento de migração para um outro equipamento que esteja disponível.

As estações de trabalho operam de forma autônoma, e o escalonamento local de processos continua funcionando mesmo que o coordenador central falhe.

7.2.2 Tolerância a falhas no Condor

O mecanismo de *checkpointing* oferecido pelo Condor permite que trabalhos interrompidos possam ser reiniciados sem que todo processamento feito até então seja perdido. Este mecanismo faz periodicamente cópias remotas do estado dos processos em execução e em caso de necessidade de restauração de um processo, a ultima cópia de seu estado é utilizada (TANNENBAUM et al., 2001).

Este mecanismo é importante para o processamento distribuído de trabalhos de longa duração. Considerando que os computadores utilizados pelo Condor são equipamentos de uso pessoal, ou máquinas de laboratório para uso geral, é possível que algumas destas possam ser desligadas ou falhem durante o processamento. O mecanismo de *checkpointing* permite que o trabalho processado até um momento próximo da falha seja recuperado.

Para que uma determinada aplicação se utilize do mecanismo de *checkpointing*, ela precisa se valer de uma biblioteca do Condor.

Para armazenar os dados relacionados com o estado dos processos o Condor oferece um servidor de *checkpoint* que funciona de forma centralizada. Este servidor recebe as informações, as armazena e as retorna no momento em que um processo for restaurado.

7.2.3 Atuais frentes de trabalho

O projeto Condor envolve uma equipe de trabalho de tempo integral, estudantes da Universidade de Wisconsin-Madison de graduação e pós-graduação. São avaliadas atualmente cinco principais frentes de trabalho (THAIN; TANNENBAUM; LIVNY, 2002):

- **pesquisa em computação distribuída:** nesta frente de trabalho, busca-se explorar o poder de processamento de recursos em regime compartilhado e dedicado; oferecer serviços de gerenciamento de *jobs* para aplicações de computação em grade; gerenciamento de serviços de baixo nível para recursos de grade; descoberta, monitoramento e gerenciamento de recursos; e tecnologia de distribuição de entrada e saída;

- **engenharia de software:** o Condor, por ser utilizado pela indústria, governo e pela academia, é tratado como um software de produção. Duas versões, a estável e a de desenvolvimento, são evoluídas simultâneamente em ambiente multiplataforma (Unix e Windows);

- **manutenção dos ambientes de produção:** a equipe do projeto é também responsável pela instalação do Condor no Departamento de Ciências da Computação na Universidade de Wisconsin-Madison;

- **ensino para estudantes:** o treinamento de estudantes para que se tornem cientistas da computação também faz parte das atividades do projeto. Parte deste treinamento ocorre em um ambiente de produção.

7.2.4 Relacionando o Condor e o TiPS

O Condor foi desenvolvido para aplicações de alto desempenho executando em redes corporativas prevendo a utilização de equipamentos de forma compartilhada com usuários locais. O TiPS apresenta um perfil mais eclético compreendendo aplicações não só da área de alto desempenho, fazendo parte de um ambiente de execução.

Dentre outras diferenças, destaca-se que, o Condor faz a escolha dos recursos a serem utilizados em função do estado instantâneo dos mesmos. O TiPS, por sua vez, utiliza uma abordagem probabilística para seleção dos recursos, construindo ao longo da execução, um perfil comportamental dos mesmos.

7.3 APST: *AppLeS Parameter Sweep Template*

O APST é um *middleware* para ambientes de grade que utiliza técnicas de escalonamento no nível da aplicação. Este *middleware* é direcionado para aplicações paralelas de troca de parâmetros (*parameter sweep*). Estas aplicações são tipicamente estruturadas como conjuntos de "experimentos", cada qual é executado com um conjunto distinto de parâmetros. Dentre as preocupações de projeto do APST estão o escalonamento adaptativo e a distribuição dos dados de forma eficiente (CASANOVA et al., 2000).

O principal objetivo do APST é oferecer um *framework* para que de forma facilitada e eficiente possam ser desenvolvidas, escalonadas e executadas aplicações de troca de parâmetros em larga escala.

7.3.1 Modelo de aplicação e do ambiente

No APST uma aplicação de troca de parametros é definida como um conjunto de tarefas seqüênciais independentes, não existe ordem de precedência entre as tarefas. Assume-se que a entrada de dados para cada tarefa é um conjunto de arquivos e um mesmo arquivo pode servir de entrada de dados para diversas tarefas.

O ambiente é constituído de uma grade computacional composta por *sites* acessíveis pela rede. Cada *site* contém recursos computacionais chamados *hosts*, assim como recursos de armazenamento. É crucial em termos de desempenho que, nas aplicações do tipo de troca de parâmetros, os *hosts* de um determinado *site* compartilhem áreas de armazenamento. Normalmente o número de tarefas computacionais na aplicação será bem maior que o número de processadores disponíveis.

Para acesso remoto aos recursos está prevista a utilização de diversos projetos orientados para a infraestrutura de grade (FOSTER; KESSELMAN, 1997; CASANOVA; DONGARRA, 1996; LITZKOW; LIVNY; MUTKA, 1988). Uma das alternativas para implementação do sistema distribuído de armazenamento parte da integração com o GASS, o serviço de arquivos do Globus (BESTER et al., 1999).

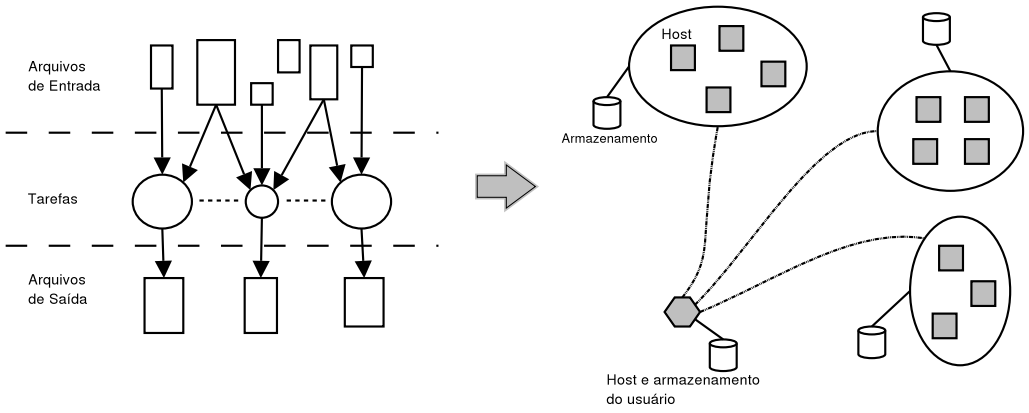

Figura 7.2: Modelo de aplicação e de ambiente

A Figura 7.2 apresenta tanto o modelo de aplicação como o modelo do ambiente distribuído. Não sendo imposta nenhuma restrição com relação as características de desempenho dos recursos. Porém, são utilizados algoritmos de escalonamento que necessitam de estimativas de poder de processamento e tempos de transferência de arquivos. Estas estimativas podem ser informadas pelo usuário, por sistemas como o NWS (WOLSKI; SPRING; HAYES, 1999) e Remos (LOWEKAMP et al., 1998) ou por serviços de grade tais como os oferecidos pelo Globus (FOSTER; KESSELMAN, 1997).

7.3.2 O algoritmo de escalonamento

O algoritmo de escalonamento empregado no APST, chamado `schedule()`, é apresentado por Henri Casanova *et al.* (2000). A estratégia do algoritmo baseia-se em estimativas do desempenho dos recursos. A partir destas, é gerado um plano determinando quais transferências de arquivos irão ocorrer sobre quais ligações da rede, bem como quais tarefas serão alocadas a quais *hosts*. A cada vez que o `schedule()` é ativado o escalonador deve possuir conhecimento sobre: (i) a topologia corrente da grade, isto é, número de agregados, número de *hosts* em cada agregado, carga

de CPU e de rede; (ii) o número e a localização das cópias de todos os arquivos de entrada; e (iii) a lista de computações e transferências em andamento ou já completadas.

```
schedule() {
  (1) compute do próximo evento de escalonamento
  (2) gere um gráfico de Gantt, G
  (3) para cada computação e transferência em andamento
        gere uma estimativa do tempo de término
        preencha os espaços correspondentes em G
  (4) selecione um subconjunto de tarefas que ainda não iniciaram: T
  (5) até que cada host possua trabalho suficiente
        atribua de forma heurística tarefas a hosts (preenchendo os
        espaços em G)
  (6) converta G em um plano
}
```

Figura 7.3: Esqueleto do algoritmo de escalonamento

O esqueleto do algoritmo `schedule()` é apresentado na Figura 7.3. A seguir é detalhado o funcionamento de cada passo do algoritmo:

(1) determina o próximo momento em que o escalonador entrará em ação. O comportamento do ambiente pode fazer com que a freqüência de eventos de escalonamento seja aumentada ou reduzida. Uma freqüência maior oferece maior capacidade de adaptação, porém aumenta os custos de escalonamento;

(2) cria um gráfico de Gantt (G) que será usado para manter conhecimento das atribuições de tarefas para as *hosts*;

(3) insere espaços em G, correspondentes aos tempos para término das tarefas em execução e das transferências de dados;

(4) faz uma seleção das tarefas a serem executadas;

(5) é o núcleo do algoritmo, determinando qual tarefa será atribuída a qual *host*;

(6) G é convertido em uma seqüência de instruções; esta seqüência provê um escala para distribuição das ações junto aos serviços de grade.

A forma como o esqueleto do algoritmo foi concebido possibilita a implementação independente de cada um dos passos a serem executados, tornando possível a experimentação com diferentes técnicas e estratégias. O objetivo principal desta abordagem é o de instanciar algoritmos especializados por aplicações e por tipos específicos de ambiente. Está previsto que a instanciação seja a mais dinâmica e automática possível para que o algoritmo possa se reconfigurar sem que a execução do sistema tenha que ser interrompida.

O APST está em fase de desenvolvimento, e os resultados de escalonamento apresentados até o presente momento estão mais relacionados ao desenvolvimento do passo número (5) da Figura 7.3.

Henri Casanova *et al.* (2000) propõe que o passo (5) do `scheduler()` seja instanciado com três diferentes heurísticas, todas considerando tarefas independentes para

um ambiente uniforme e com apenas um usuário: *Min-min*, *Max-min* e *Sufferage*. Estas três estratégias atribuem iterativamente tarefas a processadores considerando todas tarefas não escalonadas e calculando os tempos mínimos de conclusão (TMC). Para cada tarefa é calculada uma estimativa de tempo de execução da mesma sobre cada um dos recursos e a partir destas informações obtém-se o tempo mínimo de conclusão da tarefa. Para cada tarefa é calculada uma métrica baseada nos TMCs e a tarefa com melhor métrica é então atribuída ao recurso que lhe oferece esta possiblidade. Este processo é repetido até que todas tarefas tenham sido escalonadas.

A heurística *Min-min* utiliza o TMC mínimo como métrica, desta forma atribuindo maior prioridade às tarefas que podem ser completadas mais cedo. A motivação por trás da utilização de *Min-min* é de atribuir tarefas aos *hosts* que as executarão o mais rápido possível, objetivando reduzir desta forma o tempo total de execução.

A métrica gerada pela heurística *Max-min* é o máximo TMC. A idéia por trás da heurística é de sobrepor tarefas de longa duração com tarefas de curta duração.

A heurística *sufferage* procura atribuir tarefas aos *hosts* de forma que estas sejam "prejudicadas" ao mínimo. A medida do prejuízo é dada pela diferença entre os dois melhores TMCs obtidos. As tarefas com maior índice de prejuízo tem preferência.

As heurísticas consideram, para cálculo dos TMCs, os tempos de transferência de dados e que alguns destes dados já podem estar presentes nos locais de armazenamento remoto. Esta característica modifica um pouco o comportamento das heurísticas em relação a suas definições originais.

7.3.3 Relacionando o APST com o TiPS

Um aspecto significativo na diferença entre o TiPS e o APST, é este segundo ser direcionado para aplicações de troca de parâmetros, enquanto que o TiPS não apresenta comprometimento com um tipo particular de aplicação.

Outro aspecto que o diferencia do TiPS é que o APST não contempla um mecanismo de previsão do estado dos recursos no que diz respeito ao sensoriamento de recursos.

7.4 PaRT

O PaRT (*Parallel Ray Tracer*) é um *ray tracer* desenvolvido por Luís Paulo Peixoto dos Santos & Alberto Proença (2002) para avaliação de estratégias de gerenciamento de dados e carga de processamento.

Aplicações como o *ray tracing* podem ser decompostas em tarefas paralelas, com processamento independente. Além disto, também é característica deste tipo de problema possuir um número máximo de tarefas em que o mesmo pode ser dividido.

No PaRT foram incluidas cinco diferentes estratégias de escalonamento em nível de aplicação, são elas: distribuição uniforme de carga, alocação de trabalho por demanda, uma política de escalonamento determinística baseada em sensores e duas abordagens probabilísticas. A seleção da abordagens a ser empregada é feita no disparo da aplicação.

7.4.1 Distribuição uniforme de carga

Na abordagem de escalonamento com distribuição uniforme de carga a alocação de tarefas é feita de forma estática e o escalonador não tenta otimizar o processa-

mento da aplicação ao longo da execução da mesma. A imagem é dividida em tantas colunas quantos nodos houverem, sendo todas colunas de igual largura. Cada uma das colunas é atribuída a um dos nodos e nenhum outro tipo de ajuste de carga é realizado.

7.4.2 Alocação de trabalho por demanda

Nesta abordagem o escalonador central inicialmente subdivide a imagem em um determinado número de sub-regiões, as quais constituem as tarefas. O número de tarefas deve ser maior que o número de nodos disponíveis para processamento. No início do processamento, o escalonador atribui uma tarefa para cada nodo e ao término da tarefa, cada nodo irá solicitar outra para processamento. Este processo se repete até que todas as tarefas tenham sido processadas.

7.4.3 Estratégia determinística baseada em sensores

Nesta abordagem o escalonador utiliza informações sobre o estado corrente do sistema e também informações sobre o perfil da carga a ser processada. Inicialmente a imagem é dividida em tantas colunas quantos forem os nodos de processamento, todas de igual tamanho, da mesma forma como é feito na distribuição uniforme de carga. Porém, nesta abordagem os nodos de processamento enviam informações sobre o estado do ambiente ao escalonador central. Baseado nestas informações, o escalonador central classifica os nodos como receptores ou fornecedores de trabalho, para que sejam feitas eventuais migrações de tarefas.

7.4.4 Estratégia baseada em redes de decisão

Esta abordagem utiliza as mesmas informações sobre o estado do sistema que a estratégia determinística, porém o tratamento dado às informações coletadas é probabilístico. O trabalho é previamente subdividido e depois ocorrem trocas de carga de trabalho entre os nodos. O ambiente é modelado em um conjunto de variáveis aleatórias e as relações entre estas variáveis são especificadas através de tabelas de probabilidade condicional. Para cada par de nodos é avaliada a efetividade de uma troca de trabalho entre os mesmos, bem como qual será o fornecedor e o receptor.

7.4.5 Estratégia baseada em redes de decisão com adaptação

A estratégia de redes de decisão com adaptação é uma evolução da estratégia de redes de decisão sem adaptação. Neste caso foi utilizada a mesma rede de decisão, porém alguns parâmetros da rede eram atualizados dinamicamente ao longo da execução. Este processo é um mecanismo de aprendizado *online*. Estas adaptações permitem que erros cometidos na parametrização inicial da rede sejam corrigidos ao longo do tempo, e que o modelo do ambiente se adapte às variações dinâmicas que ocorrem no mesmo.

7.4.6 Avaliação das estratégias empregadas

Luís Paulo Peixoto dos Santos & Alberto Proença (2002) apresenta como piores resultados aqueles obtidos com a utilização de distribuição uniforme de carga. Os resultados imediatamente melhores foram os baseados na estratégia de distribuição por demanda. Por fim, ganhos significativos foram obtidos com a utilização da

estratégia determinística baseada em sensores.

De modo geral a aplicação escalonada utilizando redes de decisão apresentou melhores resultados que as baseadas em estratégias determinísticas (SANTOS; PROENÇA, 2002).

Além disto, Luís Paulo Peixoto dos Santos & Alberto Proença (2002) apresenta a estratégia adaptativa com redes de decisão como sendo a que obteve melhores resultados na execução da aplicação de *ray trace*, considerando a execução em um ambiente com variações dinâmicas de carga externa de processamento.

7.4.7 Avaliando o TiPS em relação ao PaRT

O principal ponto em comum entre o TiPS e o PaRT é o emprego de mecanismos probabilísticos, porém no TiPS as redes de decisão são utilizadas para seleção dos melhores nodos para processamento.

Diferentemente do TiPS, o escalonamento efetuado pelo PaRT acontece no nível de aplicação, portanto comprometido com a natureza do problema que está sendo resolvido. Por sua vez, no TiPS foi dado um tratamento mais genérico, que pudesse ser utilizado por aplicações de um número maior de tipos de aplicação.

7.5 Comparativo dos trabalhos em relação ao TiPS

Nesta seção é apresentado um resumo comparativo entre o TiPS e os outros sistemas avaliados. Na tabela 7.1 são apresentadas características desejáveis no tratamento do escalonamento em um ambiente de computação pervasiva e marcados os trabalhos que apresentam tais características.

Tabela 7.1: Comparativo de características entre o TiPS e os demais sistemas analisados

Característica	**NWS**	**Condor**	**APST**	**PaRT**	**TiPS**
Utiliza estratégias probabilísticas para tomada de decisão				•	•
Atende diversos tipos de aplicações	•	•			•
Substituição de componentes durante a execução					•
Parametrização por parte da aplicação	•		•		•
Publicação de dados sensorados disparada por limiares					•
Abordagem de *framework*	•		•		•
Direcionado para a computação pervasiva					•

8 CONCLUSÃO E TRABALHOS FUTUROS

Este capítulo revisita os quesitos de pesquisa considerados no desenvolvimento deste trabalho, ressaltando as principais contribuições e caracterizando as oportunidades de trabalho futuro na frente de pesquisa pertinente ao TiPS.

8.1 Conclusão

Neste início de década, observa-se a transformação da área de computação pervasiva de um conotação de interesse emergente para outra caracterizada por uma demanda real e qualificada de produtos, serviços e pesquisas. O projeto ISAM, em desenvolvimento no II/UFRGS, está alinhado com esta tendência internacional, e os problemas de pesquisa neste contexto motivaram trabalhos que vão desde a linguagem de programação até as camadas mais baixas do ambiente de execução.

A complexidade do escalonamento na computação pervasiva é fruto da heterogeneidade dos recursos e da dinamicidade de seu estado, seus usuários e das aplicações em execução.

Os escalonamentos do TiPS atingiram resultados melhores que os obtidos com um escalonador baseado no NWS e, por sua vez, resultados significativamente melhores que aqueles produzidos por um escalonador baseado em informações instantâneas sobre o ambiente. Estes resultados caracterizaram a adequação do emprego de estratégias probabilísticas, particularmente redes bayesianas, para tratamento de incertezas nas tomadas de decisão em ambientes dinâmicos

Considerando que estratégias de escalonamento utilizando o NWS e informações instantâneas são amplamente utilizadas, a possibilidade de emprego do TiPS fica potencializada no cenário de computação pervasiva e em grade.

8.1.1 Contribuições da pesquisa

Cientificamente, esta dissertação tem como principal contribuição a proposição de uma heurística de escalonamento direcionada à Computação Pervasiva.

Outra contribuição significativa é a concepção de *framework* para integrar esta heurística com o mecanismo de escalonamento do EXEHDA. De forma análoga aos outros serviços do *middleware* com perfil adaptativo, o *framework* proposto para o TiPS provê estratégia para troca de seus componentes durante a execução.

A seguir estão destacados os principais resultados atingidos no desenrolar das atividades de pesquisa atinentes a este mestrado:

- identificação das principais pesquisas cujos temas abordam aspectos que dizem respeito à construção de escalonadores para a computação pervasiva. Observe-

se que quando do início das atividades de pesquisa do TiPS, propostas explicitamente comprometidas com o escalonamento de recursos para computação pervasiva não estavam indexadas na literatura. Um resumo desta revisão de literatura constitui o capítulo 7;

- discussão dos requisitos que devem ser supridos por um escalonador destinado ao cenário da computação pervasiva;
- concepção de um modelo de rede bayesiana para construção dinâmica do perfil dos equipamentos envolvidos na computação. O modelo flexibiliza construção de perfis para diferentes elementos de contexto, por exemplo: ocupação de memória, processador, etc;
- concepção de um diagrama de influência que aglutina os resultados produzidos pela rede bayesiana, produzindo o índice denominado "utilidade" que é empregado pelo TiPS no momento de tomada de decisão;
- modelagem do TiPS na forma de um *framework* cujos componentes podem ser substituídos e parametrizados a critério do usuário, inclusive em tempo de execução;
- concepção da integração do mecanismo de coleta de dados sensorados com a heurística de escalonamento proposta;
- especificação e implementação de um módulo integrado ao EXEHDA-AMI para gerenciamento do TiPS;
- participação na definição das diretrizes do Projeto ISAM, particularmente do EXEHDA;
- integração com grupo de pesquisa da comunidade internacional, tendo sido realizada visita para discussão técnica do modelo geral a ser adotado no TiPS. Foi feita visita ao Departamento de Informática da Universidade do Minho (UM) localizada na cidade de Braga em Portugal. Os trabalhos foram supervisionados pelo Prof. Dr. Luís Paulo Santos, responsável pelo projeto PaRT;
- divulgação dos resultados do trabalho realizado durante o desenvolvimento da dissertação através de publicações. As principais estão relacionadas na seção 8.1.2;
- participação na submissão do projeto Projeto GRADEp: *middleware* para gerenciar um ambiente de grade pervasiva. O mesmo foi selecionado para constituir tema de grupo de trabalho na RNP. Início das atividades em agosto de 2004.

8.1.2 Publicações durante o mestrado

Esta seção lista os artigos em que o autor participou como autor ao longo do curso.

Publicações do TiPS

- Rodrigo Real, Adenauer Corrêa Yamin, Iara Augustin and Luciano da Silva, Gustavo Frainer, Jorge Barbosa e Cláudio Geyer. **Tratamento da incerteza no escalonamento de recursos em pervasive computing**. In: *Actas da Conferência IADIS Ibero-Americana WWW/Internet 2003.* Algarve, Portugal: IADIS, 2003.

- Rodrigo Real, Adenauer Yamin, Luciano da Silva and Gustavo Frainer, Iara Augustin, Jorge Barbosa e Cláudio Geyer. **Resource scheduling on grid: handling uncertainty**. In: *4th International Workshop on Grid Computing.* Phoenix, Arizona, USA: IEEE/ACM, 2003.

- Rodrigo Real, Adenauer Yamin, Iara Augustin, Luciano da Silva and Gustavo Frainer e Cláudio Geyer. **Uma proposta de escalonamento de recursos para Pervasive Computing**. Cadernos de Informática. Porto Alegre: PPGC/UFRGS. V. 3, n. 1, p.103-108, junho de 2003.

- Rodrigo Real, Adenauer Yamin, Luciano da Silva and Gustavo Frainer, Iara Augustin e Cláudio Geyer. **Tratamento de incertezas no escalonamento de recursos na pervasive computing**. In: *VII Oficina de Inteligência Artificial.* Pelotas, RS: Editora da Universidade Católica de Pelotas, 2003.

- Rodrigo A. Real, Cláudio Geyer. **Uma proposta de escalonamento de recursos para Pervasive Computing**. In: *ERAD 2004 - Fórum de Pós-Graduação.* Pelotas, RS: SBC, 2004.

Publicações no âmbito do Projeto ISAM

Nesta seção, estão relacionadas as publicações pertinentes aos Projetos ISAM e EXEHDA. Foram selecionadas aquelas que tiveram participação efetiva do autor. A ordem considera o ano de publicação.

- Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano C. da Silva, Rodrigo A. Real, Gerson Cavalheiro, Cláudio Geyer. **A Framework for Exploiting Adaptation in High Heterogeneous Distributed Processing**. In: *14th IEEE Symposium on Computer Architecture and High Performance Computing.* Vitória, Brazil. 2002.

- Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano C. da Silva, Rodrigo A. Real, Gerson Cavalheiro, Cláudio Geyer. **Towards Merging Context-aware, Mobile and Grid Computing**. *In: International Journal of High Performance Computing Applications.* London: Sage Publications. v.17, n.2, p.191-203, June 2003.

- Iara Augustin, Adenauer Yamin, Luciano C. Silva, Rodrigo Real, Gustavo Frainer, Gerson Cavalheiro, Cláudio Geyer. **ISAM, joing context-awareness and mobility to building pervasive applications**. **Mobile Computing Handbook**. I. Mahgoub and M. Ilyas Ed. Florida. CRC Press, 2003.

- Adenauer Yamin, Iara Augustin, Jorge Barbosa, Luciano C. da Silva, Rodrigo A. Real, Cláudio Geyer. **EXEHDA: Um Ambiente de Execução**

para Adaptação Dinâmica ao Contexto de Aplicações na Computação Pervasiva. Cadernos de Informática. Porto Alegre: PPGC/UFRGS. V. 3, n. 1, p.115-120, junho de 2003.

- Gustavo Frainer, Rodrigo Real, Adenauer Yamin, Luciano da Silva, Iara Augustin, Cláudio Geyer.**Perfis Adaptativos para Balanceamento de Carga no ISAM**. In: Workshop em Sistemas Computacionais de Alto Desempenho, 4, 2003, São Paulo, Brasil. Proceedings 6 São Paulo: USP, Novembro, 2003. p.164-167.

- Iara Augustin, Adenauer Yamin, Luciano C. Silva, Rodrigo Real, Cláudio Geyer. **ISAMadapt: abstractions and tools for designing general-purpose pervasive applications**. **Software, Practice & Experience. Special Issue on Experiences with Auto-adaptive and Reconfigurable Systems**. Wiley InterScience (ainda não publicado).

- Iara Augustin, Adenauer Yamin, Luciano C. Silva, Rodrigo Real, Cláudio Geyer. **ISAMadapt - um Ambiente de Desenvolvimento de Aplicações para a Computação Pervasiva**. In: Simpósio Brasileiro de Linguagens de Programação (SBLP 2004), Niterói, Maio, 2004.

- Lucas Brusamarello et al.. **Timing Verification Based on floating vector simulation: a distributed approach**. In: X WORKSHOP IBERCHIP, 2004, Cartagena de Indias, Colombia. Março 2004.

- Adenauer Yamin, Iara Augustin, Luciano C. Silva, Rodrigo Real, Jorge Barbosa, Cláudio Geyer. **ISAM: Uma Arquitetura de Software para Pervasive Computing**. CLEI 2004. Arequipa, Peru. Setembro 2004.

- Gustavo Frainer, Rodrigo Real, Adenauer Yamin, Luciano da Silva, Iara Augustin, Cláudio Geyer. **Perfis Adaptativos para Balanceamento de Carga no ISAM**. In: ESCOLA REGIONAL DE ALTO DESEMPENHO (ERAD 2004), 2004. Pelotas: SBC/UFPel/UCPel, 2004. p. 225-228.

8.2 Trabalhos futuros

O TiPS enquanto parte do EXEHDA apresenta diversas oportunidades a serem exploradas. Algumas dessas oportunidades já foram traduzidas em propostas de trabalho. Neste sentido, as principais propostas são:

1. avaliação do comportamento do TiPS nas novas aplicações da computação pervasiva;

2. avaliação do emprego de estratégias de aprendizado por reforço junto ao *framework* do TiPS;

3. avaliação do TiPS dentro do projeto GRADEp.

A frente de trabalho de novas aplicações já possui atividades em andamento, e está sendo explorada na concepção do editor de textos `WalkEd` (AUGUSTIN, 2004.

194p). O `WalkEd` é um editor de textos adaptativo, voltado para execução no ambiente da computação pervasiva. Este editor possui um comportamento adaptativo em dois aspectos: (i) um primeiro relacionado à interface gráfica, a qual se adapta as condições do dispositivo que o está executando; (ii) e um segundo ativado sempre que o editor está executando em processadores com baixo poder de processamento, neste caso, os `OX`es `Spell` e `Dict` são disparados em nodos remotos, a escolha destes nodos é feita pelo TiPS. O `WalkEd` está sendo prototipado em plataforma PC e no PDA Sharp Zaurus 6500.

O aprendizado por reforço (SUTTON; BARTO, 1999) tem sido explorado para escalonamento (ZOMAYA; CLEMENTS; OLARIU, 1998) em sistemas de processamento paralelo. Em ambientes de elevada dinamicidade, ele pode ser empregado tanto para aquisição do perfil de comportamento dos recursos e seus usuários, como também das aplicações. Particularmente, do ponto de vista do TiPS, o aprendizado por reforço poderia ser utilizado para ajustar os parâmetros da rede bayesiana ao longo de diversas execuções.

A organização modular projetada para o *framework* do TiPS permite que os componentes sejam projetados, implementados e testados de forma independente, sendo possível atualizar apenas o módulo em desenvolvimento. Com isso, é possível evoluir individualmente cada um dos componentes, conforme a demanda de pesquisa exigir a avaliação de um, ou outro aspecto.

REFERÊNCIAS

AUGUSTIN, I. **Abstrações para uma linguagem de programação visando aplicações móveis em um ambiente de *pervasive computing***. 2004. 194p. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; GEYER, C. Towards a Taxonomy for Mobile Applications with Adaptive Behavior. In: INTERNATIONAL SYMPOSIUM ON PARALLEL AND DISTRIBUTED COMPUTING AND NETWORKING, PDCN, 2002, Innsbruck, Austria. **Proceedings...** [S.l.]: International Association of Science and Technology for Development (IASTED), 2002. p.54–59.

AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; SILVA, L.; REAL, R.; FRAINER, G.; CAVALHEIRO, G.; GEYER, C. ISAM, joining context awareness and mobility to building pervasive applications. In: **Mobile Computing Handbook**. Florida: CRC Press, 2003.

BAKER, M.; BUYYA, R.; LAFORENZA, D. **Grids and Grid Technologies for Wide-Area Distributed Computing**. Monash, Australia: Monash University, 2001.

BAKRE, A.; BADRINATH, R. Handoff and System Support for Indirect TCP/IP. In: USENIX SYMPOSIUM ON MOBILE AND LOCATION-INDEPENDENT COMPUTING, 2., 1995, Monterey CA. **Proceedings...** [S.l.:s.n.], 1995.

BARBOSA, J. L. V. **Holoparadigma**: um modelo multiparadigma orientado ao desenvolvimento de software distribuído. 2002. 213p. Tese (Doutorado em Ciência da Computação) — Informatics Institute, Federal University of Rio Grande do Sul, Porto Alegre.

BAUMGARTNER, K. M.; WAH, B. W. Computer scheduling algorithms: past, present, and future. **Information Sciences: an International Journal**, Cambridge, MA, v.57-58, p.319–345, 1991.

BECKER, W. Dynamic balancing complex workload in workstation networks - challenge, concepts and experience. In: HPCN EUROPE, 1995, Milan, Italy. **Proceedings...** [S.l.]: Springer, 1995. p.407–412.

BECKER, W.; WALDMANN, G. Exploiting Inter-Task Dependencies for Dynamic Load Balancing. In: HPDC, 1994, San Francisco, Califórnia. **Proceedings...** [S.l.]: IEEE-CS, 1994. p.157–165.

BECKER, W.; WALDMANN, G. Adaption in Dynamic Load Balancing: potential and techniques. In: FACHTAGUNG ARBEITSPLATZ-RECHENSYSTEME, 1995, Hanover, Germany. **Proceedings...** [S.l.: s.n.], 1995.

BECKER, W.; ZEDELMAYR, J. Scalability and Potential for Optimization in Dynamic Load Balancing - Centralized and Distributed Structures. In: MITTEILUNGEN GI, PARALLEL ALGORITHMEN UND RECHNERSTRUKTUREN, 1994, University of Stuttgart, Germany. **Proceedings...** [S.l.: s.n.], 1994.

BERMAN, F. D. et al. Application-Level Scheduling on Distributed Heterogeneous Networks. In: SUPERCOMPUTING, 1996, Pittsburg, USA. **Proceedings...** [S.l.]: IEEE Computer Society Press, 1996.

BESTER, J. et al. GASS: a data movement and access service for wide area computing systems. In: WORKSHOP ON INPUT/OUTPUT IN PARALLEL AND DISTRIBUTED SYSTEMS, 6., 1999, Atlanta, GA. **Proceedings...** New York: ACM Press, 1999. p.78–88.

BHAGWAT, P.; TRIPATHI, S. K.; PERKINS, C. **Network Layer Mobility**: an architecture and survey. Maryland, USA: University of Maryland, 1995. (CS-TR-3570).

BORG, A. et al. Fault Tolerance Under UNIX. **ACM Transactions on Computer Systems, TOCS**, New York, NY, v.7, n.1, p.1–24, 1989.

BREWER, E. A network Architecture for Heterogeneous Mobile Computing. **IEEE Personal Communications Magazine**, Washington, v.5, n.5, p.8–24, Oct 1998.

CASANOVA, H.; DONGARRA, J. NetSolve: a network server for solving computational science problems. In: ACM/IEEE CONFERENCE ON SUPERCOMPUTING, 1996, Pittsburgh, Pennsylvania, USA. **Proceedings...** New York: ACM Press, 1996. p.40.

CASANOVA, H. et al. Heuristics for Scheduling Parameter Sweep applications in Grid environments. In: HETEROGENEOUS COMPUTING WORKSHOP, HCN, 9., 2000, Cancun, Mexico. **Proceedings...** [S.l.]: IEEE Computer Society, 2000. p.349–363.

CASANOVA, H. et al. The AppLeS Parameter Sweep Template: user-level middleware for the grid. In: SUPERCOMPUTING, 2000, Dallas, USA. **Proceedings...** [S.l.]: IEEE Computer Society Press, 2000.

CASAVANT, T. L.; KUHL, J. G. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems. **IEEE Transactions on Software Engineering**, Washington, D.C., USA, v.14, n.2, p.141–154, Nov. 1988.

CASAVANT, T. L.; KUHL, J. G. Effects of Response and Stability on Scheduling in Distributed Computer Systems. **IEEE Transactions on Software Engineering**, Washington, D.C., USA, v.14, n.11, p.1578–1588, Nov. 1988.

CHANG, Y.-C.; SHIN, K. G. A Coordinated Location Policy for Load Sharing in HyperCube-Connected Multicomputers. **IEEE Transactions on Computers**, Washington, D.C., USA, v.44, n.5, p.669–682, May 1995.

COULORIS, G.; DOLLIMORE, J.; KINDBERG, T. **Distributed Systems–Concepts and Design**. 3rd.ed. New York: Addison Wesley, 2001.

DAVIES, N. et al. Limbo: A tuple space based platform for adaptive mobile applications. In: JOINT INTERNATIONAL CONFERENCE ON OPEN DISTRIBUTED PROCESSING AND DISTRIBUTED PLATFORMS, ICODP/ICDP, 1997, Toronto, Canada. **Proceedings...** [S.l.]: Chapman and Hall, 1997.

DONGARRA, J. J.; VORST, H. A. V. der. **Performance of Various Computers Using Standard Sparse Linear Equations Solving Techniques**. Tenesse, USA: Department of Computer Science, University of Tennessee, 1992. (UT-CS-92-168).

EAGER, D. L.; LAZOWSKA, E. D.; ZAHORJAN, J. Adaptive Load Sharing in Homogeneous Distributed Systems. **IEEE Transactions on Software Engineering**, Washington, D.C., USA, p.662–675, May 1986.

FABERO, J. C. et al. Dynamic load balancing in a heterogeneous environment under PVM. In: EUROMICRO WORKSHOP ON PARALLEL AND DISTRIBUTED PROCESSING, PDP, 1996, Braga, Portugal. **Proceedings...** Washington: IEEE Computer Society Press, 1996. p.414–419.

FERRARI, D.; ZHOU, S. An Empirical Investigation of Load Indices for Load Balancing Applications. In: IFIP WG 7.3 INTERNATIONAL SYMPOSIUM ON COMPUTER PERFORMANCE MODELLING, MEASUREMENT AND EVALUATION, PERFORMANCE, 12., 1987, Brussels, Belgium. **Proceedings...** [S.l.]: North-Holland, 1987. p.515–528.

FLINN, J.; SATYANARAYANAN, M. Energy-aware adaptation for mobile applications. In: SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 1999, New York, USA. **Proceedings...** New York: ACM Press, 1999. p.48–63.

FOSTER, I.; KESSELMAN, C. Globus: a metacomputing infrastructure toolkit. **The International Journal of Supercomputer Applications and High Performance Computing**, London, v.11, n.2, p.115–128, Summer 1997.

FOSTER, I.; KESSELMAN, C. (Ed.). **The Grid**: blueprint for a new computing infrastructure. San Francisco, CA: Morgan Kaufmann, 1999.

FOX, A. et al. Adapting to Network and Client Variability via On-Demand Dynamic Distillation. In: INTERNATIONAL CONFERENCE ON ARCHITECTURE SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS, ASPLOS, 7., 1996, Cambridge, MA. **Proceedings...** New York: ACM Press, 1996.

GRIMSHAW, A. et al. **Legion**: an operating system for wide-area computing. Virginia: Department of Computer Science, University of Virginia, 1999. (CS-99-12).

HARCHOL-BALTER, M.; DOWNEY, A. B. Exploiting Process Lifetime Distributions for Dynamic Load Balancing. **ACM Transactions on Computer Systems**, New York, USA, v.15, n.3, p.253–285, 1997.

ISAM. **Projeto ISAM**. Disponível em <http://www.inf.ufrgs.br/~isam>. Acesso em: dez. 2003.

JACQMOT, C.; MILGROM, E. A Systematic Approach to Load Distribution Strategies for Distributed Systems. In: IFIP WG10.3 INTERNATIONAL CONFERENCE ON DECENTRALIZED AND DISTRIBUTED SYSTEMS, 1993, Palma de Mallorca, Spain. **Proceedings...** Amsterdam: North-Holland, 1993.

JENSEN, F. V. **Bayesian Networks and Decision Graphs**. New York: Springer, 2001.

KISTLER, J. J.; SATYANARAYANAN, M. Disconnected Operation in the Coda File System. **ACM Transactions on Computer Systems**, New York, v.10, n.1, p.3–25, Feb. 1992.

KREMIEN, O.; KRAMER, J. Methodical analysis of adaptive load sharing algorithms. **IEEE Transactions on Parallel and Distributed Systems**, Washington, D.C., USA, v.3, n.6, p.747–760, Nov. 1992.

KREMIEN, O.; KRAMER, J.; MAGEE, J. Scalable, adaptive load sharing for distributed systems. **IEEE Parallel and Distributed Technology: systems and applications**, Washington, D.C., USA, v.1, n.3, p.62–70, Aug. 1993.

KUNZ, T. The Influence of Different Workload Descriptions on a Heuristic Load Balancing Scheme. **IEEE Transactions on Software Engineering**, Washington, DC, v.17, n.7, p.725–730, July 1991.

KWOK, Y.-K.; AHMAD, I. Benchmarking and Comparison of the Task Graph Scheduling Algorithms. **Journal of Parallel and Distributed Computing**, Cambridge, MA, v.59, n.3, p.381–422, 1999.

LADEIRA, M.; VICARI, R. M.; COELHO, H. M. F. Redes Bayesianas Multiagentes. **REIC - Revista Eletrônica de Iniciação Eletrônica**, Porto Alegre, v.2, n.1, p.3–25, mar. 2002.

LEBECK, A. R. et al. Power Aware Page Allocation. In: INTERNATIONAL CONFERENCE ON ARCHITECTURAL SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS, ASPLO, 9., 2000, Cambridge, Massachusetts, United States. **Proceedings...** New York: ACM Press, 2000. p.105–116.

LITZKOW, M. J.; LIVNY, M.; MUTKA, M. W. Condor : A hunter of idle workstations. In: INTERNATIONAL CONFERENCE ON DISTRIBUTED COMPUTING SYSTEMS, 8., 1988, Washington, D.C., USA. **Proceedings...** Washington: IEEE Computer Society Press, 1988. p.104–111.

LOH, P. K. K. et al. How Network Topology Affects Dynamic Load Balancing. **IEEE Concurrency**, Washington, D.C., USA, v.4, n.3, p.25–35, Fall 1996.

LOWEKAMP, B. B. et al. A Resource Query Interface for Network-Aware Applications. In: IEEE SYMPOSIUM ON HIGH-PERFORMANCE DISTRIBUTED COMPUTING, 7., 1998, Chicago, Illinois. **Proceedings...** [S.l.]: IEEE Computer Society, 1998. p.189–196.

LU, Q.; LAU, S.-M. A negotiation protocol for dynamic load distribution using batch task assignments. **Journal of Parallel and Distributed Computing**, Cambridge, MA, v.55, n.2, p.166–191, 1998.

LULING, R.; MONIEN, B. A Dynamic Distributed Load Balancing Algorithm with Provable Good Performance. In: ACM SYMPOSIUM ON PARALLEL ALGORITHMS AND ARCHITECTURES, 1993, New York, USA. **Proceedings...** New York: ACM Press, 1993. p.164–172.

LULING, R.; MONIEN, B.; RAMME, F. A study of dynamic load balancing algorithms. In: IEEE SPDP, 3., 1991, Venice, Italy. **Proceedings...** Washington: IEEE-CS, 1991. p.686–689.

MEHRA, P. **Automated learning of load-balancing strategies for a distributed computer system**. Illinois: University of Illinois, 1992.

MUMMERT, L. B.; EBLING, M. R.; SATYANARAYANAN, M. Exploiting weak connectivity for mobile file access. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 1995, Copper Mountain, Colorado, United States. **Proceedings...** New York: ACM Press, 1995. p.143–155.

MUTKA, M. W.; LIVNY, M. Profiling Workstations' Available Capacity for Remote Execution. In: IFIP WG 7.3 INTERNATIONAL SYMPOSIUM ON COMPUTER PERFORMANCE MODELLING, MEASUREMENT AND EVALUATION, 12., 1988. **Proceedings...** [S.l.]: North-Holland, 1988. p.529–544.

NAHRSTEDT, K.; CHU, H. hua; NARAYAN, S. QoS-aware resource management for distributed multimedia applications. **J. High Speed Netw.**, Amsterdam, v.7, n.3-4, p.229–257, 1999.

NEEDHAM, R. M.; SCHROEDER, M. D. Using encryption for Authentication in Large Networks of Computers. **Communications of the ACM**, New York, USA, v.21, n.12, p.993–999, Dec. 1978.

NI, L. M.; XU, C.-W.; GENDREAU, T. B. A Distributed Drafting Algorithm for Load Balancing. **IEEE Transactions on Software Engineering**, Washington, D.C., USA, v.SE-11, p.1153–1161, Oct. 1985.

NOBLE, B. System Support for Mobile, Adaptive Applications. **IEEE Personal Communications**, Washington, D.C., USA, Feb. 2000.

NOBLE, B. D. et al. Agile Application-Aware Adaptation for Mobility. In: ACM SYMPOSIUM ON OPERATING SYSTEMS PRINCIPLES, 6., 1997, Saint Malo, France. **Proceedings...** New York: ACM Press, 1997. p.276–287.

ORAM, A. **Peer-to-peer**: harnessing the benefits of a disruptive technology. Sebastopol, CA: O'Reilly & Associates, 2001.

OZDEN, A.; SILBERSCHATZ, A. Scalable and non-intrusive load sharing in owner-based distributed systems. In: IEEE SYMPOSIUM ON PARALLEL AND DISTRIBUTED PROCESSING, 5., 1993, Dallas, Texas. **Proceedings...** [S.l.]: IEEE-CS, 1993.

PARSONS, S.; GMYTRASIEWICZ, P.; WOOLDRIDGE, M. (Ed.). **Game Theory and Decision Theory in Agent-Based Systems**. Dordrecht, Holanda: Kluwer Academics Publishers, 2002. p.1–28.

PEARL, J. **Probabilistic Reasoning in Intelligent systems**: network of plausible inference. San Mateo: Morgan Kaufmann, 1988. 571p.

PEARL, J. Decision making under uncertainty. **ACM Computing Surveys**, New York, USA, v.28, n.1, Mar. 1996.

PRESS, W. H. et al. **Numerical Recipes in C**: the art of scientific computing. 2nd ed. Cambridge, UK: Cambridge University Press, 1992.

REAL, R.; DUARTE FILHO, N.; YAMIN, A. **Unicluster**: uma proposta de *internet computing platform* para processamento de alto desempenho. 2001. Trabalho para conclusão (Curso de Engenharia de Computação) - Fundação Universidade Federal do Rio Grande (FURG), Rio Grande.

REAL, R.; YAMIN, A. C.; AUGUSTIN, I.; SILVA, L. C. da; FRAINER, G.; BARBOSA, J. L. V.; GEYER, C. Tratamento da Incerteza no Escalonamento de Recursos em Pervasive Computing. In: CONFERêNCIA IADIS IBERO-AMERICANA WWW/INTERNET, 2003. **Proceedings...** [S.l.]: IADIS, 2003. p.167–170.

REAL, R.; YAMIN, A.; SILVA, L. da; FRAINER, G.; AUGUSTIN, I.; BARBOSA, J.; GEYER, C. Resource scheduling on grid: handling uncertainty. In: INTERNATIONAL WORKSHOP ON GRID COMPUTING, 4., 2003, Arizona, USA. **Proceedings...** [S.l.]: IEEE/ACM, 2003.

ROUGHGARDEN, T. Stackelberg scheduling strategies. In: ANNUAL ACM SYMPOSIUM ON THEORY OF COMPUTING, 33., 2001, Hersonissos, Crete, Greece. **Proceedings...** New York: ACM Press, 2001. p.104–113.

ROYER, E. M.; TOH, C.-K. A Review of Current Routing Protocols for Ad Hoc Mobile Wireless Networks. **IEEE Personal Communications**, Washington, D.C., USA, p.46–55, Apr. 1999.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence**: a modern approach. Upper Saddle River, New Jersey: Prentice-Hall, 1995.

RYOU, J. C.; JUANG, J. Y. An Efficient Load Balancing Algorithm in Distributed Computing Systems. In: SYMPOSIUM ON PARALLEL AND DISTRIBUTED SYSTEMS, 1994, Los Alamitos, Ca., USA. **Proceedings...** Washington: IEEE-CS, 1994. p.233–240.

SANTOS, L. P. dos; PROENÇA, A. A Bayesian RunTime Load Manager on a Shared Cluster. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, 2001, Brisbane, Australia. **Proceedings...** [S.l.]: IEEE-CS, 2001. p.6.

SANTOS, L. P. dos; PROENÇA, A. A Systematic Approach to Effective Scheduling in Distributed Systems. In: INTERNATIONAL MEETING ON HIGH PERFORMANCE COMPUTING FOR COMPUTATIONAL SCIENCE, VECPAR, 5., 2002, Porto, Portugal. **Proceedings...** [S.l.]: IEEE-CS, 2002. p.813–825.

SATYANARAYANAN, M. A Survey of Distributed File Systems. **Annual Reviews of Computer Science**, New York, USA, n.4, p.73–104, 1990.

SATYANARAYANAN, M. Fundamental Challenges in Mobile Computing. In: SYMPOSIUM ON PRINCIPLES OF DISTRIBUTED COMPUTING, 1996, Philadelphia, USA. **Proceedings...** New York: ACM Press, 1996. p.1–7.

SATYANARAYANAN, M. Pervasive Computing: vision and challenges. **IEEE Personal Communications**, Washington, D.C., USA, p.10–17, Aug. 2001.

SCHEURER, C.; SCHEURER, H.; KROPF, P. **Load balancing driven process migration**. [S.l.]: University of Berne, 1995.

SCHILIT, B. N.; THEIMER, M. M.; WELCH, B. B. Customizing Mobile Applications. In: USENIX MOBILE AND LOCATION-INDEPENDENT COMPUTING SYMPOSIUM, 1993, Cambridge, MA. **Proceedings...** [S.l.]: Usenix Association, 1993. p.129–138.

SCHOPF, J. M.; BERMAN, F. Stochastic Scheduling. In: ACM/IEEE CONFERENCE SUPERCOMPUTING, 1999. **Proceedings...** [S.l.]: IEEE Computer Society, 1999. 21p.

SHIVARATRI, N. G.; KRÜGER, P.; SINGHAL, M. Load Distribution for Locally Distributed Systems. **IEEE Computer**, Washington, D.C., USA, v.25, n.12, p.33–44, Dec. 1992.

SILVA, L. C. da. **Primitivas para Suporte à Distribuição de Objetos Direcionados à Pervasive Computing**. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

STANKOVIC, J. A. An Application of Bayesian Decision Theory to Decentralized Control of Job Scheduling. **IEEE Transactions on Computers**, Washington, D.C., USA, v.C-34, n.2, p.117–130, Feb. 1985.

STROM, R. E.; YEMINI, S. A. Optimistic Recovery in Distributed Systems. **ACM Transactions on Computer Systems**, New York, USA, v.3, n.3, p.204–226, Aug. 1985.

SUEN, T.; WONG, J. Efficient Task Migration Algorithm for Distributed Systems. **IEEE Transactions on Parallel and Distributed Systems**, Washington, D.C., USA, v.3, n.4, p.488–499, July 1992.

SUTTON, R. S.; BARTO, A. G. **Reinforcement Learning**: an introduction. Cambridge, MA: MIT Press, 1999. 322p.

TAIT, C. D.; DUCHAMP, D. An Efficient Variable-Consistency Replicated File Service. In: USENIX FILE SYSTEMS WORKSHOP, 1992, Berkeley, USA. **Proceedings...** [S.l.]: USENIX Association, 1992. p.111–126.

TANNENBAUM, T. et al. Condor – A Distributed Job Scheduler. In: STERLING, T. (Ed.). **Beowulf Cluster Computing with Linux**. Cambridge, MA: MIT Press, 2001.

THAIN, D.; TANNENBAUM, T.; LIVNY, M. Condor and the Grid. In: BERMAN, F.; FOX, G.; HEY, T. (Ed.). **Grid Computing**: making the global infrastructure a reality. Indianapolis, USA: John Wiley & Sons Inc., 2002.

THEIMER, M.; LANTZ, K. Finding Idle Machines in a Workstation Based Distributed System. **IEEE Transactions on Software Engineering**, Washington, D.C., USA, v.15, n.11, p.1444–1458, Nov. 1989.

VOELKER, M. **Mobisaic - An Information System for a Mobile Wireless Computing Environment**. 1995. Dissertação (Mestrado em Ciência da Computação) — Department of Computer Science and Engineering, University of Washington, Washington, USA.

WANT, R. et al. **The Active Badge Location System**. Cambridge: Olivetti Research, 1992.

WARD, A.; JONES, A.; HOPPER, A. A New Location Technique for the Active Office. **IEEE Personal Communications**, Washington, D.C., USA, v.4, n.5, p.42–47, Oct. 1997.

WATTS, J.; TAYLOR, S. A Practical Approach to Dynamic Load Balancing. **IEEE Trans. Parallel Distrib. Syst.**, Washington, D.C., USA, v.9, n.3, p.235–248, 1998.

WEISER, M. The Computer for the Twenty-First Century. **Scientific American**, New York, USA, v.265, n.3, p.94–104, Sept. 1991.

WEISER, M. et al. Scheduling for Reduced CPU Energy. In: OPERATING SYSTEMS DESIGN AND IMPLEMENTATION, 1994, Moterey, USA. **Proceedings...** [S.l.]: USENIX Association, 1994. p.13–23.

WEISS, G. (Ed.). **Multiagent Systems**: A modern approach to distributed artificial intelligence. Cambridge, MA: MIT Press, 1999.

WILLEBEEK-LEMAIR, M. H.; REEVES, A. P. Strategies for dynamic load balancing on highly parallel computers. **IEEE Transactions on Parallel and Distributed Systems**, Washington, D.C., USA, v.4, n.9, p.979–993, Sept. 1993.

WOLSKI, R.; SPRING, N. T.; HAYES, J. The network weather service: a distributed resource performance forecasting service for metacomputing. **Future Generation Computer Systems**, ENS Lyon, v.15, n.5–6, p.757–768, Oct. 1999.

XU, C.; LAU, F. **Load Balancing in Parallel Computers**: theory and practice. Boston: Kluwer Academic Publishers, 1997.

XU, C. et al. An Analytical Comparison of Nearest Neighbor Algorithms for Load Balancing in Parallel Computers. In: INTERNATIONAL PARALLEL PROCESSING SYMPOSIUM, 1995, Germany. **Proceedings...** [S.l.: s.n.], 1995. p.472–479.

YAMIN, A. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. 2004. 194p. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L. da; GEYER, C. Collaborative Multilevel Adaptation in Distributed Mobile Applications. In: INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY, 22., 2002, Atacama, Chile. **Proceedings...** Los Alamitos: IEEE, 2002. p.105–115.

YAMIN, A.; AUGUSTIN, I.; BARBOSA, J.; SILVA, L.; REAL, R.; GEYER, C. EXEHDA: um ambiente de execução para adaptação dinâmica ao contexto de aplicações na pervasive computing. **Cadernos de Informática**, Porto Alegre, v.3, n.1, p.115–120, jun. 2003.

YAMIN, A. C.; BARBOSA, J. L. V.; AUGUSTIN, I.; SILVA, L. C. da; REAL, R.; GEYER, C.; CAVALHEIRO, G. Towards Merging Context-aware, Mobile and Grid Computing. **The International Journal of High Performance Computing Applications**, London, v.17, n.2, p.191–203, 2003.

ZHOU, S. A Trace-Driven Simulation Study of Dynamic Load Balancing. **IEEE Transactions on Software Engineering**, Washington, D.C., USA, v.14, n.9, p.1327–1341, Sept. 1988.

ZOMAYA, A. Y.; CLEMENTS, M.; OLARIU, S. A Framework for Reinforcement-Based Scheduling in Parallel Processor Systems. **IEEE Transactions on Parallel and Distributed Systems**, Washington, D.C., USA, v.9, n.4, p.249–260, Mar. 1998.

APÊNDICE A UMA BREVE REVISÃO DA ÁREA DE TEORIA DA DECISÃO

Neste capítulo será apresentada uma revisão dos temas relacionados a teoria da decisão que se ligam de forma mais próxima a este trabalho.

A teoria clássica da decisão é composta por um conjunto de ferramentas matemáticas para tomada de decisão sobre quais ações devem ser tomadas quando os ganhos das ações possíveis não são conhecidos (PARSONS; WOOLDRIDGE, 2002).

A.1 Teoria das probabilidades

Um agente operando em um ambiente complexo possui incertezas com relação ao ambiente, é impossível ter informação suficiente para determinar precisamente o estado corrente em que o ambiente se encontra, assim como a forma como ele evoluirá. Portanto, para cada variável X_i que incorpora algum aspecto do estado corrente do ambiente, existe um valor x_i instanciando a variável e também uma probabilidade $Pr(x_i)$ associada, que indica a probabilidade de x_i ser o valor corrente de X_i.

Sendo $\mathbf{x}$ o conjunto de todos possíveis valores de x_i, tem-se:

$$Pr : x \in \mathbf{x} \mapsto [0, 1]$$

e

$$\sum_j Pr(x_{i_j}) = 1$$

Em outras palavras, a probabilidade $Pr(x_{i_j})$ é um número entre 0 e 1 e a soma das probabilidades de todos possíveis valores de X_i é 1. Se é sabido que X_i tem valor x_{i_j}, então $Pr(x_{i_j}) = 1$ e se é sabido que X_i não possui um valor x_{i_j}, então $Pr(x_{i_j}) = 0$.

Enquanto que esta definição matemática de probabilidades é bastante direta, o mesmo não pode ser dito com relação a semântica das probabilidades. Não existe um acordo universal que explicite o que significam probabilidades. Dentre as diversas escolas de pensamento nesta linha, existem duas posições principais. A primeira, historicamente interpreta as probabilidades como a freqüência de ocorrência. Na abordagem freqüentista considera-se que um evento a com probabilidade de 0.356 ocorrerá 0.356 do tempo. A segunda abordagem, chamada de bayesiana, sugere que a probabilidade está relacionada com os valores de probabilidade que uma pessoa daria para a ocorrência do evento em questão.

Dadas duas variáveis, X_1 e X_2, as probabilidades dos vários valores de X_1 e X_2 podem estar inter-relacionadas. Se elas não estão relacionadas, as variáveis são chamadas independentes, para valor x_{1_i} e x_{2_j}, tem-se:

$$Pr(x_{1_i} \wedge x_{2_j}) = Pr(x_{1_i})x_{2_j})$$

caso as variáveis não sejam independentes, tem-se:

$$Pr(x_{1_i} \wedge x_{2_j}) = Pr(x_{1_i} \mid x_{2_j})Pr(x_{2_j})$$

onde $Pr(x_{1_i} \mid x_{2_j})$ é a probabilidade de X_1 possuir um valor x_1, dado que X_2 possui um valor x_2. Essa probabilidade condicional expressa a relação entre X_1 e X_2, por exemplo, representando o fato de que x_1 se torna mais provável quando sabe-se que x_2 é verdadeiro.

Se for considerado o conjunto $\mathbf{X}$ das variáveis X_i, que contém as informações de interesse para a tomada de decisão. Para cada par variáveis de $\mathbf{X}$, pode-se estabelecer se elas são independentes ou não. A partir das relações de dependências pode-se então, construir um grafo no qual cada nodo constitui um variável de $\mathbf{X}$ e os arcos unem nodos que não são independentes. O grafo resultante é chamado de rede bayesiana (PEARL, 1996), esta estrutura gráfica compõe uma boa ferramenta computacional para o cálculo das probabilidades de interesse. Em geral algumas variáveis têm valores conhecidos e outras precisam ter probabilidades calculadas para cada um dos possíveis valores.

A.2 Redes causais

Uma rede causal consiste de um conjunto de variáveis e um conjunto de ligações dirigidas entre variáveis. Em termos matemáticos, a estrutura é chamada de grafo dirigido. Para se referir aos elementos do grafo, é comum que se utilize relações familiares, por exemplo, se há uma ligação de A para B, diz-se que B é filho de A, e conseqüentemente que A é pai de B.

As variáveis representam eventos, ou proposições. Uma variável pode possuir um número qualquer de estados. Uma variável pode, por exemplo, ser a cor de um carro, neste caso os estados poderiam ser: azul, vermelho, verde, preto; o número de filhos em uma família, estados possíveis: $0, 1, 2, 3, 4, 5, > 6$; ou uma doença, tendo como estados: bronquite, tuberculose ou câncer de pulmão. As variáveis podem possuir um conjunto de estados discreto ou contínuo, porém neste trabalho só serão consideradas variáveis com um número finito de estados.

Em uma rede causal, uma variável representa um conjunto de possíveis estados. Uma variável está em exatamente um destes estados, este estado pode ser desconhecido. Uma rede causal pode ser usada para determinar como uma mudança na certeza do estado de uma variável pode afetar a certeza do estado de outra (JENSEN, 2001).

Alterações de certeza em uma determinada variável de uma rede causal podem influenciar nas certezas de outras variáveis. Considerando alguns tipos básicos de redes causais, podem ser obtidas algumas regras quanto a propagação das alterações em variáveis. A seguir serão apresentados três tipos básicos de redes causais, assim como a caracterização, nestes casos, da influência entre as variáveis.

A.2.1 Conexões seriais

Considerando a situação apresentada na figura A.1 A possui influência em B, que por sua vez possui influência em C. Obviamente, a evidência em A vai influenciar a certeza de B, a qual irá influenciar a certeza em C. Similarmente, a evidência em C irá influenciar na certeza em A, através de B. Por outro lado, se o estado de B é conhecido, então o canal é bloqueado e A e C se tornam independentes. Diz-se então que, A e C são *d-separated* dado B.

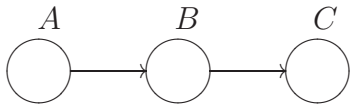

Figura A.1: Conexão serial. Quando B está instanciado, a comunicação entre A e B fica bloqueada

Conclui-se então que a evidência pode ser transmitida através de uma conexão serial, a não ser que o estado da variável que faz a conexão é conhecido.

A.2.2 Conexões divergentes

A situação apresentada na figura A.2 é chamada de conexão divergente. A influência pode passar por entre os filhos de A, a não ser que, o estado de A seja conhecido. Diz-se que $B, C, \ldots, E$ são *d-separadas* dado A.

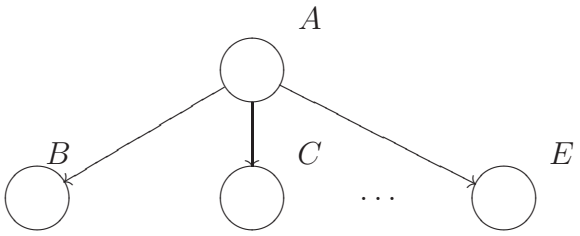

Figura A.2: Conexão divergente. Se A está instanciado, a comunicação entre os filhos fica bloqueada

A evidência portanto, pode ser transmitida por uma conexão divergente, a menos que ela esteja instanciada.

A.2.3 Conexões convergentes

A figura A.3 apresenta uma rede causal com conexão convergente. Se nada é conhecido sobre A, a exceção do que pode ser inferido a partir do conhecimento dos pais $B, \ldots, E$, então os pais são independentes: uma evidência em um deles não influência na certeza sobre os outros. O conhecimento sobre uma possível causa de um evento não diz nada com relação as outras possíveis causas. Por outro lado, se algo é conhecido sobre as conseqüências, as informações de uma possível causa podem contribuir para o conhecimento de outra possível causa.

Como conclusão, obtém-se que a evidência só pode ser transmitida através de uma conexão convergente se ou a variável na conexão, ou uma de suas filhas recebe uma evidência.

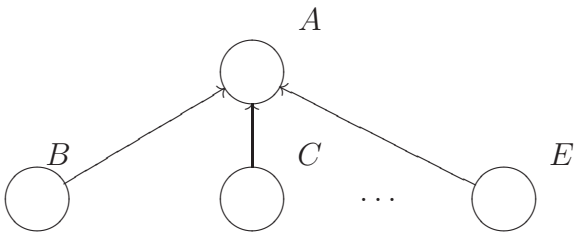

Figura A.3: Conexão convergente. Se A tiver a certeza alterada, a comunicação entre os pais é aberta

A.3 Redes bayesianas

Uma rede bayesiana é um modelo de representação gráfica para relações probabilísticas entre um conjunto de variáveis. As redes bayesianas podem ser construídas e parametrizadas por especialistas no assunto, podem ter seus parâmetros e seus relacionamentos construídos através da análise de dados amostrais, ou ainda, podem ser construídas através de um misto entre a opinião do especialista e a análise de dados amostrais.

Uma rede bayesiana é constituída de (JENSEN, 2001):

- um conjunto de variáveis e um conjunto de arcos dirigidos entre as variáveis;
- cada variável possui um conjunto finito de estados mutuamente exclusivos;
- as variáveis, juntamente com os arcos dirigidos formam um grafo acíclico dirigido (GAD);
- para cada variável A com pais $B1, \ldots, B_n$ existe em anexo uma tabela de probabilidades $P(A \mid B1, \ldots, B_n$. No caso de A não possuir pais no grafo, a tabela se reduz as probabilidades incondicionais $P(A)$.

A rede não possui nodos correspondentes a todas as causas de um evento, os fatores pouco relevantes são resumidos na incerteza associada às probabilidades de algumas variáveis. Com poucas variáveis, é possível tratar um grande universo de causas. O grau de aproximação pode ser elevado inserindo-se mais informação relevante (LADEIRA; VICARI; COELHO, 2002).

A topologia de uma rede bayesiana representa um modelo probabilístico completo, apresentando informações qualitativas (dependências), quantitativas (função de distribuição de probabilidades) e uma estrutura de controle para a inferência (LADEIRA; VICARI; COELHO, 2002). Esta estrutura determina qual estratégia será utilizada na propagação das crenças, em redes simplesmente conectadas a propagação das crenças pode ser aplicada diretamente. Porém, muitos problemas reais exigem a utilização de redes multiconectadas, nestas redes podem existir mais de um caminho entre o nodo que recebeu a evidência e o nodo que deve ser avaliado, esta situação não é prevista nos métodos diretos de propagação das evidências, aplicáveis em redes simplesmente conectadas. Os métodos de agregado e condicionantes, propostos por Pearl (PEARL, 1988), tratam o problema transformando as redes multiconectadas em redes simplesmente conectadas e então, aplicando métodos diretos.

A.3.1 A regra de Bayes

O ponto central das técnicas de inferência bayesiana é a fórmula da inversão, universalmente conhecida como a regra de Bayes,

$$P(H \mid E) = \frac{P(E \mid H)P(H)}{P(E)} \tag{A.1}$$

que estabelece que a crença em uma hipótese H, dada a evidência disponível, E pode ser calculado pela multiplicação da crença anterior em uma hipótese, $P(H)$ pelas probabilidades que E irá materializar a hipótese $P(E \mid H)$.

A.3.2 A regra da cadeia para redes bayesianas

Seja $U = (A_1, \ldots, A_n)$ o universo de variáveis. Se é possível acessar a tabela de probabilidades conjuntas $P(U) = (A_1, \ldots, A_n)$, é possível calcular $P(A_i)$, assim como $P(A_i \mid e)$, onde e é a evidência. Porém, $P(U)$ cresce exponencialmente com o número de varáveis e não é preciso que U seja muito grande para que a tabela fique intratável. Portanto, busca-se encontrar formas de armazenar esta tabela de forma mais compacta, uma representação que permite que $P(U)$ possa ser calculada.

Uma rede bayesiana sobre U é uma destas representações, na rede bayesiana $P(U)$ pode ser calculado a partir dos potenciais especificados na rede.

Formalmente (JENSEN, 2001), diz-se que sendo BN uma rede bayesiana sobre $U = A_1, \ldots, A_n$, a distribuição de probabilidades conjunta $P(U)$ é o produto de todos potenciais especificados em BN. A aplicação da equação (A.2) representa este cálculo a ser realizado, considera-se na equação $pa(A_i)$ como sendo o conjunto de nodos pais de A_i na rede bayesiana.

$$P(U) = \prod_i P(A_i \mid pa(A_i)) \tag{A.2}$$

A.3.3 Exemplo de aplicação de redes bayesianas

Nesta seção será apresentado, como exemplo, um estudo de caso simples envolvendo a propagação das probabilidades em uma rede bayesiana. Este exemplo foi originalmente apresentado em (RUSSELL; NORVIG, 1995).

A rede apresentada envolve quatro variáveis: *Nublado*, *Regador*, *Chuva* e *GramaMolhada*. A rede bayesiana relativa ao exemplo é apresentada na figura A.4, esta rede bayesiana permite por exemplo inferir as probabilidades das causas da grama estar molhada. Pode-se observar na figura que *Nublado* influencia o estado do *Regador* e de *Chuva*, estes por sua vez influenciam a variável *GramaMolhada*.

As tabelas A.1, A.2, A.3 e A.4, apresentam as probabilidades condicionais para cada uma das variáveis e suas relações, os valores nestas tabelas representam a intensidade das relações entre as variáveis. Na tabela A.3, por exemplo, é possível perceber que foi estabelecida uma probabilidade bem baixa (0.1) para o regador estar ativo considerando que está nublado. Na tabela A.4, também pode-se observar um caso típico em que a probabilidade de a grama não estar molhada (0.01) em uma situação em que está chovendo e o regador também está ligado.

Através da regra da cadeia, apresentada na seção A.3.2 que é resumida pela equação (A.2), é possível determinar a probabilidade conjunta dos nodos da rede bayesiana:

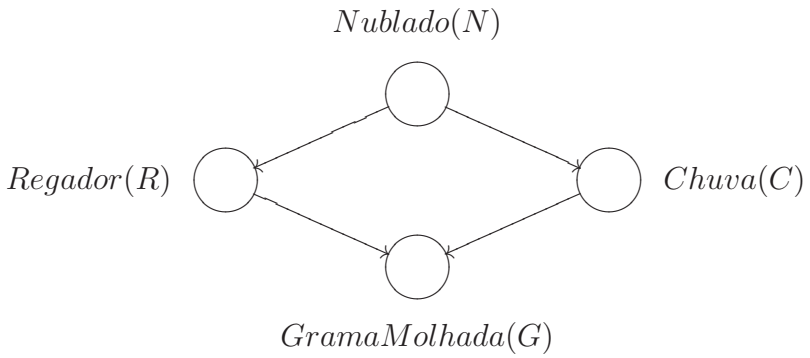

Figura A.4: Rede bayesiana para inferência sobre as causas da grama molhada

Tabela A.1: Tabela de probabilidade condicional para nodo Nublado

$P(N = F)$	$P(N = V)$
0.5	0.5

Tabela A.2: Tabela de probabilidade condicional para nodo Chuva

N	$P(C = F)$	$P(C = V)$
F	0.8	0.2
V	0.2	0.8

Tabela A.3: Tabela de probabilidade condicional para nodo Regador

N	$P(R = F)$	$P(R = V)$
F	0.5	0.5
V	0.9	0.1

Tabela A.4: Tabela de probabilidade condicional para nodo Grama Molhada

R	C	$P(G = F)$	$P(G = V)$
F	F	1.0	0.0
V	F	0.1	0.9
F	V	0.1	0.9
V	V	0.01	0.99

$$P(N, R, C, G) = P(N) * P(R \mid N) * P(C \mid N, R) * P(G \mid N, R, C)$$

Usando a independência condicional entre as relações, a equação pode ser reescrita da seguinte forma:

$$P(N, R, C, G) = P(N) * P(R \mid N) * P(C \mid N) * P(G \mid R, C)$$

na qual foi possível simplificar o terceiro termo, pois C é independente de R dado o pai N, e o último termo, pois G é independente de N dados os pais R e C.

A partir desta rede bayesiana é possível efetuar uma inferência probabilística para, por exemplo, determinar quais as probabilidades de estar chovendo ou de o regador estar ligado dado que a grama está molhada. Cada probabilidade pode ser calculada através da regra de Bayes:

$$\begin{aligned} Pr(R = 1 \mid G = 1) &= \tfrac{Pr(R=1,G=1)}{Pr(G=1)} \\ = \tfrac{\sum_{n,c} Pr(N=n,R=1,C=c,G=1)}{Pr(G=1)} &= 0.2781/0.6471 = 0.4298 \end{aligned}$$

$$\begin{aligned} Pr(C = 1 \mid G = 1) &= \tfrac{Pr(C=1,G=1)}{Pr(G=1)} \\ = \tfrac{\sum_{n,r} Pr(N=n,R=r,C=1,G=1)}{Pr(G=1)} &= 0.4581/0.6471 = 0.7079 \end{aligned}$$

$$Pr(W = 1) = \sum_{n,c,r} Pr(N = n, R = r, C = c, G = 1) = 0.6471$$

A probabilidade resultante de que a causa da grama estar molhada seja a chuva é maior que a probabilidade de que a causa seja o regador ligado.

A.4 Teoria da utilidade

Assume-se inicialmente que cada tomador de decisão possui preferências próprias e interesses sobre como o "mundo" deve ser ou deve se tornar. Assume-se também, que existe um conjunto $\Omega = \omega_1, \ldots, \omega_n$ de alternativas ou estados que representa as preferências dos tomadores de decisão (PARSONS; WOOLDRIDGE, 2002).

As preferências de um tomador de decisão serão definidas através de uma função de utilidade, esta função atribui um número real a cada uma das alternativas, este número indica o quão boa é a alternativa. Do ponto de vista do tomador de decisões, quanto maior o valor do número, melhor. Portanto, as preferências de um tomador de decisões i podem ser determinadas pela função

$$u_i : \Omega \rightarrow \Re$$

Uma função de utilidade conduz para uma ordenação de preferência com relação às alternativas. Por exemplo, se ω e ω' são os dois alternativas em Ω e $u_i(\omega) > u_i(\omega')$, então a alternativa w é preferida pelo tomador de decisão i em relação a alternativa w'.

A.5 Utilidade esperada

No cálculo da utilidade esperada considera-se que o tomador de decisões possui um conjunto de ações possíveis $\mathbf{A}$, sendo que cada membro A_i do mesmo possui

alternativas possíveis. O valor relacionado a tomada de uma ação em particular é dependente do estado em que o "mundo" se encontra. É de pouco valor carregar uma prancha de *surf* no deserto, mas de alto valor carregá-la em uma viagem ao Havaí. Para tomar uma decisão, o agente responsável deverá verificar o valor de $U(S_j)$, onde S_j é o estado antes da ação. Fazendo isto para cada uma das ações possíveis, o tomador de decisão poderá escolher a ação que leva a um estado de maior valor, sendo este valor determinado pela função de utilidade.

Para tornar os tomadores de decisão mais sensíveis, pode-se combinar cálculos de probabilidades e de utilidade para cada ação, e calcular a utilidade esperada para cada alternativa. Considerando que cada uma delas corresponde a um estado, tem-se que:

$$EU(A_i) = \sum_{s_j \in S} Pr(S_j \mid A_i) U(S_j) \tag{A.3}$$

onde **S** é o conjunto de todos estados. O tomador de decisões seleciona a ação A^*, sendo:

$$A^* = arg \max_{A_i \in \mathbf{A}} \sum_{s_j \in S} Pr(S_j \mid A_i) U(S_j)$$

As redes bayesianas podem ser instrumentalizadas com nodos de decisão e de ganho, transformando-se em diagramas de influência. O processamento do diagrama de influência envolve a aplicação da regra de Bayes e o cálculo da utilidade de cada alternativa.