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

EDER STONE FONTOURA

AvSchedP: Previsão de Disponibilidade Para Escalonamento de Tarefas em Grades Oportunistas

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

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, outubro de 2012

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Fontoura, Eder Stone

AvSchedP: Previsão de Disponibilidade Para Escalonamento de Tarefas em Grades Oportunistas / Eder Stone Fontoura. – Porto Alegre: PPGC da UFRGS, 2012.

135 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2012. Orientador: Cláudio Fernando Resin Geyer.

1. Padrão de disponibilidade. 2. Disponibilidade de recursos. 3. Grades oportunistas. 4. Previsão de disponibilidade de recursos. I. Geyer, Cláudio Fernando Resin. II. Título.

“Destiny is not a matter of chance, it is a matter of choice
it is not a thing to be waited for, it is a thing to be achieved.”
— WILLIAM JENNINGS BRYAN

AGRADECIMENTOS

Apesar da aparente simplicidade em escrever tal seção, quando comparada a complexidade de escrita de uma dissertação, tem-se que a simplicidade desaparece mediante a sensação de *medo* de esquecer de citar alguém que tenha contribuído para a elaboração deste trabalho.

Nesta seção, opto por agradecer de forma aleatória as pessoas ou entidades que me auxiliaram de alguma forma, seguindo uma "ordem"aleatória, evitando-se assim a necessidade de estabelecer graus de importância.

- A Universidade Federal Do Rio Grande do Sul (UFRGS) por me acolher nos estabelecimentos do Instituto de Informática, durante 2 anos de estudo e pesquisa que realizei.

- A toda secretaria de pós-graduação do Instituto de Informática da UFRGS (PPGC) pela paciência e auxílio prestado durante todo o período em que estive no programa de mestrado da universidade.

- Ao meu pai e minha mãe que, mesmo desconhecendo o ambiente universitário ou quais seriam as diferenças entre um mestrado e um doutorado, me incentivaram desde o início de minha vida estudantil. Ao meu pai, representante da ordem dentro da família, soube me mostrar um caminho difícil, demorado, exaustivo porém, honesto e gratificante de "ganhar"a vida sem prejudicar outras pessoas. A minha mãe por me acolher nas dificuldades e tropeços da vida e juventude, sabendo colocar-me novamente nos melhores caminhos da vida.

- A meus já distantes avôs que, certamente de onde estão, torcem pelo meu sucesso. A meu avô Afonso, por um dos ensinamentos mais significativos que tive em minha vida: a bondade que pode existir em um ser humano. Bondade esta que nos faz acreditar que o mundo realmente poderia ser diferente. A meu avô Arlindo, pelas suas brincadeiras na varanda da casa da praia, por conseguir manter o humor com seu 80 anos de idade e principalmente por ser um exemplo de luta, devido as inúmeras dificuldades vividas durante a infância e juventude.

- A minhas avós, que partiram em busca do encontro de meus avôs, ainda durante a realização de meu mestrado. Peço desculpas pela minha ausência neste período.

- Ao professor e orientador Geyer pela paciência em esperar a quase eterna data de entrega desta dissertação. Pela confiança depositada, pelas oportunidades oferecidas e pelos ensinamentos passados na sala de aula.

- A Patrícia Kayser, pela confiança que me foi depositada. Agradeço pelo incentivo e por ajudar retomar a motivação para a conclusão deste trabalho. Pelo tempo dedicado para a escrita de um artigo. Pela vontade de cooperar. Agradeço também pela oportunidade que me foi dada, oferecendo-me a possibilidade de atuar como professor convidado no curso de graduação da universidade Unilasalle. Oportunidade esta que tem me proporcionado um significativo crescimento profissional e a realização de um de meus objetivos profissionais: atuar como professor em uma instituição de ensino superior.

- A minha filha, pela longa ausência que significou a realização deste mestrado. Pela falta que fiz para a condução de seus estudos neste período.

- Ao CNPQ pelo apoio financeiro que me foi prestado durante o mestrado. Apoio este que possibilitou executar com dedicação exclusiva boa parte de meu curso de mestrado.

- A minha esposa, Luciari, por me auxiliar em todos os aspectos possíveis durante a realização deste trabalho. Atuou como minha monitora pessoal, me recomendando trabalho ou descanso quando necessário. Pela paciência em aguentar os dias de mau humor ocasionados pelo trabalho excessivo. Pelo amor que me foi dado. Pelo cuidado separando minha roupas. Por me deixar dormir um pouco mais nos dias em que eu estava cansado. Por preparar o meu café da manhã, mesmo tendo que sair para trabalhar no mesmo horário que eu. Por me estimular a terminar o meu trabalho e principalmente por acreditar nisso.

- Aos colegas dos laboratórios 205 e 207 por compartilharem suas angústias durante a pesquisa e pelas conversas no RU. Em especial ao Diego Gomes e Luciano Cavalheiro. O primeiro pela parceria e o segundo pela transferência de conhecimentos.

- Ao chefe dos laboratórios, Luis Otávio, pela sempre presente disponibilidade em ajudar e por sempre receber com um sorriso os que o procuram. Agradeço-te por disponibilizar os laboratórios para execução de testes, quando necessitei.

- A Deus por nos (família) acompanhar durante toda a caminhada.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CPU	Central Processor Unit
DAG	Directed Acyclic Graphs
DG	Desktop Grids
DNA	DeoxyriboNucleic Acid
EA	Execution Availability
EDG	Enterprise Desktop Grids
EU	Execution Unavailability
FCFS	First Come First Served
FTA	Failure Trace Archive
GIMPS	Great Mersenne Prime Search
GC	Global Computing
HA	Host Availability
HTTP	Hyper Text Transfer Protocol
HU	Host Unavailability
IO	Input/Output
LHC	Large Hadron Collider
NAT	Network Address Translation
QOS	Quality Of Service
VC	Volunteer Computing

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Atualmente, os computadores pessoais são equipados com processadores de vários núcleos possuem alto poder de processamento. Algumas pesquisas realizadas indicam que estes mesmos computadores permanecem em média 80% do tempo ociosos. A união destas duas informações fomenta à pesquisa de um tipo de computação distribuída, conhecida como grades oportunistas.

A utilização de um ambiente de grade oportunista permite a implementação de um computador paralelo capaz de processar grandes volumes de tarefas, com baixo custo financeiro.

Apesar do baixo custo financeiro proporcionado por este ambiente, tem-se um alto custo computacional devido a utilização de recursos não dedicados. Estes ambientes sofrem de um problema que tem servido de objeto de diversas pesquisas, a indisponibilidade de recursos.

O problema da indisponibilidade de recursos deve-se a volatilidade deste recursos. A volatilidade deve-se a liberdade de tornarem-se ociosos ou ocupados a qualquer instante, sem a necessidade de aviso prévio ao servidor. A volatilidade é a responsável por gerar uma série de interrupções de execução de tarefas e a consequente necessidade de novos escalonamentos desta mesma tarefa.

Para reduzir o impacto deste problema, o escalonador de tarefas poderia considerar informações sobre o comportamento futuro dos recursos, ou seja, poderia considerar informações provenientes de um preditor de disponibilidade.

Este trabalho propõe o modelo **AvSchedP**, modelo este capaz de realizar a previsão de disponibilidade baseando-se em dados binários e de forma dinâmica. O modelo AvSchedP foi projetado para ser integrado a escalonadores de grades oportunistas.

O AvSchedP foi implementado e diversos experimentos foram realizados. Os experimentos foram realizados para avaliar a qualidade de previsão e a adequação do modelo a um sistema de grade oportunista, o XtremWeb.

Os resultados obtidos demonstram que o **AvSchedP** apresenta bons resultados ao realizar previsões de duração de disponibilidade em ambientes formados por recursos voláteis.

Palavras-chave: Padrão de disponibilidade, disponibilidade de recursos, grades oportunistas, previsão de disponibilidade de recursos.

ABSTRACT

Nowadays, personal computers have multi-core processors with high processing power. Some researches show that these computers remain 80 % of the time idle. The union of these two information instigates a kind of distributed computing, known as opportunistic grids.

The opportunistic grid environment usage allows the implementation of a parallel computer like, capable of processing large volumes of tasks, with a low cost.

Despite the low financial cost, it has a high computational cost due to use of non-dedicated resources. Opportunistic grid environments have a problem that has been the subject of several researches: the resource availability.

The resource availability problem is caused by the volatility of resources. The resource is free to become idle or busy at any time without notice the server. The volatility is responsible for generating a lot of task execution interruptions and consequently, task rescheduling.

To alleviate this problem, the task scheduler could consider information about the future behavior of resources provided by an avalibility predictor.

This paper proposes a new model, called **AvSchedP** , to perform dynamic resource availability prediction based on binary data. The model was designed to be integrated into opportunistic grid schedulers.

The AvSchedP was implemented and several experiments were performed. These experiments were conducted to evaluate the prediction quality and the model integrability to the XtremWeb opportunistic grid system.

The results show that the **AvSchedP** has good results when making availability duration predictions for environments composed by volatile resources.

Keywords: Resource Availability, Opportunistic Grids, Resource Availability Prediction.

1 INTRODUÇÃO

As ciências em geral como a Biologia, Química e Física são grandes consumidoras de recursos computacionais. Os algoritmos desenvolvidos para auxiliar na resolução de problemas relacionados a estas áreas geralmente efetuam cálculos extremamente complexos, cálculos estes muitas vezes impossíveis de serem solucionados utilizando-se apenas um simples computador. Algoritmos para a identificação da cadeia de *DNA*, ou para o tratamento de dados resultantes da colisão de partículas efetuadas no *LHC* servem de exemplo.

Supercomputadores, com arquiteturas paralelas e toda a tecnologia disponível, são máquinas de alto desempenho normalmente apropriadas para executar tais algoritmos. No entanto, apresentam um alto custo de aquisição e por isso não podem ser adquiridos por muitos dos centros de pesquisa existentes.

Com a crescente necessidade de máquinas poderosas, outras importantes tecnologias foram implementadas. Entre estas, encontram-se as tecnologias que tem por princípio, o compartilhamento de recursos já existentes, sendo uma delas chamada de **Grade Computacional**.

A computação em grade (*Grid Computing*) tem por objetivo o compartilhamento coordenado de uma variedade de recursos de forma dinâmica e entre diversas organizações (FOSTER, 2001) (NABRZYSKI; SCHOPF; WEGLARZ, 2003) [1]. Os recursos compartilhados em uma grade computacional são normalmente pertencentes a domínios institucionais, sendo máquinas dedicadas e com a disponibilidade de administradores para intervir nas diversas configurações sofisticadas exigidas por este ambiente. Recursos com pouco poder computacional não são considerados muito adequados a este tipo de grade.

No entanto, outras pesquisas buscavam utilizar recursos convencionais, com menor poder de processamento e principalmente, pertencentes a usuários convencionais, usuários estes sem qualquer conhecimento sobre configuração de equipamentos. Estas pesquisas buscavam a melhor utilização de recursos já existentes, através da doação de ciclos de processamento quando ociosos. Estes recursos tinham como característica, o baixo custo.

Destas pesquisas surgiram as Grades Computacionais Oportunistas (*Opportunistic Grids*), tendo como um de seus objetivos, o uso eficiente de recursos, reduzindo-se a subutilização de recursos computacionais já existentes.

Na literatura é possível encontrar diversas termos relacionados a este mesmo conceito. *Desktop Grids*, *Volunteer Computing*, *Global Computing*, *Public Resource Computing* e *Enterprise Desktop Grids* são termos frequentemente encontrados tendo em comum, principalmente, a utilização de recursos que doam processamento quando ociosos.

[1] No ambiente de computação em grade, entende-se por recurso supercomputadores, fontes de dados e outros tipos de dispositivos.

O desenvolvimento e consolidação das grades oportunistas possibilitou a utilização deste ambiente para a execução de diversos algoritmos das áreas anteriormente citadas. Aplicações como o estudo de modelos de previsão do tempo (*projeto* ***Climatepredict-tion.net***) (CLIMATEPREDICTION.NET, 2010), o estudo de dados relacionados a colisão de partículas (*projeto* ***LHC@HOME***) (LHC@HOME, 2010) e o estudo de vacinas para controle e cura Malária (*projeto* ***Malariacontrol.net***) (MALARIACONTROL.NET, 2010), entre outras, são exemplos que utilizam este ambiente.

Esta dissertação está inserida no contexto de grades computacionais oportunistas, tendo como objetivo de pesquisa a previsão de disponibilidade de recursos doados para este ambiente.

1.1 Motivação

Os recursos que participam de grades oportunistas, geralmente são utilizados de forma compartilhada, concorrente. Esta concorrência se dá entre o sistema de grade oportunista, que deseja executar tarefas neste recurso, e o usuário local do recurso, que deseja executar tarefas próprias. No entanto, este compartilhamento não é equilibrado, tendo as tarefas locais prioridade de execução sob tarefas externas, que neste caso, são tarefas da Grade Oportunista.

Este desequilíbrio faz com que os escalonadores de tarefas das grades oportunistas, inseridos neste contexto, tenham suas decisões de escalonamento afetadas pela interrupção na execução de tarefas atribuídas a um recurso, devido a prioridade em executar tarefas locais surgidas neste recurso. Por apresentarem este comportamento, estes recursos são ditos voláteis.

A volatilidade tem como consequência a ineficiência do sistema de grade, onde as principais causas são, a redução do números de tarefas concluídas em um determinado intervalo de tempo (*Turnaround Time*), o desperdício de recursos gastos em comunicação e processamento, e o aumento do número de erros de escalonamento. Assim, a volatilidade pode ser considerada como a principal fonte de incerteza de escalonadores de Grade Oportunista.

O escalonamento ótimo em sistemas distribuídos é conhecido como um problema NP-completo (VIDYARTHI et al., 2008). Este problema é potencializado em escalonadores da grades oportunistas, devido a característica volátil dos recursos. Uma forma alternativa a este problema seria obter um escalonamento subótimo, que neste contexto poderia ser alcançado com o emprego de técnicas de previsão dos intervalos de disponibilidade que ocorrem nos recursos.

No entanto, esta não é uma tarefa de fácil realização. A identificação destes períodos incorre na previsão do futuro, que no caso de recursos de grades oportunistas significa a previsão do comportamento do usuário ao utilizar o recurso.

O alto custo imposto ao sistema, ou até mesmo a impossibilidade, para identificar estes períodos de forma determinística, induz ao desenvolvimento e utilização de heurísticas para auxiliar o escalonador nas tomadas de decisões de escalonamento. De fato, outros escalonadores utilizando heurísticas de escalonamento já foram propostos e bons resultados foram obtidos.

Considerando o acima exposto, tem-se como motivação para este trabalho, auxiliar escalonadores de grades oportunistas em suas tomadas de decisões, através do fornecimento de informações sobre a disponibilidade dos recursos participantes da grade oportunista. Com esta informação, reduz-se a incerteza para o escalonador de tarefas, permitindo ao

escalonador tomar melhores decisões sobre como o escalonamento será realizado.

A previsão de disponibilidade é normalmente realizada com base em dados históricos de utilização de recursos. Diversos algoritmos de previsão de disponibilidade foram implementados, considerando-se realizar a previsão sobre o percentual de utilização de CPU dos recursos, como alguns dos trabalhos apresentados na seção de trabalhos relacionados deste trabalho.

Apesar destes algoritmos apresentarem bons resultados, eles são específicos para uso de CPU. O problema dessa abordagem está no fato de existirem outras políticas de seleção de recursos para grades oportunistas, como por exemplo, a utilização do mouse e teclado ou até mesmo a entrada e saída do protetor de tela (*Screen Saver*). Assim, estes algoritmos não são aptos para realizar a previsão de disponibilidade para outras políticas de seleção de recursos, além da seleção por percentual de uso de CPU.

Por esta razão, agrega-se também como motivação desse trabalho a possibilidade de realizar a previsão de disponibilidade para diversas políticas de seleção de recursos, possibilitando a adoção do algoritmo proposto com as mais diversas políticas de seleção de recursos de grades oportunistas.

1.2 Objetivos

Assim, destaca-se como objetivo principal deste trabalho, a proposta de um modelo de previsão de disponibilidade para fornecer informações sobre a disponibilidade de recursos e capaz de realizar estas previsões com as mais diversas políticas de seleção de recursos, de modo a permitir que escalonadores de grades oportunistas utilizem os recursos disponíveis de forma otimizada.

Como objetivos específicos, tem-se:

- analisar a disponibilidade de recursos em ambientes reais;
- propor um modelo dinâmico, capaz de prever comportamentos de exceção, para realizar previsões sobre a disponibilidade dos recursos;
- evitar a adição de custos computacionais, ou de armazenamento, excessivos para realizar a previsão de disponibilidade do recurso;
- evitar a adição de custos computacionais ou de comunicação ao servidor;
- implementar um protótipo contendo o modelo proposto;

1.3 Contribuições do Autor

O desenvolvimento deste trabalho levou a uma série de contribuições. Estas contribuições originaram-se tanto da análise dos dados de disponibilidade de recursos, quanto do projeto e implementação do modelo proposto. Entre as principais contribuições destacam-se:

- dados de utilização de recursos pertencentes a uma empresa e uma escola;
- resultados da análise de disponibilidade dos dados de utilização de nove ambientes reais;

- modelo de previsão de disponibilidade baseado na similaridade de comportamentos - AvSchedP;
- protótipo do modelo de previsão AvSchedP;
- resultados de experimentos realizados para avaliação do protótipo implementado.
- protótipo do modelo AvSchedP integrado ao sistema de grade oportunista *XtremWeb*, chamado de *XtremWebAv*.
- resultados preliminares obtidos com o *XtremWebAv*.

1.4 Organização do Texto

Este trabalho está disposto em 6 capítulos. O capítulo 2 apresenta um breve histórico, a arquitetura, o escalonamento de tarefas e sistemas de grades oportunistas. O capítulo 3 aborda o problema que motiva a realização deste trabalho, que é a disponibilidade de recursos. Neste capítulo é apresentada a análise de disponibilidade de recursos apresentada neste trabalho bem como, os principais trabalhos relacionados. O capítulo 4 apresenta os componentes e algoritmos do modelo de previsão de disponibilidade **AvSchedP**. No capítulo 5, são apresentados o protótipo do modelo **AvSchedP**, os experimentos realizados com este protótipo e os resultados obtidos. O capítulo 6 apresenta as conclusões e trabalhos futuros gerados neste trabalho.

O trabalho conta também com o apêndice A onde são apresentados alguns coeficientes de similaridade existentes na literatura e utilizados na determinação da similaridade existente entre duas sequências binárias. Neste apêndice apresenta-se também, o principal motivo para a escolha do *Coeficiente de Hamann* como coeficiente de similaridade no modelo proposto.

2 GRADES COMPUTACIONAIS OPORTUNISTAS

Este capítulo apresenta uma visão geral sobre Grades Computacionais Oportunistas.

Nesta visão, é apresentado um breve histórico dos sistemas de grade oportunista mais conhecidos no mundo científico, como o Condor, o BOINC e o XtremWeb.

Cada um destes sistemas é apresentado na literatura com um termo diferente associado. O BOINC, por exemplo, é considerado um sistema de *Volunteer Computing*, enquanto o XtremWeb é referenciado como um sistema de *Global Computing*.

A realidade é que diversos termos permeiam o conceito de Grade Oportunista. *Global Computing* (FEDAK et al., 2001), *Volunteer Computing* (SARMENTA, 2001), *Desktop Grids* (CONSTANTINESCU-FULØP, 2008), *Enterprise Desktop Grids* (KONDO et al., 2006), *Public Resource Computing* (ANDERSON, 2004) e *Internet Computing* (BYUN et al., 2005), são alguns dos termos encontrados na literatura, e que apesar de apresentarem diferenças em alguns aspectos conceituais, convergem para um mesmo modelo, o oportunista.

Além do histórico, esta visão apresenta a arquitetura de um modelo oportunista, os principais sistemas e o modelo de escalonamento de tarefas frequentemente utilizado nestes sistemas.

Por fim, o capítulo destaca o problema de escalonamento de tarefas, encontrado em sistemas de Grade Oportunista, onde a principal causa é a alta volatilidade dos recursos pertencentes a estes ambientes.

2.1 Introdução

A análise e projeto de algoritmos para a solução de problemas originados por alguma ciência, é chamada de **Computação Científica** (*Scientific Computing*). A Computação Científica desenvolve uma série de aplicações, onde cada aplicação corresponde ao modelo matemático de uma realidade estudada.

Estas aplicações costumam gerar milhares de tarefas. Este grande número de tarefas deve-se, principalmente, à possibilidade de aplicar a um mesmo algoritmo, diferentes valores de entrada [1].

A execução de uma tarefa de uma aplicação, fornece os resultados de uma determinada situação em um cenário estudado (CONSTANTINESCU-FULØP, 2008).

Um grande problema está na impossibilidade de executar um grande número de tarefas, em um único recurso e em tempo hábil.

Para possibilitar a execução de grandes conjuntos de tarefas, frequentemente utiliza-se algum tipo de sistema que aglomera um conjunto de recursos, como os sistemas paralelos

[1] Valores de entrada são os parâmetros de entrada necessários a execução de uma determinada aplicação.

e distribuídos.

Computação em Grade (*Grid Computing*) é um sistema distribuído que pode ser utilizado como uma alternativa para o processamento de grandes quantidades de tarefas computacionais, em um ambiente que promove o compartilhamento recursos pertencentes a diferentes entidades (FOSTER, 2001).

Como alternativa a computação em grade, grades oportunistas podem ser utilizadas quando as tarefas são independentes entre si (SARMENTA, 2001). Grades oportunistas oferecem um menor custo e são facilmente configuráveis, tornando-se um alternativa altamente atrativa, principalmente quando os requisitos de escalabilidade, usabilidade, tolerância a falhas, estiverem presentes (BEHSAZ; JAFERIAN; MEYBODI, 2006).

As **Grades Computacionais Oportunistas** (*Opportunistic Grids*) tem como característica o recrutamento de computadores convencionais[2], quando estes encontram-se em estado ocioso. Estes recursos são utilizados com o propósito de executar o maior número de tarefas a um baixo custo financeiro[3].

Pode ser vista como uma plataforma computacional, geralmente constituída por um elevado número de recursos computacionais, os quais podem estar localizados em redes locais ou largamente distribuídos interconectados por redes de longa distância. De fato, um sistema de grade oportunista não faz qualquer distinção sobre a localização dos recursos participantes.

Os recursos podem estar localizados em ambientes controlados, como os proporcionados por empresas, onde existem funcionários habilitados para a realização de configurações avançadas, redundância de equipamentos e monitoramento, mas também podem estar localizados em ambientes residenciais, desprovidos de recursos sofisticados e sob utilização de usuários não capacitados para a realização de configurações complexas.

A utilização de grades oportunistas torna-se ainda mais atrativa ao encontrar-se estudos indicando um elevado nível de ociosidade dos recursos existentes, que chegam a permanecer 80% do tempo, ou mais, livres (ANURAG; GUY; JOEL, 1997).

Outra forte característica de grades oportunistas, é a de assumir a possibilidade de alocar tarefas a recursos que não permanecem o tempo todo disponíveis para a grade, ou seja, assumir a possibilidade de alocar tarefas a recursos, cujo o tempo de disponibilidade pode ser menor que o tempo necessário para executar a tarefa (THAIN; TANNENBAUM; LIVNY, 2004)(WRIGHT, 2001).

Ao pesquisar sobre grades oportunistas, o pesquisador depara-se com uma série de nomenclaturas distintas. Os termos *Global Computing*, *Volunteer Computing*, *Desktop Grids*, *Enterprise Desktop Grids*, *Public Resource Computing* e *Internet Computing*, são alguns dos termos existentes. Apesar de apresentarem algumas diferenças conceituais não tão evidentes, devido a falta de definições fortes, todos estes termos convergem para o modelo oportunista.

Para facilitar a exposição do texto, neste trabalho os termos **Grade Oportunista** ou **Computação Oportunista** (*Opportunistic Computing*) serão utilizados em substituição aos demais termos. Faz-se importante salientar que o emprego destes dois termos não tem por objetivo indicar que todos os demais termos tem o mesmo significado, mas sim, que todos os termos seguem o modelo oportunista considerado neste trabalho.

No que tange a história de grades oportunistas, a literatura indica que os primeiros sis-

[2]Por recursos convencionais entenda-se o uso de estações de trabalho (*Desktops*) e computadores pessoais (*Personal Computers*)

[3]Atinge-se baixo custo financeiro devido à utilização de recursos ociosos de uma infraestrutura já existente.

temas de Computação Oportunista surgiram nas décadas de 80 e 90. Os mais conhecidos no meio acadêmico são os projetos **Condor** (LITZKOW; LIVNY; MUTKA, 1988), **Great Mersenne Prime Search (GIMPS)** (CALDWELL, 2010), (CONSTANTINESCU-FULØP, 2008), **Distributed.net** (SULLIVAN WOODRUFF T. et al., 1997) e **SETI@home** (SULLIVAN WOODRUFF T. et al., 1997).

O projeto Condor (LITZKOW; LIVNY; MUTKA, 1988), um dos mais populares no meio acadêmico, foi um dos mais importantes para o desenvolvimento da Computação Oportunista. Entretanto, o seu modelo foi projetado para executar em redes privadas apresentando dificuldades de utilização em ambientes altamente distribuídos, como o proporcionado na Internet.

Já o projeto SETI@home (SULLIVAN WOODRUFF T. et al., 1997) foi concebido para executar em ambientes largamente distribuídos, como a Internet. Neste ambiente é utilizado um servidor *HTTP* para permitir o *download* e *upload* de tarefas e resultados, sem sofrer bloqueios por *firewalls*. Este projeto atraiu milhares de participantes espalhados pelo mundo, chegando a atingir mais de 70 TeraFLOPS de processamento (ANDERSON, 2004).

2.2 Arquitetura de Grades Oportunistas

Em uma Grade Oportunista têm-se três papéis distintos. O servidor que controla as atividades da arquitetura, o trabalhador que executa as tarefas existentes e o cliente que submete as tarefas.

Segundo Choi et al. (2006), a arquitetura de uma Grade Oportunista pode ser descentralizada ou centralizada.

Na arquitetura descentralizada, o servidor não é tratado como um componente central, podendo qualquer recurso assumir este papel. É possível que um recurso assuma qualquer combinação de papéis entre os papéis de servidor, trabalhador e cliente. Nesta arquitetura, os recursos apresentam-se a recursos vizinhos, formando grupos, onde cada membro de um grupo pode gerenciar, submeter ou executar tarefas do grupo.

Em uma arquitetura centralizada, existe a figura do servidor como elemento central. Os demais recursos podem ser trabalhadores, clientes ou a combinação concorrente destes dois papéis. No entanto, não são habilitados a assumir o papel de servidor.

Derrick Kondo (2005) apresenta uma arquitetura genérica e em conformidade com a arquitetura centralizada, descrita por Choi et al. (2006).

Os principais sistemas de grades oportunistas como (*Condor*, *BOINC* e *XtremWeb*)[4] enquadram-se nesta arquitetura e por este motivo, a arquitetura descrita neste capítulo será apenas a arquitetura centralizada.

Neste trabalho, foi assumida a seguinte nomenclatura para os componentes da arquitetura de uma grade oportunista:

- **Server** para o componente servidor;
- **Worker** para o componente trabalhador;
- **Client** para o componente que submete tarefas.

De fato, esta nomenclatura já é empregada por alguns sistemas de Grade Oportunista, como o *XtremWeb*.

[4]Estes sistemas serão descritos posteriormente.

Os três componentes da arquitetura de uma Grade Oportunista estão dispostos na figura 2.1. Além dos componentes *Server*, *Worker* e *Client*, a figura também apresenta o banco de dados, que tem como principal finalidade, a persistência de informações sobre *Workers*, *Clients*, aplicações e tarefas destas aplicações.

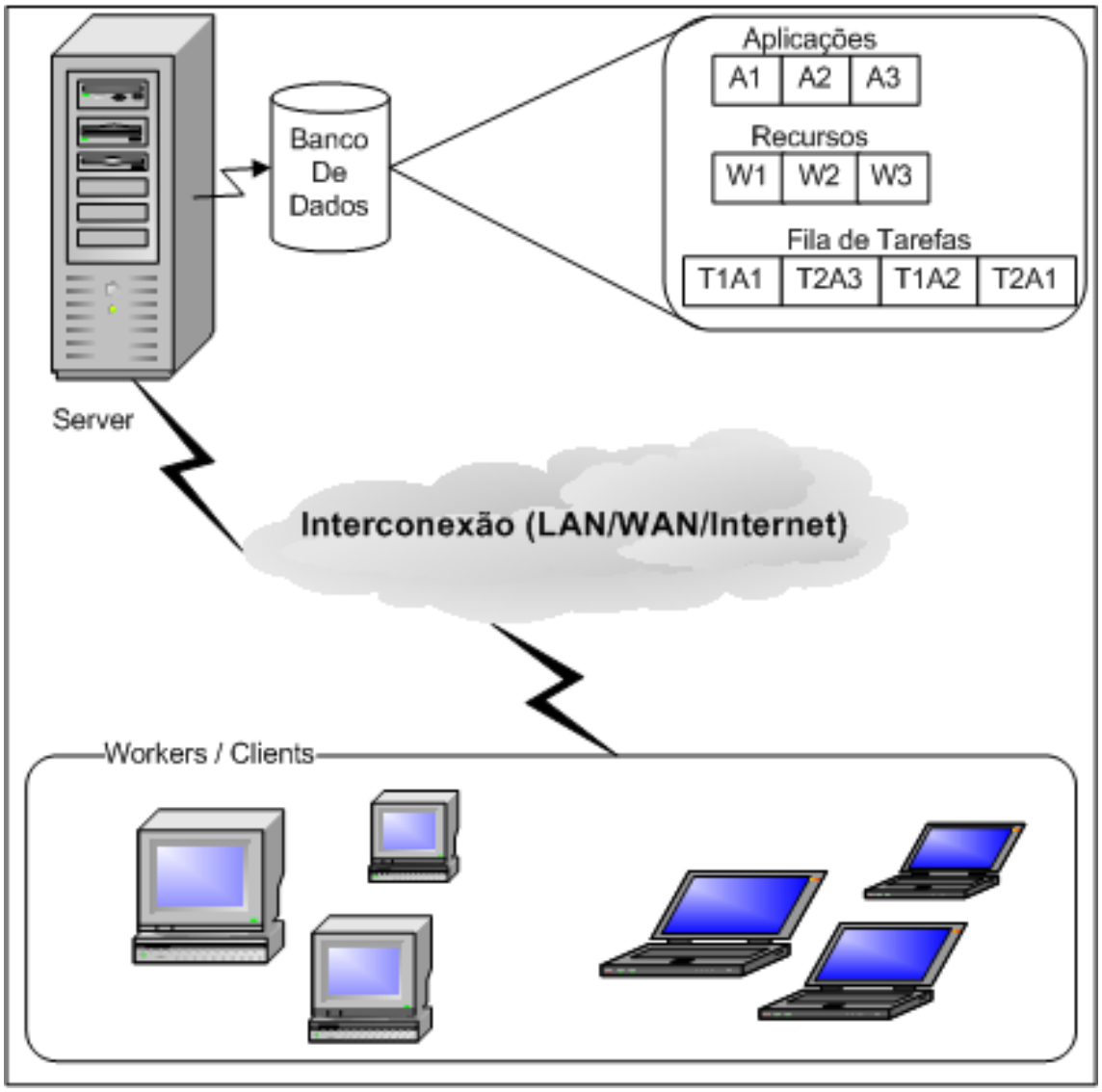

Figura 2.1: Componentes da arquitetura de uma Grade Oportunista

O componente *Client* tem como principais responsabilidades, a submissão de tarefas ao servidor para execução e a coleta de resultados obtidos com a conclusão destas tarefas. Sempre que uma aplicação deve ser executada com um determinado conjunto de parâmetros de entrada, o *Client* gera uma tarefa e a submete ao *Server*.

O *Client* pode ainda, monitorar tarefas submetidas através do envio de consultas encapsuladas em requisições enviadas diretamente ao componente *Server* da arquitetura.

O *Server*, por ser o elemento central da arquitetura, realiza o monitoramento e controle de todos os demais componentes da arquitetura. Este componente geralmente encontra-se em um ambiente controlado e por este motivo, pode utilizar-se de recursos e configurações mais avançadas (CHOI et al., 2006).

No que diz respeito a gerência de aplicações, o *Server* pode ter uma ou muitas aplicações sob sua responsabilidade, como demonstra a figura 2.1.

Cada aplicação pode possuir uma série de tarefas de processamento intensivo (*CPU - Bound tasks*), tarefas estas que realizam poucas operações de entrada e saída (KONDO, 2005). Na figura 2.1, a tarefa *T1A1* é a tarefa de número 1 da aplicação *A1*.

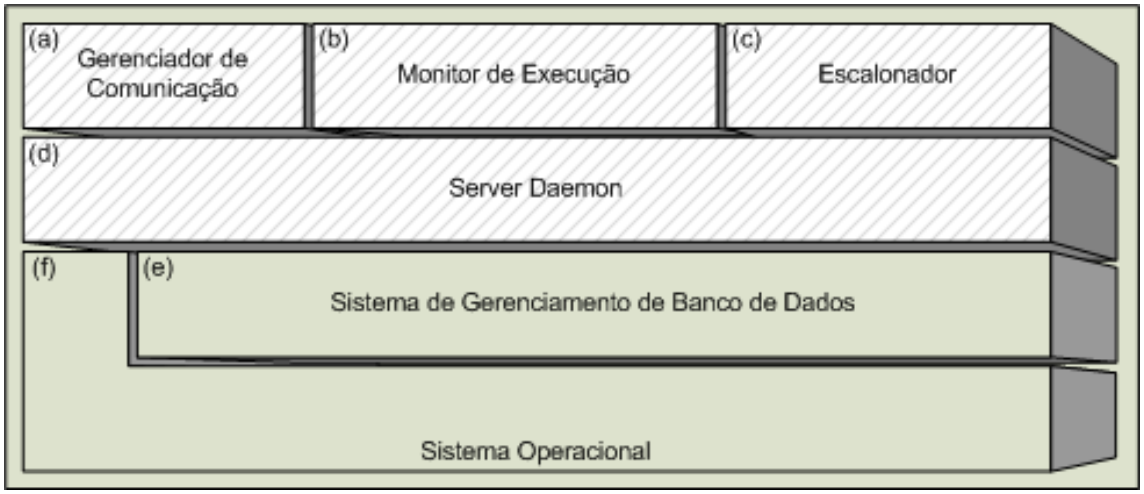

Figura 2.2: Arquitetura de um *Server* baseada na implementação do *XtremWeb*

Para apresentar algumas das responsabilidades do *Server*, neste trabalho estudou-se a implementação do sistema de Grade Oportunista **XtremWeb versão 1.8**. Na implementação do *XtremWeb Server* identificou-se uma série de subcomponentes arquiteturais, apresentados na figura 2.2 e descritos abaixo:

- *Gerenciador de Comunicação* - o componente Gerenciador de Comunicação (a) é responsável pelo tratamento das diversas mensagens enviadas por *Workers* e *Clients*;

- *Monitor de Execução* - o componente Monitor de Execução (b) opera em conjunto com o Gerenciador de Comunicação, e é responsável pelo monitoramento das tarefas existentes no *Server*. De acordo com as informações sobre o estado das tarefas, recebidas pelo Gerenciador de Comunicação, e com as informações sobre os recursos, enviadas pelos *Workers*, o Monitor de Execução atualiza o estado das tarefas e também, o estado dos *Workers*.

  Uma importante atribuição do Monitor de Execução é a detecção dos recursos que falharam. Cabe ao Monitor de Execução detectar se um recurso não está mais ativo ou se uma tarefa foi perdida. Ele deve dispor de mecanismos para detectar falhas e notificar o componente Escalonador sobre a necessidade de realizar um novo escalonamento para estas tarefas;

- *Escalonador* - O componente Escalonador (c) é o responsável pela seleção e entrega de tarefas aptas para execução, mediante o recebimento de requisições enviadas por *Workers*. Este componente opera em conjunto com o Gerenciador de Comunicação, que recebe e encaminha as requisições enviadas pelos *Workers*;

- *Server Daemon* - O componente *Server Daemon* (d) é o componente responsável por coordenar a operação dos demais componentes do *Server*.

- *Banco de Dados* - O componente *Banco de Dados* (e) é utilizado na persistência de informações pertinentes à arquitetura.

- *Sistema Operacional* - O *Sistema Operacional* (f) é utilizado pelos demais componentes durante a realização de suas atividades.

O *Worker*, o último elemento da arquitetura, tem como responsabilidades, solicitar tarefas ao *Server*, executar estas tarefas e manter o *Server* informado sobre a situação das tarefas alocadas ao *Worker*, bem como, sobre a situação do próprio *Worker*.

Assim como na demonstração das responsabilidades do *Server*, optou-se por apresentar as principais responsabilidades do *Worker*, baseando-se no estudo realizado sobre a implementação do *XtremWeb*. A figura 2.3 apresenta os principais subcomponentes identificados em um *XtremWeb Worker*.

Uma breve descrição destes componentes encontra-se abaixo:

- *Gerenciador de Comunicação* - o componente Gerenciador de Comunicação (a) coordena a comunicação, ou seja, a troca de mensagens realizada entre o *Worker* e o *Server*. Nesta comunicação, as principais mensagens são: (1) a requisição de registro do *Worker* na grade (*join*); (2) a requisição de tarefas para execução; (3) mensagem de entrega dos resultados gerados; (4) mensagem de informação de estado do *Worker*. Este componente em conjunto com o Gerenciador de Execução, possibilita manter o *Server* atualizado sobre o estado das tarefas em execução e do estado dos *Workers*;

- *Gerenciador de Execução* - o componente Gerenciador de Execução (b) realiza a gerência local da execução de uma tarefa no *Worker*. Entre as principais atividades do Gerenciador de Execução estão: (1) iniciar a execução de tarefas e (2) suspender a execução de tarefas. As atividades do Gerenciador de Execução são baseadas nas informações geradas pelo componente Monitor de Atividade;

- *Monitor de Atividade* - o Monitor de Atividade (c) determina o estado de ociosidade do recurso. Ele fornece a informação sobre a disponibilidade ou indisponibilidade do recurso. Para determinar a disponibilidade, o Monitor de Atividade considera a política de recrutamento [5] configurada no *Worker*;

- *Worker Daemon* - O *Worker Daemon* (d) é o componente responsável pela coordenação e união dos demais componentes.

- *Sistema Operacional* - O componente (e) corresponde ao Sistema Operacional (SO) do *Worker*, utilizado pelos demais componentes durante a execução de suas atividades.

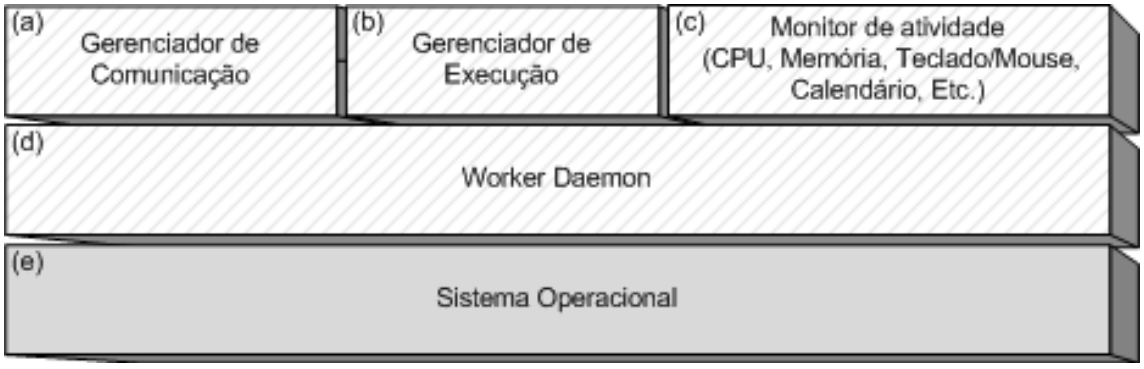

Figura 2.3: Arquitetura de um *Worker* baseada na implementação do *XtremWeb Worker*

[5] Política de recrutamento é a regra utilizada para determinar se o recurso encontra-se ocioso ou ocupado.

Apesar da apresentação de componentes de software baseados na implementação do sistema *XtremWeb*, a arquitetura aqui descrita é genérica e condizente com a arquitetura encontrada em diversos sistemas de grade oportunista. Alguns destes sistemas serão apresentados na Seção 2.4.

2.3 Escalonamento em Grades Oportunistas

Sistemas distribuídos em geral, dispõem de grande capacidade de processamento. No entanto, a utilização eficiente desta capacidade não é uma tarefa trivial.

Para obter sucesso na utilização desta capacidade, estes sistemas contam com um Juiz para determinar a distribuição adequada de tarefas aos recursos disponíveis. A este Juiz dá-se o nome de **Escalonador**. De fato, o ponto chave para que uma aplicação seja executada de forma eficiente é o escalonamento das tarefas desta aplicação (CONSTANTINESCU-FULØP, 2008).

Assim como nas demais categorias de sistemas distribuídos, um dos componentes mais importantes de grades oportunistas é o escalonador. Um escalonador é composto de um mecanismo e uma política. O mecanismo possibilita a ação de escalonar, enquanto a política define como será realizado o escalonamento (VIDYARTHI et al., 2008). Pode-se dizer que o mecanismo de escalonamento utiliza uma política de escalonamento para realizar a ação de escalonar.

Além do escalonamento de tarefas, o escalonador tem a responsabilidade de coordenar a execução destas tarefas (VIDYARTHI et al., 2008).

O escalonamento em grades oportunistas enquadra-se na classificação, ou taxonomia, proposta por Casavant e Kuhl (1988). Esta é uma das mais conhecidas e referenciadas taxonomias de escalonamento em sistemas distribuídos. Ela define e apresenta hierarquicamente o escalonamento em sistemas distribuídos.

A figura 2.4 é uma representação parcial desta taxonomia, onde os elementos em destaque referem-se a ramificação que melhor descreve o escalonamento em grades oportunistas.

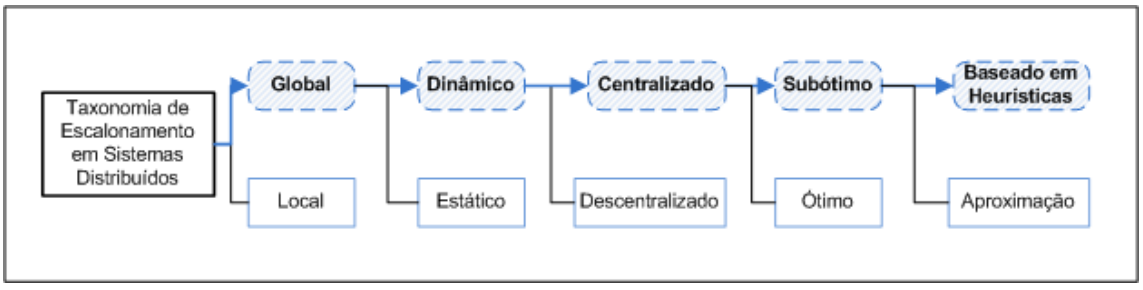

Figura 2.4: Apresentação parcial da taxonomia de escalonamento de Casavant

Um **escalonador local** define a ordem e tempo de execução de cada tarefa existente no recurso computacional. Um **escalonador global** distribui ou aloca tarefas a diferentes recursos computacionais. Após a realização do escalonamento global, o escalonamento local é utilizado para determinar a execução local da tarefa. Em uma Grade Oportunista, o escalonador do *Server* é classificado como global, já no *Worker* ocorre o escalonamento local, realizado pelo próprio sistema operacional e possivelmente por alguma implementação de escalonador local existente no sistema de grade oportunista utilizado.

O **escalonamento estático** é aquele realizado antes de ser iniciada a execução de qualquer tarefa no sistema. Neste tipo de escalonamento, o primeiro passo a ser realizado é a

alocação das tarefas aos recursos computacionais disponíveis. O segundo passo é a distribuição das tarefas aos recursos, na ordem em que foi estabelecida no escalonamento. A definição da alocação é frequentemente realizada utilizando-se grafos acíclicos direcionados (*Directed Acyclic Graphs*) ou DAGs (WU, 1998).

Para determinar a alocação de tarefas a recursos computacionais, um escalonador estático conta com a existência de informações sobre as tarefas e recursos para realizar as tomadas de decisão de escalonamento (WU, 1998).

Uma constatação importante é que o escalonamento estático é considerado mais adequado para ambientes formados por recursos homogêneos (CONSTANTINESCU-FULØP, 2008; WU, 1998), o que é totalmente diferente do ambiente de uma grade oportunista.

O **escalonamento dinâmico** é aquele realizado em tempo de execução, ou seja, no escalonamento dinâmico a alocação de tarefas a recursos computacionais é realizada já com o sistema em funcionamento. Neste tipo de escalonamento, a alocação e distribuição de tarefas podem ser realizadas a qualquer instante, mesmo com tarefas já em execução.

No escalonamento dinâmico, o escalonador requer pouca ou nenhuma informação sobre tarefas e recursos do ambiente (WU, 1998).

Observando-se as exigências definidas por um modelo de escalonamento estático, nota-se que o escalonamento dinâmico é considerado o mais adequado para uma grade oportunista, onde os recursos são heterogêneos e as tarefas têm tempo de execução variável (WU, 1998).

Kumar et al. (1994) apresentaram uma extensão à taxonomia de Casavant e Kuhl (1988). Propuseram a extensão da classificação de escalonamento estático e dinâmico, introduzindo aspectos como a existência ou não de preempção de tarefas.

O **escalonamento sem preempção** aloca tarefas a recursos, devendo estas tarefas serem executadas por completo no recurso selecionado e sem sofrer interrupções.

No **escalonamento com preempção**, a execução das tarefas alocadas a recursos pode ser interrompida. Neste caso, uma tarefa interrompida pode ser removida do recurso ao qual foi alocada inicialmente e ser realocada em outro. Além disso, a execução de tarefas interrompidas pode ser reiniciada.

É possível perceber que escalonadores dinâmicos e preemptivos são adequados ao ambiente de grades oportunistas, já escalonadores estáticos não são tão adequados. Um ponto significativamente negativo do escalonamento estático, quando considerado uma grade oportunista, é que a definição da alocação de tarefas é realizada uma única vez.

A forma de alocação de tarefas em modelos de escalonamento estático não condiz com as características de grades oportunistas, onde ocorrem constantes entradas e saídas de recursos, além da chegada de novas tarefas a qualquer instante. Estas características dificultam, ou até mesmo impedem, a utilização do escalonamento estático em grades oportunistas.

O escalonamento dinâmico adapta-se melhor a ambientes onde o número de recursos disponíveis pode variar durante a execução do sistema (WU, 1998).

Conforme apresentado por Wu (1998), um algoritmo de escalonamento dinâmico frequentemente emprega cinco políticas durante a tomada de decisão de escalonamento. São elas:

- **Política de Inicialização**: A política de inicialização indica quem deve iniciar um evento de escalonamento no sistema. A inicialização pode ser realizada pelo recurso mais ocioso, recebendo o nome de *Receiver-Initiated*, ou pelo recurso mais ocupado, recebendo o nome de *Sender-Initiated*.

- **Política de Transferência**: A política de transferência determina se o recurso encontra-se em estado apropriado para realizar a transferência de tarefas. Normalmente são utilizados limites (*tresholds*) na determinação do estado do recurso. Quando a carga do recurso ultrapassa o limite superior especificado, a política de transferência determina a remoção de tarefas do recurso. Quando a carga está abaixo do limite inferior, a política de transferência determina que o recurso está apto a receber novas tarefas para execução.

- **Política de Seleção**: A política de seleção determina qual a tarefa mais adequada para execução em um determinado recurso.

- **Política de Localização**: A política de localização determina como encontrar recursos para compartilhar tarefas.

- **Política de Informação**: A política de Informação determina quando, onde e quais informações devem ser coletadas. A política de informação define quais são as informações necessárias para realizar a distribuição eficiente de tarefas. A coleta de informações é classificada como periódica, quando as informações são coletadas em um intervalo de tempo definido, ou não periódica, quando a coleta de informações é realizada de acordo com a necessidade do algoritmo de escalonamento.

No **escalonamento descentralizado**, as decisões de escalonamento são determinadas por diversos recursos distribuídos. Neste caso, o mecanismo de controle fica fisicamente distribuído em cada recurso computacional do sistema (WU, 1998). No **escalonamento centralizado** a decisão de escalonamento é executada por um único recurso. Este recurso coleta as informações necessárias sobre o estado do sistema como um todo e realiza as tomadas de decisão para determinar a alocação de tarefas (WU, 1998).

O escalonamento também pode ser classificado como **escalonamento ótimo** e **escalonamento subótimo**. O escalonamento ótimo possibilita a definição da melhor alocação de tarefas matematicamente possível, segundo algum critério estabelecido. Um exemplo seria obter um escalonamento ótimo quando consideramos a taxa de tarefas prontas por unidade de tempo (*throughput*) como critério. Neste caso o escalonamento ótimo definiria a melhor alocação possível para obter o maior número de tarefas executadas em um espaço de tempo. O escalonamento subótimo é todo o escalonamento não ótimo e resultam em escalonamentos próximos ao escalonamento ótimo.

A natureza dos sistemas distribuídos faz com que o escalonamento de tarefas neste ambiente seja classificado como NP-Completo (WU, 1998), sendo então o escalonamento subótimo adequado para estes sistemas.

O escalonamento subótimo é especializado em **escalonamento por aproximação** ou **escalonamento por heurística**. No escalonamento por aproximação, são utilizados algoritmos capazes de determinar uma boa alocação, próxima à alocação ótima. A proximidade da alocação em relação a alocação ótima deve estar dentro de limites aceitáveis e com comprovação formal.

O escalonamento por heurística realiza alocação de tarefas segundo um possível modelo da realidade, apresentando uma solução aceitável, porém, sem uma comprovação formal. Os algoritmos de escalonamento heurísticos frequentemente utilizam parâmetros para possibilitar maior aproximação da realidade. De fato, diversas propostas de escalonamento para grades oportunistas utilizam algoritmos heurísticos parametrizáveis, devido a complexidade do ambiente.

Kumar et al. (1994) também classificaram o escalonador conforme o número de aplicações a serem escalonadas. O algoritmo de escalonamento pode tratar da alocação de tarefas de uma única aplicação ou de múltiplas aplicações. No escalonamento com **múltiplas aplicações**, o algoritmo de escalonamento deve preocupar-se em selecionar tarefas de diversas aplicações, o que insere mais elementos de complexidade ao algoritmo de escalonamento. O **escalonamento com uma única aplicação**, o escalonador não têm a necessidade de se preocupar com a justiça ou prioridade de escalonamento de tarefas de múltiplas aplicações, não introduzindo complexidades extras ao algoritmo de escalonamento.

As grades oportunistas frequentemente operam com múltiplas aplicações e por esta razão, os escalonadores destes ambientes utilizam o escalonamento de múltiplas aplicações.

Kumar et al. (1994) propuseram ainda uma classificação referente a capacidade de adaptação do escalonador ao ambiente. No **escalonamento não adaptativo**, a política de escalonamento não se altera, independente das variações que venham a ocorrer no sistema.

Já o **escalonamento adaptativo** altera a política de escalonamento de acordo com as variações ocorridas no sistema. A troca de políticas de escalonamento é realizada considerando-se informações obtidas de sensores contidos no sistema em que o escalonador encontra-se.

Choi et al. (2008) propuseram uma taxonomia no contexto de grades oportunistas. A taxonomia proposta define o escalonamento sob três perspectivas: Aplicação, Recurso e Escalonador.

(I) - **Aplicação**: O escalonamento sob perspectiva de aplicação, identifica quais aspectos relacionados à aplicações, devem ser considerados durante o projeto e construção de escalonadores para grades computacionais. Neste aspecto o escalonador deve considerar a existência ou não de dependência entre tarefas, o tipo de tarefas (*IO/CPU bound*, a divisibilidade das tarefas, o padrão de submissão ao escalonador (tarefas são submetidas antes ou durante o escalonamento) e a existência de *QoS* (tarefas exigem qualidade de serviço durante a execução). As Aplicações formadas por tarefas independentes também são conhecidas como *Bag-Of-Tasks Applications*. Aplicações deste tipo são mais adequadas para a execução em ambientes de grades oportunistas (SILVA; CIRNE; BRASILEIRO, 2002).

(II) - **Perspectiva de Recurso**: O escalonamento sob perspectiva de recurso, identifica quais aspectos dos recursos devem ser considerados durante o projeto e construção de escalonadores. A dedicação do recurso define se os recursos participantes são dedicados ou não dedicados ao escalonador da grade. A heterogeneidade dos recursos define se os recursos participantes são heterogêneos ou homogêneos. O padrão de registro no escalonador determina se todos os recursos são registrados no escalonador antes do início do escalonamento ou durante o escalonamento. A qualidade de serviço determina se existe compartilhamento ou balanceamento de carga para manter uma distribuição de carga justa entre os recursos.

(III) - **Perspectiva do Escalonador**: O escalonamento sob perspectiva de escalonador, identifica quais aspectos do escalonador devem ser considerados durante o projeto e construção de escalonadores. A organização define a localização das decisões de escalonamento (centralizada, descentralizada e hierárquica). O modo

define como o escalonamento é iniciado, podendo ser: *i* - pull, quando o escalonamento é iniciado pelo Worker quando se torna ocioso e solicita tarefas ao servidor ou, *ii* - push, quando o escalonamento é iniciado pelo servidor que coleta informações de Workers e envia tarefas para execução nos Workers selecionados (CHOI et al., 2008). A política define se a política de escalonamento utiliza abordagem simples ou heurística. O dinamismo determina se o escalonador é dinâmico ou estático. A tolerância a falhas define os artefatos utilizados pelo escalonador para possibilitar a realização do escalonamento na presença de recursos que falham. A adaptação define se o escalonador altera ou não sua política de escalonamento conforme o estado do ambiente.

Apesar da existência destas possíveis classificações apresentadas por Choi et al. (2008), os sistemas de grades oportunistas, em sua maioria, utilizam a política de escalonamento **FCFS** (Fisrt Come First Served) (KONDO, 2005; DOMINGUES; MARQUES; SILVA, 2006). Isto deve-se principalmente à facilidade de implementação e bons resultados obtidos com o emprego desta política. Além disso, o emprego do modo de escalonamento *pull* também é o mais utilizado.

No modo de escalonamento *push*, o servidor seleciona uma tarefa e a submete para execução em um determinado *Worker*. Já no modelo *pull*, o *Worker* é quem realiza a requisição por tarefa (CHOI et al., 2008). O modo de escalonamento *pull* é utilizado para reduzir a necessidade de realizar configurações de rede, como *Firewall* e NAT, pois o modelo pull pode ser implementado sobre o protocolo HTTP que é amplamente utilizado, não sendo necessário realizar configurações de rede para inclusão de um novo *Worker* no sistema (KONDO, 2005).

A utilização de um modo de escalonamento *push*, exigiria conhecimento técnico avançado dos proprietários dos recursos doados para a realização de configurações necessárias para por o sistema em funcionamento.

2.3.1 O modelo de Execução de Grades Oportunistas

Considerando a arquitetura e classificações ora expostas, sob a perspectiva de recursos as grades oportunistas utilizam-se de recursos não dedicados, heterogêneos e com padrão de registro não determinístico. Sob a perspectiva de aplicação, as tarefas são independentes, do tipo *CPU bound*, atômicas e com padrão de submissão não determinístico. Por fim, sob a perspectiva de escalonador, a organização é centralizada, com escalonamento modo *pull*, dinâmico, com política de escalonamento simples, tolerante a falhas e não adaptativo.

Após esta caracterização de grades oportunistas, faz-se necessário apresentar o modelo de execução deste ambiente. Este modelo pode ser dividido em 6 fases principais. São elas:

- **Registro**: nesta fase, os *Workers* registram suas informações no *Server*. Nome do recurso, endereço de rede, capacidades (processamento, memória, disco), sistema operacional, arquitetura de *hardware*, são algumas informações registradas;

- **Submissão de Tarefas**: *Clients* podem submeter tarefas para execução. A submissão de uma tarefa é o registro desta tarefa e a transferência dos arquivos de entrada necessários para a execução desta;

- **Requisição de Tarefas**: o *Server* recebe requisições por tarefas dos *Workers* registrados e como resposta, entrega tarefas aos requisitantes conforme a política de

escalonamento. Na entrega da tarefa, os arquivos de entrada desta tarefa são transferidos para o *Worker* executá-la;

- **Execução de Tarefas**: cada *Worker* executa as tarefas recebidas;
- **Entrega dos Resultados ao Server**: após o término de execução de uma tarefa, o *Worker* transfere para o *Server* os resultados obtidos;
- **Recuperação dos Resultados**: o *Client* solicita os resultados de tarefas submetidas ao *Server*. Em resposta, o *Server* entrega os resultados solicitados ao *Client*.

Um ciclo de escalonamento é iniciado quando um *Worker* torna-se disponível e envia uma requisição por tarefa ao *Server*.

Em resposta à requisição, o escalonador do *Server* seleciona a primeira tarefa apta na fila de tarefas e a transfere para *Worker*. Tanto a seleção de tarefas, quanto o atendimento de requisições de *Workers* são, na maioria dos sistemas de grades oportunistas, realizados em ordem de chegada, ou seja, realizados seguindo a política *(FCFS)*.

O *Worker* dá início a execução da tarefa. Durante a execução, o *Worker* monitora o estado de utilização do recurso e informa o seu estado e o das tarefas sob sua responsabilidade ao *Server*.

Ao concluir a execução da tarefa, o *Worker* envia os resultados obtidos para o *Server*. O *Server* então atualiza o estado da tarefa concluída e do *Worker*.

O *Client* pode consultar o estado da tarefa através de requisições submetidas ao *Server*. Os resultados disponíveis no servidor podem ser recuperados pelo *Client*.

Se após o término de execução de uma tarefa o *Worker* permanecer ocioso, poderá iniciar um novo ciclo de escalonamento.

Este ciclo de escalonamento apresentado corresponde à um cenário ideal ou perfeito. No entanto, em ambientes de grades oportunistas, este não é o cenário predominante.

O cenário predominante em grades oportunistas é aquele onde falhas ocorrem e são consideradas naturais devido as características dos recursos utilizados. Este cenário deve-se principalmente à utilização de recursos não dedicados. Uma das principais características destes recursos é a alta volatilidade gerada pela execução de tarefas locais do usuário do recurso (KONDO, 2005; DOMINGUES; MARQUES; SILVA, 2006; REN et al., 2006; HUANG et al., 2007).

Falhas em recursos podem ser ocasionadas por qualquer situação que o impossibilite de concluir a execução de tarefas. Estas falhas podem ocorrer devido a atividade local do proprietário do recurso, defeitos de *hardware* do recurso e falhas de rede (KONDO, 2005).

No entanto, a utilização do recurso pelo usuário local é o principal fator de geração de falhas em grades oportunistas (CONSTANTINESCU-FULØP, 2008). Isto ocorre devido a necessidade de priorizar tarefas locais geradas pela atividade do usuário local do recurso. Estas tarefas são prioritárias em relação à tarefas da Grade Oportunista.

Os sistemas de grades oportunistas existentes dispõem de mecanismos capazes de possibilitar a detecção destas falhas e a recuperação do sistema. Por exemplo, nos casos de desligamento de um *Worker*, o *Server* identifica a ausência de mensagens do *Worker* e conclui que o mesmo falhou.

Já nos casos em que o recurso torna-se indisponível devido ao usuário local iniciar alguma atividade no recurso, o *Worker* interrompe as tarefas da grade em execução e comunica o *Server* sobre a falha ocorrida.

Para determinar o estado de disponibilidade do recurso, o *Worker* é configurado com uma determinada *Política de Recrutamento*. No caso do XtremWeb, o componente *Monitor de Atividade*, apresentado anteriormente neste capítulo, considera a política configurada para determinar o estado de ociosidade de um recurso.

Algumas políticas de recrutamento disponíveis no XtremWeb versão 1.8 são:

- **Sempre Disponível**: esta política torna o *Worker* sempre disponível para executar tarefas, desconsiderando qualquer tipo de utilização do recurso;

- **Nível de utilização de CPU**: esta política determina o estado do *Worker*, baseando-se no nível de utilização de (CPU) do recurso. Para identificar o estado do recurso, são definidos limites (*tresholds*) de utilização de (CPU). A definição destes limites é configurável, podendo ser realizada pelo próprio usuário. Nos casos em que o nível de utilização não ultrapassa o limite especificado, o *Worker* é considerado apto para executar tarefas da grade. Nos casos em que este limite for ultrapassado, o recurso é considerado indisponível;

- **Calendário**: esta política determina o estado do *Worker* segundo a configuração em forma de calendário fornecida pelo usuário do recurso. Esta configuração é realizada através da especificação dos períodos de disponibilidade, com hora de início e fim deste período de disponibilidade. Quando o relógio local do recurso coincide com algum período de disponibilidade especificado, o recurso é considerado disponível. Nesta política, mesmo que o usuário local utilize o recurso durante um período de disponibilidade configurado, o recurso será considerado como ocioso e executará tarefas da grade concorrentemente com as do usuário local;

- **Uso de Teclado/Mouse**: esta política determina o estado do recurso segundo a utilização do teclado ou mouse, conectados ao *Worker*. Sempre que estes periféricos estiverem em uso, esta política de recrutamento determina que o recurso está indisponível. Quando o usuário permanece sem utilizar estes periféricos por um intervalo de tempo, o recurso passa a ser considerado ocioso e portanto, apto a executar tarefas da Grade Oportunista.

Ao detectar ou ser notificado sobre a falha de um recurso, o *Server* ajusta o estado das tarefas que estavam em execução neste recurso, tornado-as aptas a serem reescalonadas em outros recursos.

2.4 Sistemas de Grades Computacionais Oportunistas

Com a pesquisa em Grades Computacionais Oportunistas, alguns sistemas livres e de código aberto, foram desenvolvidos e disponibilizados à comunidade, respeitando uma série de características pertinentes ao ambiente.

Segundo Kondo (2005) e Choi et al. (2006) um sistema de Grade Oportunista deve ser:

- *Escalável* - o número de tarefas executadas pelo sistema, conhecido como *throughput*, deve aumentar proporcionalmente ao aumento do número de recursos pertencentes ao sistema. Além disso, o aumento do número de recursos não deve causar degradação significativa do sistema;

- *Tolerante a Falhas* - a ocorrência de uma falha em algum dos componentes do sistema não pode levar o sistema a um estado inconsistente;
- *Seguro e confiável* - o sistema deve ser seguro para quem submete e para quem executa tarefas. Quem submete tarefas deve ter a garantia da não modificação da aplicação bem como da não visualização dos dados. Quem executa tarefas deve ter a garantia de que se uma tarefa for introduzida ao sistema com código errado ou malicioso, esta tarefa não causará danos ao recurso (*sandbox*);
- *Facilidade de uso, instalação e integração* - o sistema deve ser fácil de instalar, atualizar e ter preferencialmente, configuração automática. Deveria oferecer também, um conjunto de ferramentas para permitir o gerenciamento de recursos e tarefas, além de facilidade de integração com aplicações já existentes;
- *Não intrusivo* - o sistema não deve interferir nas atividades diárias do usuário, muito menos prejudicá-lo na realização de suas tarefas diárias. Como o *Worker* é um recurso doado, a capacidade de processamento deste recurso é compartilhada entre a execução de tarefas geradas pelo usuário local e a execução de tarefas da grade. É imprescindível priorizar a execução de tarefas do usuário local em detrimento a execução de tarefas da grade. Com outras palavras, as tarefas da grade devem ser interrompidas sempre que o usuário necessitar realizar alguma atividade no recurso. As exceções a esta regra ficam por conta do próprio usuário local, pois ele têm a possibilidade de configurar níveis de compartilhamento diferenciados com a aplicação de diferentes políticas de recrutamento;
- *Incentivos* - o sistema deve oferecer alguma forma de recompensa aos doadores de recursos, como incentivo à doação de ciclos de processamento;
- *Desempenho* - é extremamente importante que o sistema apresente bom desempenho. Um bom desempenho pode ser obtido através de boas decisões ao alocar tarefas. Neste caso, o escalonador assume um papel importante na arquitetura. Ele deve ser eficiente e eficaz na escolha da tarefa a ser entregue ao recurso solicitante.

Entre os sistemas de Grade Oportunista desenvolvidos, destacam-se o *Condor* (LITZKOW; LIVNY; MUTKA, 1988), o *BOINC* (ANDERSON, 2004) e o *XtremWeb* (FEDAK et al., 2001). Com a popularização deste tipo de grade, surgiu também o sistema proprietário *Entropia* (CHIEN, 2003).

Um breve detalhamento destes sistemas será apresentado a seguir.

2.4.1 Condor

O projeto **Condor** (LITZKOW; LIVNY; MUTKA, 1988) surgiu nos anos 80 na Universidade de Wisconsin. Miron Livny combinou sua tese de doutorado em processamento cooperativo com projetos de multi computação, e sistemas de acesso remoto, culminando em um novo sistema de computação distribuída, desenvolvido com a linguagem de programação C (THAIN; TANNENBAUM; LIVNY, 2004).

Condor sustenta-se em dois conceitos altamente relacionados: Computação Oportunista (*Opportunistic Computing*) e Sistemas de Computação em Larga Escala (*High Throughput Computing*) (THAIN; TANNENBAUM; LIVNY, 2004). *High Throughput Computing* tem por objetivo proporcionar a seus consumidores, alta capacidade de processamento computacional por longos períodos de tempo, utilizando-se de recursos conectados à rede (BASNEY; LIVNY, 1999).

A arquitetura do Condor é formada pelos componentes **Central Manager**, **Execution Machine** e *Submission Machine*. Cada componente é composto por uma série de *daemons* responsáveis por manter os serviços necessários a um **Condor Pool**.

Condor Pool é a arquitetura funcional de um sistema *Condor*, sendo este composto por apenas um único *Central Manager* e uma ou muitas máquinas de submissão e execução, *Submission Machine* e *Execution Machine* respectivamente (WRIGHT, 2001).

O *Central Manager* executa o componente **Collector**, *daemon* que mantém o repositório de informações da arquitetura. O *Collector* recebe informações das demais entidades da arquitetura, com o objetivo de manter a arquitetura atualizada sobre o estado e características destas entidades. As características e requerimentos de uma entidade são definidas em uma estrutura de dados chamada de *classAd*. Algumas entidades podem consultar o *Collector* para obter informações sobre outras entidades de interesse, consultando cada *classAd* registrado.

O **Negotiator**, outro *daemon* que executa no *Central Manager*, é responsável por encontrar pares de entidades que contenham características e requerimentos compatíveis entre si, em especial aqueles que oferecem e aqueles que solicitam recursos. Ao encontrar entidades compatíveis, o par de entidades é notificado para proceder com as operações necessárias.

Uma máquina de execução (*Execution Machine*) é aquela que oferta recursos para processamento de tarefas. Em cada *Execution Machine* é executado o *daemon* **startd** que é responsável pelo monitoramento do recurso onde está executando, pela informação de disponibilidade para execução de tarefas e principalmente, pela aplicação da política de recrutamento especificada pelo proprietário do recurso. O *startd* gerencia os momentos em que um recurso deve iniciar, finalizar e suspender a execução de tarefas.

Uma máquina de submissão *Submission Machine* é aquela que submete *jobs*[6] para execução.

O usuário submete ao *Schedd* cada *job* a ser executado.

O *schedd* é um *daemon* localizado na *Submission Machine*. Ele é responsável pela criação e manutenção da fila de *jobs*, por publicar junto ao *collector* o *classAd* de cada *job* da fila, por negociar um recurso para execução e por realizar o escalonamento local de *jobs*.

Ao encontrar um recurso que oferece processamento com o *classAd* compatível ao *classAd* de algum *job* submetido, o *Negotiator* notifica a *Execution Machine* e a *Submission Machine* referentes aos *classAd* compatíveis.

Após receber uma notificação de compatibilidade, o *schedd* da respectiva *Submission Machine* passa a interagir diretamente com o *startd* da respectiva *Execution Machine*, solicitando o controle temporário do recurso.

Como pode ser observado, a arquitetura do *Condor* é semelhante à arquitetura genérica apresentada anteriormente. Os componentes *Central Manager*, *Execution Machine* e *Submission Machine* correspondem, respectivamente, aos componentes *Server*, *Worker* e *Client* ora apresentados.

No entanto, os modelo de execução e escalonamento do *Condor* diferem-se dos modelos descritos anteriormente. Além da existência de escalonamento local, executado pela *Submission Machine*, *Submission Machines* e *Execution Machines* podem comunicar-se diretamente. Esta característica é decorrente do projeto voltado para ambientes controlados, onde administradores de rede realizam as configurações necessárias para possibilitar esta comunicação.

[6]Um *job* é uma entidade da arquitetura e por isso é especificado através de um *classAd*.

2.4.2 Boinc

BOINC (Berkley Open Infrastructure for Network Computing) é um sistema de grade oportunista livre e de código aberto, desenvolvido na Universidade de Berkley, Estados Unidos (ANDERSON, 2004; ANDERSON; MCLEOD, 2007).

O BOINC foi implementado com a linguagem de programação C e tem como principais elementos de sua arquitetura, o servidor chamado de *BOINC Server* e o cliente, chamado de (*BOINC Client*).

O *BOINC* emprega o conceito de projetos. Um projeto pode ser desenvolvido por uma universidade, um laboratório de pesquisa, uma empresa ou até mesmo, por um indivíduo (CHOOSING BOINC PROJECTS, 2010). Um projeto possui um propósito específico, definido pela entidade que o criou. Alguns exemplos de projetos BOINC são apresentados abaixo. Uma lista completa de projetos BOINC pode ser obtida em (CHOOSING BOINC PROJECTS, 2010).

- **Climateprediction.net**: projeto que tenta encontrar o melhor modelo de previsão de tempo. São realizadas comparações dos resultados obtidos de diversas execuções de diferentes modelos de previsão de tempo. A descrição mais completa do projeto pode ser obtida em (CLIMATEPREDICTION.NET, 2010).

- **LHC@HOME**: projeto com o objetivo de melhorar a qualidade dos detectores de colisão de partículas, bem como melhorar a qualidade do próprio acelerador de partículas (LHC). A descrição mais completa do projeto pode ser obtida em (LHC@HOME, 2010).

- **Malariacontrol.net**: projeto criado para auxiliar no desenvolvimento de vacinas contra a malária, bem como na identificação de métodos de controle da doença. Para isso, são realizados mapeamentos da forma de disseminação da Malária e dos efeitos da doença. A descrição mais completa do projeto pode ser obtida em (MALARIACONTROL.NET, 2010).

- **Superlink@Technion**: projeto criado para fornecer informações sobre genes que poderiam determinar a incidência de diabetes em indivíduos. A descrição mais completa do projeto pode ser obtida em (SUPERLINK@TECHNION, 2010).

Um projeto é composto por uma ou mais aplicações. Cada aplicação é um programa registrado no *BOINC Server*. Para desenvolver uma aplicação *BOINC*, o desenvolvedor deve utilizar a interface de programação fornecida, chamada de *BOINC API*.

O *BOINC Server* é composto por, pelo menos, um servidor HTTP para tratamento de *download* e *upload*, e um servidor de banco de dados para manter o registro e estado das tarefas (ULRIK; JAKOB; PEDERSEN, 2005). Cada tarefa é chamada de *Workunit*. Cada *Workunit* é formada por um conjunto de informações de entrada (parâmetros de execução) e uma linha de comando para realizar a execução da aplicação (programa executável) com as informações de entrada.

O *BOINC Server* opera com uma série de *daemons* para tratamento e execução de *Workunits* (ULRIK; JAKOB; PEDERSEN, 2005). O *Work Generator* é responsável pela criação de *Workunits* e informações de entrada desta *Workunit*. Ele é responsável também por registrar estas *Workunits* no banco de dados do *BOINC Server*. O *Transitioner* coordena as transições de estado das *Workunits* e os resultados obtidos. O *Validator* realiza a validação dos resultados entregues, comparando os resultados obtidos de execuções

redundantes de uma mesma *Workunit*. O *Assimilator* verifica periodicamente se existem novas *Workunits* para extração dos resultados obtidos. Após assimilada a *Workunit* passa para o estado de completa. O *File Deletion* exclui as informações de entrada e saída das *Workunits* executadas. O *Feeder* cria um segmento de memória compartilhada, para fornecer as informações necessárias ao *CGI Scheduler*. A criação do segmento de memória é feita para oferecer melhor desempenho, evitando ou reduzindo o número de consultas ao banco de dados (ULRIK; JAKOB; PEDERSEN, 2005).

O *CGI Scheduler* é um programa *CGI* que é executado sempre que um *BOINC Client* efetua uma requisição por *Workunit*.

O usuário desejando compartilhar seu recurso com um projeto *BOINC*, deve realizar o download e instalação do *BOINC Client* e após, inscrever-se no projeto de interesse.

O *BOINC Client* é o responsável pela execução de *Workunits* de projetos. O *BOINC Client* pode ser registrado em múltiplos projetos e pode executar tarefas de qualquer um destes projetos. No entanto, a forma como esse recurso será compartilhado pode ser configurada e estas configurações podem ser globais ou específicas de projetos.

Em uma preferência global, o indivíduo pode especificar se o recurso poderá executar *Workunits* quando o recurso estiver em uso, ou se somente poderá executar tarefas quando o recurso permanecer ocioso durante um intervalo de tempo especificado, por exemplo. Já em uma preferência específica de projeto, o usuário pode configurar a quantidade máxima e mínima de *Workunits* contidas na fila de tarefas local do *BOINC Client*. Esta preferência faz com que o recurso efetue um controle sobre a fila local de tarefas, realizando a solicitação de tarefas ao *BOINC Server*, sempre que o limite mínimo for atingido (ULRIK; JAKOB; PEDERSEN, 2005).

A comunicação entre o *BOINC Client* e o *BOINC Server* é feita através de conexões HTTP, possibilitando o envio de mensagens em ambientes dotados de *Firewall*.

Diferentemente de muitas outras arquiteturas, o *BOINC* não apresenta o responsável pela submissão de tarefas e recuperação de resultados, como um componente da arquitetura. A submissão de tarefas é realizada através do aplicativo auxiliar *Work Generator*. Apenas usuários autorizados podem submeter *WorkUnits* para execução.

Assim, diferentemente da arquitetura genérica apresentada, o *BOINC* não possui três componentes e sim dois. Os componentes *BOINC Server* e *BOINC Client* referem-se respectivamente aos componentes *Server* e *Worker* apresentados.

O modelo de execução também difere-se devido a existência de escalonamento local no *BOINC Client*.

2.4.3 XtremWeb

O *XtremWeb* (FEDAK et al., 2001) é uma plataforma de *software* livre e de código aberto, para exploração de questões científicas e aplicações de grades oportunistas, mais especificamente *Desktop Grid, Global Computing* e *Peer-to-Peer*.

O XtremWeb foi desenvolvido na Universidade de Paris-Sud, França, sendo inicialmente projetado para estudar modelos de execução em ambientes de *Global Computing*. Posteriormente, com a evolução da pesquisa em torno deste projeto, O *XtremWeb* tornou-se uma plataforma de produção (LODYGENSKY et al., 2003).

O *XtremWeb* foi implementado em Java e a arquitetura é formada por um servidor, chamado de *XtremWeb Server*, por recursos que atuam como geradores de tarefas, chamados de *XtremWeb Client* e por recursos que executam tarefas, chamados de *XtremWeb Worker* (CAPPELLO et al., 2005). No entanto, um recurso pode ser tanto *XtremWeb Client*, quanto *XtremWeb Worker* (FEDAK et al., 2001; LODYGENSKY et al., 2003).

O *XtremWeb Server* é o elemento central da arquitetura e é responsável pela gerência de tarefas submetidas para execução. Toda comunicação existente na arquitetura passa pelo *XtremWeb Server*, ou seja, o *XtremWeb Server* realiza a interligação entre *XtremWeb Clients* e *XtremWeb Workers*. Portanto, não existe submissão de tarefas nem entrega de resultados entre *XtremWeb Clients* e *XtremWeb Workers*, sem passar pelo *XtremWeb Server* (ABDENNADHER; BOESCH, 2007).

A comunicação entre o *XtremWeb Worker* e o *XtremWeb Server* é sempre iniciada pelo *XtremWeb Worker*. As principais comunicações realizadas entre o *XtremWeb Worker* e o *XtremWeb Server* são:

- ***workRequest***: mensagem enviada para o *XtremWeb server*, para solicitar uma tarefa para execução. Esta mensagem é enviada sempre que o recurso for considerado disponível;
- ***workAlive***: mensagem enviada periodicamente ao *XtremWeb Server* para atualizar o estado do *XtremWeb Worker*. Esta mensagem também é enviada para manter o *XtremWeb Server* atualizado sobre o estado das tarefas em execução no *XtremWeb Worker*;
- ***sendResult***: mensagem enviada ao *XtremWeb Server* contendo o resultado obtido com a execução de uma tarefa.

Para solicitar tarefas o *XtremWeb Worker* deve estar ocioso. A determinação do estado de um *XtremWeb Worker* é dada pela política de recrutamento nele configurada. A política de recrutamento pode ser alterada pelo proprietário do recurso, conforme a necessidade.

As aplicações do *XtremWeb* são registradas no *XtremWeb Server*. No entanto, o *XtremWeb* não utiliza o conceito de projetos como utilizado no *BOINC*. No *XtremWeb* pode-se encontrar múltiplas aplicações instaladas no *XtremWeb Server*.

Cada tarefa do *XtremWeb*, de forma semelhante ao apresentado no sistema *BOINC*, é formada por um conjunto de informações de entrada e uma linha de comando para a execução dessa tarefa com estas entradas.

2.4.4 Entropia

DCGrid é uma plataforma de computação em grade formada por computadores pessoais, que tornou-se popularmente conhecida como **Entropia**. Entropia era o nome da empresa que mantinha o *DCGrid* como uma plataforma comercial até ser descontinuado.

Por ser comercial, o *Entropia* oferece uma série de funcionalidades para portar aplicações já existentes para o sistema de Grade Oportunista. Com isso, era exigido muito pouco esforço na realização desta tarefa. Esta e outras funcionalidades fizeram com que este sistema fosse instalado em diversos ambientes comerciais abrangendo uma grande variedade de tipos de aplicações (CHIEN, 2003).

O *Entropia* possui uma arquitetura formada por um elemento central, chamado de *Entropia Server* e recursos que executam as tarefas, chamados de *Desktop Clients* que são recursos convencionais com o Sistema Operacional Windows instalado.

O *Entropia Server* possui um gerenciador de *jobs* (***Job Manager***). O sistema permite além da utilização deste gerenciador, a utilização de outros gerenciadores de jobs existentes, como o PBS (CHIEN, 2003). O *Job Manager* realiza a divisão dos *jobs* em tarefas. Logo após, submete estas tarefas ao escalonador de recursos (***Job Scheduler***) (CHIEN et al., 2003).

Tabela 2.1: Tabela resumo dos sistemas de Grade Oportunista abordados

Sistema	Componentes	Organização	Modo	Internet	LP
Condor	Central Manager, Execution Machine e Submission Machine	Centralizado	Push	Não	C
BOINC	Server e client	Centralizado	Pull	Sim	C
XtremWeb	Server, Worker e Client	Centralizado	Pull	Sim	Java
Entropia	Server e Client	Centralizado	Pull e Push	Sim	C

O escalonador de recursos do *Entropia Server*, seleciona os recursos disponíveis na arquitetura, considerando os requerimentos especificados pelo *job*. O *Entropia Server* escalona tarefas windows nos *Desktop Clients* baseando-se nas políticas de recrutamento especificadas pela empresa e pelo proprietário do recurso, e pelos requerimentos impostos pela tarefa a ser escalonada (CALDER et al., 2005) (CHIEN et al., 2003).

A manutenção das informações sobre as características e o estado de um *Desktop Client* é de responsabilidade do gerente de recursos (***Resourse Manager***). O gerente de recursos é uma camada de software existente tanto no *Entropia Server* quanto no *Desktop Client*.

O *Entropia*, semelhantemente ao *BOINC*, não possui um componente para submissão de tarefas e obtenção de resultados. Os resultados devolvidos ao *Entropia Server* pelos *Desktop Clients* são recuperados manualmente pelo usuário final (CHIEN, 2003).

Para finalizar esta seção, apresenta-se a tabela 2.1, onde é apresentado o resumo dos sistemas de Grade Oportunista apresentados.

A coluna **Componentes** apresenta os principais componentes arquiteturais de cada sistema. Já a coluna **Organização** apresenta o tipo de organização do escalonador de tarefas do sistema. A coluna **Modo** especifica se o escalonador de tarefas utiliza o modelo *push* ou *pull* para distribuição das tarefas. Por fim, a coluna **Internet** indica se o sistema é capaz de executar em escala de *Internet*.

2.5 Considerações Finais

Neste capítulo foram apresentados os principais aspectos que contextualizam Grades Computacionais Oportunistas.

Optou-se por utilizar grades oportunistas neste trabalho devido ao contexto desafiador proporcionado por estes ambientes, quando o que se trata é a disponibilidade de recursos[7].

Apresentou-se diversos aspectos relacionados a arquitetura de grades oportunistas, bem como, aspectos relacionados ao escalonamento de tarefas neste ambiente. Dentre os modelos arquiteturais existentes, optou-se por utilizar o modelo centralizado, sendo este o modelo adotado pelos sistemas de grades oportunistas mais conhecidos. E a escolha do modelo de escalonamento pull também baseou-se neste mesmo critério, sendo este o modelo utilizado nos sistemas mais conhecidos.

[7] A disponibilidade de recursos pertencentes a estes ambientes será abordada no próximo capítulo

Neste capítulo, diversos ambientes de grades oportunistas foram apresentados. Destes ambientes, o XtremWeb foi selecionado para a realização dos experimentos realizados neste trabalho. O XtremWeb foi utilizado como base para a integração do modelo de disponibilidade proposto e implementado, possibilitando a realização do conjunto de experimentos planejados.

A escolha do deste sistema foi baseada na familiaridade com a linguagem de programação Java e na facilidade de entendimento do ambiente, visto que o mesmo não dispõe de funcionalidades extremamente sofisticadas como as existentes nos sistemas BOINC e Condor.

O próximo capítulo apresenta a disponibilidade de recursos em um contexto de grades oportunistas. São apresentadas as definições de disponibilidade adotadas neste trabalho e uma análise de disponibilidade de recursos realizada sobre conjuntos de dados coletados neste trabalho e os obtidos em outros sítios.

3 A DISPONIBILIDADE DE RECURSOS COMPUTACIONAIS

Este capítulo apresenta a análise de disponibilidade de recursos realizada neste trabalho. O capítulo inicia pela definição de disponibilidade adotada neste trabalho e segue com a realização da análise propriamente dita.

Foram investigados aspectos como a relação entre o comportamento monitorado em um recurso e o horário de funcionamento do ambiente onde este recurso encontra-se inserido e a duração e frequência dos intervalos de disponibilidade ocorridos nos recursos.

Esta análise identificou aspectos importantes relacionados à utilização de recursos computacionais por seus usuários. Estes aspectos foram utilizados para a análise dos trabalhos relacionados e serviram de fundamentação para a proposta de um novo modelo de previsão de disponibilidade de recursos.

A apresentação dos trabalhos relacionados e a comparação destes com os aspectos identificados também é realizada neste capítulo.

3.1 Introdução

Em grades oportunistas, durante os períodos de ociosidade dos recursos trabalhadores (*Worker*), os ciclos de processamento destes recursos são destinados à execução de tarefas existentes no *Server*.

Como citado no capítulo anterior, a utilização destes recursos para o processamento de tarefas é motivadora devido ao grau de ociosidade apresentado (ANURAG; GUY; JOEL, 1997). No entanto, a utilização destes recursos introduz um grande desafio: amenizar as consequências do uso de recursos voláteis.

A volatilidade destes recursos é tão alta que passa a ser considerada como uma característica inerente a recursos de grades oportunistas (BREVIK; WOLSKI; NURMI, 2003; KONDO et al., 2004). Estes recursos são ditos voláteis devido às frequentes alterações no estado de disponibilidade de cada recurso.

Além de frequente, a alteração entre os estados de disponibilidade apresentados pelo recurso não é realizada de forma consensual com o *Server*, ou seja, não é exigido do recurso que ele comunique previamente ao sistema de Grade Oportunista, as alterações de estado nele ocorridas.

Como consequência da volatilidade, tem-se o alto grau de instabilidade do sistema, causada pelas frequentes interrupções na execução de tarefas. Estas interrupções contribuem para a má utilização dos recursos e para o aumento do tempo total de execução das tarefas de uma aplicação (CONSTANTINESCU-FULØP, 2008; BYUN et al., 2005).

Este capítulo apresenta as definições adotadas neste trabalho, para os principais termos

que cingem disponibilidade.

Neste capítulo também é apresentada uma análise de disponibilidade de recursos, análise esta realizada como parte deste trabalho. A realização desta analise trouxe importantes contribuições utilizadas na definição do modelo de previsão de disponibilidade proposto.

Por fim, são apresentados os trabalhos relacionados ao objetivo deste trabalho e a identificação da necessidade de se propor um novo modelo de previsão de disponibilidade, capaz de realizar previsões individuais, locais e dinâmicas, além de ser capaz de prever também, exceções.

3.2 Disponibilidade no Contexto Deste Trabalho

Disponibilidade é o estado ou qualidade de estar disponível (LETRAS, 2008). Também pode ser interpretada como a qualidade de algo estar presente ou, pronto para uso.

No entanto, quando definimos disponibilidade devemos defini-la sob algum aspecto, sendo este aspecto o responsável pela determinação da qualidade de algo estar apto ou não a realizar uma determinada atividade. Segundo Kondo et al. (2007), definir disponibilidade não se trata apenas de encontrar na literatura uma boa descrição do termo, mas sim, defini-la de acordo com o contexto em que está sendo estudada.

Afim de fundamentar as definições de disponibilidade aqui adotadas, investigou-se na literatura algumas definições e modelos utilizados para caracterizar a disponibilidade de recursos em grades oportunistas.

Rood e Lewis (2007) apresentaram um modelo formado por cinco diferentes estados de disponibilidade possíveis de serem assumidos por um recurso. Neste modelo o recurso pode estar disponível (***Available***) para executar tarefas da grade, com a presença do usuário (***User Presence***), ter excedido o limite de utilização de CPU estabelecido (***CPU Treshold Exceeded***), estar impossibilitado de executar tarefas da grade (***Job Eviction State***), ou estar impossibilitado de se comunicar com o servidor por algum motivo, sendo então considerado como indisponível para a grade (***Unavailable***).

Byun et al. (2006) apresentaram um modelo de disponibilidade com quatro estados de disponibilidade possíveis. ***(HU) Host Unavailability*** indica a falta de participação do recurso no sistema. A falta de participação do recurso pode ser causada pelo desligamento do recurso ou pela impossibilidade de acessar este recurso através da rede. ***(EU) Execution Unavailability*** é a impossibilidade do recurso executar tarefas da grade, e pode ser causada pela utilização do recurso por seu proprietário. ***(HA) Host Availability*** significa que o recurso está participando do sistema e ***(EA) Execution Availability*** significa que o recurso encontra-se apto para executar tarefas da Grade Oportunista. É importante observar que um recurso no estado (*HA*) não necessariamente encontra-se em um estado (*EA*), ou seja, os estados modelados podem ser combinados.

Kondo et al. (2006; 2007) modelaram a disponibilidade em 3 estados possíveis. ***Host Availability*** é um valor binário que define se um recurso está acessível ou não à Grade Oportunista. ***Task Execution Availability***, é um valor binário que define se uma tarefa pode ou não ser executada no recurso, considerando a política de recrutamento adotada neste recurso. ***CPU Availability*** quantifica a fração de CPU que pode ser explorada por uma aplicação da Grade Oportunista.

Ren et al. (2006; 2006) estudaram a indisponibilidade de recursos sob dois aspectos. O primeiro, chamado de ***Unavailability due to Excessive resource Contention***, considera as situações onde uma tarefa da grade é interrompida em um *Worker*, devido a concor-

rência entre tarefas da grade e tarefas geradas pelo usuário local do recurso. O segundo, chamado de ***Unavailability due to Resource Revocation***, considera as situações onde o usuário local do recurso retira o recurso da grade oportunista, ou quando falhas de *Hardware* ou *Software* ocorrem.

Estas definições são importantes para a caracterização e estudo da disponibilidade de recursos. Analisando-as percebe-se que apesar da utilização de diferentes nomenclaturas, as definições apresentam pontos em comum. Dentre estes pontos destaca-se a distinção entre estar disponível, no sentido de ligado, funcional e conectado à rede, e disponível para a grade, no sentido de estar apto à executar tarefas da grade.

Dadas estas definições, faz-se necessário definir a disponibilidade segundo o contexto deste trabalho.

Para este trabalho optou-se por definir a disponibilidade de um recurso, considerando a visão de um escalonador de grade oportunista. Para o escalonador, o recurso pode estar apto ou inapto a executar tarefas da grade. Assim, definiu-se disponibilidade como um valor binário representando os dois estados que o recurso pode assumir: **disponível** ou **indisponível**.

Para representar o estado de disponibilidade dos recursos e possibilitar a realização de estudos quantitativos sobre os dados de disponibilidade dos recursos, assumiu-se a utilização de uma variável binária que assume o valor **Um (1)**, quando o recurso encontra-se ocupado e o valor **Zero (0)**, quando o recurso encontra-se ocioso.

A principal vantagem em se considerar apenas dois estados de disponibilidade é a redução da complexidade ao modelar a disponibilidade do recurso e consequentemente, ao propor um modelo de previsão de disponibilidade. Em contrapartida, tem-se como desvantagem a perda da informação existente nos estados de disponibilidade intermediários identificados pelos demais autores. No entanto, os resultados obtidos neste trabalho demonstram que é possível realizar boas previsões de disponibilidade, considerando-se apenas dois estados de disponibilidade.

Equivalente a definição adotada para o estado **Disponível**, tem-se o estado *Task Execution Availability* do modelo apresentado por Kondo et al. (2007). O estado *Host Availability*, no entanto, não pode ser considerado equivalente, visto que um recurso pode estar acessível mas não apto a executar uma tarefa da grade. O mesmo pode ser dito sobre os estados *(EA) Execution Availability* e *(HA) Host Availability* do modelo apresentado por Byun et al. (2006).

A tabela resumo 3.1, apresenta os possíveis estados de disponibilidade identificados pelos autores citados anteriormente, e o mapeamento que poderia ser realizado entre estes estados e os propostos neste trabalho.

Observando a participação de um recurso ao longo do tempo em uma Grade Oportunista, nota-se a existência de alterações não determinísticas entre os estados disponível e indisponível. O período de tempo observado pode ser representado por uma sequência de valores binários (sequência de 0s e 1s), que identificam as alterações de estado ocorridas no recurso ao longo do tempo.

Nesta sequência de valores binários pode-se ainda identificar subsequências onde o estado de disponibilidade do recurso manteve-se contiguamente igual por um determinado período, ou seja, os casos onde um recurso permaneceu **continuamente** em um dos estados de disponibilidade por mais de uma unidade de tempo.

Definiu-se como **Intervalo de Disponibilidade (D)**, a subsequência, com início e fim, na qual o recurso permaneceu constantemente disponível.

A figura 3.1 apresenta os intervalos de disponibilidade e indisponibilidade ocorridos

em determinado recurso, durante três dias da semana, onde $D1$, $D2$ e $D3$ representam os intervalos de disponibilidade, enquanto $I1$ e $I2$ representam os intervalos de indisponibilidade do recurso.

Ao observar atentamente a figura 3.1 é possível perceber a existência de intersecções entre os intervalos de disponibilidade ocorridos nos dias apresentados. Esta intersecção foi chamada de **Padrão de Disponibilidade (PD)**, visto que retrata a ocorrência de um intervalo de disponibilidade em diversos dias, em um mesmo horário.

Tabela 3.1: Tabela resumo de definições de disponibilidade e mapeamento para a definição adotada

Autor	**Estados de Disponibilidade**	**Definição adotada**	
		Disponível	**Indisponível**
(ROOD; LEWIS, 2007)	Available, User Presence, CPU Treshold Exceeded, Job Eviction State e Unavailable	Available	User Presence, CPU Treshold Exceeded, Job Eviction State e Unavailable
(BYUN et al., 2006)	Host Availability, Execution Availability, Host Unavailability e Execution Unavailability	Execution Availability	Host Unavailability, Execution Unavailability
(KONDO et al., 2007)	Host Availability, Task Execution Availability e CPU Availability	Task Execution Availability	-
(REN et al., 2006)	Unavailability Due Resource Contention e Unavailability Due Resource Revocation	-	Unavailability Due Resource Contention e Unavailability Due Resource Revocation

Na figura 3.1 são apresentados os padrões de disponibilidade $PD1$, $PD2$ e $PD3$.

Cada intervalo de disponibilidade tem um tamanho que representa a quantidade de unidades de tempo em que o recurso permaneceu disponível. Este tamanho foi chamado de **Duração (Dur)**, e pode ser dado em qualquer unidade de tempo. Este termo foi empregado com o mesmo significado por outros autores (NADEEM et al., 2008a; ROOD; LEWIS, 2008; BREVIK; NURMI; WOLSKI, 2004; NADEEM; PRODAN; FAHRINGER, 2008; BREVIK; WOLSKI; NURMI, 2003).

Na figura 3.1, as durações dos intervalos de disponibilidade $D1$, $D2$ e $D3$ ocorridos na Segunda-Feira foram de 45, 60 e 30 minutos respectivamente.

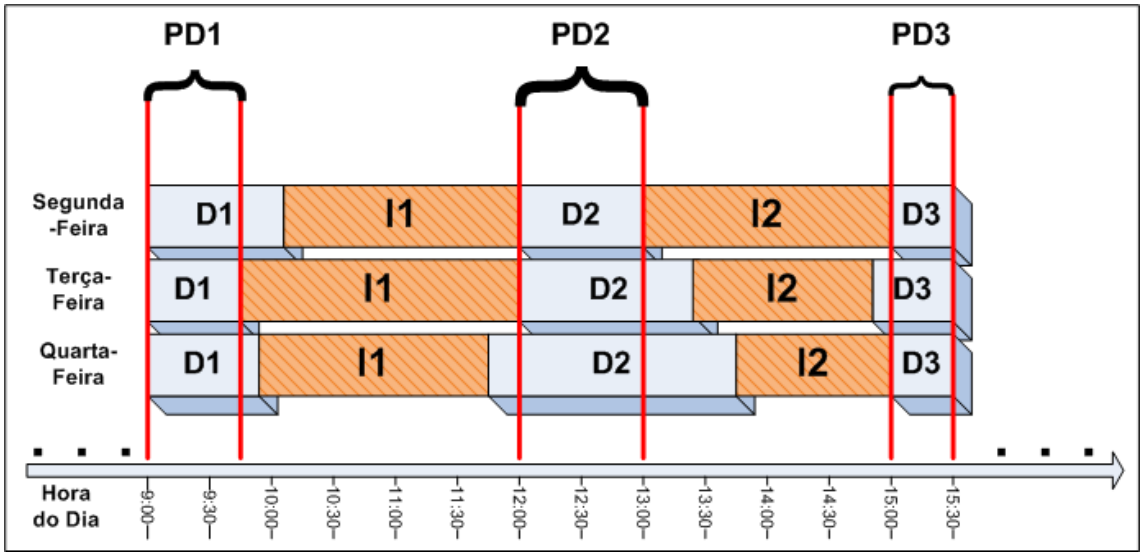

Figura 3.1: Padrão de disponibilidade de um recurso.

3.3 Analise Exploratória da Disponibilidade de Recursos

A previsão de disponibilidade é a principal motivação deste trabalho, como relatado no Capítulo 1.

A disponibilidade de um recurso participante de uma grade oportunista é um a informação importante para o escalonador destes tipos de sistema. Para que um escalonador possa realizar o escalonamento, informações sobre a disponibilidade dos recursos podem ser fornecidas com antecedência. O fornecimento antecipado desta informação requer a previsão de disponibilidade dos recursos.

Previsão é uma forma de analise de dados onde tem-se como objetivo, determinar o futuro através da identificação de tendências nestes dados, possibilitando assim a tomada de decisões inteligentes (HAN; KAMBER, 2006).

Assim, a previsão pode ser vista como um processo de construção e uso de um modelo que tem como entrada um conjunto de dados e disponibiliza como saída um valor que indica o comportamento futuro do atributo considerado. A estes dados de entrada, dá-se o nome de dados de treinamento (*Training Data Set*).

A previsão é utilizada em diversos segmentos como na liberação de crédito para o consumidor através da análise de riscos, na área médica para identificação de possíveis doenças para um determinado paciente, na previsão do tempo, no controle de estoque através da previsão de falta de produtos.

Neste trabalho, a previsão é de disponibilidade de recursos. Para a realização da previsão é requerido um modelo. A concepção deste tipo de modelo é frequentemente iniciada pela análise de dados já existentes, ou seja, pela análise de dados históricos, com o objetivo de identificar propriedades estatísticas existentes nestes dados. As propriedades identificadas podem servir de base do modelo usado na previsão.

Por este motivo, para a concepção do modelo de previsão de disponibilidade proposto neste trabalho optou-se por realizar a análise exploratória de dados de disponibilidade de recursos.

O objetivo de realizar esta análise é extrair informações importantes que venham à auxiliar na especificação de um novo modelo de previsão de disponibilidade. Segundo Morettin e Bussab (2010), quando analisamos dados, estamos buscando um padrão ou

forma de regularidade que possibilitem a identificação de um modelo que represente o ambiente estudado.

Para realizar esta análise, foram utilizados dados resultantes do monitoramento de recursos pertencentes a nove ambientes diferentes: Dbcc, PlanetLab, Overnet, Microsoft, DNS, Grid5000, NotreDame, WebSites e Seti@Home. Os dados coletados destes ambientes, exceto os do ambiente Dbcc, foram obtidos no repositório de traces ***Failure Trace Archive*** (KONDO et al., 2010).

O *Failure Trace Archive* (FTA) é um repositório que fornece dados sobre a disponibilidade de recursos pertencentes a sistemas distribuídos, além de ferramentas para análise destes dados. Este repositório é de acesso público e seus dados são fornecidos em formato padronizado (KONDO et al., 2010).

Outros trabalhos como, (JAVADI et al., 2009; KONDO; ANDRZEJAK; ANDERSON, 2008; MICKENS; NOBLE, 2006; IOSUP; OZAN SONMEZ; EPEMA, 2007), também utilizaram alguns dos conjuntos de dados utilizados neste trabalho, com o intuito de analisar a disponibilidade ou avaliar os modelos de previsão de disponibilidade propostos.

Para a realização da análise, os dados dos ambientes passaram por transformações. Foi gerado um arquivo para cada dia de monitoramento, para cada recurso monitorado. Sobre estes arquivos, aplicou-se um filtro para excluir os arquivos onde não ocorreram mais de 5 oscilações entre os estados de disponibilidade e indisponibilidade. Este filtro foi aplicado pelo fato de ter-se identificado casos em que os recursos permaneciam disponíveis ou indisponíveis ao longo de todo um dia de monitoramento. Esta situação não é a de interesse para esta análise, visto que o objetivo é identificar as características presentes em um ambiente com recursos altamente voláteis. Optou-se por 5 oscilações entre estados como volatilidade mínima, devido a pequena volatilidade apresentada na maioria dos ambientes. Ao aumentar o número de oscilações, o filtro acabava por gerar uma amostra de dados pouco significativa, podendo comprometer a análise realizada.

Para facilitar a realização da análise, considerou-se apenas o subintervalo compreendido entre às 08:00 e 18:00 horas.

A frequência dos intervalos de disponibilidade varia conforme o tempo de duração do intervalo. Em alguns trabalhos sobre a caracterização de disponibilidade de recursos de grades oportunistas, como Kondo et al. (2005; 2007; 2008; 2006), Rood e Lewis (2007), Ren e Eingeman (2006), Toth e Finkel (2007) e Javadi et al. (2009), foi identificado que a medida que aumenta a duração de um intervalo de disponibilidade, menor é a sua frequência. Chegaram a esta conclusão, analisando os dados de alguns dos ambientes utilizados no presente trabalho. No entanto, o ambiente Dbcc não foi objeto destes estudos. Por esta razão, decidiu-se analisar a distribuição de frequência dos intervalos de disponibilidade do ambiente Dbcc.

A figura 3.2 apresenta a distribuição cumulativa das durações de disponibilidade de todos os recursos monitorados no ambiente Dbcc. Observa-se que aproximadamente 50% dos intervalos de disponibilidade são menores ou iguais à 5 minutos. Também é perceptível que durações de disponibilidade de 30 minutos representam apenas 5% de todas as durações monitoradas.

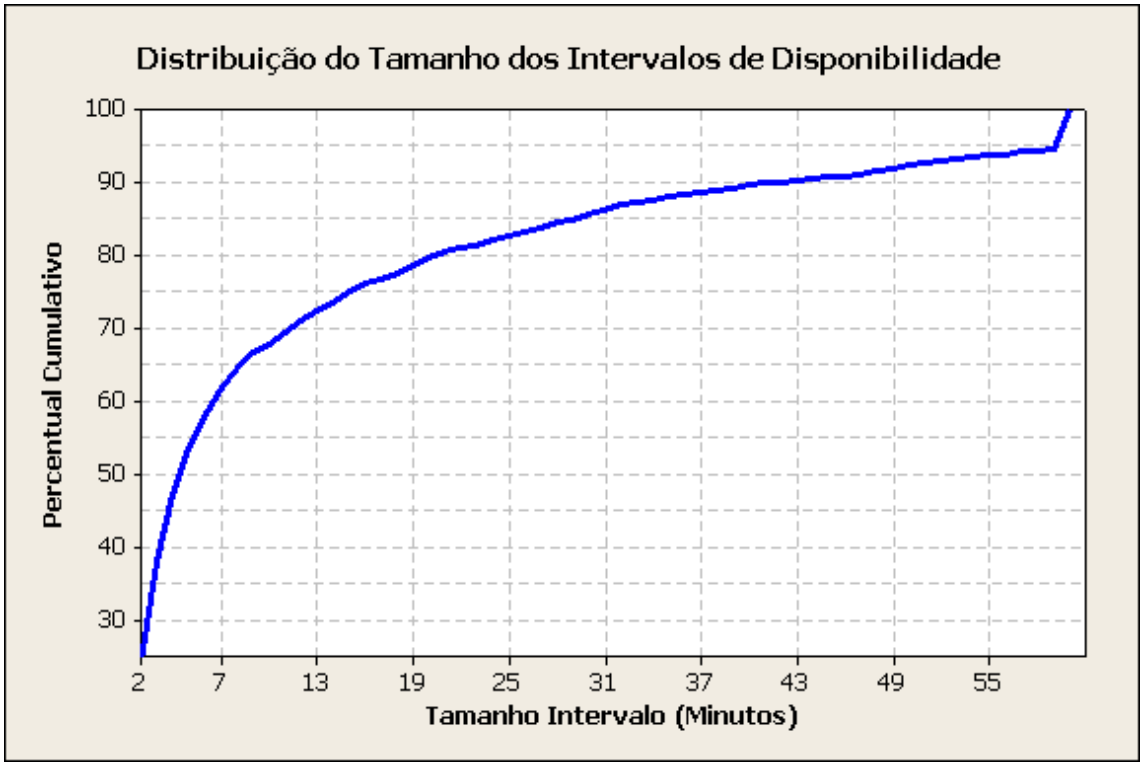

Figura 3.2: Distribuição acumulada da duração dos intervalos de disponibilidade do ambiente Dbcc.

Entende-se então que quanto maior for a duração de um intervalo de disponibilidade menor será a sua frequência, ou seja, intervalos com duração de disponibilidade menores são mais frequentes. Este resultado demonstra que o ambiente Dbcc está em concordância com os demais ambientes, no que diz respeito à frequência dos intervalos de disponibilidade.

Para o projeto de escalonadores de grades oportunistas, esta é uma informação de grande relevância. As tarefas existentes na fila de escalonamento normalmente necessitam de tempos de execução diferenciados. Como constatado por Kondo et al. (2005; 2007), quanto maior for o tempo necessário para executar uma tarefa, menor será a chance desta tarefa ser executada.

Neste caso, o escalonador poderia considerar o tempo necessário para executar uma tarefa e a duração dos intervalos de disponibilidade dos recursos [1].

Sabe-se, no entanto, que o tempo necessário para executar uma tarefa nem sempre é conhecido com antecedência pelo escalonador. Além disso, a existência de recursos heterogêneos dificulta ainda mais o fornecimento de tal informação. No entanto, pode-se utilizar uma aproximação, como o tempo médio obtido com a execução de tarefas pertencentes a uma determinada aplicação. Esta informação já é fornecida nos sistemas de Grade Oportunista *XtremWeb* e *BOINC*.

3.3.1 Análise de Disponibilidade do Ambiente Dbcc

O conjunto de dados do ambiente chamado Dbcc, contém dados de utilização de recursos pertencentes a uma escola e uma empresa.

[1] A duração de um intervalo de disponibilidade poderia ser fornecida por algum modelo de previsão de disponibilidade

Para realizar a coleta dos dados do ambiente Dbcc, foi implementado o componente **Coletor** que posteriormente foi incluído ao modelo proposto neste trabalho. Estes dados foram coletados como parte da pesquisa realizada neste trabalho.

Realizou-se o monitoramento de vinte e cinco recursos (25), por um período de dois (2) meses, onde cada recurso teve o componente **Coletor** instalado e configurado para registrar a utilização do recurso a cada minuto do dia. A cada minuto, o componente registra a utilização do recurso, persistindo a informação em arquivos.

Ocorrendo a utilização do recurso durante o minuto monitorado, o **Coletor** registra o recurso como indisponível durante aquele minuto. O estado indisponível é identificado pelo valor (1) persistido no arquivo. Não ocorrendo a utilização do recurso durante o período monitorado, o componente registra o recurso como disponível durante aquele minuto. O estado disponível é identificado pelo valor (0) persistido no arquivo.

A definição do estado de disponibilidade de um recurso é dada pela política de recrutamento empregada pelo recurso, como apresentado anteriormente na Seção 2. Assim, empregou-se a política de recrutamento *Uso de Teclado/Mouse*, também apresentada anteriormente na Seção 2.

Com a adoção da política de recrutamento *Uso de Teclado/Mouse*, o componente **Coletor** registrou o estado (1) quando durante o minuto monitorado, o usuário utilizou o mouse ou o teclado do recurso, e caso contrário, registrava (0).

Para evitar perda de informações devido a alteração comportamental dos usuários dos recursos, os mesmos não foram informados sobre a realização do monitoramento no recurso. Assim, acredita-se ter capturado o comportamento real do usuário ao utilizar seu recurso à cada dia.

3.3.1.1 Intervalos de Disponibilidade e o Horário de Trabalho

Como o horário de trabalho da escola e empresa monitorados no conjunto de dados Dbcc era conhecido, verificou-se a correspondência entre os intervalos de disponibilidade apresentados pelos recursos e o horário de trabalho destas instituições. O horário de funcionamento dos demais ambientes não era conhecido e por esta razão, a análise foi realizada apenas sobre os dados do ambiente Dbcc.

Tanto a escola quanto a empresa possuem horário de funcionamento bem definido. A tabela 3.2 apresenta o horário de funcionamento da empresa e escola.

Chama-se a atenção para os períodos de intervalo apresentados nesta tabela. Por serem intervalos das instituições, os usuários deveriam deixar os recursos, ou seja, nos intervalos especificados para cada instituição os recursos deveriam apresentar intervalos de disponibilidade correspondentes aos intervalos de funcionamento. Assim, os intervalos das instituições poderiam ser geradores de **Padrões de Disponibilidade (PD)** nos dados dos recursos monitorados.

Para verificar a correspondência entre os intervalos de funcionamento e os intervalos de disponibilidade apresentados pelos recursos, foram gerados os gráficos apresentados nas figuras 3.3 e 3.4.

Tabela 3.2: Horário de funcionamento da empresa e escola

Início	**Fim**	***Descrição do Período***
Empresa		
08:30	12:00	Expediente
12:00	13:00	Intervalo
13:00	17:30	Expediente
Escola		
08:00	09:40	Expediente
09:40	10:00	Intervalo
10:00	11:40	Expediente
11:40	14:00	Intervalo
14:00	15:40	Expediente
15:40	16:00	Intervalo
16:00	17:40	Expediente
17:40	19:00	Intervalo

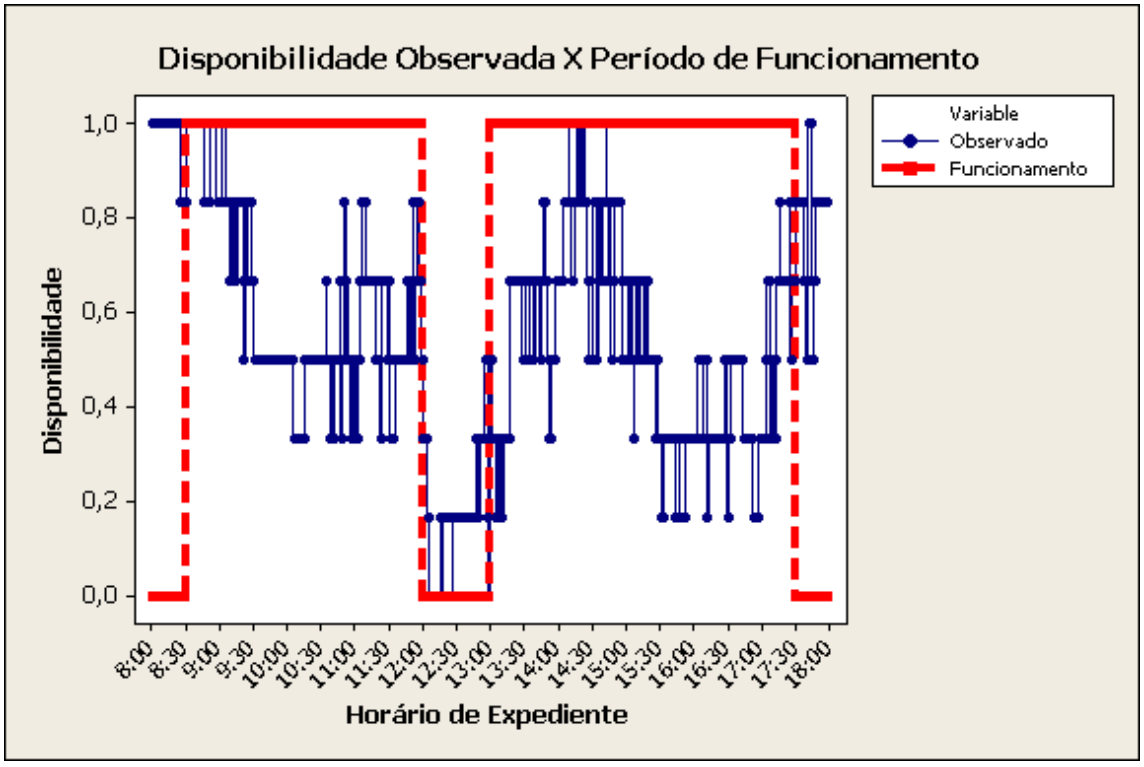

Figura 3.3: Período de funcionamento X disponibilidade observada (Empresa)

A linha chamada **Funcionamento** representa como deveria ser a disponibilidade do recurso, conforme a tabela de funcionamento da instituição em que este recurso se encontra (escola ou empresa). A linha chamada **Observado** corresponde à disponibilidade observada nos dados coletados. O valor de disponibilidade referente ao observado corresponde a média de dias em que os recursos estiveram disponíveis naquele horário.

Conforme apresentado na tabela 3.2, entre às 8:30 e 12:00 os recursos deveriam estar indisponíveis. Logo após, no intervalo entre às 12:00 e 13:00 estes recursos estariam disponíveis. De fato, a disponibilidade observada apresentou contornos que se assemelham

aos dados da tabela. Analisando o gráfico da figura 3.3 percebe-se que coletivamente, existe um comportamento similar entre o que foi observado e o funcionamento da empresa.

No entanto, é importante perceber que o comportamento observado não foi fiel ao horário de funcionamento especificado. No intervalo compreendido entre às 9:30 e 10:30, os recursos permaneceram disponíveis em, pelo menos, 50% dos dias monitorados, enquanto segundo o horário de funcionamento da instituição, deveriam estar indisponíveis. Já no intervalo entre às 12:00 e 13:00, onde os recursos deveriam estar sempre disponíveis, observa-se que em aproximadamente 20% dos dias monitorados houve utilização dos recursos nesse horário.

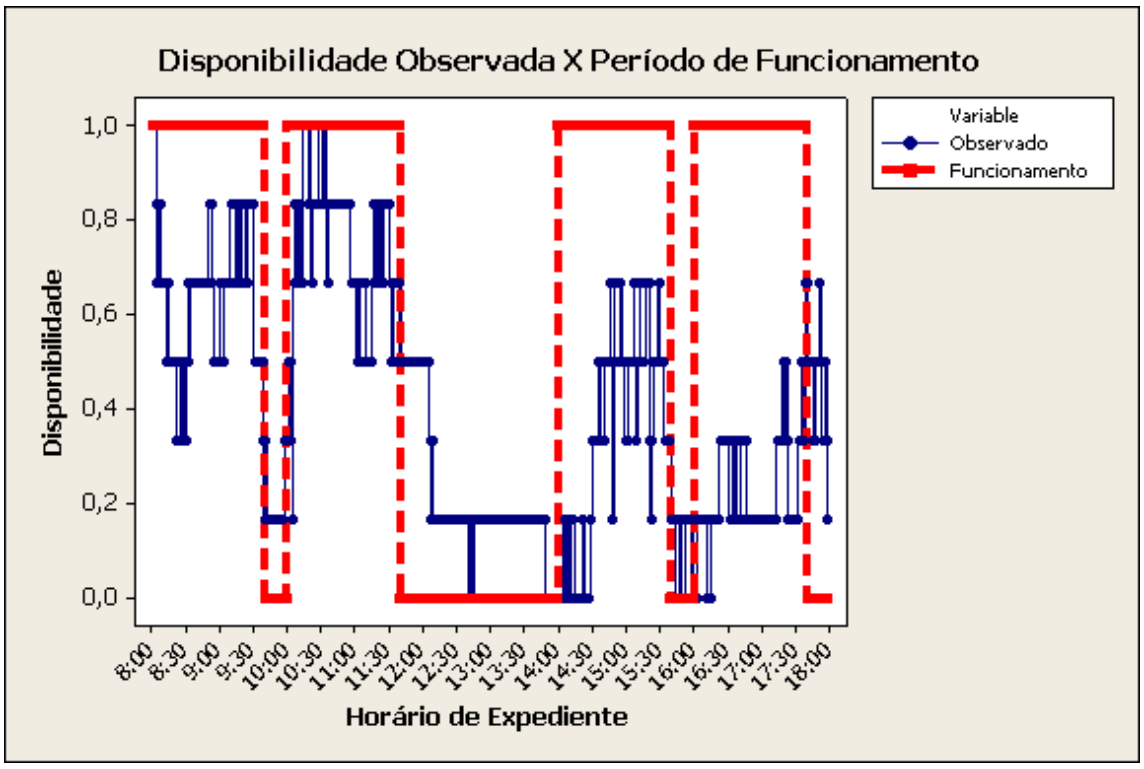

Figura 3.4: Período de funcionamento X disponibilidade observada (Escola)

Resultado similar foi obtido com os recursos da escola. Como apresentado no gráfico da figura 3.4, a tendência a comportar-se como definido pelo horário de funcionamento é visível. No entanto, também é possível observar que no intervalo entre às 12:00 e 14:00 os recursos permaneceram em uso em aproximadamente 20% dos dias monitorados.

Estes gráficos demonstram também que são poucos os casos em que os recursos realmente permaneceram disponíveis ou indisponíveis conforme o horário de funcionamento especificado pela instituição.

A análise destes resultados traz três informações importantes:

- Realmente existem padrões de disponibilidade nos dados monitorados. Estes padrões são similares ao definidos pelo funcionamento do ambiente porém, não iguais.

- Existem intervalos de disponibilidade e indisponibilidade em horários que não condizem com o horário de funcionamento especificado pela instituição, ou seja, não se deve assumir a não existência de intervalos de disponibilidade no recurso durante

o horário de expediente da instituição, assim como não se deve assumir que não existe indisponibilidade em horários de intervalo.

- Um recurso tende a apresentar um comportamento de disponibilidade em conformidade ao ambiente em que está inserido. Isso mostra que a disponibilidade dos recursos pode variar de ambiente para ambiente, dificultando a definição de um único modelo de disponibilidade para todos os ambientes.

Se os recursos apresentassem o comportamento de disponibilidade fiel à instituição em que se encontram, a previsão de disponibilidade não seria necessária. Bastaria apenas considerar o horário de funcionamento dos ambientes onde os recursos estão inseridos. No entanto, conforme os resultados obtidos, não se pode assumir que o recurso sempre se comportará de forma fiel ao período de funcionamento do ambiente. Isso reforça a necessidade de existir a previsão de disponibilidade dos recursos, quando se pretende utilizar um recurso de forma compartilhada, como ocorre em grades oportunistas.

Ainda sobre os dados do ambiente Dbcc, a figura 3.5 apresenta a aproximação do horário de almoço de um recurso da escola. Neste figura são apresentados 6 dias de dados de disponibilidade coletados deste recurso, entre o intervalo dás 11:30 e 12:35 horas. Foi utilizado apenas este intervalo, com o objetivo de apresentar o comportamento de um recurso quando aproxima-se de um horário que tende a ser um padrão de disponibilidade.

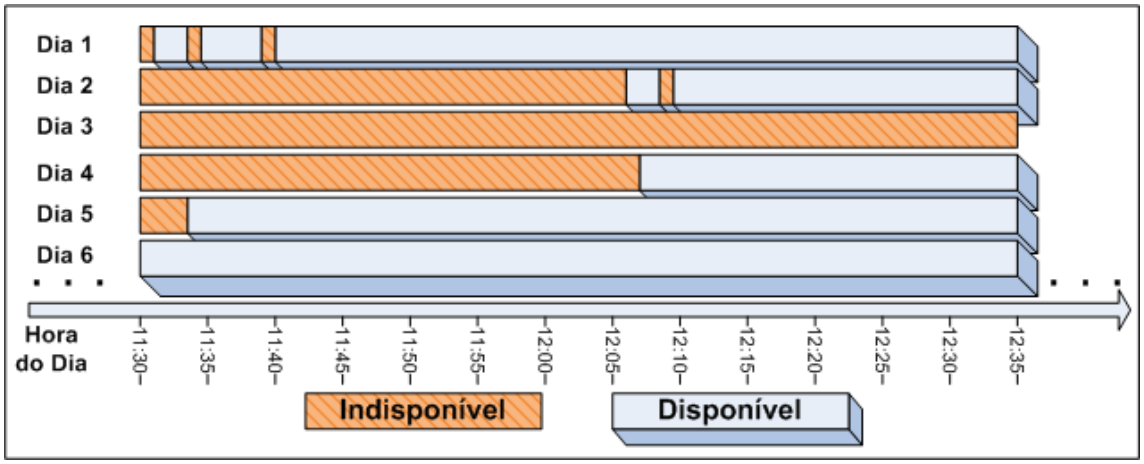

Figura 3.5: Disponibilidade de um recurso da escola, aproximando-se de um intervalo de disponibilidade

Nos dados do dia 6, o recurso apresentou disponibilidade durante todo o período apresentado na figura. No entanto, no dia 3 ocorreu justamente o contrário, ficando o recurso indisponível durante todo o período apresentado. Observa-se também que nos dias 1, 5 e 6 os intervalos de disponibilidade ocorridos são semelhantes entre si. O mesmo ocorre nos dias 2 e 4. A exceção ficou no dia 3 que não apresentou-se semelhante aos demais dias.

A análise desta figura traz outras duas informações importantes:

- Existe um padrão de utilização do recurso e deste padrão se pode extrair Padrões de Disponibilidade (PD). No caso da figura 3.5, existe um padrão PD entre às 12:10 e 12:35 horas.

- Realizar previsões baseando-se exclusivamente nos padrões PD encontrados geraria erros de previsão por perda de períodos de disponibilidade. Um exemplo seria se no dia 5 da figura 3.5 fosse realizada a previsão de disponibilidade do recurso às 11:34 horas (quando o recurso tornou-se disponível). Conforme o padrão de disponibilidade PD do recurso para este horário, o recurso estaria disponível somente entre às 12:10 e 12:35 horas, ou seja, o recurso seria considerado indisponível dás 11:34 às 12:09 horas enquanto na verdade, o mesmo esteve disponível durante todo esse intervalo.

3.3.1.2 Frequência de Padrões de Disponibilidade

Decidiu-se prosseguir a análise sobre os padrões de disponibilidade e verificar a ocorrência destes padrões em todos os ambientes citados anteriormente. O padrão procurado refere-se à ocorrência de intervalos de disponibilidade, nos diferentes arquivos de dados de recursos.

Para cada ambiente, recurso e arquivo de dados, percorreu-se os dados no intervalo compreendido entre às 08:00 e 18:00 horas, em busca de intervalos de disponibilidade com duração maior ou igual a cinco minutos [2].

Para cada intervalo de disponibilidade encontrado, chamado de **Intervalo de Referência (IR)**, verificou-se nos demais arquivos do mesmo recurso, a existência de intervalos de disponibilidade no mesmo horário do intervalo IR.

Foram definidas 4 medidas diferentes com o objetivo de quantificar a ocorrência média dos intervalos de disponibilidade existentes nos arquivos de um recurso, dado um intervalo de referência.

A primeira medida, chamada de **freqD**, quantifica a frequência de intervalos de disponibilidade com a mesma duração do IR. Esta medida é conservadora, pois busca por intervalos de disponibilidade exatamente iguais a IR.

Para facilitar o entendimento das medidas utilizadas, a figura 3.6 apresenta as situações em que cada medida seria aplicada, dado um intervalo de referência IR.

Na figura, IR tem duração de 9 minutos, tendo como horário de início e fim, 13:05 e 13:13 horas respectivamente. São ilustrados 8 arquivos, representados por retângulos preenchidos com 0 (disponível) e 1 (indisponível).

Na figura 3.6, a medida **freqD** indica que intervalos de disponibilidade de mesma duração que IR (Dur=9) ocorreram em 37.50% dos arquivos de um recurso. Foi inserido um retângulo interno à cada retângulo de arquivo que apresentou o intervalo de disponibilidade compatível com a medida. A frequência foi determinada com a média aritmética de ocorrências dos intervalos encontrados.

Além desta medida conservadora, investigou-se a ocorrência de intervalos de disponibilidade com medidas mais relaxadas, com o objetivo de identificar aproximações do intervalo de referência.

A medida **freqContinua(95)** considera os intervalos de disponibilidade com duração maior ou igual a 95% da duração do intervalo IR. Esta medida foi chamada de **freqContinua(95)**. Na figura 3.6, a medida **freqContinua(95)** indica a frequência de 50% de intervalos com duração igual ou maior à 95% da duração de IR. O arquivo 8 apresentado na figura não foi selecionado por não apresentar duração contínua correspondente à 95% da duração de IR. Como houve indisponibilidade no intervalo, a disponibilidade contínua foi de apenas 4 minutos.

[2] Utilizou-se a restrição de 5 minutos com o objetivo de descartar pequenos intervalos de disponibilidade, não importantes para a análise realizada.

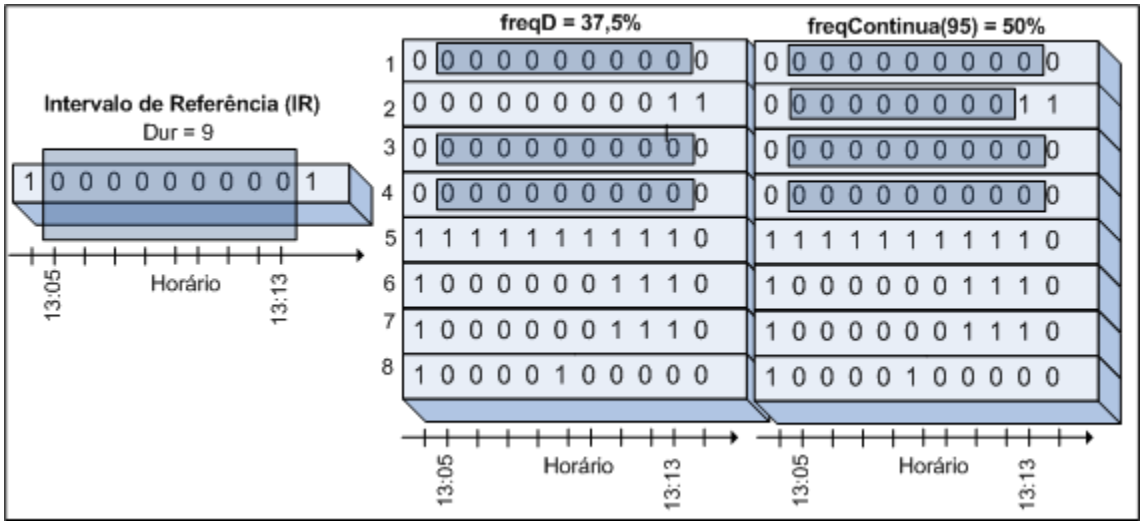

Figura 3.6: Representação das medidas freqD e freqContinua(95)

Outra medida adotada foi chamada de **freqNãoContinua(95)** e representa a frequência média de intervalos de disponibilidade, cujo somatório das ocorrências de disponibilidade ocorridas entre os horários de início e fim da sequencia analisada chega a 95% da duração de IR. Esta medida desconsidera o que poderia ser uma interferência na sequência, ou seja, desconsidera a ocorrência de indisponibilidade em um intervalo analisado, desde que esta ocorrência não represente mais do que 5% do tempo de duração de IR. Na figura 3.7, esta medida indica a frequência de 62.50% de intervalos de disponibilidade apresentando duração de disponibilidade não contínua igual ou maior à 95% da duração de IR. A situação de interferência pode ser verificada no arquivo 8 da figura onde, no intervalo analisado ocorre a indisponibilidade (1) no arquivo porém, o somatório dos minutos de disponibilidade ocorridos atinge 95% da duração apresentada no IR.

A última medida considerada é a **freqOcorrência**. Esta medida quantifica a ocorrência de intervalos de disponibilidade ocorridos no intervalo analisado, desde que este intervalo ultrapasse 5 minutos [3]. Na figura 3.7, esta medida indica a frequência de 87.50% de intervalos de disponibilidade com duração maior ou igual a 5 minutos nos arquivos analisados.

[3]Utilizou-se 5 minutos para manter o mesmo critério de disponibilidade mínima considerada para a obtenção de IR.

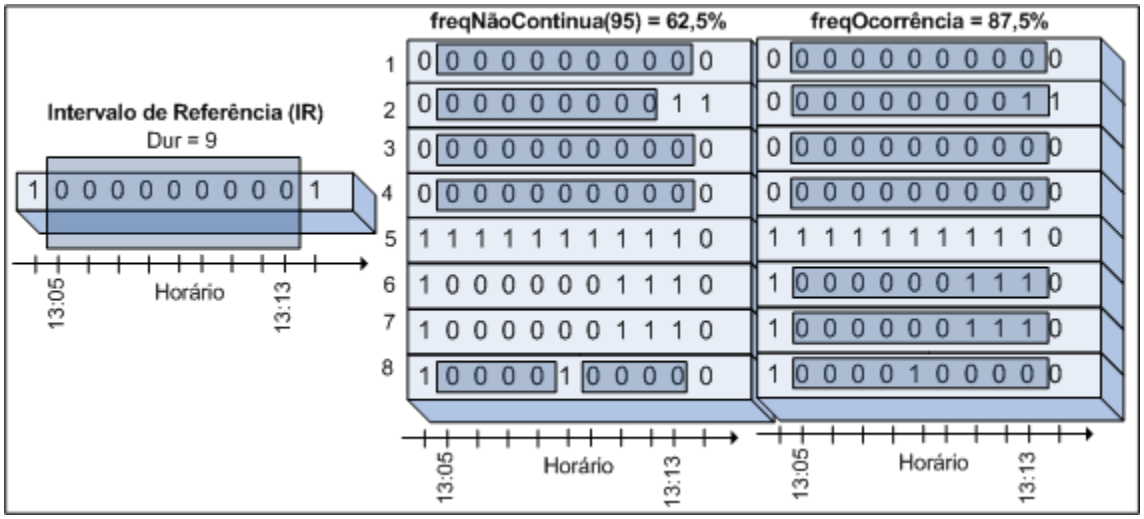

Figura 3.7: Representação das medidas freqNãoContinua(95) e freqOcorrência

Os dados apresentados nas figuras 3.6 e 3.7 são apenas ilustrativos. Aplicaram-se estas medidas sobre os dados de cada recurso dos ambientes ora apresentados, para verificar a existência de padrões de disponibilidade.

A tabela 3.3 apresenta os resultados destas medidas aplicadas a estes dados. A tabela sumariza os dados na linha **Total**. A frequência média de intervalos de disponibilidade de mesma duração foi de 33.01% (**freqD**), ou seja, dado um intervalo de disponibilidade qualquer, ele se repete em 33.01% dos aquivos verificados. Além de ser uma média um tanto baixa para estabelecer um padrão de disponibilidade, observando-se o desvio padrão e o coeficiente de variabilidade apresentados entende-se que os dados formadores desta média apresentam uma grande variabilidade. De fato, a amplitude existente nos dados foi de 36.81 pontos percentuais.

Observando os valores totais das demais medidas é possível perceber um aumento na frequência de intervalos encontrados.

Este resultado complementa a análise da figura 3.5, anteriormente realizada. Considerar um padrão de disponibilidade rígido faz com que períodos de disponibilidade sejam perdidos.

Ao usar o padrão rígido, foi obtida a média de 33.01%. No entanto, ao relaxar este padrão e adotar uma medida que exige a disponibilidade contínua de 95% do intervalo de referência (**freqContinua(95)**), a média foi elevada a 56.03% dos casos.

Ao considerar a medida **freqNãoContinua(95)** esta média foi maior, chegando a 72.58% dos casos. Outro fato importante a se observar está na homogeneidade dos resultados obtidos. Ao utilizar uma medida mais relaxada, as médias obtidas para cada ambiente tornaram-se mais próximas e estáveis. Isso pode ser percebido na redução do coeficiente de variação de cada medida.

Analisando os resultados obtidos para cada ambiente, nota-se que o coeficiente de variação de cada ambiente é diferente e principalmente que em muitos casos permanece acima de 25% [4].

Como a estatística apresentada para cada ambiente é formada pelos resultados obtidos para cada recurso, pode-se deduzir que a frequência média de intervalos de disponibilidade encontrados, varia muito de recurso para recurso em um ambiente e também que

[4]Dados que apresentam um coeficiente de variação de até 25%, são considerados estáveis ou homogêneos.

esta variação é diferente conforme o ambiente em que o recurso está localizado.

Tabela 3.3: Resultado das medidas de frequência de disponibilidade

	freqD			freqContinua(95)		
Ambiente	**M(%)**	**Dp**	**CV**	**M(%)**	**Dp**	**CV**
Dbcc	19.62	4.03	20.55	39.1	5.86	14.98
DNS	25.11	11.44	45.55	48.45	20.77	42.88
Grid5000	48.12	13.46	27.97	72.14	15.47	21.45
Microsoft	37.89	11.36	29.99	59.03	16.00	27.11
Notre Dame	47.96	5.37	11.20	83.14	10.53	12.66
Overnet	16.07	7.32	45.56	42.07	14.73	35.01
PlanetLab	30.90	15.99	51.74	45.22	25.25	55.84
Seti@Home	18.56	14.77	79.59	31.38	22.16	70.61
Websites	52.88	14.10	26.66	83.77	20.06	23.94
Total	**33.01**	**14.20**	**43.00**	**56.03**	**19.48**	**34.77**
	freqNãoContinua(95)			freqOcorrência		
Dbcc	67.26	8.04	11.96	54.55	8.01	14.69
DNS	71.81	19.27	26.84	72.11	19.23	26.67
Grid5000	87.90	10.56	12.02	87.62	9.91	11.31
Microsoft	75.89	14.42	19.01	75.89	14.42	19.01
Notre Dame	93.12	6.90	7.41	92.73	7.09	7.65
Overnet	60.67	17.42	28.71	58.89	17.52	29.75
PlanetLab	57.51	27.57	47.94	59.08	27.24	46.10
Seti@Home	46.40	25.16	54.23	40.89	23.62	57.77
Websites	92.68	20.81	22.45	92.01	23.43	23.43
Total	**72.58**	**16.41**	**22.61**	**70.42**	**18.30**	**25.98**

M = Média aritmética.
Dp = Desvio padrão.
CV = Coeficiente de variação.

A análise destes resultados traz mais algumas informações importantes:

- A média de ocorrências de um padrão de disponibilidade rígido, representado por **freqD**, reforça que utilizar apenas o padrão de disponibilidade para a realização de previsões pode levar a altas taxas de erros, visto que um intervalo de disponibilidade igual ao de referência ocorreu em somente 33.01% dos casos.

- um modelo de previsão de disponibilidade considerando estatísticas coletivas [5] pode não ser a melhor alternativa, devido a diferença de comportamento na utilização de recursos de ambientes distintos, ou até mesmo, conforme o perfil do usuário utilizador do recurso [6].

[5] Estatísticas entre todos os ambientes ou entre todos os recurso de um mesmo ambiente, por exemplo.

[6] Informação deduzida dos altos índices apresentados pelo coeficiente de variação de cada ambiente, apresentados na tabela 3.3.

3.3.1.3 Perfil de Utilização dos Recursos

Para verificar a existência de diferenças no perfil de utilização dos recursos, foram selecionados 2 recursos de cada ambiente.

Para cada recurso foi obtida a duração dos intervalos de disponibilidade maiores que 5 minutos, por hora do dia. Para tanto, percorreu-se cada arquivo de disponibilidade de cada recurso, no intervalo entre às 8:00 e 18:00 horas. Para cada hora deste intervalo, foi calculada a duração de disponibilidade média dos intervalos de disponibilidade que iniciaram naquela hora.

A figura 3.8 apresenta o gráfico de disponibilidade média por hora do dia dos recursos Dbcc_Nodo_1 e Dbcc_Nodo_2 do ambiente Dbcc. É possível perceber que mesmo fazendo parte de um mesmo ambiente, estes recursos apresentam médias de disponibilidade diferentes e em horários diferentes.

O recurso Dbcc_Nodo_1 mostra uma tendência em apresentar intervalos de disponibilidade com maior duração de disponibilidade, nos intervalos que têm início às 11:00 horas. Estes intervalos apresentaram, em média, duração de 68.84 minutos.

Já o recurso Dbcc_Nodo_2 apresenta esta tendência nos intervalos que têm início às 12:00 horas e às 15:00 horas, intervalos estes com duração média de *39.00* e *37.88* minutos, respectivamente.

Além dessa constatação, é possível perceber maior variabilidade das durações de disponibilidade encontradas no recurso Dbcc_Nodo_1. O intervalo de confiança (IC) do recurso Dbcc_Nodo_1 em relação a média de duração de disponibilidade é em geral maior que o IC obtido para os intervalos existentes no recurso Dbcc_Nodo_2. Em outras palavras, as médias de duração de disponibilidade do recurso Dbcc_Nodo_2 foram obtidas de dados menos variáveis do que as médias obtidas para o recurso Dbcc_Nodo_1.

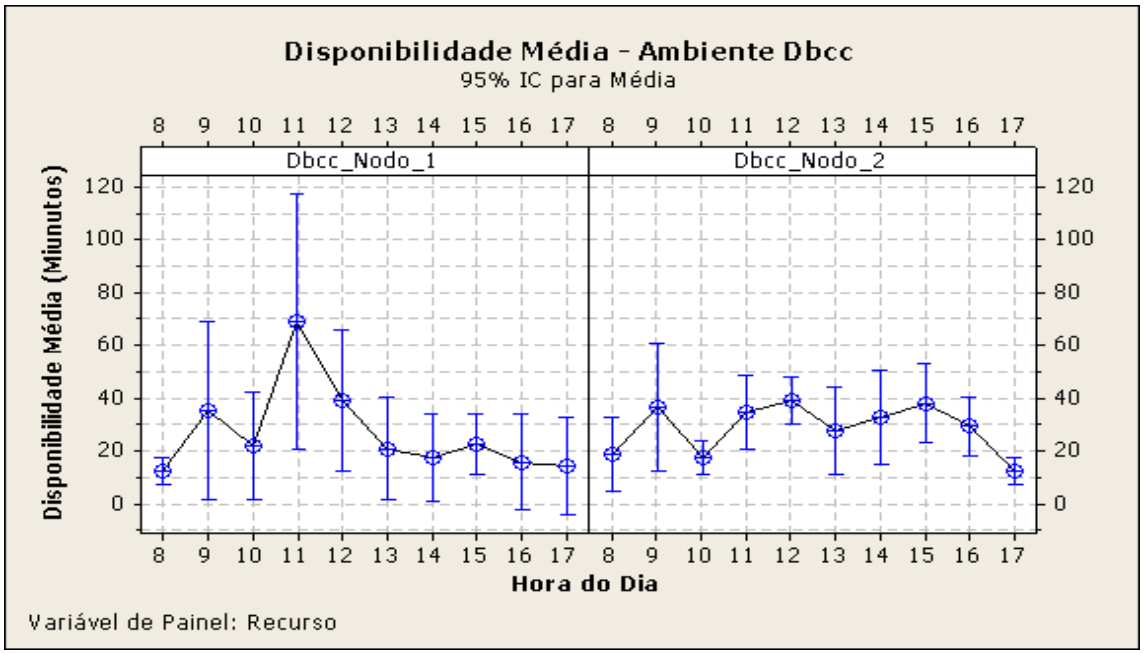

Figura 3.8: Duração de disponibilidade média por hora do dia para o ambiente Dbcc

A figura 3.9 apresenta o gráfico de disponibilidade média por hora do dia dos recursos DNS_Nodo_1 e DNS_Nodo_2 do ambiente DNS.

O recurso DNS_Nodo_1 apresentou alta variabilidade nas durações de disponibilidade encontradas, o que pode ser verificado observando-se o intervalo de confiança apresen-

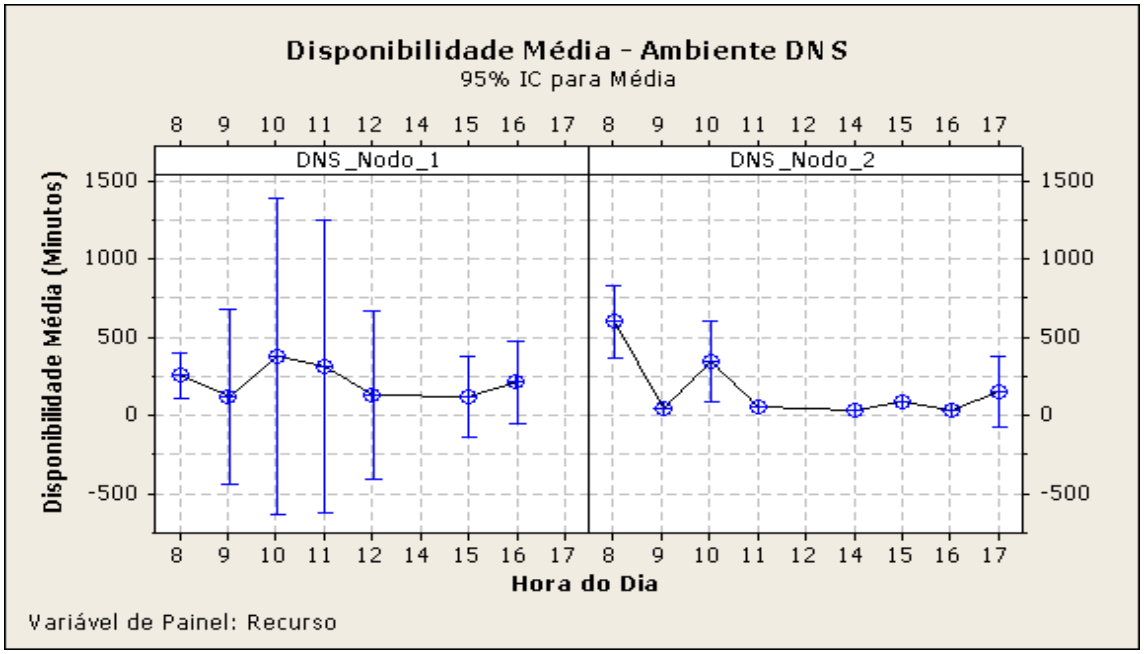

Figura 3.9: Duração de disponibilidade média por hora do dia para o ambiente DNS

tado no gráfico. O recurso DNS_Nodo_2 por sua vez, apresentou-se mais estável com pequenos intervalos de confiança.

No entanto, as maiores médias de disponibilidade foram encontradas no recurso DNS_Nodo_1, onde foi obteve-se as médias de *255.30*, *377.00* e *311.33* minutos nos intervalos iniciados às 8:00, 10:00 e 11:00 horas.

Observa-se também que em alguns horários do dia os recursos nunca apresentaram durações de disponibilidade maiores que 5 minutos, como ocorre com o recurso DNS_Nodo_1 às 14:00 horas e com o recurso DNS_Nodo_2 às 12:00 horas.

A figura 3.10 apresenta o gráfico de disponibilidade média por hora do dia dos recursos Grid5000_Nodo_1 e Grid5000_Nodo_2 do ambiente Grid5000. Neste ambiente, tanto o recurso Grid5000_Nodo_1 quanto o recurso Grid5000_Nodo_2 apresentam alta variabilidade na duração média dos intervalos de disponibilidade ocorridos.

No recurso Grid5000_Nodo_1 é possível perceber um pico de duração média para os intervalos iniciando às 8:00 e 14:00 horas, intervalos estes com as respectivas médias de *283.36* e *261.50* minutos. Já no recurso Grid5000_Nodo_2 identifica-se a tendência a aumentar a duração do intervalo de disponibilidade, partindo das 10:00 horas (167 minutos), até às 14:00 horas (457 minutos).

Nos gráficos das figuras 3.12, 3.9, 3.13, 3.14, 3.15 e 3.16, as situações de alta variabilidade e tendências de disponibilidade também podem ser observadas.

No entanto, alguns recursos apresentam características consideravelmente distintas dos demais. Os recursos NotreDame_Nodo_2, Overnet_Nodo_1, PlanetLab_Nodo_1, Seti@Home_Nodo_1 e WebSites_Nodo_1 apresentaram certa homogeneidade na média de duração de disponibilidade, como pode ser observado nos pequenos intervalos de confiança gerados. Por outro lado, observando o intervalo de confiança dos recursos Overnet_Nodo_2 e NotreDame_Nodo_1, percebe-se altíssima irregularidade quando comparados com os demais.

A análise destes gráficos demonstra a heterogeneidade comportamental existente entre recursos de ambientes distintos, e até mesmo entre recursos de um mesmo ambiente.

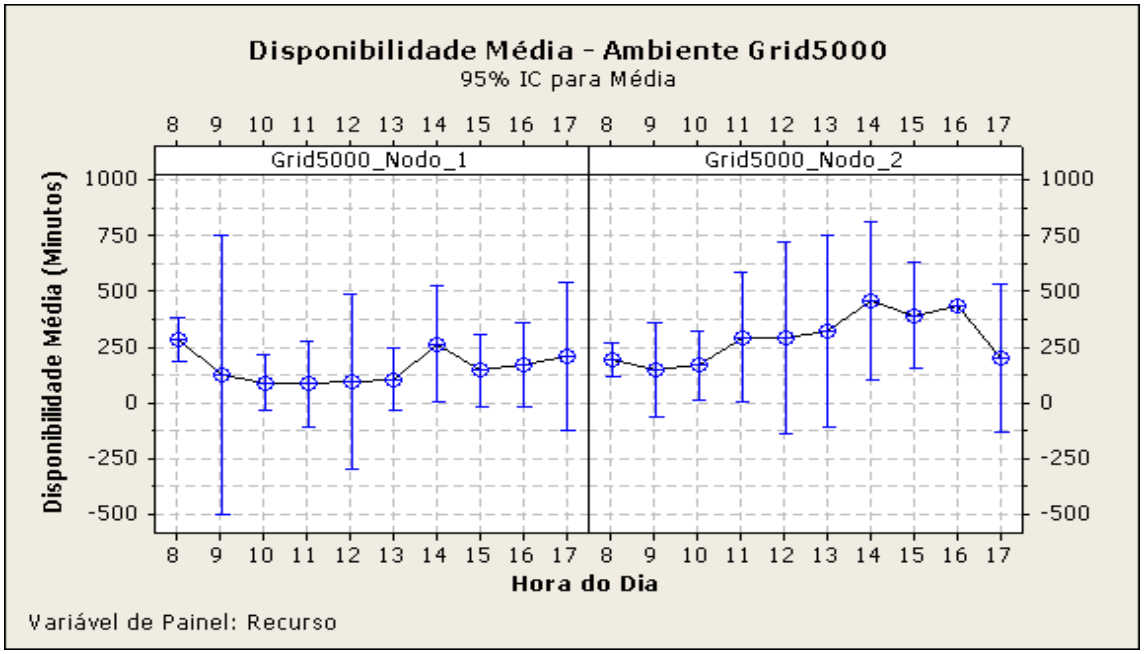

Figura 3.10: Duração de disponibilidade média por hora do dia para o ambiente Grid5000

3.3.1.4 A Recorrência de um Intervalo de Disponibilidade

A definição de um modelo de previsão de disponibilidade pode basear-se em diversas técnicas da probabilidade e estatística. Estas técnicas identificam padrões comportamentais nos dados. A análise de séries temporais por exemplo, apoia-se na existência de padrões e tendências existentes nos dados (MORETTIN; TOLOI, 2006). Frequentemente, emprega-se a estatística para identificar um modelo, uma distribuição, que seja capaz de descrever os dados e possibilitar a previsão do comportamento futuro com base no modelo identificado (MORETTIN; BUSSAB, 2010).

No entanto, os resultados desta análise mostram que a utilização de um modelo representando todos os recursos pode gerar erros de previsão devido as características diferenciadas dos ambientes onde estes recursos estão localizados ou até mesmo pelas características particulares de cada recurso.

Conclusão semelhante também foi destacada por Naaden et al. (2008a), onde defendeu-se que a definição do modelo de disponibilidade, bem como a previsão baseada neste modelo, apresentava melhores resultados quando o modelo baseava-se na classe ao qual o recurso pertence. Foi identificada a existência de diferentes classes de disponibilidade para recursos situados em ambientes distintos (NADEEM et al., 2008a). Para fundamentar esta conclusão, os autores realizaram avaliações de compatibilidade entre os principais modelos clássicos de distribuição de probabilidade e os dados de utilização dos recursos. Por fim, os autores concluíram que a existência desta diversidade de classes de comportamento, tornava inapropriada a utilização de um único modelo probabilístico para a realização de previsões de disponibilidade e que por serem estáticos, estes modelos de distribuição não seriam eficientes para realizar previsões adequadas ao comportamento dinâmico dos recursos.

Rahman, Hassan e Buyya (2010) chegaram a conclusão similar. Os autores analisaram a disponibilidade de um conjunto de recursos e afirmaram que os recursos pertencentes a ambientes como os de grades oportunistas, mudam o seu comportamento continuamente, dificultando a identificação da distribuição existente nos dados destes recursos e conse-

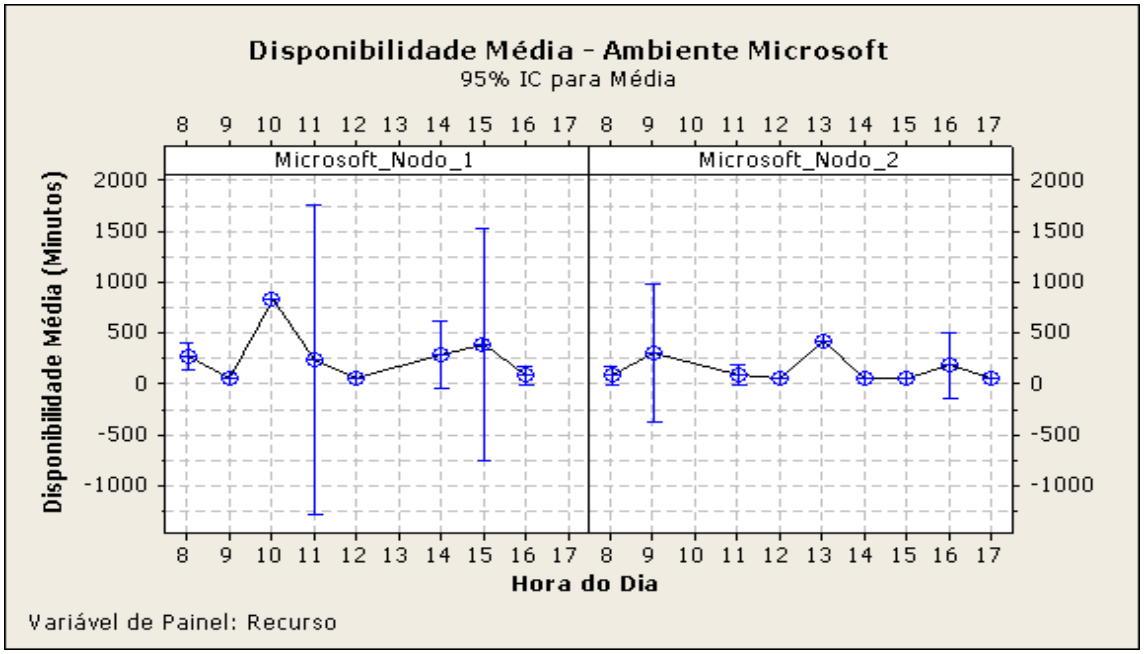

Figura 3.11: Duração de disponibilidade média por hora do dia para o ambiente Microsoft

quentemente, a definição de um modelo probabilístico.

Camenisch et al. (1999) optaram pela utilização de um modelo teórico de distribuição. Este modelo seria empregado em cada recurso, individualmente e não coletivamente, onde os parâmetros deste modelo seriam refinados ao longo do tempo pelo próprio recurso.

A análise de disponibilidade realizada neste trabalho mediu a frequência média de intervalos de disponibilidade (**freqD**), dado um intervalo de referência (IR), com o objetivo de identificar a existência de padrões de disponibilidade nos dados.

O resultado desta medida torna explícita a dificuldade de definição de um padrão, visto que um intervalo de referência (IR) ocorre em apenas *33.01%* dos arquivos de cada recurso dos ambientes analisados.

No entanto, uma informação não tão explícita também foi observada: **A chance de um Intervalo de Referência ter ocorrido *pelo menos uma vez* em outro dia de utilização do recurso é consideravelmente elevada**, como apresenta a tabela 3.4.

A tabela 3.4 apresenta a média e o coeficiente de variação obtidos. Para obter estes números, verificou-se se um Intervalo de Referência encontrado teria ocorrido pelo menos uma (1) vez em outro dia no mesmo recurso.

Esta informação é muito importante pois permite elaborar modelos de previsão que não somente baseados em padrões.

As informações obtidas nesta análise, em conjunto com as conclusões obtidas nos outros trabalhos citados acima, identificou-se que o modelo de previsão de disponibilidade adequado para uma Grade Oportunista formada por recursos com características de utilização semelhantes às apresentadas nesta seção, deveria estar de acordo com os seguintes critérios:

- **Individual** - para identificar as particularidades de cada recurso, a previsão deve ser realizada individualmente e não coletivamente;

- **Dinâmico** - para identificar as diferenças de utilização do recurso, a previsão deve ser dinâmica. Deve ser capaz de realizar previsões conforme a hora do dia, por

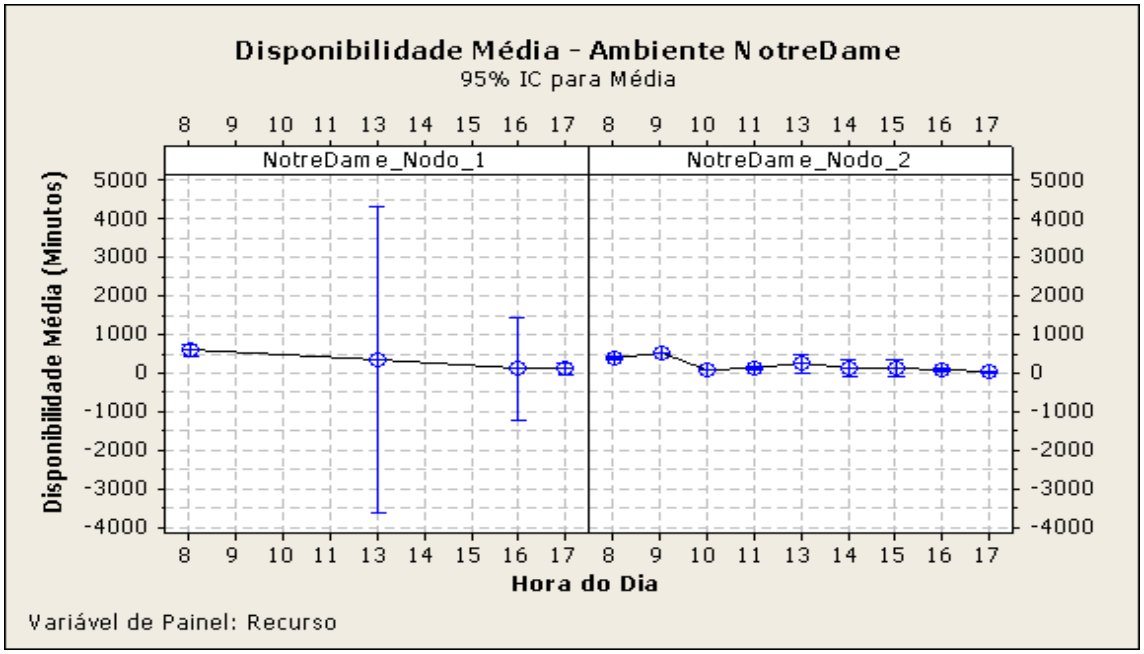

Figura 3.12: Duração de disponibilidade média por hora do dia para o ambiente Notre-Dame

exemplo;

- **Capaz de prever sem a existência de padrões** - a previsão de disponibilidade não deve ser baseada unicamente na existência de padrões de utilização de um recurso;
- **Capaz de prever exceções** - a previsão deve ser capaz de identificar períodos de disponibilidade atípicos e não somente os intervalos de disponibilidade que ocorrem com mais frequência..

Estes critérios foram utilizados na avaliação dos trabalhos relacionados.

3.4 Trabalhos Relacionados

Esta seção tem como objetivo, apresentar os trabalhos relacionados ao escopo de pesquisa desta dissertação. Nestes trabalhos, diferentes definições para disponibilidade foram apresentadas e consequentemente, diferentes formas de previsão foram modeladas.

3.4.1 A Previsão de Disponibilidade de Begole, Tang e Hill

No artigo de Begole, Tang e Hill (2003), foi proposto um algoritmo para a identificação de padrões temporais no comportamento de usuários de computadores, para prever a presença de um usuário em seu recurso computacional.

Analisando dados de comportamento de usuários, os autores identificaram que os usuários costumam chegar e sair regularmente no mesmo horário. Identificaram também que os padrões comportamentais destes usuários podem mudar conforme o dia da semana e a localização em que estes usuários encontram-se (BEGOLE et al., 2002; BEGOLE; TANG; HILL, 2003).

O padrão de presença de um usuário foi modelado considerando-se três tipos de transições:

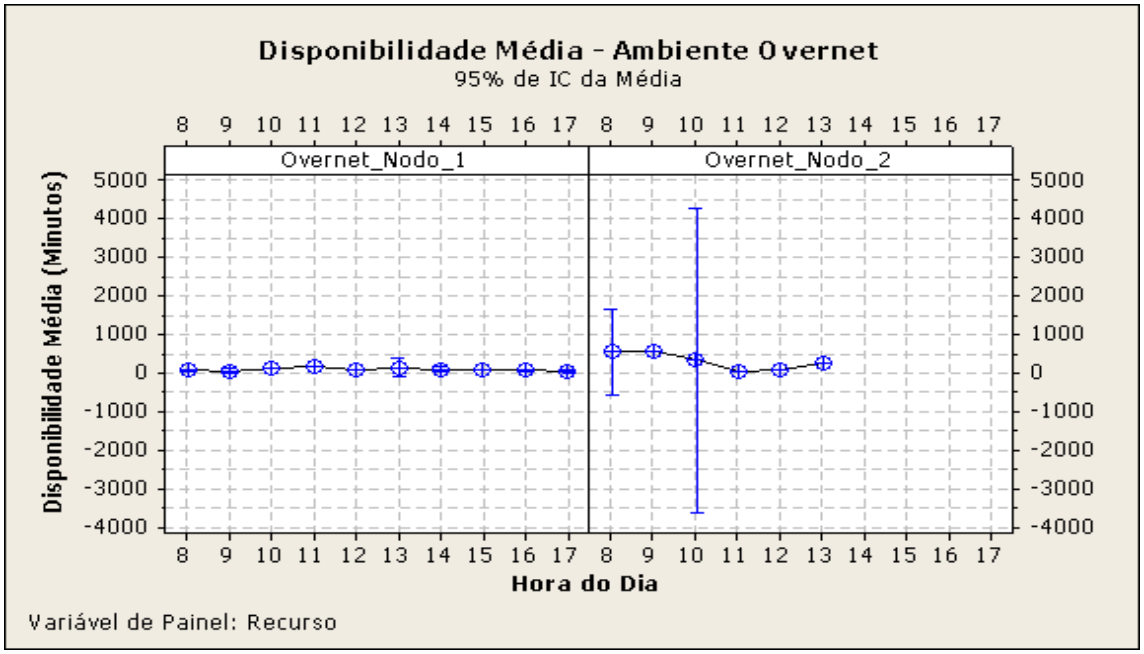

Figura 3.13: Duração de disponibilidade média por hora do dia para o ambiente Overnet

1 transições recorrentes entre localidades;

2 transições de início e fim do dia;

3 transições recorrentes causadas por reuniões, almoço, café e outros eventos.

Segundo os autores, as transições dos tipos 1 e 2 são facilmente identificáveis. No entanto, a identificação das transições do tipo 3 requer um algoritmo mais elaborado e complexo.

Para prever a disponibilidade de um usuário, foi proposto um algoritmo que identifica dinamicamente as transições do tipo 3. Este algoritmo é dividido em três etapas:

- Descoberta de Transições: Neste passo busca-se estabelecer o início, fim e duração de transições, baseando-se na alteração dos níveis de atividade no recurso monitorado.

- Agrupamento de períodos de inatividade similares: Na fase de agrupamento (*Clustering*), o algoritmo agrupa períodos de inatividade conforme sua similaridade a alguma transição descoberta na fase anterior.

- Refinamento da Estimativa: Finalmente o algoritmo refina as estimativas de inicio e fim da transição descoberta, baseando-se nos grupos gerados.

Os objetivos da previsão realizada no algoritmo proposto por Begole, Tang e Hill (2003) são diferentes dos objetivos de um escalonador de um ambiente de *Desktop Grid*. A utilização da métrica de disponibilidade proposta tem o objetivo de oferecer a previsão de presença do usuário e também, formas de visualização gráfica dos períodos padrões de presença do usuário em seu recurso.

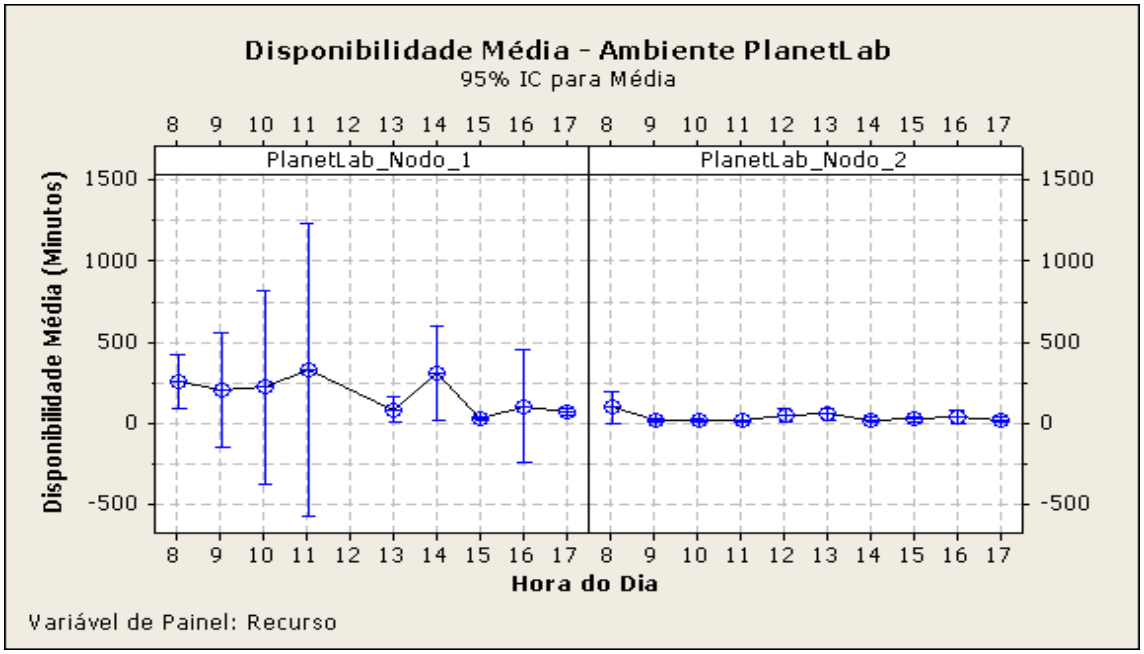

Figura 3.14: Duração de disponibilidade média por hora do dia para o ambiente PlanetLab

3.4.2 A Previsão de Disponibilidade de Taufer et al.

Taufer et al. (2005) propuseram um modelo de disponibilidade de recursos para *Global Computing*.

Neste modelo a disponibilidade de um recurso foi medida conforme a produtividade deste recurso em um período de tempo. Foi proposto então que a disponibilidade dos recursos fosse medida conforme o número de tarefas distribuídas para este recurso bem como, o número de tarefas que ele conseguiu completar. Assim, a disponibilidade de um $recurso_i$ é obtida com a aplicação da seguinte equação:

$$Disponibilidade_i(time - int) = \frac{WU_{completasi}(time - int)}{WU_{distribuidasi}(time - int)} \tag{3.1}$$

O total de tarefas completadas pelo recurso no intervalo de tempo $time - int$ é representado por $WU_{completas}$. O total de tarefas distribuídas para o recurso no intervalo de tempo $time - int$ é representado por $WU_{distribuidas}$.

O número de tarefas distribuídas e o número de tarefas executadas por recurso são persistidos na base de dados do servidor. A responsabilidade de determinar a disponibilidade de cada recurso é do servidor.

Para o escalonamento da tarefa o recurso é classificado conforme o resultado obtido com a aplicação da equação. Em tempo de execução, quando o worker requisita uma tarefa, o servidor atualiza a disponibilidade do recurso e os limites de disponibilidade contidos no servidor. Considerando estes limites, o recurso é classificado como altamente disponível HA (High Available) ou pouco disponível LA (Low Available).

O escalonador realiza então a seleção de *Workers* que receberão tarefas para execução, baseando-se na classificação obtida pelo recurso.

3.4.3 A Previsão de Disponibilidade de Ren et al.

Ren et al. (2006) propuseram um modelo para prever a disponibilidade de recursos que utiliza uma extensão dos processos de Markov, chamada de *Semi-Markov Process*

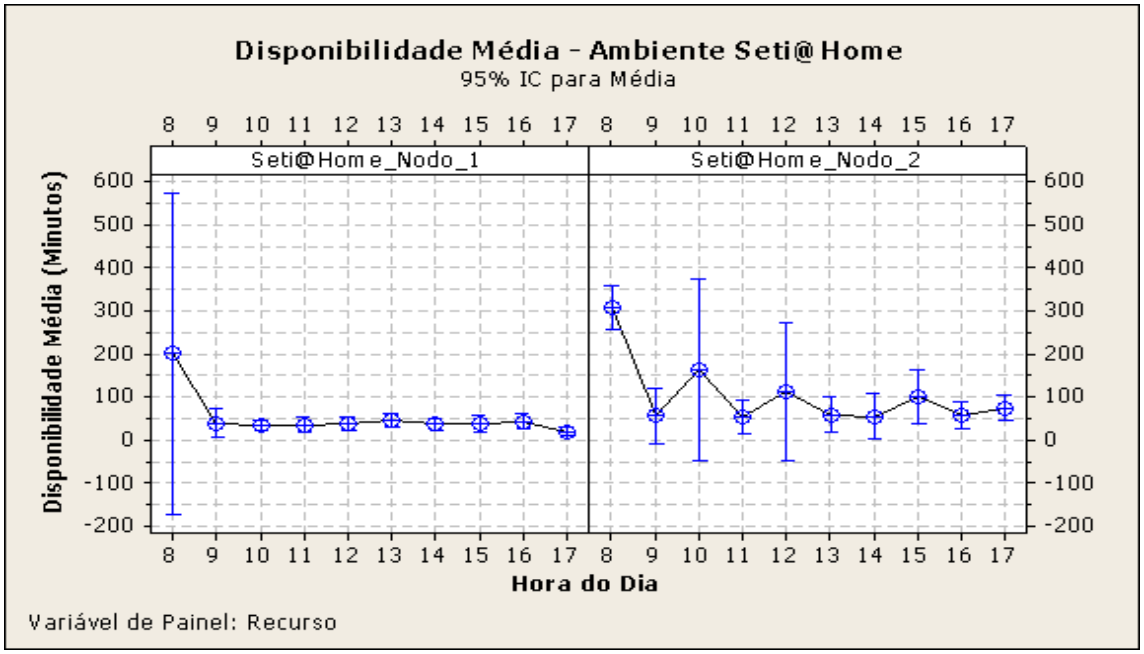

Figura 3.15: Duração de disponibilidade média por hora do dia para o ambiente Seti@Home

Model (SMP). *SMP* é um modelo probabilístico utilizado na análise de sistemas dinâmicos complexos que envolve o conceito de estados e transições de estados, sendo este adequado para aplicação no modelo de disponibilidade proposto.

Neste artigo definiram um modelo de disponibilidade de recursos. Neste modelo, um recurso poderia apresentar cinco estados distintos. Em dois destes estados, o recurso estaria disponível para ser utilizado pelo sistema e nos demais o recurso estaria indisponível.

Os estados foram identificados por S1, S2, S3, S4 e S5. S1 e S2 são os estados em que o recurso está disponível. S1 representa que o recurso pode executar tarefas do sistema externo sem qualquer comprometimento do recurso. S2 representa que a execução concorrente de processos locais e da tarefa externa, está causando a redução de desempenho do recurso, podendo afetar o trabalho do usuário deste recurso. Neste caso, a tarefa é suspensa, sua prioridade é reduzida e sua execução reiniciada, favorecendo os processos locais.

A indisponibilidade de um recurso indica que o mesmo não pode executar tarefas do sistema externo e é representada por três estados, S3, S4 e S5. S3 indica que a tarefa do sistema externo está ultrapassando o limite tolerável e deve ser terminada. S4 indica que não existe memória livre no recurso para execução da tarefa externa e S5 indica os momentos em que o recurso é removido do sistema externo por algum motivo, como o desligamento, falha de *hardware* ou falha de *software* do recurso.

Para determinar se o recurso está em um dos estados S1, S2 ou S3, foram utilizados limites, definidos no próprio trabalho. Dois limites, chamados de th1 e th2, foram obtidos. O limite th1 indica o percentual inferior de uso de *CPU*, onde os processos locais começam a ser prejudicados pela execução de uma tarefa externa. Se a carga da *CPU* estiver abaixo de th1, o recurso está no estado S1. O limite th2 indica o percentual máximo de uso de *CPU*, onde os processos locais são significativamente afetados pela execução de uma tarefa externa. Se o percentual de uso de *CPU* de um recurso encontra-se entre th1 e th2, o recurso é classificado como S2. Se o percentual de uso de *CPU* for maior que th2

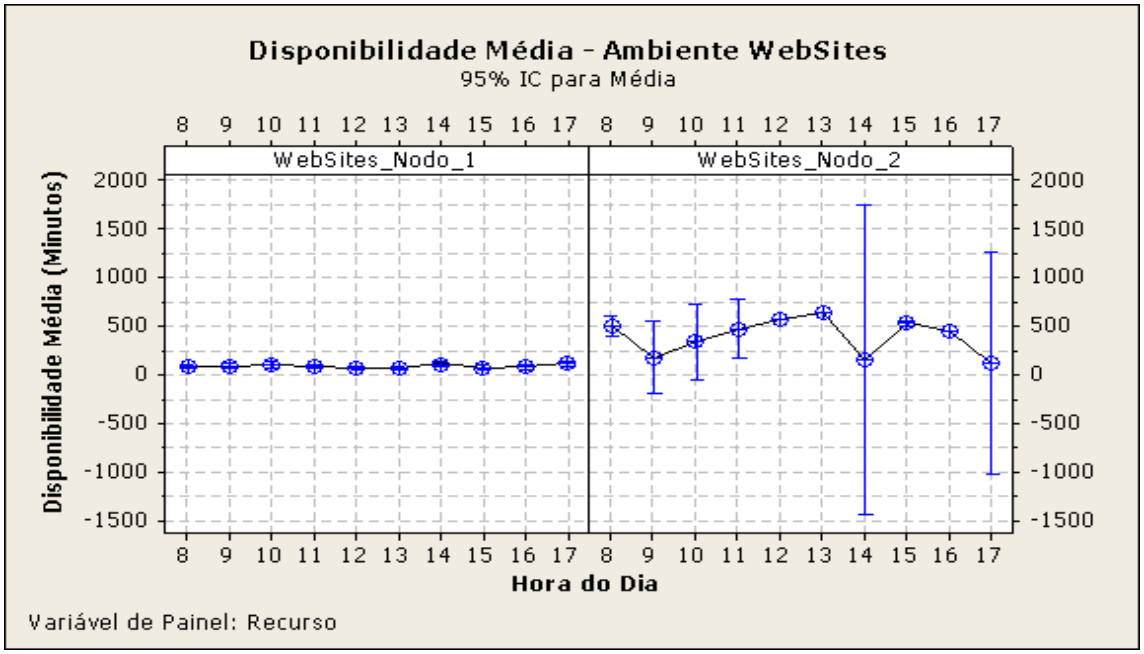

Figura 3.16: Duração de disponibilidade média por hora do dia para o ambiente WebSites

o recurso é classificado como S3.

Com base nesta classificação são gerados traces de disponibilidade contendo o histórico de utilização do recurso. Cada entrada neste histórico corresponde a uma ocorrência de um dos estados de indisponibilidade (S3, S4 ou S5).

Para prever a disponibilidade do recurso, é calculada a confiabilidade temporal do recurso Tr (*Temporal Reliability*). Dado um estado inicial, Tr é a probabilidade que o recurso tem de não alterar seu estado para S3, S4 ou S5 durante um determinado período de tempo.

Para escalonar uma tarefa, o escalonador consulta a disponibilidade prevista de cada recurso disponível, para um período de tempo suficiente para concluir a execução da tarefa a ser escalonada.

3.4.4 A Previsão de Disponibilidade de Huang et al.

Para promover qualidade de serviço na execução de tarefas em ambientes de *Desktop Grid*, Huang et al. (2007) propuseram um mecanismo que inclui um modelo de disponibilidade de recursos.

O modelo de disponibilidade proposto, chamado de *available time model*, é composto por três partes de informação. A primeira parte é chamada de informação básica do recurso e corresponde as características de *Hardware* e *Software* do recurso como memória, número de operações por segundo, largura de banda para envio e recebimento e versão do sistema operacional.

A segunda parte é chamada de informação estatística do tempo de disponibilidade do recurso. Os dados de disponibilidade do recurso são coletados, sintetizados e armazenados em um banco de dados, onde cada registro corresponde a uma hora do dia de um dia da semana. A estrutura da tabela que armazena estes dados contém campos que informam a hora do dia, o tempo médio dos intervalos de disponibilidade ocorridos naquela hora do dia *MTOA* (*Mean Time of Availability*), o número médio de períodos de disponibilidade daquela hora do dia *ATimes* (Availability Times), a média de uso de *CPU*, o desvio pa-

Tabela 3.4: Número médio de ocorrências de pelo menos 1 intervalo de disponibilidade igual ao Intervalo de Referência

Ambiente	**M(%)**	**CV**
Dbcc	86.75	2.09
DNS	81.25	2.63
Grid5000	97.26	0.45
Microsoft	89.77	2.92
Notre Dame	98.91	0.24
Overnet	58.66	11.41
PlanetLab	94.48	8.41
Seti@Home	95.47	0.87
Websites	97.73	1.10
Total	**88.92**	**14.34**

M = Média aritmética.
CV = Coeficiente de variação.

drão dos elementos que compõem *MTOA* e desvio positivo e negativo dos elementos que compõem *MTOA*.

A terceira parte é chamada de histórico de execução de tarefas. Os dados de histórico de execução de tarefas são sintetizados em uma tabela que apresenta em seus campos o número de tarefas requisitadas pelo recurso, o número de tarefas terminadas dentro do tempo estimado, o número de tarefas terminadas com atraso e o número de tarefas migradas.

Com estas informações obtém-se a métrica *Emergency Degree* ($D_{emergency}$) que determina o nível de urgência que uma tarefa tem para terminar a sua execução no tempo estimado. A equação 3.2 apresenta o cálculo do nível de urgência.

$$D_{emergency} = \frac{ETA}{RET} \tag{3.2}$$

Na equação 3.2, *ETA*(*Expected Time of Availability*) corresponde ao tempo de disponibilidade total estimado, desde o tempo atual (*nowtime*) até o tempo final de execução da tarefa (*deadline*) e é obtido através da equação 3.3.

Na equação 3.3, *MTOA* e *ATimes* são obtidos da tabela de tempo de disponibilidade das informações estatísticas de tempo de disponibilidade citadas anteriormente.

$$ETA = \sum_{i=nowtime}^{deadline} (MTOAxATimes) \tag{3.3}$$

Para a utilização da métrica de disponibilidade foi definido um modelo chamado de *QoS Engine*. Neste modelo estão incluídos os componentes necessários para a obtenção da métrica de disponibilidade e os componentes necessários para decisão de que ação deve ser tomada com uma determinada tarefa.

O modelo foi implementado e testado utilizando o XtremWeb. Como o XtremWeb não implementa nenhum mecanismo de *checkpoint* os autores definiram que no contexto do trabalho, migração corresponde a interrupção da execução da tarefa e reescalonamento em outro recurso, enquanto replicação corresponde a submeter a tarefa para executar em um novo recurso sem interromper a execução da tarefa no recurso atual, ou seja, de forma

paralela. Para determinar se uma tarefa deve ser migrada, replicada ou se nenhuma medida necessita ser tomada avalia-se em conjunto a confiabilidade e o $D_{emergency}$ do recurso, conforme a tabela 3.5.

Tabela 3.5: Discretização de *Emergency Degree*

Confiabilidade \ Emergência	Alta	Media+	Media-	Baixa
Alta	M	R	R	R
Media+	N	N	M	R
Media-	N	N	N	M
Baixa	N	N	N	N

Ações: N = Normal, M = Migrar e R = Replicar

Caso seja detectada a necessidade de tomar alguma ação, um sinal é enviado ao servidor, para que o mesmo tome a ação correspondente.

3.4.5 A Previsão de Disponibilidade de Rahman, Hassan e Buyya

Rahman, Hassan e Buya (2010) propuseram um algoritmo de aprendizagem preguiçoso (*Lazy Learning Algorithm*) baseado no Coeficiente de Jaccard *Jaccard Coefficient* para realizar a previsão de disponibilidade de recursos de *Enterprise Desktop Grids*.

O coeficiente de Jaccard indica a similaridade ou dissimilaridade existente entre duas sequências, ou janelas, de números binários de mesmo tamanho.

A previsão de disponibilidade é realizada sobre o histórico de utilização de um recurso, armazenado em arquivos que contém o valor 0 e 1 para indicar indisponibilidade e disponibilidade, respectivamente.

Para obter o coeficiente de Jaccard, o algoritmo proposto obtém dos dados históricos de utilização, duas janelas de mesmo tamanho representando o comportamento recente e o comportamento passado. É obtido o coeficiente aplicando a equação de *Jaccard* sobre as janelas obtidas.

Após obter o valor do coeficiente, a janela representando o comportamento passado é deslocada uma unidade à frente (direcionando-se para o tempo presente) e o coeficiente é calculado novamente. Esse processo segue até que a janela de passado atinja a penúltima posição da janela que contém o comportamento recente. Então, a janela de passado que obteve o maior coeficiente é então selecionada. O valor binário subsequente a janela escolhida é considerado como o estado previsto para o recurso.

Os autores apresentaram também, uma extensão ao modelo incluindo votação. Variando o tamanho da janela para a mesma previsão, o resultado da previsão também era alterado. Para oferecer melhores previsões, o algoritmo realizava a votação, escolhendo o resultado que mais se repetiu para todas as variações de tamanho de janela.

O modelo proposto foi comparado com duas outras implementações de algoritmos de previsão: (a) *K-Nearest Neighbors algorithm* e (b) *Naive Bayes*. Os resultados obtidos com o modelo proposto superaram os demais algoritmos implementados, quando o tamanho das janelas utilizadas pelo algoritmo era pequeno.

3.5 A Necessidade de um Novo Modelo para Previsão de Disponibilidade

O escalonamento considerando a disponibilidade é, de fato, um dos grandes desafios de pesquisa na área de grades oportunistas (CHOI et al., 2008, 2006; KONDO et al., 2007; KONDO, 2005; CONSTANTINESCU-FULØP, 2008; BYUN et al., 2005). Desta forma, um dos desafios para a realização deste trabalho está na possibilidade de aumentar a eficiência na utilização dos recursos de grades oportunistas, através do fornecimento de informações de previsão de disponibilidade de recursos ao escalonador de tarefas da grade oportunistas.

Com a análise de dados de disponibilidade realizada e com a análise dos diferentes trabalhos relacionados, identificou-se a necessidade de propor um novo modelo de previsão de disponibilidade para recursos participantes de grades oportunistas.

Na análise de disponibilidade dos recursos realizada, foram identificados alguns critérios que julgou-se fundamentais para um modelo de previsão de disponibilidade. Cada trabalho relacionado foi então avaliado considerando-se estes critérios.

Além desses critérios, definiu-se um critério adicional, que respeito ao local onde a previsão é realizada. Este critério foi nomeado como **Local**, e pode assumir os valores "Servidor", nos casos em que a previsão de disponibilidade é realizada pelo servidor, "Recurso"nos casos onde a previsão é realizada pelo recurso e "Desconhecido"para os casos em que o local de realização das previsões não é citado.

Ambientes de grades oportunistas operam com milhares de recursos efetuando requisições por tarefas, submetendo tarefas, devolvendo resultados, atualizando o seu estado junto ao servidor. Realizar a previsão de disponibilidade no servidor pode inserir um alto custo computacional ao servidor, fazendo com que o mesmo tenha que executar o algoritmo de previsão sobre todos os recursos registrados, desviando-se de suas atividades principais.

Outro critério importante identificado é a capacidade de realizar previsões sobre uma grande variedade de diferentes medias de disponibilidade (*Diversidade*). Este critério é importante pois avalia a possibilidade de aplicar o mecanismo de predição sobre diferentes bases de dados de históricos de utilização de recursos. Assim, o mecanismo de previsão não fica limitado a prover previsões sobre apenas um tipo de medida de utilização.

Tabela 3.6: Comparativo dos trabalhos relacionados

Proposta	Ind.	Din.	Loc.	Pad.	Exc.	Div.
Begole, Tang e Hill (2003)	S	S	N	S	N	N
Taufer et al. (2005)	S	N	Srv	N	N	N
Ren et al. (2006)	S	S	Rsc	S	N	N
Huang et al. (2007)	S	S	Srv	S	N	N
Rahman, Hassan e Buyya (2010)	S	S	N	N	N	S

Individual (Ind.): S = Sim, N = Não
Dinâmico (Din.): S = Sim, N = Não
Local (Loc.): Srv = Servidor, Rsc = Recurso, N = Não se Aplica
Padrão (Pad.): S = Sim, N = Não
Exceção (Exc.): S = Sim, N = Não
Diversidade (Div.): S = Sim, N = Não

A tabela 3.6 apresenta a classificação dos trabalhos relacionados, conforme os critérios especificados.

O critério Individual foi identificado em todas propostas apresentadas, o que reforça a necessidade de realizar previsões individuais e não coletivas.

O critério Dinâmico não foi atendido na proposta de Taufer et al. (2005), visto que os autores indicam que a previsão utiliza um espaço de tempo, como por exemplo as últimas 24 horas do dia, para qualquer previsão realizada. O problema desta abordagem é a perda de informação gerada pela falta de dinamismo do modelo de previsão proposto, que não considera o momento em que o recurso tornou-se ocioso. Aplicando-se a proposta de Taufer et al. (2005), pode-se chegar a seguinte situação: um recurso (A) obteve maior índice de participação do que outro recurso (B), em um período de 24 horas. Segundo a proposta, o recurso (A) é mais disponível que o recurso (B) e neste caso, o escalonador selecionará o recurso (A) para entregar um tarefa para execução. No entanto, analisando-se esta situação é possível perceber que não se pode afirmar que o recurso (A) foi mais disponível que (B) no intervalo das 13:00 às 14:00. Este informação é importante, pois se o recurso (B) apresentou maior disponibilidade neste intervalo, seria melhor que o escalonamento de tarefas que ocorrer nesse intervalo, selecione o recurso (B) e não o (A).

As propostas de Taufer et al. (2005) e Huang et al. (2007) tem como local de realização da previsão, o servidor. Este tipo de abordagem insere sobrecarga de comunicação na rede e de processamento no servidor, por exigir a transferência de informações de disponibilidade dos recursos para o servidor, aumentando o tráfego de rede. A necessidade de realizar a previsão de disponibilidade para cada recurso disponível, insere sobrecarga de processamento no servidor.

Considerando-se o critério Padrão de Disponibilidade, observa-se que na maioria dos modelos propostos a existência de padrões de disponibilidade nos dados é requisito.

Referente ao critério exceções, nenhuma proposta mostrou-se capaz de realizar a previsão de exceções. Apenas a proposta Rahman, Hassan e Buyya (2010) seria capaz de realizar uma previsão mais adequada, devido a verificação de similaridade. No entanto, por utilizar janelas de tamanho pequeno para obter maior acurácia, dificilmente seria capaz de identificar uma exceção.

Referente a capacidade de realizar previsões utilizando diferentes tipos de dados (Diversidade), com exceção de Rahman, Hassan e Buyya (2010), os demais mecanismos não são capazes de utilizar diferentes tipos de dados para realizar a previsão de disponibilidade, visto que são extremamente relacionados ao tipo de dado quer utilizam para realizar a previsão. Já Rahman, Hassan e Buyya (2010) utilizam um mecanismo capaz de realizar previsões sobre diferentes tipos de dados, desde que estes dados possam ser, de alguma forma, classificados de forma binária.

Nota-se a necessidade e espaço para propôr um modelo de previsão de disponibilidade dinâmico, que realiza à previsão no próprio recurso, que não dependa da existência de padrões de disponibilidade no recurso, capaz de realizar previsões mesmo em casos onde venham a ocorrer exceções no que diz respeito à utilização de recursos e principalmente, diverso no que diz respeito a capacidade de realizar previsões sobre diferentes tipos de dados .

Deste espaço encontrado, surgiu a proposta deste trabalho: **AvSchedP**, um modelo de previsão de disponibilidade para grades oportunistas.

3.6 Considerações Finais

A disponibilidade de um recurso, aqui definida como um dos estados que um determinado recurso pode assumir ao longo do tempo, sendo estes estados apto ou inapto a executar uma tarefa da Grade Oportunista, foi estudada detalhadamente neste capítulo.

Considerando os intervalos de disponibilidade existente nos dados utilizados nesta análise, estudou-se a existência de padrões de repetição destes intervalos ao longo do tempo, aqui chamados de padrões de disponibilidade.

Realizou-se tal analise, tendo em vista a base dos modelos estatísticos que é a existência de padrões em dados. Foi possível identificar e demonstrar que nos dados utilizados existe a repetição de intervalos de disponibilidade ao longo do tempo, mas que a frequência de ocorrência destas repetições é baixa para ser considerada como um padrão de disponibilidade.

Percebeu-se então que um modelo baseado em padrões poderia não ser o mais adequado. Além disso, identificou-se como importante para um modelo de previsão de disponibilidade, considerar as particularidades de cada recurso.

Foram então identificadas algumas características entendidas como necessárias para a concepção de um modelo de previsão de disponibilidade de recursos, destacando-se entre elas a capacidade de realizar previsões de disponibilidade em dados que não apresentam padrões significativos e a localidade do componente de previsão com o objetivo de considerar a individualidade comportamental de cada recurso, sem sobrecarregar o servidor da Grade Oportunista.

Estas características foram utilizadas para avaliar alguns dos modelos de previsão de disponibilidade existentes, o que culminou na necessidade de propor um novo modelo de previsão de disponibilidade.

O próximo capítulo apresenta o modelo de previsão de duração de disponibilidade para grades oportunistas, AVSchedP concebido neste trabalho e que fundamenta-se nas características identificadas neste capítulo.

4 AVSCHEDP: PREVISÃO DE DISPONIBILIDADE PARA GRADES OPORTUNISTAS

O presente capítulo apresenta a descrição do modelo de previsão de disponibilidade para grades oportunistas, elaborado neste trabalho.

São apresentados os principais componentes do modelo, entre eles o **Preditor**, considerado como o principal componente do modelo proposto. O algoritmo de previsão de duração de disponibilidade localizado no **Preditor** é apresentado detalhadamente e suas características são relacionadas aos aspectos identificados na análise apresentada na Seção 3.3 do Capítulo 3, análise esta desenvolvida no contexto deste trabalho.

Apresenta-se também o tratamento de heterogeneidade dos recursos proposto no modelo AvSchedP, finalizando com a descrição do funcionamento como um todo dos componentes propostos no modelo.

4.1 Introdução

O *AvSchedP* (*Availability Predictor for Scheduling*) foi concebido para realizar a previsão de duração de intervalos de disponibilidade dos recursos (*Workers*) participantes de grades oportunistas.

Para o AvSchedP, o conhecimento do tempo de duração de cada intervalo de disponibilidade que venha a ocorrer em um recurso, é a maior fonte de incerteza para um escalonador de tarefas inserido em um contexto de grade oportunista.

Em geral o tratamento de incertezas remete a um problema complexo. Esta complexidade deve-se principalmente a dificuldade de prever o comportamento de uma ou mais fontes de incerteza (REAL et al., 2003).

O AvSchedP foi concebido com o propósito de reduzir o nível de incerteza existente em um escalonador de tarefas de grades oportunistas.

O modelo tem como produto, a informação de tempo duração de disponibilidade prevista para um intervalo de disponibilidade ocorrido em um recurso. Para escalonadores de tarefas a informação de tempo de duração de disponibilidade de um recurso não é por si só suficiente, devido a heterogeneidade dos recursos envolvidos, ou seja, considerar somente o tempo de duração de disponibilidade em ambientes com recursos heterogêneos não seria o mais adequado devido a presença de diferentes capacidades computacionais entre os recursos participantes (KONDO, 2005).

Para oferecer uma informação mais adequada ao escalonador, o modelo propõe a transformação da medida de tempo de disponibilidade para quantidade de processamento disponível. Esta informação é mais adequada ao escalonador, pois nela encontram-se encapsuladas as informações de tempo de disponibilidade do recurso e capacidade de

processamento do recurso, ou seja, esta informação considera a heterogeneidade de processamento dos recursos envolvidos.

Apesar de ter como principal objetivo realizar a previsão de disponibilidade para servir de informação utilizada por escalonadores de tarefas de grades oportunistas, a informação de tempo de duração de disponibilidade poderia ser utilizada para outros fins.

A informação de tempo de duração de um intervalo de disponibilidade, resultante da previsão realizada pelo AvSchedP, poderia ser utilizada na determinação da periodicidade de realização de *checkpoints*, ou para decidir o melhor momento para migrar uma tarefa, ou para escolher o recurso mais adequado a manter réplicas de arquivos em sistemas P2P de arquivos distribuídos, entre outras possibilidades.

Um exemplo é o FreeMMG 2 (CECIN et al., 2004), projeto do Grupo de Processamento Paralelo e Distribuído (GPPD) do Instituto de Informática da UFRGS. Esse projeto consiste na proposta de uma nova arquitetura de suporte a jogos massivamente multijogador ou "MMOGs"(*Massively multiplayer online games*), baseada no modelo Par-a-Par. Uma rede FreeMMG 2 pode ter até centenas de milhares de pares, que serão sempre recursos computacionais voluntários como PCs desktop. Os pares são as máquinas dos jogadores. Os jogadores, quando presentes no computador, estão jogando o jogo e, quando não estão utilizando o PC de forma interativa, o disponibilizam para disparar um "par simulador"da arquitetura. O "par simulador"é um daemon não interativo que, juntamente com os outros pares simuladores, mantém a vasta malha de simulação da rede e atua como "servidor"para outros "pares interativos", ou seja, os PCs onde o usuário humano está presente interagindo com o jogo. A seleção de um "par simulador"deve ser realizada com cautela pois, se este recurso tornar-se indisponível exigirá a troca de contexto do ambiente, impactando drasticamente no desempenho do ambiente. Em discussões no GPPD, descobriu-se que a contribuição central desta dissertação, o modelo de predição de disponibilidade de recursos, supre uma lacuna na arquitetura FreeMMG 2, que não dispõem de previsão de disponibilidade para seleção dos "pares simuladores".

Outro exemplo é o JavaRMS (GOMES, 2008), um sistema de gerência de dados para grades baseado num modelo P2P. O JavaRMS mantém réplicas de fragmentos de arquivos espalhados pela rede. A disponibilidade de um dado neste sistema está ligada diretamente a disponibilidade dos recursos selecionados para manter as réplicas. Desta forma, a previsão de disponibilidade é um serviço imprescindível para este modelo.

Finalmente, tem-se o modelo GRAND (MANGAN, 2006). O modelo GRAND propõem um escalonador de tarefas estático e hierárquico. Nesta hierarquia, recursos da rede local são selecionados para gerenciar a execução de tarefas. É interessante para este modelo que o recurso selecionado apresente um certo nível de disponibilidade, para que não ocorra a perda de execução de tarefas, devido a indisponibilidade deste recurso.

4.2 Grades Oportunistas, Disponibilidade e o AvSchedP

Escalonadores de grades oportunistas têm como atribuição a determinação da alocação de tarefas a recursos compartilhados (não dedicados). Estes recursos são a base de operação destas grades. A grade aproveita-se dos momentos de ociosidade destes recursos para executar tarefas, como apresentado no capítulo 2.

Um dos principais problemas enfrentados por escalonadores inseridos nestes ambientes é a ausência de determinismo dos períodos de disponibilidade ocorridos nos recursos. Os períodos de disponibilidade ocorrem de forma estocástica.

Para escalonadores estáticos, a previsão dos períodos de disponibilidade de um recurso

é fator fundamental para atingir o objetivo de escalonamento, pois todas tarefas devem ser alocadas antes do sistema iniciar a execução (ANURAG; GUY; JOEL, 1997).

A previsão de disponibilidade, para um escalonador estático em uma Grade Oportunista, necessitaria de um modelo de previsão que realizasse a previsão de início e duração de um intervalo de disponibilidade. Somente com a obtenção destas informações, que permitem a determinação de qual tarefa e quando esta será executada, seria possível realizar o escalonamento de tarefas.

No entanto, conforme apresentado na seção 2.3, em um cenário de grades oportunistas o escalonador é considerado dinâmico. Por esta razão, assume-se que o AvSchedP é um modelo de previsão de disponibilidade para escalonadores dinâmicos inseridos no contexto de grades oportunistas.

Escalonadores dinâmicos, quando comparados a escalonadores estáticos são considerados mais complexos (ANURAG; GUY; JOEL, 1997). No entanto, quando o problema é previsão de disponibilidade, a previsão em si pode ser mais simples quando o objetivo é oferecer a informação de disponibilidade a um escalonador dinâmico.

Para um escalonador dinâmico em uma Grade Oportunista, a previsão de disponibilidade pode consistir, unicamente, em prever o tempo de duração de um intervalo de disponibilidade que já iniciou. O algoritmo de previsão de disponibilidade pode favorecer-se da característica do modelo de escalonamento *pull* deste ambiente e realizar a previsão de disponibilidade somente no instante em que o recurso torna-se ocioso.

Assim, o AvSchedP não realiza a previsão de quando um intervalo de disponibilidade ocorrerá mas sim, dado o início de um intervalo de disponibilidade, por quanto tempo o recurso permanecerá disponível.

O AvSchedP realiza a previsão de disponibilidade sobre sequências de números binários. Uma sequência é a representação da utilização do recurso ao longo do tempo, onde esta utilização é discretizada em apenas dois estados possíveis: (0) quando o recurso esteve ocioso e (1) quando o recurso esteve ocupado.

A possibilidade de realizar previsões sobre uma sequência de números binários confere ao AvSchedP a possibilidade de aplicá-lo a qualquer política de recrutamento empregada em um recurso.

O AvSchedP foi projetado para realizar previsões dinâmicas, em recursos voláteis e não é baseado em padrões de disponibilidade.

4.3 A Previsão de Disponibilidade

Recursos participantes de uma Grade Oportunista são recursos doados por seus proprietários. A execução de tarefas da grade depende da ociosidade destes recursos, ou seja, depende do proprietário do recurso estar ou não utilizando o recurso. Por esta razão, prever a disponibilidade de um recurso pertencente a uma Grade Oportunista incorre em prever o comportamento de um ser humano ao utilizar o recurso computacional dele.

A previsão do comportamento humano é uma tarefa que exige modelos extremamente complexos (PENTLAND; LIN, 1995).

De fato, realizar previsões na tentativa de definir o tempo de duração de um intervalo de disponibilidade é uma tarefa complexa. Por esta razão, a duração de um período de disponibilidade é tratada neste modelo como a principal fonte de incerteza para um escalonador de tarefas de Grades Oportunista.

Considerando-se o cenário de Grade Oportunista, a utilização de modelos extremamente complexos pode ser inviável devido a diversas limitações de execução do algoritmo

que representa este modelo, como tempo hábil e capacidades computacionais exigidos para execução deste algoritmo.

Para realizar a previsão de disponibilidade, o AvSchedP utiliza-se de um novo algoritmo de previsão de disponibilidade, proposto neste trabalho. Este algoritmo realiza a previsão do tempo de duração de um intervalo de disponibilidade baseando-se na similaridade existente entre o comportamento atual do recurso e comportamentos armazenados em uma base de históricos de utilização do recurso. A base de históricos de utilização retrata o perfil de utilização do recurso por seu usuário ao longo do tempo.

O AvSchedP propõe um algoritmo capaz de ser executado em recursos convencionais e realizar previsões em tempo hábil, além de apresentar bons resultados quando aplicado a previsão de disponibilidade de recursos de grades oportunistas.

Para manter a escalabilidade do *Server* de uma grade oportunista, o AvSchedP propõe uma arquitetura de componentes para o recurso computacional (*Worker*), ou seja, o próprio recurso é quem realiza a previsão de disponibilidade. A opção por esta abordagem está de acordo com os critérios requeridos para um modelo de previsão de disponibilidade, identificados na Seção 3.5 deste trabalho.

O modelo de previsão proposto pode ser dividido em uma série de atividades. Estas atividades vão desde a coleta de dados de utilização do recurso, até a obtenção da informação de disponibilidade proveniente da previsão. As atividades do modelo proposto são descritas detalhadamente a seguir.

4.3.1 Manutenção do Histórico de Utilização

O modelo AvSchedP realiza a previsão baseando-se no histórico de utilização deste recurso. O histórico de utilização do recurso é obtido registrando-se os períodos de disponibilidade e indisponibilidade do recurso.

A informação de disponibilidade do recurso é persistida considerando-se apenas 2 estados possíveis. O estado (0) para os casos em que o recurso encontra-se ocioso e (1) para os casos em que o recurso encontra-se ocupado. Esta informação é persistida junto com uma marca temporal que contém as informações necessárias para identificar o dia, mês, ano, hora, minuto e segundo em que ocorreu aquele estado[1].

Os registros de histórico de utilização são persistidos no próprio recurso. A persistência não requer a instalação de *software* adicional, devido a necessidade de sempre evitar a instalação de *software* em recursos da grade oportunista. A instalação de um sistema de gerenciamento de banco de dados (SGBD), por exemplo, além de consumir mais recursos computacionais do recurso, exigiria conhecimento técnico do proprietário do recurso para realizar a instalação, configuração e manutenção deste *software*.

O espaço consumido para o armazenamento de informações relativas a utilização do recurso é uma preocupação do modelo AvSchedP. Ao longo do tempo, a base de históricos cresce, podendo consumir recursos de maneira excessiva. O espaço de armazenamento no recurso é um ponto importante, visto que, este recurso é doado pelo proprietário.

Para amenizar o consumo de armazenamento, o modelo propõe a limpeza periódica destes dados. O problema neste caso consiste em determinar quais dados históricos podem ser eliminados.

O AvSchedP propõe a limpeza desta base de históricos utilizando o algoritmo de definição de similaridade entre comportamentos, proposto neste trabalho, e que será apresentado posteriormente.

[1] A frequência de armazenamento destes dados (minuto-a-minuto, segundo-a-segundo,...) pode ser configurada.

Ao longo do tempo o comportamento de utilização de um recurso volta a ocorrer de forma similar. Por esta razão, o AvSchedP propõe a identificação e exclusão de comportamentos similares na base de históricos, mantendo apenas um único registro deste comportamento.

Aplicando-se o algoritmo de definição de similaridade à base de históricos, é possível identificar a similaridade entre dois períodos de utilização do recurso armazenados no histórico. É importante ressaltar que estes dois períodos analisados devem compreender um mesmo período de utilização, ou seja, a hora de início e fim dos intervalos comparados devem ser iguais.

Quando o obtido com a execução do algoritmo indicar similaridade total entre os intervalos de utilização selecionados, o AvSchedP opta pela exclusão do intervalo mais antigo.

Desta forma, o AvSchedP mantém a base de históricos de utilização enxuta, economizando espaço de armazenamento neste recurso. Além disso, a limpeza da base de históricos promove maior eficiência ao algoritmo de previsão de disponibilidade, pois ao realizar a previsão o algoritmo utilizará apenas intervalos de utilização distintos (não similares), o que aumentará o espaço de escolha do algoritmo de previsão.

4.3.2 A Determinação de Similaridade

A determinação de similaridade no AvSchedP é realizada através de um novo algoritmo proposto neste trabalho. Este algoritmo tem como resultado o nível de similaridade entre dois intervalos de utilização selecionados.

Conforme citado anteriormente, a disponibilidade do recurso é representada por dois estados no AvSchedP ((0) ocioso e (1) ocupado).

Um intervalo de utilização é então definido como uma sequência de 0s e 1s, ordenados temporalmente, que representam a utilização do recurso naquele intervalo. O tamanho de um intervalo é definido pela equação 4.1:

$$tamanho(pi, pf) = instante(pf) - instante(pi) \tag{4.1}$$

onde $instante(pi)$ corresponde ao *timestamp* de início (i) do intervalo (p) e $instante(pf)$ corresponde ao *timestamp* de fim (f) do intervalo (p);

O algoritmo de similaridade trata um intervalo de utilização como um vetor de dados binários, onde o índice do vetor representa o instante de tempo em que o estado foi determinado, conforme apresentado na figura 4.1.

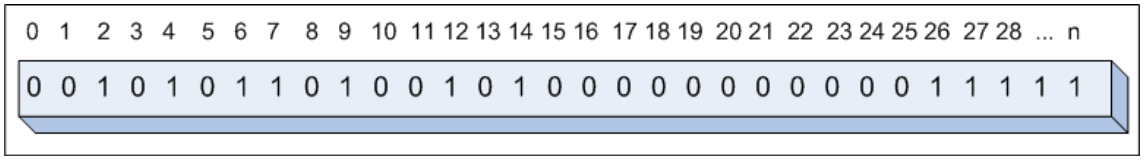

Figura 4.1: Representação de um intervalo de utilização como um vetor binário

O AvSchedP determina a similaridade entre dois intervalos de utilização distintos aplicando técnicas de determinação de similaridade entre vetores binários.

A similaridade entre vetores binários pode ser determinada por diferentes técnicas existentes na literatura. Estas técnicas geralmente resultam em coeficientes que determinam o grau de similaridade entre estes vetores. Alguns dos principais coeficientes para

determinação de similaridade entre vetores binários estão descritos no apêndice A deste trabalho.

A determinação da similaridade para o cenário do AvSchedP considera dois resultados possíveis na comparação entre elementos dos vetores binários comparados. Ao comparar um elemento i de dois vetores binários A e B, são contabilizados como sucesso os casos onde $A[i] = B[i]$ e como fracasso os casos onde $A[i] <> B[i]$. Nos casos onde $A[i]$ e $B[i]$ são diferentes, não existe nenhum nível de importância para as diferentes combinações entre $A[i]$ e $B[i]$. Por exemplo, a ocorrência de ($A[i] = 0$ e $B[i] = 1$) ou ($A[i] = 1$ e $B[i] = 0$), não é tratada de forma diferenciada pelo AvSchedP, sendo ambos os casos tratados como fracasso.

Entre os coeficientes de similaridade estudados, verificou-se que o **Coeficiente de Hamann** (FRANK; TODESCHINI, 1994) ou *Hamann Coefficient* considera os pares (1-1) e (0-0) como casos de sucesso (idênticos) e os pares (1-0) e (0-1) como casos de insucesso (não idênticos), como é requerido pelo modelo AvSchedP. Este coeficiente não diferencia entre os dois possíveis casos de fracasso como outros coeficientes fazem.

A determinação do coeficiente de *Hamann* inicia com a geração da tabela de contingência (2 x 2) contendo os dados resultantes da comparação entre dois vetores binários (FRANK; TODESCHINI, 1994).

A célula A da tabela corresponde a contagem das ocorrências onde o vetor $Va[i]$ e o vetor $Vb[i]$ no instante i apresentavam o valor 1. A célula B corresponde a contagem dos casos onde o vetor $Va[i]$ no instante i apresentava o valor 0 e a vetor $Vb[i]$ no instante i apresentava o valor 1. A célula C corresponde a contagem dos casos onde o vetor $Va[i]$ no instante i apresentava o valor 1 e a vetor $Vb[i]$ no instante i apresentava o valor 0. A célula D da tabela corresponde a contagem das ocorrências onde o vetor $Va[i]$ e o vetor $Vb[i]$ no instante i apresentavam o valor 0.

Após a geração da tabela de contingência, aplica-se a equação de *Hamann* 4.2 sobre os dados da tabela formada, obtendo-se como resultado o coeficiente que quantifica a similaridade existente entre os dois vetores comparados (FRANK; TODESCHINI, 1994).

$$cHamann = \frac{(A + D) - (B + C)}{(A + D) + (B + C)} \tag{4.2}$$

A figura 4.2 apresenta a tabela de contingência e o coeficiente de *Hamann* obtidos com a determinação de similaridade entre dois vetores binários Va e Vb.

O coeficiente de *Hamann* resultante pode variar entre -1 e 1. Quanto mais próximo de 1 for o coeficiente de *Hamann*, maior será a similaridade entre os vetores comparados, assim como quanto mais próximo de -1, menor será a similaridade entre os vetores.

Na figura 4.2 o coeficiente de *Hamann* resultante foi $cHamann = 0.55$. Como pode ser observado na figura, os vetores Va e Vb são significativamente similares e isto é comprovado pelo resultado obtido com a aplicação da equação de *Hamann*.

A figura 4.3 apresenta o algoritmo para a determinação de similaridade entre 2 vetores binários. Ao receber 2 vetores binários v_a e v_b, o algoritmo monta a tabela de contingência ($linhas\ 3, 4$ e 5) e aplica a equação de *Hamann* 4.2 à tabela de contingência ($linha\ 8$). Então, o algoritmo retorna o valor encontrado ($linha\ 9$). Neste algoritmo, a tabela de contingência foi modelada como uma vetor de duas dimensões (Matriz 2 X 2) chamado de cT.

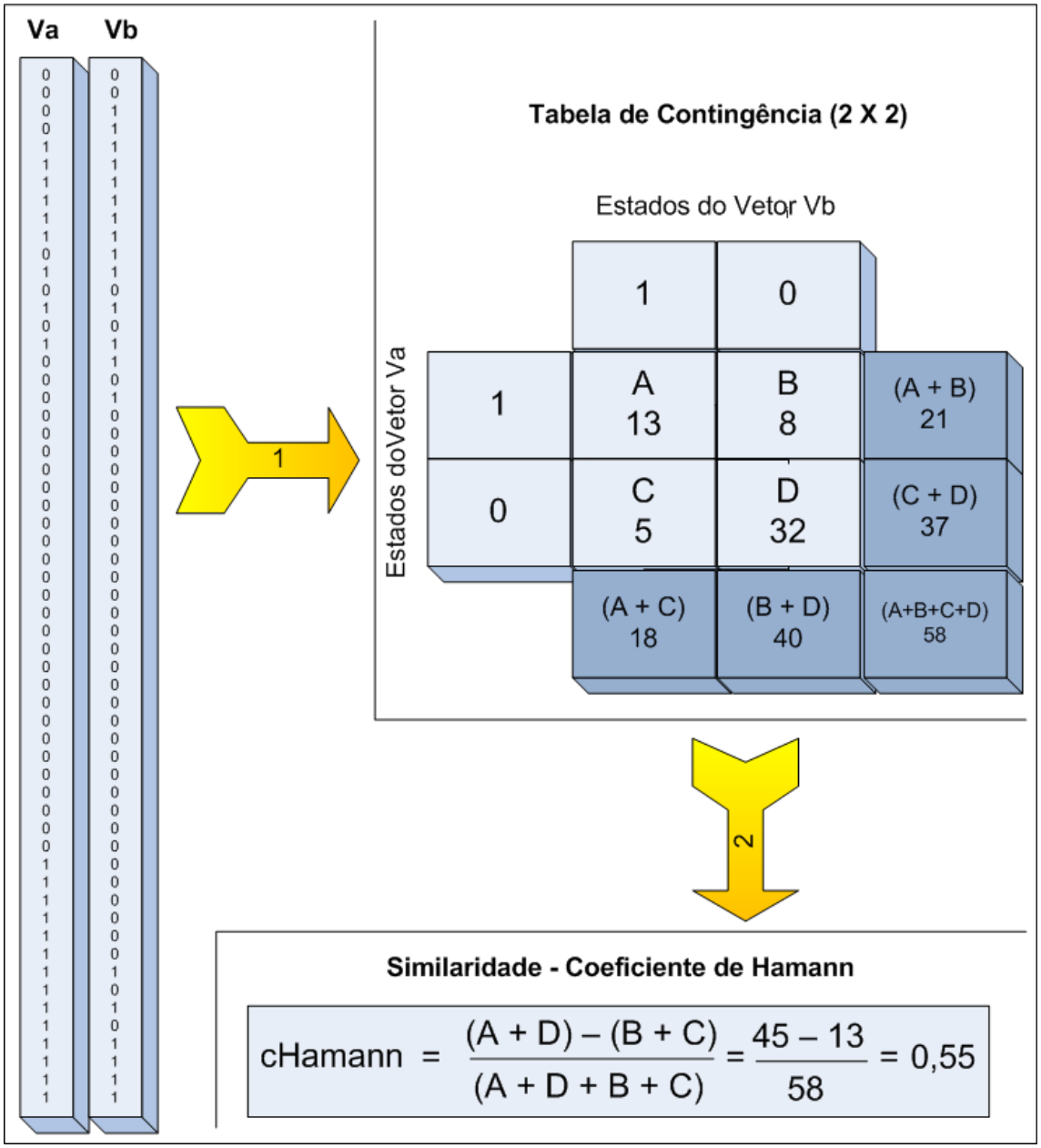

Figura 4.2: Similaridade entre vetores binários utilizando o coeficiente de Hamann

4.3.3 A Previsão de Duração de Disponibilidade

A previsão de duração de disponibilidade adotada pelo AvSchedP não exige a existência de padrões comportamentais dos usuários ao utilizarem seu recurso.

Como apresentado na Seção 3.3, dado um intervalo de referência extraído do recurso, encontrou-se um comportamento idêntico já ocorrido no recurso em aproximadamente *88%* dos casos. Além disso, Begole, Tang e Hill (2003) também apresentaram resultados indicando a existência de repetições comportamentais ao longo do tempo.

Baseando-se o acima exposto, para realizar a previsão de duração de disponibilidade o AvSchedP busca no passado o comportamento semelhante ao que está ocorrendo momentaneamente no recurso, ou seja, o algoritmo captura como o usuário está se comportando ao utilizar o seu recurso e busca no histórico de utilização do recurso o intervalo de utilização mais similar ao comportamento atual.

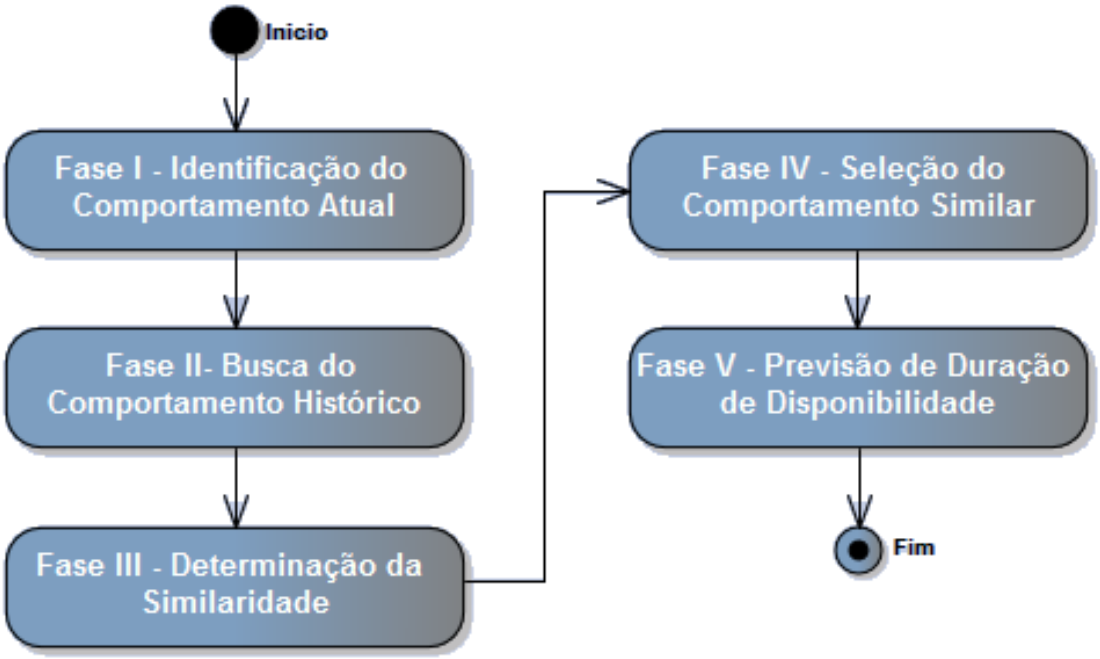

Figura 4.3: Fases do algorítmo de previsão de disponibilidade

Assim, para realizar a previsão de disponibilidade o AvShedP propoe um novo algoritmo. Este algoritmo é dividido em 5 etapas as quais executadas na ordem apresentada no diagrama de atividades da figura 4.3.

Entrada: Vetor binário v_a, Vetor binário v_b
Saída: Coeficiente *Hamann* $cHamann$ de Similaridade

1 **início**
2 /* *Geração da tabela de contingência (cT) dos vetores recebidos* */;
3 **para** $i \leftarrow 0$ **to** $(tamanho(v_a) - 1)$ **faça**
4 $cT[v_a[i]][v_b[i]] \leftarrow cT[v_a[i]][v_b[i]] + 1$;
5 **fim**
6 /* *Cálculo do coeficiente de similaridade* */;
7 /* *Utilizando o coeficiente de Hamann* */;
8 $cHamann \leftarrow \frac{(cT[0][0]+c[1][1])-(cT[0][1]+cT[1][0])}{(cT[0][0]+cT[1][1]+cT[0][1]+cT[1][0])}$
9 **retorna** $cHamann$;
10 **fim**

Figura 4.4: Algoritmo de determinação de similaridade

Como mostra a figura 4.3, o algoritmo inicia pela identificação do comportamento atual do recurso, para realizar a busca por comportamentos similares a a definição do comportamento com maior similaridade ao comportamento atual. Conhecendo o comportamento mais similar é realizada a previsão da duração de disponibilidade. A figura 4.5 exibe o algoritmo de previsão de duração de disponibilidade em pseudo-código.

As 5 fases do algoritmo de previsão de duração de disponibilidade são apresentadas em detalhes a seguir.

Entrada: $timestamp$ identificando o momento em que o recurso tornou-se disponível; m identifica o tamanho do vetor binário a ser recuperado; c número de dias do histórico.
Saída: **Tempo de duração** de disponibilidade previsto para o intervalo de disponibilidade

```
1  início
2      /* Fase I */;
3      va ← getCompAtual(timestamp, m);
4      /* Fase II */;
5      para i ← 0 to c faça
6          v_b ← getCompHst(timestamp, i, m);
7          /* Fase III */;
8          cfHam ← getCefHamann(v_a, v_b);
9          /* Fase IV */;
10         se (cfHam > cfMaxHam) então
11             maxHamI ← i;
12             cfMaxHam ← cfHam;
13         fim
14     fim
15     /* Fase V */;
16     v_b ← getHstSel(timestamp, maxHamI) ;
17     dur ← previsaoDeDuracao(v_b) ;
18     retorna dur ;
19 fim
```

Figura 4.5: Algoritmo de previsão disponibilidade

4.3.3.1 (Fase I) - Identificação do Comportamento Atual

Nesta fase o algoritmo identifica como o recurso estava sendo utilizado até o momento em que tornou-se ocioso. A identificação deste comportamento dá-se através da extração de um vetor de números binários, representando o comportamento atual do recurso.

Para realizar a extração desse vetor é necessário definir o tamanho do mesmo, ou seja, definir quantas unidades de tempo anteriores ao momento de ociosidade do recurso devem ser extraídas para formar o vetor.

O parâmetro m do modelo AvSchedP serve para determinar este tamanho. Este parâmetro deve ser ajustado ao dados contidos na base de históricos de utilização do recurso.

A determinação do valor deste parâmetro deve considerar 3 aspectos principais:

1. Qualidade da previsão: deve-se verificar a acurácia do preditor, variando-se o valor deste parâmetro. A alteração deste valor deve garantir a eficácia do algoritmo de previsão.

2. Identificação do comportamento dinâmico: a escolha de valores reduzidos podem levar a perda de identificação do comportamento de utilização do recurso. É importante observar que o usuário apresenta um comportamento dinâmico ao utilizar o

seu recurso. A captura deste comportamento dinâmico é importante na determinação de um comportamento similar ocorrido no passado.

3. Consumo de recursos computacionais: a escolha de valores elevados para este parâmetro implica no maior consumo de recursos computacionais. O valor de m determinará o tamanho dos vetores utilizados nas determinações de similaridade.

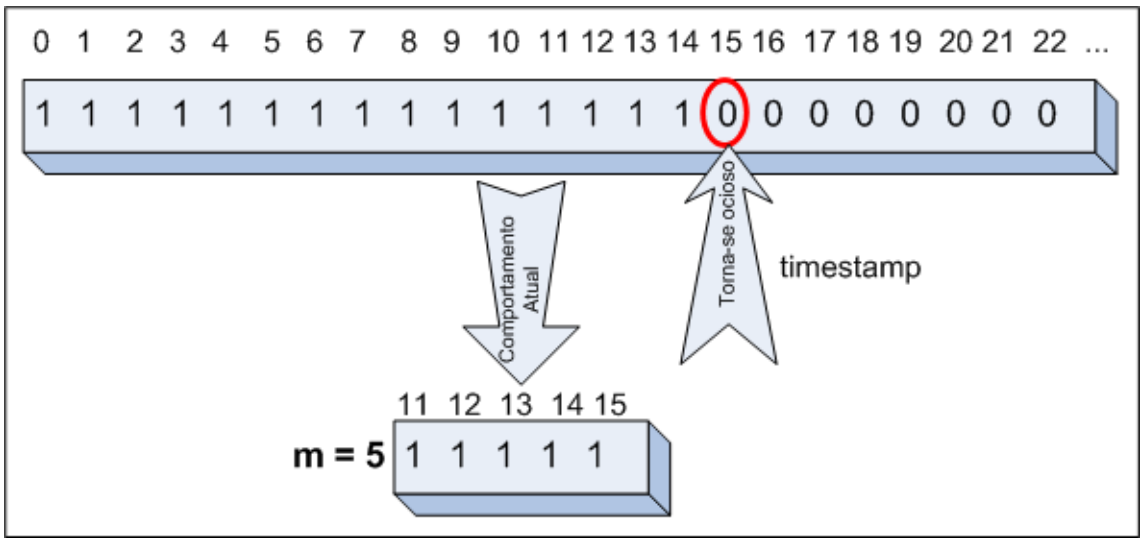

Figura 4.6: Extração do vetor de comportamento atual

O vetor extraído do comportamento atual do recurso é visto como uma sequência binária de utilização do recurso das últimas m unidades de tempo anteriores ao instante em que o recurso tornou-se ocioso. Este instante passará a ser referenciado como $timestamp$.

A extração do comportamento atual é realizado através da chamada à função $getCompAtual(timestamp, m)$ ($linha\ 3$) do algoritmo apresentado na figura 4.5.

Para exemplificar, supondo que a unidade de tempo configurada para o AvSchedP seja minutos, obter o comportamento atual do recurso com o parâmetro m configurado para 5 ($m = 5$) em um instante ($timestamp = 12:00:00$), corresponde a obter um vetor binário contendo 5 posições, uma posição para cada minuto, cada posição contendo o estado de utilização do recurso no instante ($timestamp - m + i$), onde i corresponde ao índice do vetor extraído. Resumidamente, este vetor contém a sequência binária dos 5 minutos anteriores a ocorrência do instante $timestamp$.

O processo de extração desse vetor é apresentado na figura 4.6 [2]. Como resultado desta fase obteve-se um vetor de tamanho m, que será utilizado como o vetor de referência nas próximas fases do algoritmo.

4.3.3.2 (Fase II) - Busca do Comportamento Histórico

A busca de comportamentos do histórico consiste em realizar uma busca na base de históricos de utilização do recurso, extraindo um vetor binário para cada histórico encontrado.

O vetor binário representando o histórico deve ter o mesmo valor especificado para o parâmetro m, e deve representar o mesmo intervalo de tempo do vetor de comportamento atual obtido na fase I.

[2] Exemplo com parâmetro m calibrado para 5 minutos.

A extração de cada comportamento histórico é realizada através da chamada à função $getCompHst(timestamp, i, m)$ do algoritmo ($linha\ 6$) apresentado na figura 4.5.

O número de vetores que devem ser recuperados da base de históricos é definido através do parâmetro (c). A determinação desse parâmetro é importante para o desempenho do algoritmo e também para a qualidade de previsão.

Na prática este parâmetro simboliza o tamanho do espaço amostral que deverá ser utilizado pelo algoritmo, e implica na seleção dos últimos (c) dias de utilização do recurso registrados na base de históricos. É importante observar que os vetores extraídos representam diferentes dias do passado.

A figura 4.7 apresenta a extração dos vetores da base de históricos de utilização de um recurso. Cada entrada na base de históricos, $Ch20$, $Ch19$, $Ch18$, $Ch17$ e $Ch16$ representa um dia de utilização do recurso persistido na base de históricos, sendo $Ch20$ o histórico mais recente enquanto $Ch16$ é o histórico mais antigo da base. Os vetores $ExCh20$, $ExCh19$, $ExCh18$ são os vetores resultantes da aplicação da Fase II, com os parâmetros $c = 3$, $m = 7$ e $timestamp = 12$.

4.3.3.3 (Fase III) - Determinação da Similaridade e (Fase IV)- Seleção do Comportamento Similar

Nesta fase o algoritmo determina a similaridade existente entre o vetor contendo comportamento atual[3] e os vetores recuperados da base de históricos de utilização[4].

Para calcular a similaridade, aplica-se o algoritmo de determinação de similaridade, o qual utiliza a equação de *Hamann* para tal fim. Como resultado, tem-se o coeficiente de similaridade entre o vetor de comportamento atual e cada vetor extraído da base de históricos de utilização.

A determinação da similaridade entre o vetor de comportamento atual v_a e cada comportamento histórico v_b recuperado é realizada através da chamada à função $getCefHamann(v_a, v_b)$ ($linha\ 8$) do algoritmo apresentado na figura 4.5. O algoritmo desta função foi apresentado anteriormente na figura 4.3.

Para realizar a previsão de duração de disponibilidade, o algoritmo necessita conhecer qual dos vetores recuperados da base de históricos é o mais similar ao vetor de referência.

Baseando-se no valor do coeficiente de similaridade obtido para cada comportamento, o algoritmo seleciona o vetor de comportamento histórico que obteve o maior coeficiente, ou seja, seleciona o comportamento mais similar ao comportamento atual, segundo a equação de *Hamman*. Nos casos de ocorrência de empate, o algoritmo opta pelo vetor de histórico mais atual entre os vetores empatados.

A identificação do histórico mais similar é realizada no algoritmo ($linhas$ 10,11,12) apresentado na figura 4.5.

A recuperação do comportamento mais similar ocorre com a chamada à função $getHstSel(timestamp, maxHamI)$ ($linha\ 16$) do algoritmo apresentado na figura 4.5.

O algoritmo consiste em localizar o comportamento mais similar ao comportamento recente. Pelas características empregadas no algoritmo, é possível identificar a capacidade de realizar previsões de forma dinâmica.

Cada execução do algoritmo em um instante ($timestamp$) diferente, pode levar o algoritmo a selecionar um dia diferente do histórico, visto que a escolha dependerá exclusivamente da similaridade entre os comportamentos.

[3] Vetor recuperado na fase I do algoritmo.

[4] Vetores extraídos na fase II do algoritmo.

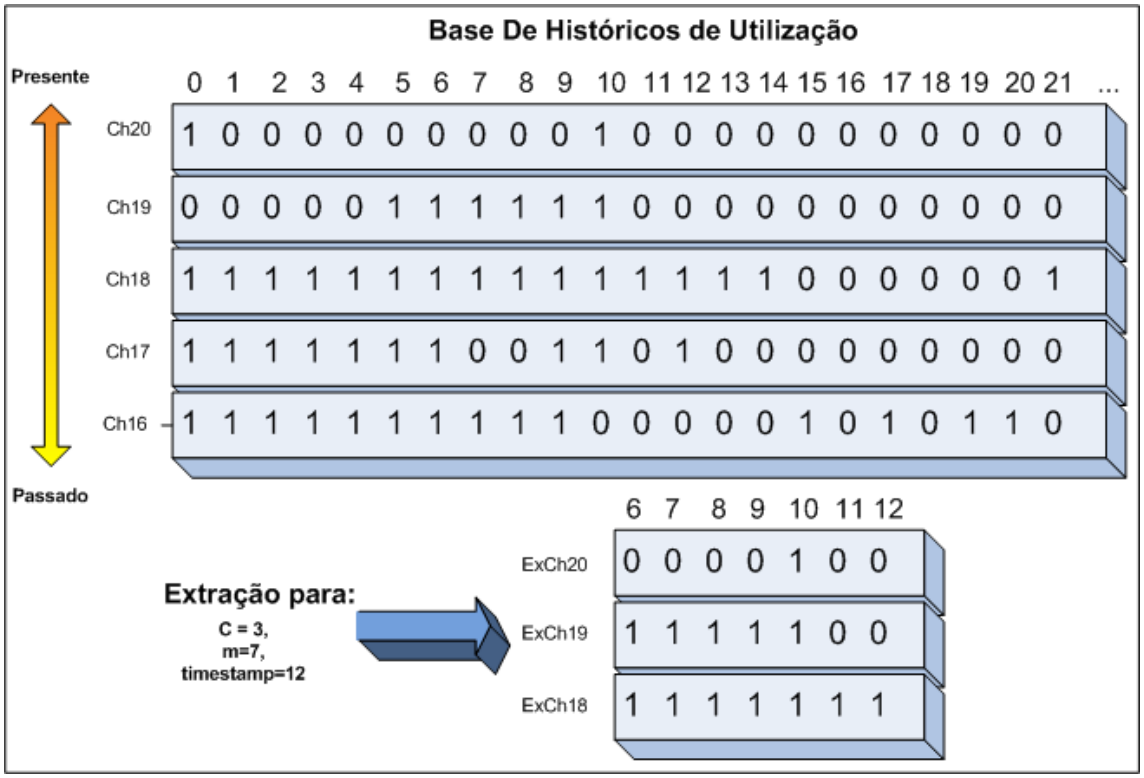

Figura 4.7: Extração dos vetores de comportamento da base de históricos

Observa-se também que o comportamento similar a ser encontrado, deve corresponder ao mesmo instante ($timestamp$) de tempo em que o recurso tornou-se disponível. Esta característica atribui ao algoritmo a capacidade de realizar diferentes previsões ao longo de um mesmo dia. Por exemplo, caso o usuário do recurso tenha o costume de ausentar-se do recurso às 12:30 horas, quando o algoritmo for executado neste instante, serão recuperados os históricos referentes ao mesmo horário, fazendo com que o intervalo de disponibilidade, comum na maioria dos dias, seja detectado pelo algoritmo.

A possibilidade de realizar a previsão no instante em que o recurso ocorre e a comparação com dados históricos do mesmo período faz do algoritmo de previsão proposto no AvSchedP, um algoritmo de previsão dinâmico. Assim, o AvSchedP também atende ao critério Dinâmico definido na Seção 3.3 deste trabalho.

O modelo também se mostra capaz de realizar previsões em caso de exceções comportamentais. Supondo que o usuário do recurso não utilize o recurso em um determinado dia e horário que sempre costuma utilizar, pois estava doente e não pode comparecer ao trabalho.

A não utilização do recurso neste horário estaria diferindo-se do padrão de utilização do recurso. Como o algoritmo realiza a busca por similaridade, seria recuperado do histórico um outro dia no qual o usuário não tenha utilizado o recurso naquele horário, como um outro dia que o usuário também faltou ao trabalho e deixou o recurso livre.

Este exemplo demonstra que o modelo proposto também considera o critério de exceções definido na Seção 3.3 deste trabalho. Esta é uma grande contribuição deste modelo, se comparado com os modelos apresentados nos trabalhos relacionados.

Modelos estáticos de previsão de disponibilidade seriam incapazes de realizar previsões para tal situação, visto que baseiam-se apenas em comportamentos que seguem um determinado padrão e não em suas exceções.

4.3.3.4 (Fase V) - Previsão de Duração de Disponibilidade

Após a definição do histórico similar, o algoritmo prossegue com a previsão do tempo de duração de disponibilidade.

Ao encontrar o comportamento histórico similar, o algoritmo assume que se o usuário comportou-se desta forma no passado, ele continuará se comportando assim no futuro.

Ao assumir esta hipótese, a previsão é realizada considerando o comportamento apresentado no histórico mais similar encontrado, a partir do instante $timestamp$.

Neste ponto o algoritmo determina a duração de disponibilidade identificando a quantidade de unidades de tempo que o recurso permaneceu ocioso no histórico, partindo do instante $timestamp$.

A quantidade de unidades de tempo que o recurso permaneceu consecutivamente ocioso no comportamento histórico (sequência de zeros (0)) corresponde à duração de disponibilidade prevista pelo algoritmo de previsão de duração de disponibilidade.

Com base em experimentos preliminares, observou-se que quanto maior for a duração do intervalo de disponibilidade a ser previsto, maior é a taxa de erros de previsão. Por esta razão, definiu-se o parâmetro f para o algoritmo, que corresponde ao tempo máximo de duração que pode ser previsto pelo algoritmo, após o instante $timestamp$. O valor deste parâmetro poderia ser fornecido pelo escalonador de tarefas, representando a medida de tempo necessário para executar uma tarefa existente na fila de tarefas do escalonador. Assim, o algoritmo poderia fornecer previsões adequadas ao tamanho das tarefas existentes na fila de execução.

A figura 4.8 apresenta o algoritmo da função de previsão de duração da disponibilidade. Esta função é parte integrante do algoritmo de previsão de disponibilidade do AvSchedP apresentado na figura 4.5 ($linha$ 17).

Entrada: Vetor de histórico selecionado v_b; Parâmetro f
Saída: Previsão de duração do intervalo de disponibilidade

```
1  início
2      /* Percorre o vetor de histórico recebido até encontrar 1 (ocupado) */;
3      para i ← timestamp to (timestamp + f) faça
4          se (v_b[i] == 0) então
5              dur ← dur + 1 ;
6          senão
7              retorna dur ;
8          fim
9      fim
10     retorna dur ;
11 fim
```

Figura 4.8: Algoritmo da função $previsaoDeDuracao(v_b)$

4.3.4 O Tratamento da Heterogeneidade dos Recursos

Como citado anteriormente, os recursos participantes de grades oportunistas são altamente heterogêneos. Esta diversidade impede que se assuma um tempo de execução uniforme para uma tarefa a ser executada em qualquer um dos recursos participantes, ou

seja, o tempo de execução de uma tarefa não é determinístico em um ambiente heterogêneo.

A medida de disponibilidade em unidade de tempo, como é fornecida pelo algoritmo do AvSchedP apresentado na figura 4.3, indica o tempo de disponibilidade que um recurso apresentará. No entanto, esta medida de tempo desconsidera a capacidade computacional do recurso, ou seja, o algoritmo do AvSchedP não considera a heterogeneidade dos recursos.

Para exemplificar, considerando-se que dois recursos, A e B, tornaram-se disponíveis em um mesmo horário ($timestamp$). Ambos executam o algoritmo de previsão de disponibilidade e obtêm como resultado 5 e 8 minutos de duração de disponibilidade, respectivamente.

O recurso B, seria a resposta correta para a pergunta - (**Qual dos recursos permanecerá mais tempo disponível?**). No entanto, esta resposta não seria a correta se a pergunta fosse - (**Qual dos recursos tem maior chance de completar a execução da tarefa Tx, cujo tempo de execução previsto é de 7 minutos?**).

A resposta a essa pergunta depende do tempo necessário para a tarefa ser completamente executada em cada recurso. O recurso A, mesmo que com menor tempo de disponibilidade previsto, pode ter maior chance de completar a execução da tarefa Tx do que o recurso B, caso A possua maior poder de processamento que B.

Com o objetivo de oferecer uma única medida comum a todos os recursos, o AvSchedP emprega a transformação de unidade de tempo para quantidade de processamento previsto.

Enquanto o tempo de duração de disponibilidade previsto fornece a informação do tempo pelo qual o recurso permanecerá disponível, a quantidade de processamento previsto fornece a quantidade de processamento que será realizada pelo recurso para o tempo de disponibilidade previsto.

A transformação da informação de tempo de disponibilidade em quantidade de processamento previsto, proposta no AvSchedP, é realizada em duas etapas:

- **Determinação da capacidade de processamento**: Nesta etapa o recurso realiza a estimativa de sua capacidade para uma unidade de tempo. Para obter esta medida cada recurso deve executar uma tarefa que explora a capacidade do recurso e apresenta como resultado a medida de desempenho médio para uma unidade de tempo daquele recurso. Para esta etapa utiliza-se, por exemplo, tarefas de um *Benchmark* sintético, como o *DhryStone* (YORK, 2002) ou o *Whetstone* (WEICKER, 1990).

  Esta tarefa seria disponibilizada pelo servidor da Grade Oportunista e seria executada por cada recurso participante, com o objetivo de determinar a sua capacidade de processamento.

  A medida de capacidade de processamento (cp), é então utilizada na transformação do tempo de duração de disponibilidade previsto para quantidade de processamento previsto.

- **Transformação**: A segunda etapa consiste em transformar a unidade de tempo proveniente do algoritmo, em quantidade de processamento considerando-se a capacidade de processamento (cp) estimada na etapa anterior para o recurso. A transformação entre as unidades segue a seguinte equação:

$$Qp = duracao * cp \tag{4.3}$$

$duracao$ corresponde ao tempo de disponibilidade previsto e cp é a capacidade de processamento estimada para o recurso.

É importante ressaltar que o AvSchedP fornece a informação de tempo de duração de disponibilidade como uma possibilidade de extensão a outras aplicações, que não só, escalonamento de tarefas para grades oportunistas.

Para um sistema de arquivos em sistemas *Peer-To-Peer* por exemplo, o tempo duração de disponibilidade é mais importante para as tomadas de decisão referentes a definição da localização de réplicas de arquivos, que a quantidade de processamento disponível em cada recurso.

Assim como no exemplo citado, para outros sistemas a informação de tempo de disponibilidade pode ser mais importante e por esta razão, o AvSchedP também fornece esta informação.

4.4 Caracterizando as Operações do AvSchedP

Esta seção apresenta uma visão geral sobre o AvSchedP inserido em um ambiente de grades oportunistas.

O modelo AvSchedP pode ser integrado a um ambiente de grade oportunista para oferecer as informações de disponibilidade ao escalonador de tarefas.

A participação de um recurso em uma grade oportunista inicia pela inscrição do recurso ao sistema de grade oportunista. Considerando-se a implementação do AvSchedP neste ambiente de grade, além do registro desse novo *Worker*, seria realizado o *download* da tarefa de *Benchmark* a ser executada pelo *Worker* com o propósito de determinar a capacidade de processamento do recurso.

Após a determinação da capacidade de processamento do recurso, o *Worker* está preparado para executar tarefas da grade oportunista, dependendo somente de tornar-se ocioso.

Ao tornar-se ocioso, o algoritmo de previsão de duração de disponibilidade entra em ação. A determinação da ociosidade de um recurso é de responsabilidade da política de recrutamento empregada no mesmo, como apresentado no capítulo 2.

O algoritmo recebe como entrada o $timestamp$ correspondente ao horário em que o recurso tornou-se ocioso e fornece como saída o tempo de duração de disponibilidade previsto, extraído da base de históricos de utilização do recurso. A informação de duração de disponibilidade é então transformada na capacidade de processamento prevista para o recurso.

A capacidade de processamento prevista é então enviada ao escalonador localizado no *Server* da Grade Oportunista. O escalonador de posse desta informação pode selecionar a tarefa a ser disponibilizada para o recurso.

A figura 4.9 apresenta um exemplo de execução do algoritmo de previsão proposto. [5]. Os triângulos numerados indicam os passos de execução do algoritmo.

O passo 1 consiste na identificação do instante $timestamp$ na base de históricos de utilização.

O passo 2 representa a extração do comportamento atual de tamanho m.

O passo 3 representa a extração de c vetores de tamanho m do histórico de utilização.

O passo 4 ilustra o valor do coeficiente de *Hamann* obtido para cada vetor de histórico.

[5] Exemplo com parâmetro m calibrado para 5 minutos e parâmetro c calibrado para 4 dias.

Por fim, o passo 5 exibe o resultado do algoritmo de previsão, que para este caso, identificou a existência de 6 unidades de tempo de ociosidade.

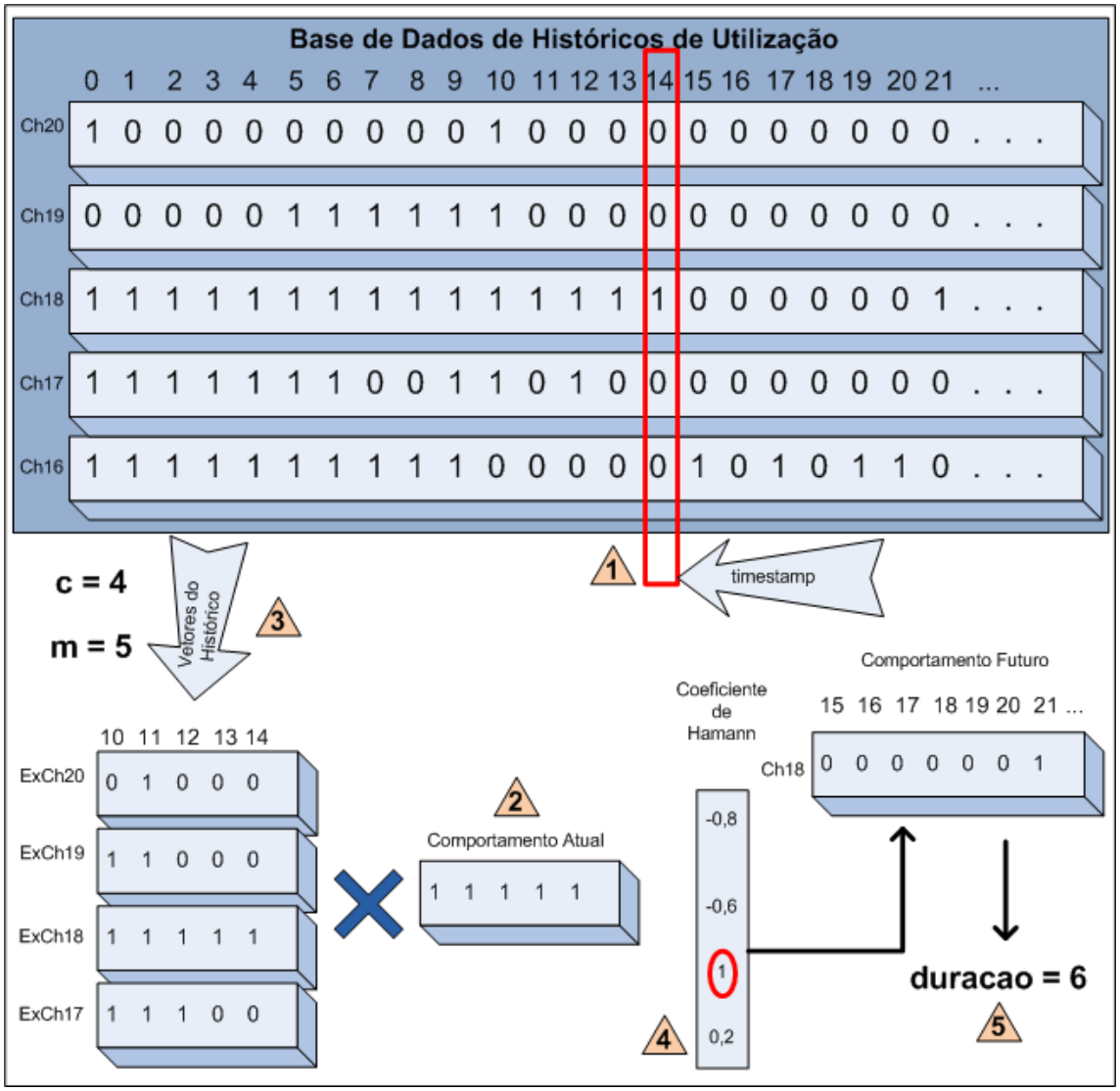

Figura 4.9: A execução do algoritmo de previsão de disponibilidade

4.5 Os Componentes do AvSchedP

O *AvSchedP* é formado por um conjunto de componentes que pode ser integrado ao *Worker* de um sistema de Grade Oportunista. Estes componentes fornecem a infraestrutura necessária para a realização da previsão de disponibilidade oferecida pelo AvSchedP. A figura 4.10 apresenta estes componentes.

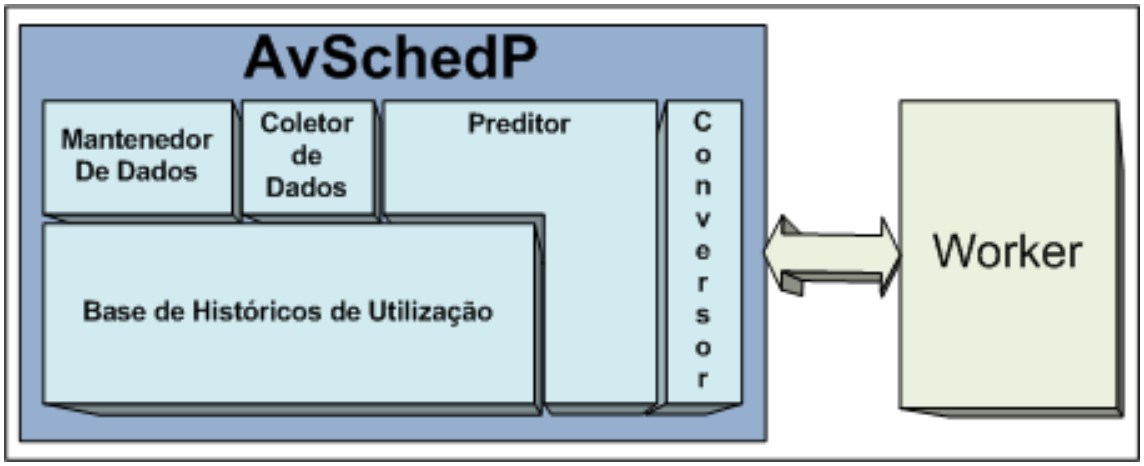

Figura 4.10: Arquitetura de componentes do AvSchedP

O componente **Base de Históricos de Utilização** é o responsável pela persistência e recuperação dos dados referentes ao histórico de utilização. Este componente responde a requisições realizadas pelos componentes **Mantenedor de Dados** e **Preditor**.

O **Coletor de Dados** é o componente responsável por interagir com o componente de monitoramento de utilização do recurso e converter a informação de utilização do recurso para os estados (0) e (1) utilizados pelo AvSchedP. Após a transformação destes dados, os mesmos são enviados ao **Mantenedor de Dados** que verifica se estes dados devem ser armazenados ou não.

O **Mantenedor de Dados** é o responsável pela organização dos dados que são persistidos. Ao receber uma requisição para armazenamento de dados proveniente do **Coletor de Dados**, o mantenedor consulta o componente de base de históricos para identificar se já existe algum histórico similar armazenado, conforme citado no início deste capítulo. Caso exista, é feito o descarte dos dados similares mais antigos.

O **Preditor** é o componente responsável pela realização das previsões de duração de disponibilidade. Ele interage com o monitor de utilização do *Worker* para identificar o momento em que uma previsão deve ser realizada. O preditor interage também com a base de históricos de utilização para extrair os vetores com os históricos de utilização e com o Coletor de Dados para recuperar o vetor contendo o comportamento atual do recurso.

Por fim, tem-se o componente **Conversor** que realiza a conversão do tempo de duração de disponibilidade prevista para a quantidade de processamento prevista.

Um ponto importante a ser observado é que tanto o **Coletor de Dados** quanto o **Preditor**, apesar de interagirem diretamente com o *Worker*, são independentes da política de recrutamento adotada pelo*Worker*. Essa estratégia permite a utilização do AvSchedP com sistemas empregando as mais diversas políticas de recrutamento.

Outro ponto importante a se observar é que a previsão de disponibilidade é descentralizada, ou seja, a previsão de disponibilidade ocorre no *Worker* e não no *Server*. Adotou-se a estratégia descentralizada para evitar a transferência de dados referentes ao histórico de utilização para o *Server*. Esta tarefa teria de ser realizada por cada *Worker* participante da grade oportunista, o que poderia gerar congestionamento nas redes de comunicação envolvidas. Outro problema seria o *overhead* imposto ao *Server*, devido a necessidade de executar o algoritmo de previsão para cada *Worker*.

4.6 Considerações Finais

O presente capítulo apresentou detalhadamente o modelo de previsão de disponibilidade para grades oportunistas, chamado AVSchedP, proposto neste trabalho.

O modelo é formado por um conjunto de componentes, onde o principal é o componente que realiza as previsões, chamado de Preditor.

Para atender as características identificadas no Capítulo 3, o algoritmo de previsão utiliza-se da equação de Hamann para determinar a similaridade entre sequências binárias selecionadas. Com esta abordagem, foi possível realizar a previsão de disponibilidade baseando-se apenas nos dois estados de disponibilidade definidos no Capítulo 3, disponível e indisponível.

Por ser aplicada a sequências de dados binárias, a equação de Hamann foi utilizada na determinação de similaridade entre dois intervalos de disponibilidade, sendo um o que representa o comportamento atual ocorrido no recurso e o outro, um comportamento extraído de uma base de históricos de disponibilidade.

O modelo proposto é aplicado de forma individual, ou seja, a cada recurso e não depende da existência de padrões de disponibilidade efetivos para realizar a previsão. Além disso, a previsão é realizada sempre no instante solicitado, o que torna o modelo de previsão dinâmico, sendo capaz de realizar previsões de disponibilidade diferenciadas conforme o instante em que é executada.

O próximo capítulo realiza a avaliação do modelo AVSchedP proposto. Inicialmente, são realizados experimentos para determinar a qualidade de previsão atingida pelo algoritmo de previsão, conforme a variação dos parâmetros existentes no modelo. Definidos os melhores valores para estes parâmetros, são realizados experimentos para avaliar a acurácia do algoritmo de previsão e a comparação dos resultados obtidos com os resultados obtidos pro um outro modelo proposto em um dos trabalhos relacionados. Por fim, avaliou-se a integração do modelo proposto com o sistema de Grade Oportunista, XtremWeb.

5 PROTÓTIPO, EXPERIMENTOS E RESULTADOS

Neste capítulo são apresentados os aspectos de implementação do modelo AvSchedP, a formatação dos experimentos realizados e a descrição dos resultados obtidos.

Os diferentes experimentos realizados serviram para avaliar os melhores valores para os parâmetros existentes no modelo e a qualidade das previsões de duração de disponibilidade realizadas com o AvSchedP.

Além de avaliar o ponto principal deste trabalho (previsão de disponibilidade), realizou-se também a experimentação e avaliação do modelo proposto quando integrado a um sistema de grade oportunista. O modelo foi integrado ao sistema *XtremWeb*, tanto no *worker* realizando previsões de duração de disponibilidade, quanto no *Server* auxiliando o escalonador de tarefas em suas tomadas de decisão.

Os resultados obtidos demonstram que o AvSchedP apresentou resultados interessantes nos diversos ambientes em que foi experimentado, sendo realmente um modelo que pode favorecer o escalonamento de tarefas em sistemas de Grades Oportunista.

5.1 Introdução

O modelo AvSchedP foi concebido com o propósito de realizar a previsão de duração de disponibilidade de recursos utilizados em grades oportunistas. O modelo é formado por um conjunto de componentes, descritos no Capítulo 4 deste trabalho, os quais cooperam para fornecer uma infraestrutura de previsão de disponibilidade.

Uma das principais contribuições do modelo AvSchedP é o algoritmo de previsão, sendo este o núcleo do componente **Preditor**. Para avaliar o componente **Preditor** uma série de experimentos foram realizados. Estes experimentos foram classificados em três tipos:

- **Avaliação de parâmetros do modelo**: experimentos realizados para avaliar a variação da taxa de acertos de previsão do **Preditor**, conforme a variação dos valores dos parâmetros c, m e f do modelo AvSchedP. Um acerto de previsão é considerado quando a duração de disponibilidade ocorrida no recurso é maior ou igual à duração de disponibilidade prevista pelo algoritmo. Rahman, Hassan e Buyya (2010) trataram o acerto de previsão da mesma forma como foi tratado neste trabalho;

- **Qualidade de previsão**: experimentos realizados para avaliar a qualidade das previsões realizadas pelo **Preditor**, sob os aspectos taxa de acertos, acurácia e efetividade apresentados ao longo deste capítulo. Nestes experimentos foi realizada a comparação entre os resultados obtidos com o AvSchedP e os obtidos com o modelo proposto por Rahman, Hassan e Buyya (2010). Para realizar a comparação, o

modelo proposto por Rahman, Hassan e Buyya (2010) também foi implementado neste trabalho, gerando o protótipo aqui chamado de *JaccardP*;

- **AvSchedP integrado à Grade Oportunista**: experimentos para avaliar a integração do AvSchedP ao sistema de Grade Oportunista *XtremWeb*.

Os resultados obtidos com a realização destes experimentos demonstraram que a previsão de disponibilidade realizada pelo AvSchedP tem qualidade considerável, justificando a sua utilização em sistemas de grades oportunistas.

Além disso, outros resultados interessantes foram obtidos com a integração do modelo AvSchedP ao sistema de grade oportunista *XtremWeb*. Neste caso, o fornecimento da informação de disponibilidade ao escalonador de tarefas demonstrou a possibilidade de se obter bons resultados, até mesmo superando um escalonador com política FCFS em alguns aspectos.

O protótipo, o detalhamento dos experimentos realizados e a análise dos resultados obtidos encontram-se detalhados a seguir.

5.2 O Protótipo

Para a realização dos experimentos, um protótipo do modelo AvSchedP foi implementado. Dos componentes existentes no modelo AvSchedP, foram implementados apenas o **Coletor de Dados** e o **Preditor**.

O componente **Coletor de Dados** foi desenvolvido para realizar a coleta de dados de utilização dos recursos pertencentes ao ambiente Dbcc, dados estes utilizados na análise de disponibilidade de recursos apresentada na Seção 3.3 do Capítulo 3.

O **Coletor de Dados** verifica a utilização do mouse ou teclado a cada minuto passado e persiste o estado de disponibilidade definido para aquele minuto em arquivos organizados por data.

Para identificar se houve a utilização de algum destes periféricos, o componente realiza chamadas de sistema específicas ao Sistema Operacional em que está instalado [1], consultando as alterações de estado ocorridas nestes periféricos.

O componente **Preditor** implementado realiza a previsão de duração de disponibilidade sobre um determinado conjunto de arquivos gerados pelo **Coletor de Dados**. O componente recebe como entrada o instante onde o recurso tornou-se ocioso e executa o algoritmo de previsão sobre os arquivos de histórico de utilização existentes, para realizar a previsão de duração de disponibilidade.

Para avaliar o modelo AvSchedP em grades oportunistas o componente **Preditor** foi integrado ao sistema *XtremWeb*. Optou-se por utilizar a linguagem de programação Java no desenvolvimento do protótipo, para facilitar a integração com o sistema de Grade Oportunista *XtremWeb*.

O protótipo resultante desta integração foi chamado de **XtremWebAv**.

A figura 5.1 apresenta os componentes do modelo AvSchedP integrados aos componentes do *XtremWeb Worker*. O *Worker* resultante desta integração foi chamado de **XtremWebAv Worker**. Na figura, os componentes marcados com linhas transversais são os já existentes na arquitetura de um *XtremWeb Worker*. Já os componentes marcados com barras horizontais e verticais, são componentes já existentes na arquitetura de um *XtremWeb Worker*, mas que sofreram alguma modificação. Por fim, os componentes sem barras são os novos componentes do *XtremWebAv Worker*.

[1] Os recursos participantes do ambiente Dbcc utilizam os Sistemas Operacionais Linux e Windows.

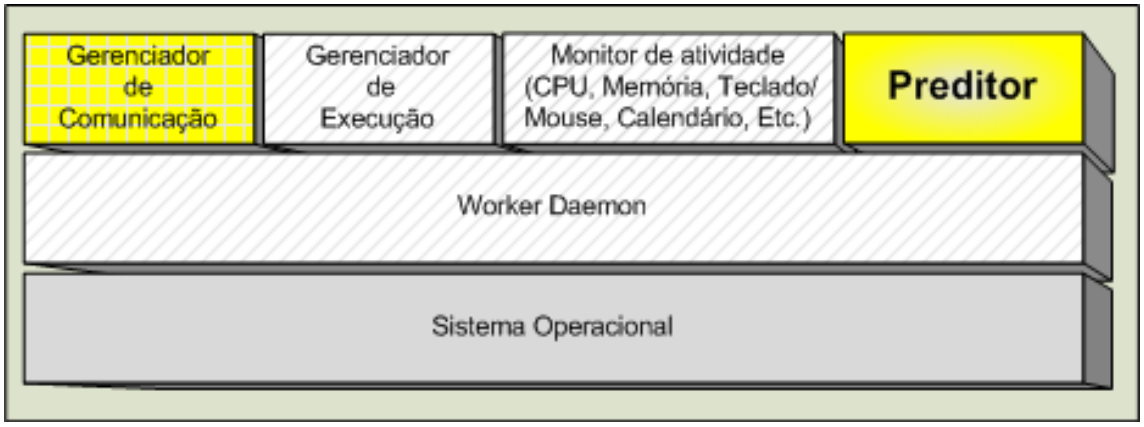

Figura 5.1: O *XtremWebAv Worker*

No *XtremWebAv Worker*, o *Worker Daemon* solicita a realização da previsão de disponibilidade ao componente **Preditor** sempre que o recurso torna-se disponível. O resultado da previsão é entregue ao *Worker Daemon* que aciona o componente **Gerenciador de Comunicação**, para realizar a requisição por tarefa.

O **Gerenciador de Comunicação** foi alterado para receber a duração de disponibilidade prevista e anexá-la à mensagem de requisição por tarefa, mensagem esta enviada ao *XtremWebAv Server*.

Alguns componentes do *XtremWeb Server* também foram alterados para possibilitar o recebimento e tratamento da informação de disponibilidade enviada pelo *Worker*. A figura 5.2 apresenta a arquitetura de componentes do *XtremWebAv Server*, arquitetura esta resultante da integração do modelo *AvSchedP* ao *XtremWeb Server*. Na figura, os componentes marcados com linhas transversais são os já existentes na arquitetura de um *XtremWeb Server*. Já os componentes marcados com barras horizontais e verticais, são componentes já existentes na arquitetura de um *XtremWeb Server*, mas que sofreram alguma modificação.

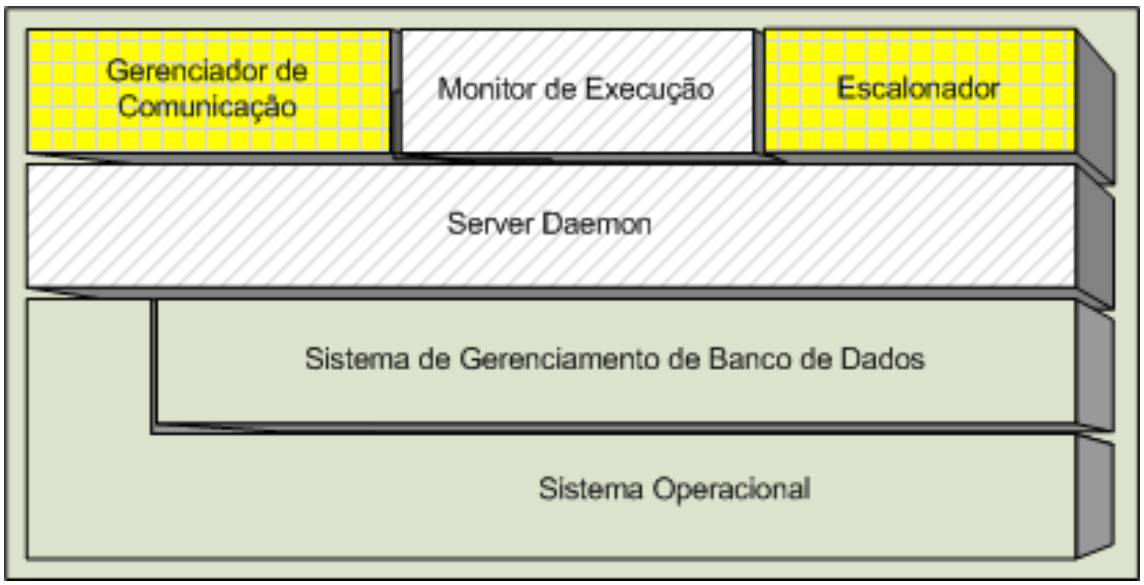

Figura 5.2: O *XtremWebAv Server*

Os componentes **Gerenciador de Comunicação** e **Escalonador** foram alterados para possibilitar a integração do AvSchedP ao sistema.

O componente **Gerenciador de Comunicação** foi alterado para possibilitar o recebimento da informação de duração de disponibilidade enviada pelo **XtremWebAv Worker** e encaminhá-la ao **Escalonador**. Já o **Escalonador** foi modificado para tomar suas decisões utilizando-se desta informação, tendo assim, sua política de escalonamento modificada.

5.2.1 A Política de Escalonamento do XtremWebAv Server

No *XtremWeb*, o escalonador utiliza a política *FCFS* no escalonamento de tarefas. O escalonador também se utiliza de outras informações nas tomadas de decisões, como as características dos recursos e os requisitos de execução de uma tarefa.

A política de escalonamento adotada no escalonador do *XtremWebAv Server* foi definida baseando-se nos resultados obtidos com a análise de disponibilidade realizada neste trabalho (Seção 3.3 do Capítulo 3).

A análise de disponibilidade realizada demonstrou que para o conjunto de dados utilizado, os intervalos de disponibilidade de maior duração ocorrem com menor frequência, quando comparada com a frequência dos intervalos de menor duração.

Esta conclusão permite assumir que quanto menor for o tempo necessário para executar completamente uma tarefa, maior será a chance desta tarefa ser executada por completo em uma grade oportunista. Esta constatação também foi realizada por Kondo, Chien e Casanova (2007).

Assim, decidiu-se considerar esta característica ao definir a política de escalonamento, priorizando a execução de tarefas maiores sempre que possível.

A política de escalonamento adotada no *XtremWebAv Server*, política esta baseada na duração de disponibilidade fornecida pelo *XtremWebAv Worker*[2] tem como propósito a seleção de tarefas.

Para realizar a seleção de tarefas, a política propõe que o escalonador percorra a fila de tarefas, consulte o tempo necessário para executar cada tarefa existente na fila, comparando-o com o tempo de duração de disponibilidade fornecido pelo requisitante da tarefa. **A primeira tarefa da fila, com tempo necessário de execução mais próximo e menor que o tempo de duração de disponibilidade previsto, é selecionada e entregue ao *Worker*.**

Nos casos onde o tempo de disponibilidade previsto é menor que o tempo necessário para executar a menor tarefa existente na fila, o escalonador seleciona a menor tarefa existente na fila e a entrega ao *Worker*.

Com esta abordagem, tarefas grandes serão selecionadas somente quando o tempo de duração de disponibilidade previsto no *Worker* solicitante for adequado para executar uma tarefa grande. Isso é importante para evitar o desperdício de tempo de processamento, ocasionado pela atribuição de tarefas que necessitam de tempo de execução maior que a duração de disponibilidade do recurso recebedor da tarefa.

Um outro aspecto importante desta política de escalonamento é sempre entregar alguma tarefa para execução mediante o recebimento de uma requisição, independente do tempo de duração de disponibilidade previsto pelo recurso solicitante. Com isso, a política de escalonamento adotada mantém a característica de sempre utilizar recursos disponíveis, como feito por uma política *FCFS*, além de aumentar a chance de acertos de escalonamento, ao entregar tarefas pequenas para recursos com baixa duração de disponibilidade prevista, visto que intervalos de disponibilidade menores são mais frequentes.

[2] A duração de disponibilidade é a informação resultante da previsão de disponibilidade executada pelo *Worker*.

Assim, **a política de escalonamento definida para os experimentos com o protótipo *XtremWebAv* tem como objetivo de escalonamento, reduzir o tempo desperdiçado na execução de tarefas canceladas devido à incompatibilidade entre o tempo necessário para concluir a execução da tarefa e o tempo disponível para executá-la em um recurso**.

Um exemplo de desperdício de tempo poderia ser observado na seguinte situação: um recurso dá início à execução de uma tarefa que necessitaria de 30 minutos para completar. Aos 25 minutos de execução desta tarefa o processamento é cancelado devido à utilização deste recurso pelo usuário. Nesse exemplo teria ocorrido o desperdício de 25 minutos de execução, visto que tudo o que foi executado até o momento do cancelamento da tarefa seria perdido.

O desperdício de tempo (*Wasted Time*) de processamento é frequentemente empregado na avaliação de escalonadores de grades computacionais e oportunistas, como realizado por Nadeen et al. (2008b) e Ballier (2005).

Considerando o exemplo acima, a política de escalonamento do *XtremWeb* selecionaria a primeira tarefa existente na fila, independente do tempo necessário para executá-la e independente da duração de disponibilidade do recurso, aumentando a chance de ocorrer o desperdício de tempo de processamento. Já a política de escalonamento adotada no *XtremWebAv* selecionaria a tarefa mais adequada ao tempo de duração de disponibilidade previsto pelo recurso ou a tarefa com o menor tempo de execução existente na fila caso o tempo de duração de disponibilidade previsto fosse menor que o tempo necessário para executar qualquer tarefa existente na fila, aumentando a chance de concluir a execução da tarefa e reduzir o desperdício de tempo de processamento.

É importante observar que com esta política, o escalonador não realiza seleção de recursos. Em outros trabalhos, como os realizados por Ren et al. (2006) e Taufer et al. (2005), a informação de disponibilidade recebida pelo escalonador de tarefas é utilizada para realizar a classificação e seleção de recursos, entregando tarefas apenas para recursos com boa classificação [3]. Neste tipo de abordagem, recursos com má classificação ficam sem receber tarefas para executar.

Para aproveitar ao máximo os recursos disponíveis, a política proposta realiza somente a seleção de tarefas. Para tanto, faz-se necessário conhecer previamente o tempo necessário para executar cada tarefa. Para obter essa informação, o *XtremWebAv* utiliza o tempo médio de execução de tarefas de uma aplicação, tempo este fornecido pelo próprio sistema *XtremWeb*. No entanto, o tempo fornecido não considera a heterogeneidade dos recursos existentes no ambiente.

O modelo AvSchedP propõe o componente **Conversor** para tratar da heterogeneidade dos recursos. Este componente não foi implementado no protótipo *XtremWebAv*. Optou-se então por alterar o cálculo do tempo médio fornecido pelo *XtremWeb* de modo a inserir as características computacionais do recurso nesta medida.

O cálculo foi alterado para fornecer o tempo médio de execução de tarefas de uma aplicação por *Worker*. O tempo médio de execução de tarefas de uma aplicação passou a ser calculado como o tempo médio de execução de tarefas que o próprio *Worker* obteve ao executar tarefas de uma determinada aplicação. Assim, cada *Worker* pode ter um tempo médio de execução de tarefas por aplicação, tempo este que pode variar conforme as características computacionais do recurso.

No entanto, esta medida não representa diretamente a capacidade computacional do recurso e sim, o tempo de execução de tarefas daquela aplicação em um determinado

[3] Boa classificação conforme os critérios adotados no modelo proposto por cada autor.

recurso. A inclusão de uma nova aplicação no servidor deixaria o escalonador de tarefas sem as informações necessárias para realizar o escalonamento, visto que qualquer recurso que solicitasse uma tarefa ainda não teria executado alguma tarefa desta aplicação e consequentemente, não teria disponível o tempo médio de execução de tarefas daquela aplicação. Como o escalonador se utiliza desta informação, o escalonamento de tarefas seria afetado. Esta situação enfatiza a utilização do componente **Conversor** do modelo AvSchedP. Com o uso deste componente, é fornecida uma medida representante da capacidade computacional do recurso, medida esta independente da aplicação existente no servidor, possibilitando a inclusão de novas aplicações sem afetar o escalonamento.

5.3 Experimentos e Resultados

A avaliação do modelo AvSchedP foi realizada de forma empírica. Foram utilizados dados resultantes do monitoramento de recursos pertencentes a 9 ambientes diferentes: Dbcc, PlanetLab, Overnet, Microsoft, DNS, Grid5000, Notre Dame, Web Sites e Seti@Home. Estes dados são os mesmos que foram utilizados na Seção 3.3 do Capítulo 3.

As próximas seções descrevem os experimentos realizados e apresentam a análise dos resultados obtidos em cada experimento realizado.

5.3.1 Avaliação dos Parâmetros do Modelo AvSchedP

No Capítulo 4 os parâmetros m, c e f do modelo AvSchedP foram apresentados e discutidos detalhadamente. No entanto, cabe revisitar suas definições. O parâmetro m define quantas unidades de tempo (Ex: minutos) serão utilizadas na extração dos vetores de comportamento que serão utilizados pelo algoritmo do modelo para quantificar a similaridade do comportamento atual com algum comportamento do passado (histórico). O parâmetro c define o número de períodos do passado (Ex: dias) que devem ser recuperados da base de históricos para comparação de similaridade com o comportamento atual. Por fim, o parâmetro f corresponde ao tempo máximo de duração que pode ser previsto pelo algoritmo, após o instante em que o recurso torna-se ocioso.

O ajuste do valor destes parâmetros é importante para o modelo, visto que estes afetam diretamente a qualidade das previsões realizadas pelo algoritmo de previsão e a quantidade de recursos computacionais necessários para realizar a previsão.

Ao examinar detalhadamente o algoritmo proposto no AvSchedP é possível identificar que quanto maior for o valor definido para os parâmetros m, c e f, maior será a quantidade de recursos computacionais consumidos com a execução do algoritmo.

Nesta seção, foram realizados experimentos para avaliar o impacto na taxa de acertos de previsões de disponibilidade fornecidas pelo componente **Preditor** conforme o valor de entrada dos parâmetros do modelo. Esta seção tem como objetivo, fornecer respostas para as questões abaixo:

1. O uso de valores maiores para os parâmetros incorre no aumento da taxa de acertos de previsão ?

2. Os valores dos parâmetros devem variar conforme o ambiente selecionado ou a especificação de um único valor forneceria boas previsões para todos os ambientes?

Para fornecer respostas a estes questionamentos, os experimentos foram conduzidos em rodadas. Para cada rodada, os valores dos parâmetros m, c e f são definidos, um

instante de tempo para realizar a previsão é escolhido e o algoritmo é executado para cada recurso de cada ambiente utilizado no experimento. O resultado de cada rodada é a média da taxa de acertos obtida em cada ambiente experimentado.

A variação dos valores de cada parâmetro seguiu a respectiva faixa de valores apresentada na tabela 5.1.

Tabela 5.1: Faixa de valores dos parâmetros c, m e f

Parâmetro	Valor Inicial	Valor Final	Unidade
c	10	90	% de arquivos de histórico
m	10	360	Minutos
f	10	360	Minutos

O resultado desse experimento pode apresentar variações na taxa de acertos obtida, conforme o instante de tempo fornecido ao algoritmo. Para evitar a obtenção de resultados distorcidos, optou-se por executar uma mesma rodada sobre quinze instantes de tempo diferentes. Estes quinze instantes diferentes são escolhidos aleatoriamente pelo algoritmo desenvolvido para realizar estes experimentos. Os valores 505, 543, 608, 617, 653, 767, 822, 848, 849, 856, 930, 952, 991, 1026 e 1066 são exemplos de instantes avaliados em uma rodada. Cada valor equivale ao minuto do dia, como por exemplo, 505 equivale às 08 horas e 41 minutos de um dia.

Para avaliar se quinze instantes diferentes seria um valor suficientemente confiável, foram executadas rodadas sobre cinco, dez, quinze, vinte e cinco, trinta e quarenta instantes diferentes. A tabela 5.2 apresenta a taxa média de acertos obtida com cada número de instantes aplicados. Como pode ser observado, o número de instantes selecionados não apresenta uma relação direta com a taxa de acertos obtida, ou seja, o aumento do número de instantes não apresentou tendência significativa nos resultados, possibilitando a escolha de qualquer número de instantes para realizar os experimentos, sem causar distorções significativas nos dados. Desta forma, optou-se por utilizar quinze instantes por ser um número intermediário de execuções e que gera uma quantidade de dados mais fácil de ser analisada.

Tabela 5.2: Taxa de acertos por número de instantes utilizados por rodada

Número de Instantes	% Acertos
5	75.23
10	74.77
15	75.22
25	75.61
30	75.95
40	75.96

As taxas de acertos de previsão obtidas são apresentadas e discutidas abaixo.

5.3.1.1 Resultados

Para facilitar a avaliação dos resultados obtidos com a variação dos valores dos parâmetros c, m e f, foram gerados gráficos apresentando a taxa de acertos de previsão de disponibilidade obtida com a variação de cada parâmetro.

Os gráficos das figuras 5.3, 5.4 e 5.5 apresentam o resultado obtido de forma coletiva, ou seja, a taxa de acertos obtida para todos os ambientes experimentados de forma agrupada.

A figura 5.3 apresenta o gráfico de média de acertos do parâmetro c (Coletivo).

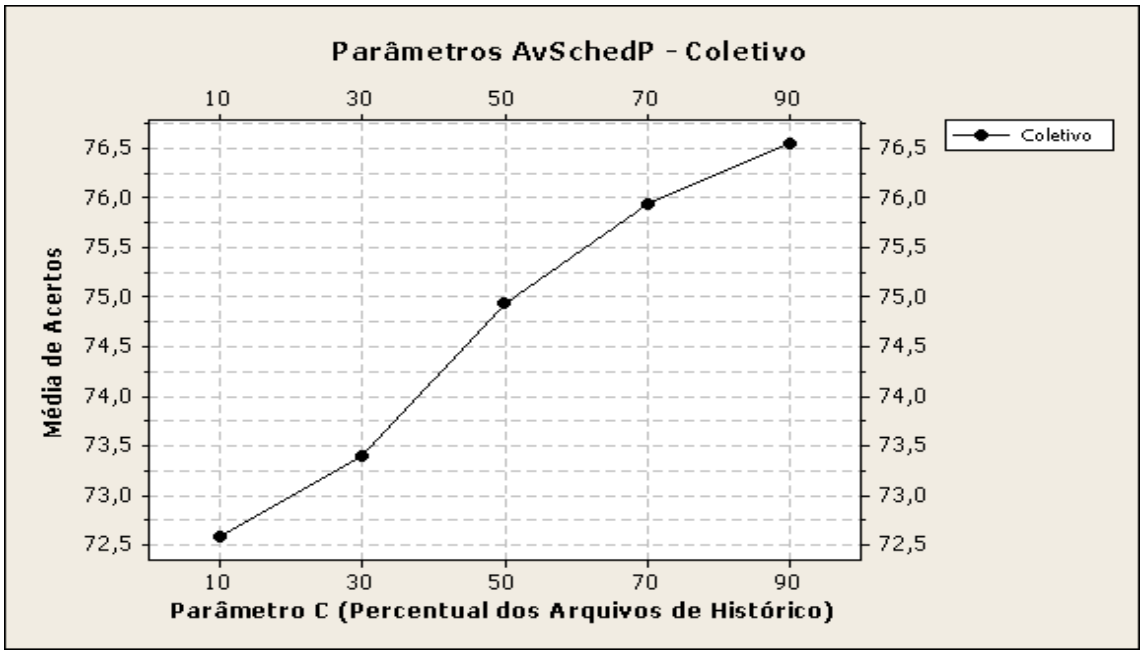

Figura 5.3: Avaliação do parâmetro c - Coletivo

Os resultados obtidos com a variação do parâmetro c demonstram o aumento da taxa de acertos de previsão de disponibilidade, à medida que o valor do parâmetro c aumenta. É importante observar que cada taxa de acertos apresentada para o parâmetro c é formada pela taxa de acertos média obtida com a variação dos valores dos parâmetros m e f do modelo.

Com este resultado pode-se dizer que ao aumentar o número de arquivos de histórico utilizados na identificação do comportamento mais similar ao comportamento atual do recurso, aumenta-se também a taxa de acertos de previsões realizadas pelo algoritmo de previsão de disponibilidade.

Este aumento na taxa de acertos ocorre em virtude do aumento da diversidade de comportamentos utilizados pelo algoritmo durante a realização da previsão, permitindo a seleção de um comportamento que retrate com maior fidelidade o comportamento futuro deste recurso.

No entanto, deve-se observar que ao utilizar a totalidade de arquivos de histórico aumenta-se o espaço de busca do algoritmo, sendo exigido mais recursos computacionais para a execução do algoritmo. Neste ponto, é importante ressaltar que o algoritmo calcula a similaridade entre o comportamento atual e cada arquivo de histórico considerado. Logo, ao aumentar o valor do parâmetro c, mais arquivos de histórico são considerados, aumentando o consumo de recursos computacionais necessários para a execução do algoritmo.

Avaliando-se os resultados observa-se que a variação no valor deste parâmetro, desde seu valor mínimo até o seu valor máximo (72,6 a 76,6), apresentou um ganho médio de quatro pontos percentuais.

Esta pequena variação no resultado obtido possibilita a configuração do parâmetro c conforme a necessidade do ambiente em que o AvSchedP estiver inserido. Se o ambiente exigir maior qualidade na informação de previsão de duração de disponibilidade fornecida em detrimento ao consumo de recursos necessários para a execução do algoritmo, pode-se estabelecer valores altos para o parâmetro c. Em contrapartida, caso o ambiente exija o menor consumo de recursos, o parâmetro pode receber um valor menor e mesmo assim fornecer previsões com nível de qualidade considerável.

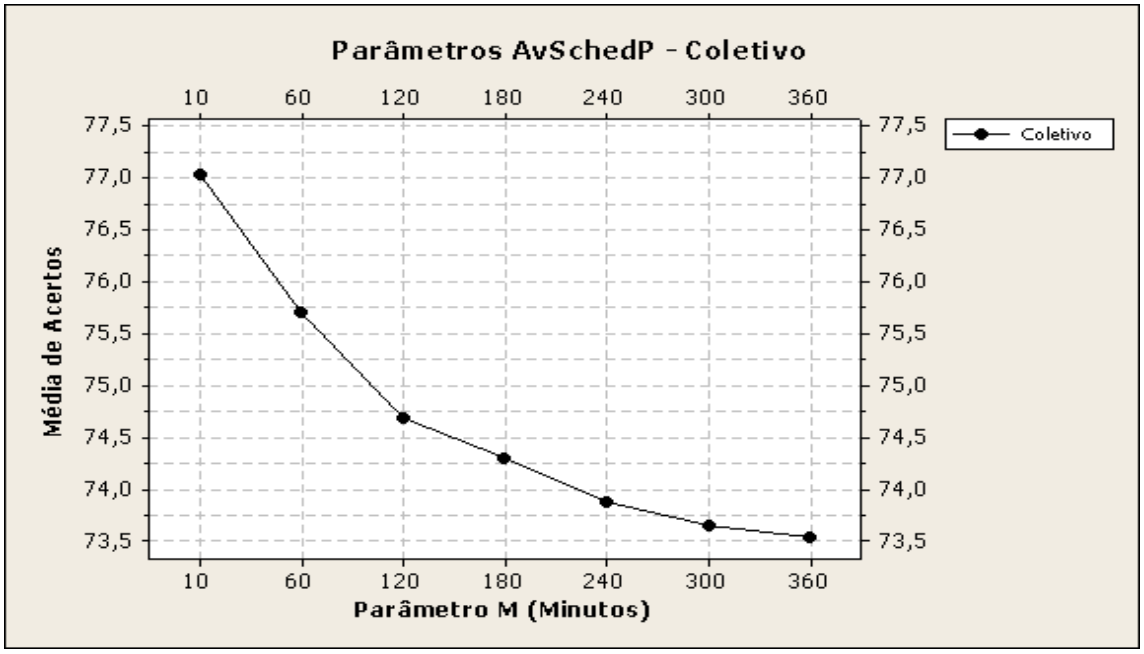

Figura 5.4: Avaliação do parâmetro m - Coletivo

Prosseguiu-se com a avaliação dos resultados obtidos com a variação dos demais parâmetros. O parâmetro m, aquele que determina o tamanho da sequência binária que será utilizada pelo algoritmo de previsão para determinar a similaridade, foi então avaliado.

A figura 5.4 apresenta o gráfico de taxas de acertos de previsão obtidas com a variação dos valores do parâmetro m. É importante observar que cada taxa de acertos apresentada para o parâmetro m é formada pela taxa de acertos média obtida com a variação dos valores dos parâmetros c e f do modelo.

O gráfico demonstra que a variação do valor do parâmetro m eleva o número de erros de previsão de disponibilidade à medida que ocorre o aumento do valor do parâmetro, ou seja, quanto maior for o valor de m, maior será o número de erros de previsão do algoritmo.

Analisando o algoritmo, isto significa que a utilização de sequências de dados binários maiores na comparação de similaridade faz com que o algoritmo selecione comportamentos que não refletem da melhor forma o comportamento futuro do recurso.

Este resultado vem a favorecer o algoritmo de previsão proposto no modelo AvSchedP, no que diz respeito a consumo de recursos computacionais. Com a utilização de pequenas sequências binárias extraídas do comportamento atual e do histórico, o algoritmo consegue obter boas taxas de acertos de previsão. Este resultado favorece o desempenho do algoritmo, pois a previsão de duração de disponibilidade pode ser realizada com a comparação de pequenos vetores de comportamento, consumindo menos recursos computacionais.

Assim como o parâmetro m, os resultados apresentados pelo parâmetro f forneceram conclusões semelhantes, favorecendo o algoritmo de previsão de disponibilidade no que diz respeito ao consumo de recursos computacionais.

A figura 5.5 apresenta o gráfico de taxas de acertos obtidas com a variação do valor do parâmetro f. É importante observar que cada taxa de acertos apresentada para o parâmetro f é formada pela taxa de acertos média obtida com a variação dos valores dos parâmetros m e c do modelo.

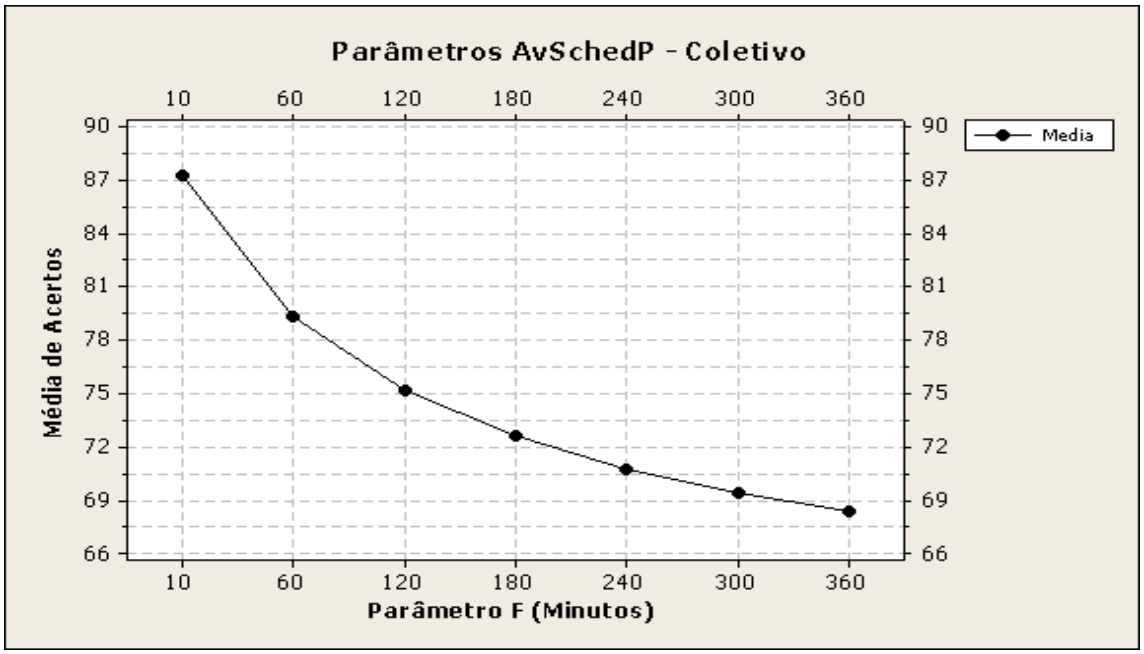

Figura 5.5: Avaliação do parâmetro F - Coletivo

A variação do valor do parâmetro f demonstrou a ocorrência de redução da taxa de acertos de previsão à medida em que aumenta-se o valor do parâmetro. Como o parâmetro f é utilizado pelo algoritmo de previsão como um limitador do futuro de predição (quantidade de unidades de tempo a serem previstas), este resultado demonstra que quanto maior for o tempo de duração de disponibilidade a ser previsto, menor será a taxa de acertos de previsão.

Com estes experimentos, avaliou-se o efeito da variação dos parâmetros sobre a qualidade de previsão do algoritmo proposto. Os resultados obtidos fornecem subsídios preliminares para responder a questão 1 levantada nesta seção. Foi possível identificar que o aumento do valor dos parâmetros não significa necessariamente que o algoritmo produzirá melhores resultados. Identificou-se sim que apenas o parâmetro c requer valores altos para oferecer melhores previsões, enquanto os parâmetros m e f demonstraram justamente o contrário.

Porém, esta avaliação demostra os resultados de forma coletiva, o que não permite responder por completo a questão 1 tampouco a questão 2. Optou-se então por verificar se estes resultados são semelhantes aos obtidos em cada ambiente experimentado.

Os gráficos apresentados nas figuras 5.6, 5.7 e 5.8 apresentam os resultados obtidos com a variação dos valores dos parâmetros, porém agora por ambiente.

A figura 5.6 apresenta a taxa de acertos de previsão de disponibilidade por ambiente, variando-se o valor do parâmetro c.

Como pode ser observado, os resultados obtidos para cada ambiente demonstraram que o aumento do valor do parâmetro c levou ao aumento na taxa de acertos do algoritmo de previsão.

Observa-se, no entanto, que nos ambientes DNS, Grid5000 e Overnet a taxa de acertos caiu quando o valor de c aumentou de 10% para 30%. Atribui-se esta queda à escolha do arquivo de comportamento realizada pelo algoritmo, ou seja, ao acaso.

Apesar do aumento no número de comportamentos históricos, a escolha do histórico utilizado na previsão depende da similaridade com o comportamento ocorrido. No entanto, mesmo sendo o comportamento mais similar não existe nenhuma garantia de que este comportamento histórico selecionado permanecerá similar ao comportamento que ocorrerá no recurso. Por esta razão, estas situações realmente podem ocorrer com a utilização deste e qualquer outro algoritmo de previsão.

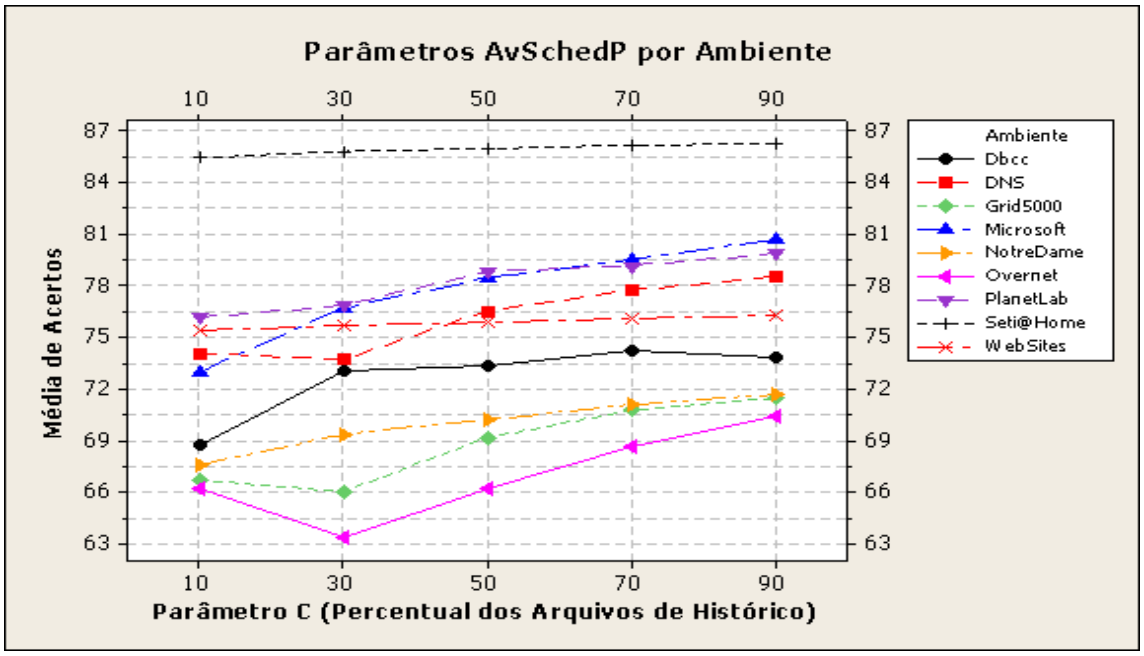

Figura 5.6: Avaliação do parâmetro c por ambiente

Apesar destas pequenas alterações, pode-se assumir que existe sim a tendência a ocorrer o aumento na taxa de acertos de previsão à medida em que ocorre o aumento do valor do parâmetro c. Além deste constatação, é possível assumir também que o resultado obtido em cada ambiente é muito similar ao obtido coletivamente.

Os gráficos das figuras 5.7 e 5.8 apresentam a taxa de acertos por ambiente obtida com a variação dos valores dos parâmetro m e f respectivamente.

Os resultados obtidos também foram extremamente semelhantes aos obtidos na avaliação coletiva. Pode observar que em qualquer um dos ambientes experimentados, ocorreu a redução da taxa de acertos de previsão à medida em que o valor destes parâmetros foi aumentado.

Com os resultados obtidos na análise da variação dos valores dos parâmetros c, m e f foi possível identificar que as taxas de acertos obtidas apresentaram valores semelhantes tanto quando analisadas coletivamente, quanto quando analisadas separadamente por ambiente.

Estes resultados possibilitam responder as questões 1 e 2 identificadas nesta seção. Referente a questão 1, pode-se assumir que a utilização de valores elevados não incorre na

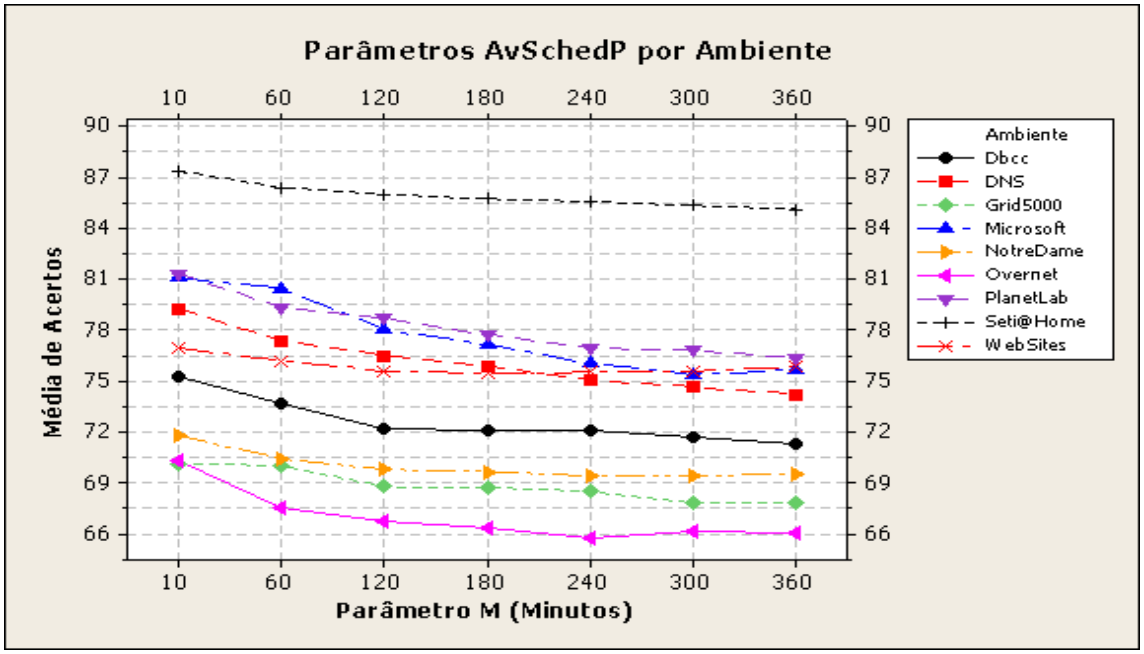

Figura 5.7: Avaliação do parâmetro m por ambiente

melhoria da qualidade de previsão. Esta resposta foi constatada tanto quando avaliou-se os resultados de forma coletiva, quanto quando avaliou-se os resultados de forma individual. Nos resultados por ambiente, em nenhum dos ambientes experimentados o resultado da previsão diferenciou-se significativamente do resultado obtido na avaliação coletiva.

Esta constatação também fornece a resposta à questão 2. Os resultados apresentados demonstraram a possibilidade de assumir valores únicos para os valores dos parâmetros, em qualquer um dos ambientes experimentados.

Como os ambientes experimentados são reais e são originados de ambientes com perfis distintos, é possível assumir a utilização dos mesmo valores em qualquer ambiente que venha a ser utilizado.

5.3.2 Qualidade de Previsão do Modelo AvSchedP

A avaliação de desempenho deve lidar com as incertezas existentes no mundo real. Essas incertezas introduzem erros nas medições obtidas, podendo induzir o pesquisador a conclusões incorretas sobre o objeto de estudo.

O AvSchedP realiza a previsão da duração de disponibilidade em recursos computacionais, para auxiliar escalonadores em suas tomadas de decisão. A incerteza introduzida nessa medida leva a ocorrência de erros de escalonamento. Por esta razão faz-se necessário quantificar esta incerteza, ou seja, avaliar a qualidade das previsões fornecidas pelo modelo proposto.

Para auxiliar na identificação da incerteza inserida nos resultados obtidos nos experimentos, costuma-se utilizar ferramentas da probabilidade e estatística. Em geral, a avaliação da qualidade dos resultados de experimentos que envolvem medidas de tempo é realizada considerando-se a acurácia e a precisão dos resultados obtidos com alguma unidade de referência (LILJA, 2000).

Para avaliar a qualidade de previsão do modelo AvSchedP três métricas foram utilizadas, a taxa de acertos de previsão, a acurácia da previsão, e a efetividade da previsão. Estas métricas são detalhas abaixo.

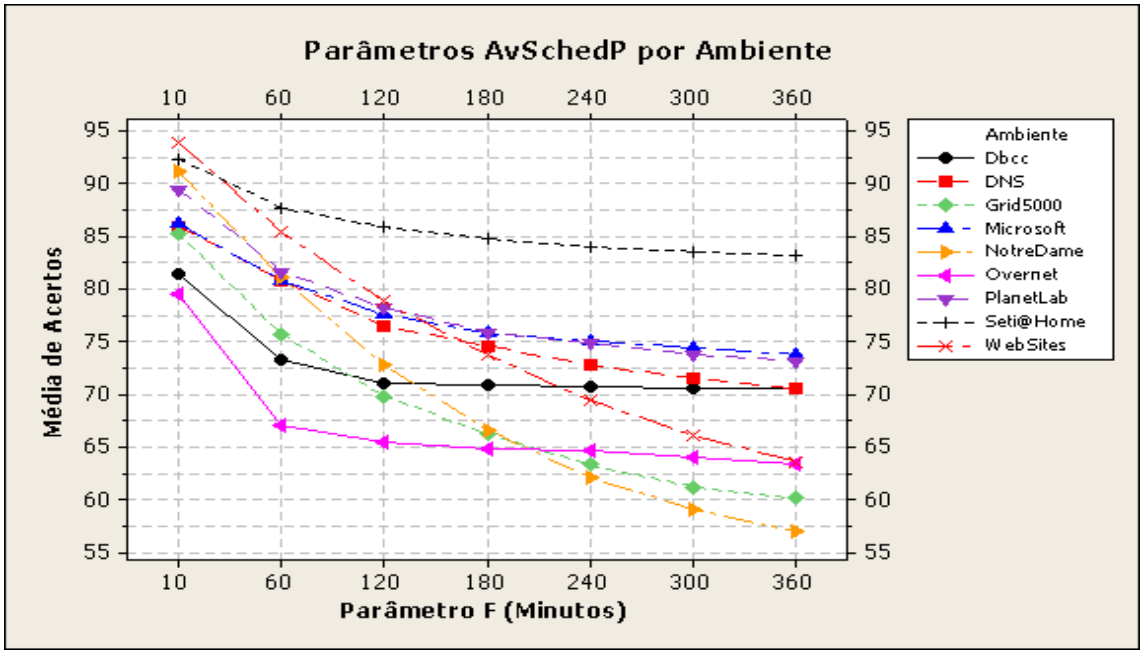

Figura 5.8: Avaliação do parâmetro f por ambiente

- **Taxa de Acertos** ($\overline{TxA}$): esta métrica apresenta a média de acertos de previsão do algoritmo. Um acerto de previsão (act) é considerado quando a duração de disponibilidade real for maior ou igual à duração de disponibilidade prevista. A equação 5.1 apresentada abaixo foi utilizada para determinar se uma previsão realizada trata-se de um acerto ou um erro.

$$act(D_{prev}, D_{real}) = \begin{cases} 1, & seD_{prev} <= D_{real} \\ 0, & seD_{prev} > D_{real} \end{cases} \tag{5.1}$$

  Esta medida foi amplamente utilizada por Rahman, Hassan e Buyya (2010), Nadeen et al. (2008b) e Ren et al. (2007) para avaliar a qualidade das previsões de disponibilidade de recursos.

  O cálculo da taxa de acertos não considera os casos onde a previsão realizada teve resultado igual a zero, ou seja, foram consideradas apenas previsões com tempo de duração maior que zero. Como este trabalho está inserido no contexto de escalonamento de tarefas, sempre que o tempo previsto for menor que o realizado, foi considerado um acerto pois o tempo previsto realmente ocorreu. Já nos casos de previsão com duração igual a zero, optou-se por desconsiderar para não tratar como um acerto de previsão, visto que o tempo previsto teria ocorrido e acabava elevando consideravelmente a taxa média de acertos.

- **Acurácia** ($\overline{Acc}$): esta métrica apresenta a acurácia média do algoritmo. Em uma pesquisa faz-se necessário o estabelecimento de medidas de confiabilidade sobre os resultados obtidos. Uma dessas medidas é a acurácia, que representa a proximidade do resultado obtido em relação à realidade (PEREIRA, 2004).

  O objetivo desta medida é quantificar o quão próxima foi a duração prevista D_{prev}, da duração que realmente ocorreu no recurso D_{real}.

Outros autores como Rahman, Hassan e Buyya (2010), Nadeen et al. (2008b) e Ren et al. (2007) também avaliaram a acurácia das previsões de disponibilidade utilizando uma equação semelhante a utilizada neste trabalho.

Mediu-se a acurácia média de previsão do modelo, obtida com a média da taxa de proximidade prx do tempo de duração previsto(D_{prev}) em relação ao tempo de duração real(D_{real}). A acurácia foi calculada considerando-se tanto previsões com duração maior que a realizada no recurso, quanto previsões com duração menor que a realizada no recurso.

Para os casos onde a duração prevista é menor que a realizada, a taxa de proximidade de cada previsão foi calculada com a equação 5.2.

$$prx(D_{prev}, D_{real}) = \frac{D_{prev}}{D_{real}} \tag{5.2}$$

No entanto, quando a duração prevista for maior que a duração realizada, o resultado deve ser tratado. Neste casos, dua situações devem ser consideradas: (i) quando a duração prevista obtida pelo preditor for até uma vez maior que a realizada; (ii) quando a duração prevista for duas ou mais vezes maior que a duração realizada.

No primeiro caso, se o preditor obter uma previsão de 25 minutos e a duração realizada no recurso for de 20 minutos, o resultado apresentado pela equação 5.2 seria 1,25. Este resultado deve ser interpretado como um erro de 25 décimos a maior. Então, neste caso, a proximidade entre o previsto e o realizado foi de 0,75. Para tratar esta situação, quando o valor previsto é maior que o realizado, aplica-se a equação 5.3, ou seja, utiliza-se a diferença entre o previsto e o realizado para obter a taxa de proximidade.

No segundo caso, o resultado da equação 5.3 seria negativo. Por exemplo, se a duração prevista fosse de 45 minutos e a duração realizada fosse de 20 minutos, a equação 5.3 forneceria um resultado negativo de -0,25. Para esta situação, definiu-se a taxa de proximidade da previsão como zero $(prox = 0)$. Como a menor duração prevista que pode ser fornecida pelo preditor é zero, nunca negativa, entendeu-se também que a proximidade entre as medidas duração realizada e prevista também não poderia ser negativa. Além disso, esta decisão foi tomada para que os valores negativos não afetem na média calculada.

$$prx(D_{prev}, D_{real}) = 1 - \frac{(D_{prev} - D_{real})}{D_{real}} \tag{5.3}$$

Assim, a proximidade entre a duração de disponibilidade prevista por um preditor e a duração de disponibilidade realizada no recurso foi dada pela função abaixo:

$$prx(D_{prev}, D_{real}) = \begin{cases} \frac{D_{prev}}{D_{real}}, & se D_{prev} < D_{real} \\ 1 - \frac{D_{prev} - D_{real}}{D_{real}}, & se D_{real} < D_{prev} < D_{real} * 2 \\ 0, & se D_{prev} > D_{real} * 2 \end{cases} \tag{5.4}$$

A acurácia média é então obtida com a aplicação da equação 5.5.

$$\overline{Acc} = \frac{\sum_{n=1}^{N} prx(D_{prev(n)}, D_{real(n)})}{N} \tag{5.5}$$

- **Efetividade ($\overline{Ef}$)**: esta métrica indica o quanto efetivo é o algoritmo de previsão, e é dada pelo produto das medidas $\overline{TxA}$ e $\overline{Acc}$, como apresentado na equação 5.6. A efetividade permite comparar dois algoritmos de previsão, considerando os aspectos número de acertos e acurácia;

$$\overline{Ef} = \overline{TxA} * \overline{Acc} \tag{5.6}$$

Para tornar mais abrangente esta avaliação, optou-se por comparar o modelo proposto com o modelo proposto por Rahman, Hassan e Buyya (2010), modelo este apresentado no Capítulo 3 na Seção de trabalhos relacionados (3.4). Como citado anteriormente, o protótipo resultante desta implementação foi chamado de *JaccardP*.

Optou-se pela seleção deste modelo pois o mesmo fora comparado com outros três modelos de previsão, apresentando melhores resultados (RAHMAN; HASSAN; BUYYA, 2010). Acredita-se que ao comparar o modelo AvSchedP com o modelo proposto por Rahman, Hassan e Buyya (2010), compara-se de forma indireta o modelo AvSchedP com os modelos comparados no trabalho relacionado.

O *JaccardP* emprega o coeficiente de *Jaccard* para determinar a similaridade entre sequências binárias. Diferentemente do modelo AvSchedP, o *JaccardP* não extrai do comportamento histórico diversos vetores de mesmos tamanho e instante de tempo para determinar a similaridade. O *JaccardP* emprega um método de janela deslizante aplicando o coeficiente de *Jaccard* para cada janela formada [4].

Para realizar a comparação entre os modelos, configurou-se o *JaccardP* com os parâmetros apresentados na tabela 5.3.

Tabela 5.3: Parâmetros aplicados ao modelo baseado no coeficiente de *Jaccard*

Parâmetro	**Valor**
window size	3 (Min.)
training data	60 (Min.)
test data	10%

O parâmetro $windowsize$ foi inicializado com o valor 3 ($windowsize = 3$) devido a este ser o valor onde os experimentos realizados por Rahman, Hassan e Buyya (2010) apresentaram os melhores resultados. Os demais parâmetros também foram definidos com base nos valores especificados no artigo de Rahman, Hassan e Buyya (2010) no entanto, adaptados à realidade dos experimentos conduzidos neste trabalho.

O modelo AvSchedP realiza a previsão de duração de disponibilidade. Já o modelo baseado no coeficiente de *Jaccard* realiza a previsão de disponibilidade apenas para o instante $timestamp + 1$. Para possibilitar a comparação entre os modelos, o *JaccardP*

[4] O modelo de índice de Jaccard foi apresentado na Seção de trabalhos relacionados 3.4 do Capítulo 3 deste trabalho.

estende o modelo de Rahman, Hassan e Buyya (2010) para permitir a previsão do tempo de duração de um intervalo de disponibilidade [5].

Os parâmetros do modelo AvSchedP foram ajustados com $c = 90(\%)$ (utilizar 90% dos arquivos de histórico existentes conforme cada recurso) e $m = 10(Minutos)$ (utilizar apenas 10 minutos anteriores ao instante $timestamp$ para calcular a similaridade). O parâmetro f (tempo máximo de duração de disponibilidade a ser previsto) foi configurado de forma variável com o propósito de demonstrar a acurácia dos preditores conforme o aumento do tamanho do tempo de duração de previsão. Ambos modelos tiveram suas previsões limitadas ao parâmetro f.

Para fornecer o instante $timestamp$ de realização de uma previsão, foi utilizado o mesmo gerador de instantes utilizado nos experimentos de avaliação dos parâmetros. Utilizou-se quinze valores de hora do dia diferentes compreendidos entre às 8:00 horas e 18:00 horas. Neste intervalo os recursos apresentavam maior atividade, oscilando entre períodos de disponibilidade e indisponibilidade. Como o modelo está sendo proposto para realizar previsões em ambientes voláteis, optou-se por considerar este intervalo nos experimentos realizados. Cada valor de hora do dia selecionado era utilizado em todas as previsões realizadas em cada rodada de experimentos.

5.3.2.1 Resultados

Antes de realizar a avaliação detalhada da previsão de disponibilidade do modelo AvSchedP segundo as métricas definidas (taxa de acertos, acurácia e efetividade), optou-se por apresentar um comparativo entre o previsto pelo AvSchedP, o previsto pelo JaccardP e o que realmente ocorreu de disponibilidade nos recursos.

Para obter este comparativo foram geradas as medidas de duração média prevista e duração média realizada.

A duração média prevista foi computada obtendo-se a média das durações de disponibilidade previstas pelos modelos AvSchedP e JaccardP para cada recurso do ambiente.

A duração média realizada foi computada obtendo-se a média das durações de disponibilidade que realmente ocorreram em cada recurso do ambiente.

O cálculo destas médias foi realizado a cada rodada do experimento e seguiu as mesmas definições dos experimentos realizados nesta seção, os quais foram detalhados anteriormente.

Assim como na seção de avaliação dos parâmetros, para facilitar a avaliação dos resultados optou-se pela geração de gráficos apresentando os resultados de forma coletiva e por ambiente.

O gráfico disposto na figura 5.9 apresenta os resultados destas medidas obtidos com o agrupamento dos resultados de todos os ambientes.

Observando a linha representando a duração média realizada no recurso (*Dreal*) percebe-se que em média as previsões de disponibilidade fornecidas pelo AvSchedP (*Dprev AvSchedP*) foram próximas ao ocorrido no recurso. Nota-se também que o tempo médio previso pelo AvSchedP manteve-se abaixo do tempo médio real para todos os valores utilizados no parâmetro f. Com esse resultado é possível presumir que o AvSchedP possui uma tendência a fornecer previsões de disponibilidade com tempo de duração menor que o realizado no recurso.

O JaccardP (*Dprev JaccardP*) no entanto, apresenta um comportamento consideravelmente diferente. Observa-se que durante o experimento, a medida em que o parâmetro f

[5] A extensão deste modelo seguiu as orientações passadas pelos próprios autores no artigo referido.

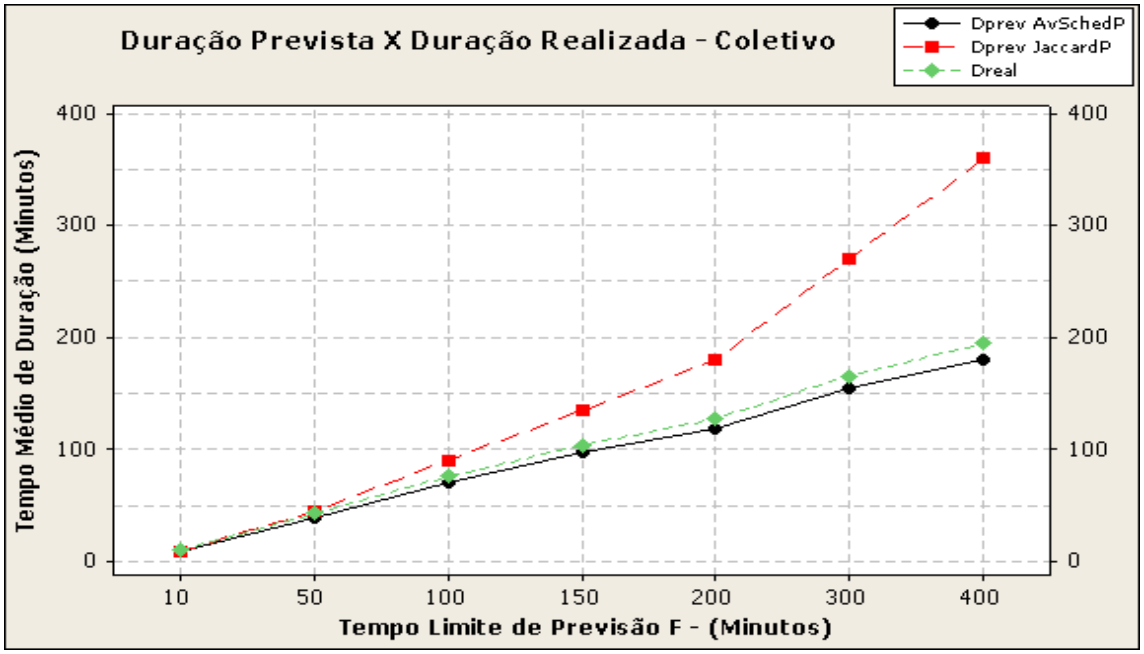

Figura 5.9: Comparativo entre Previsões e Disponibilidade Real - Coletivo.

foi aumentando as previsões foram se afastando cada vez mais do realizado no recurso. Fica evidente a tendência do preditor JaccardP fornecer previsões de disponibilidade com tempo de duração maior que o realizado no recurso.

O fornecimento de previsões de duração de disponibilidade menores que as durações que realmente ocorrem nos recursos, ou a **subestimação** da duração de disponibilidade, faz com que o sistema utilizador desta informação acabe tomando decisões que não utilizem o tempo real de disponibilidade do recurso por completo. Já o fornecimento de previsões de duração de disponibilidade maiores que as durações que realmente ocorrem nos recursos, ou **superestimação** da duração de disponibilidade, faz com que o sistema utilizador desta informação tome decisões que resultarão em erros ocasionados pela falta de tempo necessário para a execução de uma tarefa qualquer.

Para um escalonador de tarefas, a subestimação de duração de disponibilidade pode fornecer resultados melhores que a superestimação. A subestimação pode levar o escalonador a executar uma tarefa menor, onde se teria espaço para executar uma tarefa maior. Já a superestimação pode levar o escalonador a entregar tarefas que não serão completadas devido à falta de tempo no recurso. No primeiro caso, o escalonador desperdiçou tempo disponível no recurso, mas completou uma tarefa. No segundo caso, nenhuma tarefa foi executada e todo o tempo de processamento do recurso foi perdido [6].

O gráfico disposto na figura 5.10 apresenta os resultados obtidos por ambiente. Observa-se que em todos ambientes experimentados o comportamento dos preditores repete-se, tendo o AvSchedP apresentado a tendência de realizar previsões subestimando o tempo realizado, enquanto o JaccardP manteve a tendência de realizar previsões superestimando o tempo realizado.

Para complementar esta análise, a tabela 5.4 fornece os números referentes à estatística descritiva dos resultados obtidos.

Com os dados apresentados na tabela 5.4 é possível quantificar a distância média entre os valores previsto pelo AvSchedP e o realizado, assim como esta mesma distância

[6]Nos casos onde o escalonador de tarefas não conta com algum mecanismo de checkpoint.

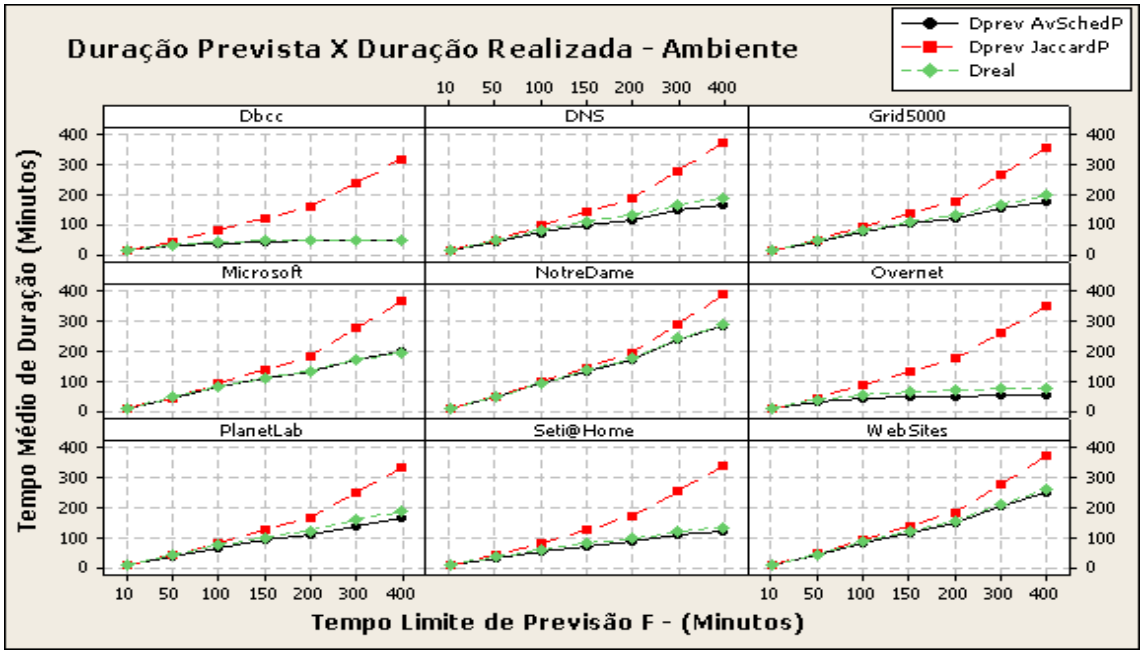

Figura 5.10: Comparativo entre Previsões e Disponibilidade Real - Ambiente.

Tabela 5.4: Estatística Descritiva do Comparativo entre Previsto e Realizado - Ambiente

	AvSchedP			**JaccardP**			**Real**	
Ambiente	$\overline{x}$	σ	C_v	$\overline{x}$	σ	C_v	$\overline{x}$	C_v
Dbcc	34,53	13,49	39,07	137,35	102,68	74,76	36,61	35,41
DNS	90,96	53,94	59,30	160,60	119,88	74,65	102,94	58,01
Grid5000	94,88	54,50	57,44	152,86	114,09	74,64	103,45	59,25
Microsoft	107,73	64,95	60,29	158,95	118,75	74,71	107,01	58,49
NotreDame	140,48	92,92	66,14	168,33	125,51	74,56	141,91	66,02
Overnet	40,62	15,51	38,18	151,81	113,47	74,75	56,38	42,87
PlanetLab	89,98	53,67	59,64	146,69	110,16	75,10	101,35	62,98
Seti@Home	70,58	41,60	58,94	149,22	111,63	74,81	78,60	55,42
WebSites	123,83	81,09	65,48	162,82	121,46	74,60	129,32	65,69

entre os valores previstos pelo JaccardP. Com estes dados pode-se observar a tendência de subestimação do AvSchedP e a tendência de superestimação do JaccardP. Nota-se que com exceção do ambiente Microsoft, o AvSchedP apresentou um tempo médio de duração de disponibilidade previsto inferior ao realizado pelo recurso em todos os ambientes. Observa-se também a superestimação ocorrida com o JaccardP em todos os ambientes.

Considerando-se uma grade oportunista com um ambiente como o Overnet, o escalonador de tarefas receberia informações de disponibilidade do preditor AvSchedP com tempo de duração, em média, de aproximadamente 40 minutos. Já com o preditor JaccardP esta informação teria um tempo médio de duração de disponibilidade de aproximadamente 151 minutos. Observando o que realmente ocorreu nos recursos deste ambiente, aproximadamente 56 minutos de duração de disponibilidade média, nota-se a distância significativa entre a previsão do JaccardP e o realizado. Com esta informação o escalonador estaria sujeito a selecionar tarefas extremamente grandes para o tempo realmente disponível, o que ocasionaria uma sucessão de erros de escalonamento e desperdício de

tempo de execução.

Outra evidência extraída destes dados é a variabilidade dos valores previstos pelo AvSchedP. Como o desvio padrão é extremamente influenciado quando os dados utilizados possuem alta variabilidade, fica difícil a comparação dos resultados. Além disso, o desvio padrão é expresso na mesma unidade de medida da variável analisada.

Para facilitar a análise e comparação dos resultados foi calculado também o coeficiente de variação, coeficiente este que expressa o seu resultado como uma taxa, um percentual (%) de variação existente nos dados.

Como pode ser observado, o coeficiente de variação obtido nas previsões do AvSchedP é mais estável. Com exceção dos ambientes Overnet e Dbcc, onde a varição dos dados foi muito menor que os demais ambientes, o AvSchedP apresentou taxas de variação próximas de 60%. Já o JaccardP apresentou taxas de variação próximas de 74%.

No entanto, neste caso, mais importante que comparar os valores obtidos entre os preditores é comparar os valores obtidos em cada preditor com o realizado. Isso é importante devido ao que foi medido. Tem-se aqui como medida tempos de duração de disponibilidade. Estes tempos podem variar consideravelmente de acordo com a hora do dia, como apresentado no Capítulo 3. A prova disso encontra-se no coeficiente de variação obtido nos valores realizados. A variação existente nos dados realizados é decorrente da variação de tamanho dos intervalos de disponibilidade considerados. Por esta razão, a comparação com o valor realizado se faz muito importante.

Comparando os valores previstos com o realizado, observa-se que o AvSchedP apresentou valores de variação, observados no coeficiente de variação, próximos ou similares aos valores obtidos no realizado. Já o JaccardP não apresentou muita semelhança entre os valores obtidos com o preditor e o realizado.

Isto sugere que as previsões realizadas pelo AvSchedP têm menor variação que as previsões fornecidas pelo JaccardP, e que as previsões fornecidas são mais similares ao que realmente ocorreu no recurso.

Dando andamento à análise de qualidade das previsões realizadas pelo preditor, identificou-se que a tendência em fornecer previsões com tempo menor que o realizado realmente faz com que o preditor AvScheP tenha uma taxa de acertos de previsão ($\overline{TxA}$) maior que a taxa de acertos de previsão obtida com o JaccardP.

O gráfico da figura 5.11 apresenta os resultados obtidos com a métrica $\overline{TxA}$, avaliados de forma coletiva.

Destaca-se no gráfico da figura 5.11, a sensibilidade da taxa média de acertos de previsão de disponibilidade $\overline{TxA}$ conforme a variação do parâmetro f (tempo limite de previsão). Esta sensibilidade é verificada em ambos preditores. A medida em que o parâmetro f é elevado, é reduzida quase que proporcionalmente a taxa de acertos dos preditores.

Mais significativo ainda é o caso das previsões realizadas com o JaccardP. O preditor mostrou-se muito mais sensível a este parâmetro, apresentando redução ainda maior da taxa de acertos.

Observa-se que com o valor do parâmetro f ajustado para $f = 10$, ambos preditores apresentavam aproximadamente 97% de acertos de previsão, ou seja, dada uma duração qualquer obtida da previsão de disponibilidade de qualquer um dos dois preditores e limitada a um tempo máximo de 10 minutos ($f = 10$), esta previsão teria aproximadamente 97% de chance de estar correta .

Com o aumento do valor do parâmetro f para $f = 150$ a chance de uma previsão correta mudou significativamente. O preditor AvSchedP passou a fornecer previsões corretas em aproximadamente 77% das previsões realizadas. Já o JaccardP, sendo significativa-

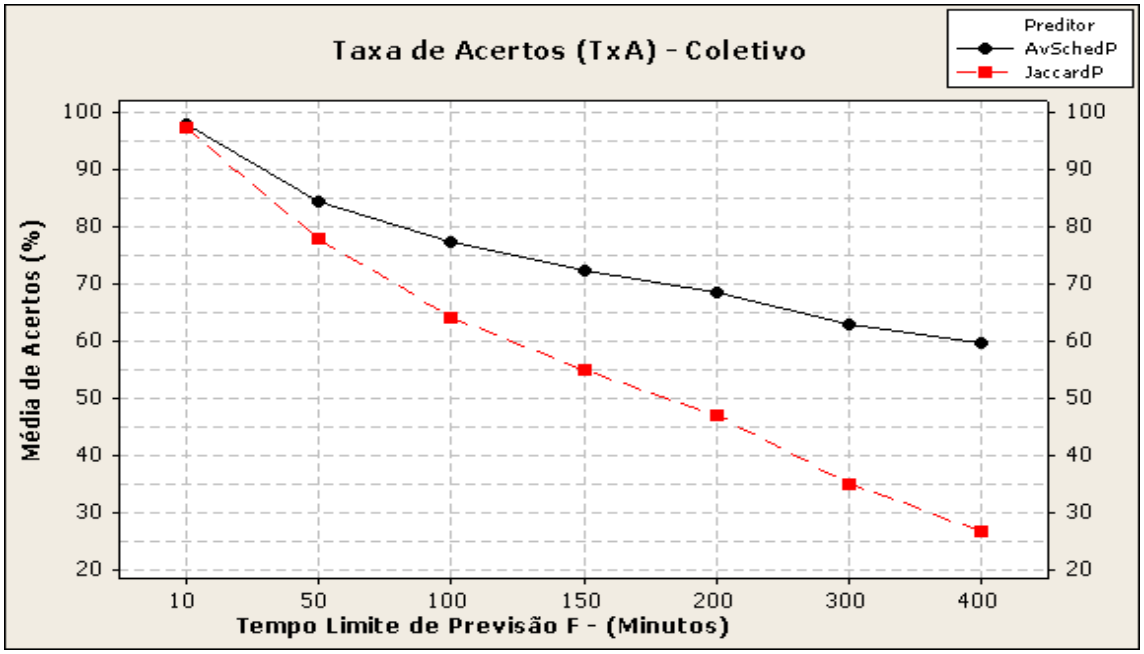

Figura 5.11: Taxa Média de Acertos de Previsão - Coletivo

mente mais sensível à variação do parâmetro, forneceu previsões corretas em aproximadamente 55% das previsões realizadas. Com o parâmetro f configurado para $f = 150$ o AvSchedP superou o JaccardP em aproximadamente 22 pontos percentuais.

Ao aumentar o valor do parâmetro f para $f = 400$, a diferença na taxa de acertos $(\overline{TxA})$ foi ainda mais significativa. O AvSchedP passou a fornecer previsões corretas para aproximadamente 60% das previsões realizadas, contra aproximadamente 27% obtidos com o JaccardP. Com esta configuração o AvSchedP acertou ainda mais, apresentando uma diferença de aproximadamente 38 pontos percentuais em relação ao JaccardP.

A tabela 5.5 apresenta a estatística descritiva coletiva dos resultados obtidos.

Tabela 5.5: Estatística Descritiva da Taxa de Acertos (Txa) - Coletivo

Preditor	$\overline{x}$	σ	C_v
AvSchedP	74.725	14.624	19.57
JaccardP	57,625	28,986	50,30

Coletivamente, a média da taxa de acertos $(\overline{TxA})$ obtida pelo preditor AvSchedP foi de aproximadamente 74% de acertos entre todas previsões realizadas para todas variações do parâmetro f. Observa-se uma diferença de aproximadamente 17 pontos percentuais em média entre a taxa de acertos do AvSchedP e do JaccardP, que obteve a média de aproximadamente 57% de acertos.

O Coeficiente de Variação C_v apresentado na tabela 5.5 também permite analisar a variabilidade dos resultados obtidos. Como o C_v do preditor AvSchedP foi menor que o obtido no JaccardP, é possível perceber que a taxa média obtida por ambiente foi mais estável nas previsões realizadas pelo preditor AvSchedP.

Como a análise de resultados da $(\overline{TxA})$ realizada até o momento contempla apenas a visão coletiva, realizou-se também a análise da taxa de acertos de previsão por ambiente. A análise em nível de ambiente permite identificar se as taxas de acertos de alguns am-

bientes são consideravelmente diferentes das obtidas nos demais ambientes e se esta taxa diferencia-se significativamente da taxa de acertos coletiva.

O gráfico da figura 5.12 apresenta a taxa de acertos de previsão $(\overline{TxA})$ por ambiente para cada preditor.

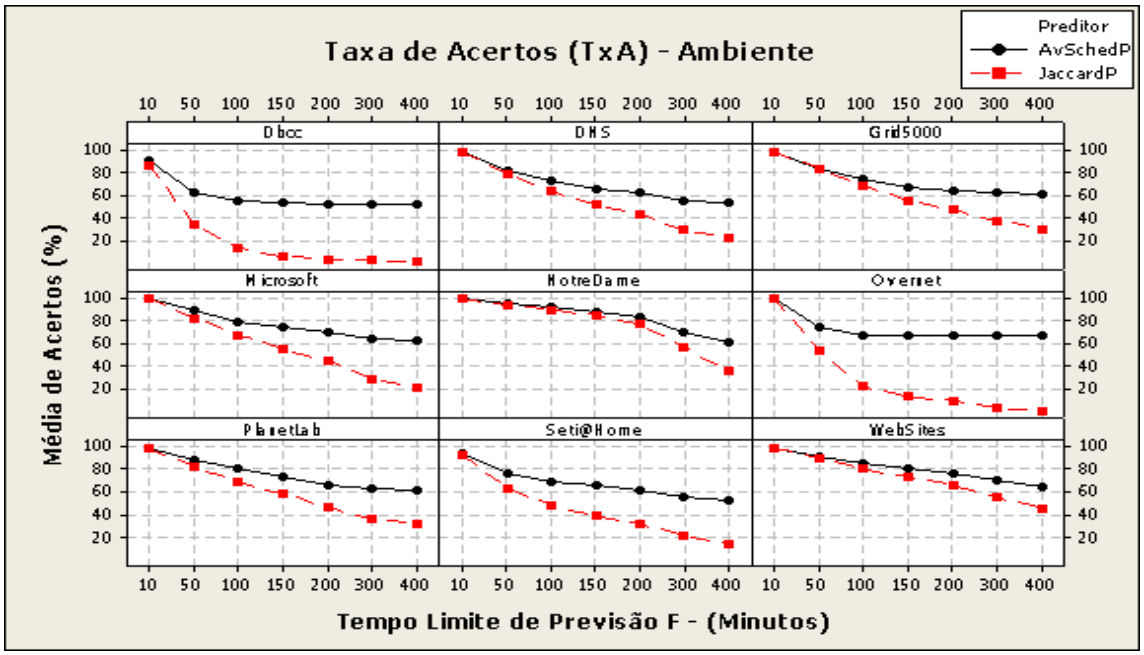

Figura 5.12: Taxa de Acertos de Previsão $(\overline{TxA})$ - Por Ambiente

É possível observar que em todos os ambientes experimentados ambos modelos mantiveram suas tendências em evidência. Tanto o AvSchedP, quanto o JaccardP, mantiveram em todos os ambientes experimentados a característica de reduzir a taxa de acertos à medida em que o parâmetro f é incrementado.

Observa-se também que em todos os ambientes o AvSchedP superou o preditor JaccardP, apresentando taxas de acertos de previsão superiores. Além disso, o AvSchedP apresentou em todos os ambientes a característica de ser menos sensível à variação de valor do parâmetro f, aumentando a diferença entre a taxa de acertos do AvSchedP e a taxa de acertos do JaccardP conforme ocorre o aumento do valor do parâmetro f.

Tabela 5.6: Estatística Descritiva da Taxa de Acertos $(\overline{TxA})$ - Ambiente

	AvSchedP			**JaccardP**		
Ambiente	$\overline{x}$	σ	C_v	$\overline{x}$	σ	C_v
Dbcc	59,33	13,35	22,50	20,46	29,13	142,34
DNS	69,65	15,31	21,98	54,88	26,18	47,71
Grid5000	72,34	12,67	17,52	59,72	23,27	38,96
Microsoft	76,80	13,03	16,97	57,21	26,80	46,86
NotreDame	83,88	13,03	15,54	76,74	20,89	27,23
Overnet	72,17	11,98	16,60	28,98	33,49	115,54
PlanetLab	76,22	14,01	18,39	60,86	24,84	40,82
Seti@Home	68,50	13,46	19,66	44,73	25,44	56,87
WebSites	81,41	11,38	13,99	73,29	18,20	24,84

Confirmando o resultado apresentado pelo Coeficiente de Variação C_v coletivo da tabela 5.5, o gráfico da figura 5.12 torna visível a menor variação das taxas de acertos obtidas em cada ambiente, por preditor. Observa-se que em todos os ambientes a taxa de acertos do AvSchedP, obtida com o parâmetro f ajustado para 400, manteve-se próxima de 60%. Já o JaccardP apresenta maior variabilidade, tendo taxas de acertos próximas de 0%, como a obtida no ambiente Dbcc e 50% como a obtida no ambiente WebSites.

Complementado os resultados apresentados na figura 5.12, a tabela 5.6 apresenta a estatística descritiva da taxa de acertos de previsão por ambiente. Chama-se atenção novamente para o coeficiente de variação apresentado na tabela para os dois preditores. Nota-se que as taxas de acertos obtidas com o AvSchedP possuem menor variância que as taxas obtidas com o JaccardP, devendo-se então considerar que a taxa de acertos média obtida com o AvSchedP é mais confiável que a taxa de acertos média obtida com o JaccardP.

Quando considerado o contexto de escalonamento em grades oportunistas, os resultados obtidos até o momento indicariam que o preditor AvSchedP ofereceria ao escalonador previsões de disponibilidade com maior chance de acertos do que as oferecidas pelo preditor JaccardP, visto que a chance de ocorrer a duração de disponibilidade prevista pelo AvSchedP foi maior que a obtida com o JaccardP.

Cabe ressaltar que foram considerados como acertos os casos onde o preditor teve a duração de disponibilidade prevista menor ou igual à duração de disponibilidade que realmente ocorreu no recurso. Considerando o escalonamento, este acerto é importante pois permitiria ao escalonador selecionar tarefas adequadas à duração de disponibilidade prevista, aumentando o número de acertos de escalonamento.

Rahman, Hassan e Buyya (2010) e Byun et al. (2005), foram apresentadas apenas as taxas de acertos como medida de qualidade de um preditor de disponibilidade em ambientes de grades oportunistas. Este trabalho vai além, apresentando a medida de proximidade da duração de disponibilidade prevista em relação à disponibilidade que realmente ocorreu no recurso, ou seja, mostra a acurácia do preditor.

A figura 5.13 apresenta a acurácia média de previsão dos preditores AvSchedP e JaccardP, representada pela medida $\overline{Acc}$.

Tabela 5.7: Estatística Descritiva da Acurácia do Preditor ($\overline{Acc}$) - Coletivo

Preditor	$\overline{x}$	σ	C_v
AvSchedP	95,76	7,25	7,58
JaccardP	60,14	35,56	59,14

Nesta medida os dois preditores também apresentaram resultados significativamente diferentes. Com a variação do parâmetro f de 10 a 400 minutos, o preditor AvSchedP mostrou grande proximidade entre o previsto e o realizado, ou seja, manteve sua acuraria quase constante e significativamente elevada. A acurácia do AvSchedP manteve-se próxima dos 95% em todos os valores aplicados ao parâmetro f.

Já o JaccardP apresentou grande perda de acurácia conforme ocorre o aumento do valor do parâmetro f. Este resultado está extremamente ligado à taxa de acertos. Mostra que além da tendência de produzir previsões com tempo de duração de disponibilidade maior que o realizado no recurso, a diferença existente entre as durações previstas e as durações realizados são significativamente grandes. Esta característica já fora identificada

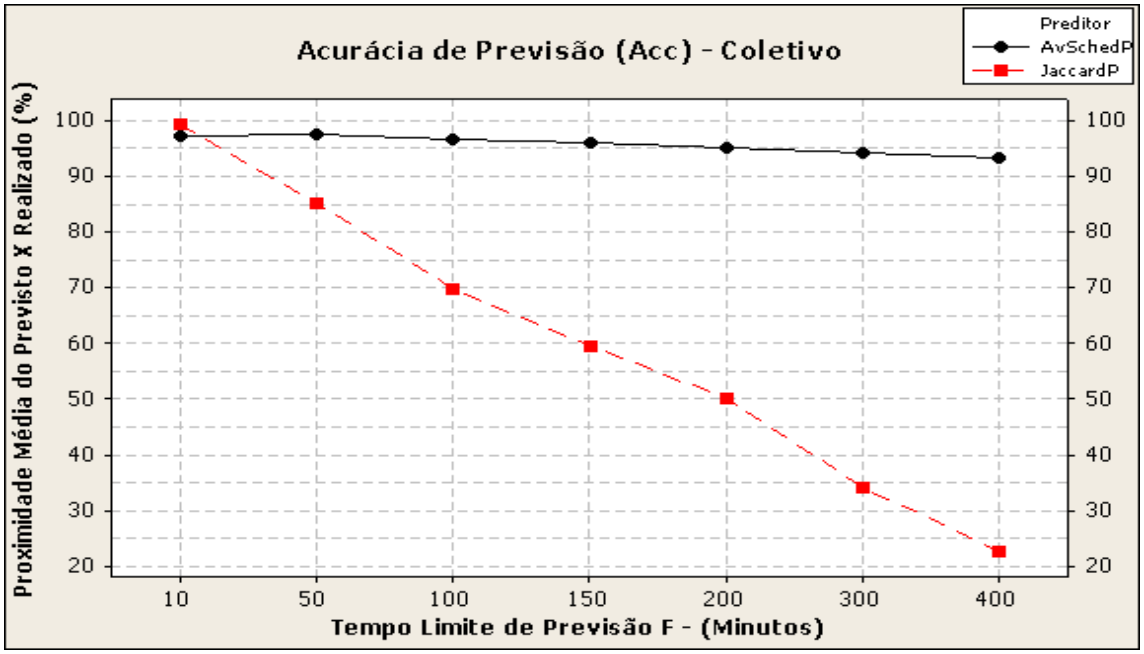

Figura 5.13: Acurácia de Previsão ($\overline{Acc}$) - Coletivo

no gráfico apresentado na figura 5.9, onde é possível observar a diferença significativa entre o que realmente ocorreu no recurso e o que foi previsto pelo preditor JaccardP.

A tabela 5.7 apresenta a estatística descritiva coletiva da acurácia ($\overline{Acc}$) obtida dos resultados dos experimentos.

Novamente o coeficiente de variação chama a atenção. No AvSchedP, a acurácia média foi de 95,76% com um coeficiente de variação de 7,58, o que indica baixa variância nos dados. No JaccardP, a acurácia foi aproximadamente 35% inferior ao AvSchedP e a variância nos dados é grande, como mostra o coeficiente de variação com o resultado de 59,14.

Para verificar se esta situação se repetia em todos os ambientes, realizou-se também a mesma análise sobre os dados separados por ambiente. Os gráficos da figura 5.14 apresentam a acurácia de previsão ($\overline{Acc}$) de cada preditor por ambiente experimentado.

Os resultados obtidos ao mensurar a acurácia de previsão ($\overline{Acc}$) demonstram a superioridade do preditor AvSchedP em todos ambientes experimentados.

O gráfico da figura 5.14 exibe diferenças significativas na qualidade de previsão dos dois preditores. O JaccardP em todos os ambientes manteve baixa acurácia. Isso se deve principalmente à superestimação das durações de disponibilidade, fazendo com que a distância entre o valor previsto e o que realmente ocorreu fosse extremamente grande e consequentemente, reduzindo a acurácia do preditor.

Para facilitar a análise quantitativa dos resultados, foi fornecida a tabela 5.8 contendo a estatística descritiva da acurácia dos preditores.

Apesar de visível no gráfico apresentado na figura 5.14, chama-se a atenção para variância nos resultados obtidos com o preditor AvSchedP, por ambiente. Observa-se na tabela 5.8 que o coeficiente de variação apresentou valores baixos em todos os ambientes com este preditor. Já com o JaccardP, o coeficiente de variação foi consideravelmente elevado principalmente, no ambiente Overnet onde obteve a mais baixa acurácia de previsão.

Até o momento, o preditor AvSchedP tem demonstrou superioridade sobre o preditor JaccardP, apresentando resultados superiores ao JaccardP em todos os experimentos re-

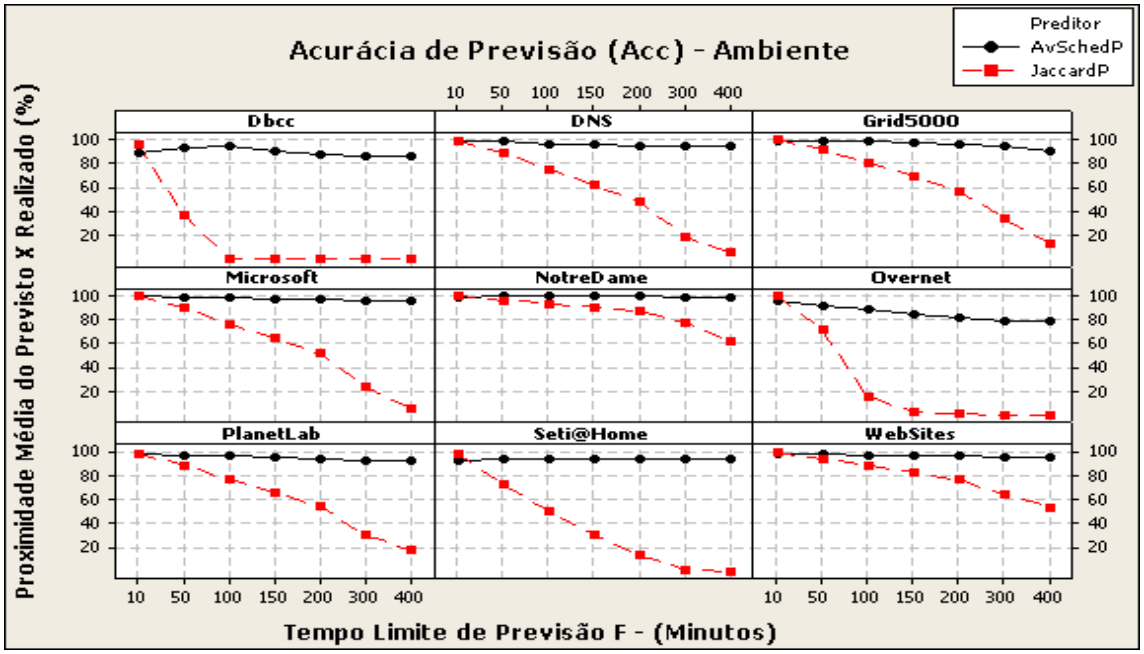

Figura 5.14: Acurácia de Previsão ($\overline{Acc}$) - Por Ambiente

Tabela 5.8: Estatística Descritiva da Acurácia do Preditor ($\overline{Acc}$) - Ambiente

	AvSchedP			**JaccardP**		
Ambiente	$\overline{x}$	σ	C_v	$\overline{x}$	σ	C_v
Dbcc	89,65	11,99	13,37	19,03	34,47	181,15
DNS	95,96	9,24	9,63	56,58	33,97	60,05
Grid5000	96,35	2,96	3,08	63,36	29,51	46,57
Microsoft	97,76	2,60	2,66	58,93	33,50	56,84
NotreDame	99,64	0,36	0,37	86,99	12,08	13,90
Overnet	85,95	10,87	12,65	27,15	38,74	142,72
PlanetLab	95,66	4,92	5,15	62,52	32,74	52,37
Seti@Home	94,59	9,85	10,42	38,40	36,35	94,65
WebSites	97,51	1,40	1,44	80,45	16,83	20,93

alizados. Destacou-se ainda mais no experimento que avaliou a acurácia das previsões, demonstrando que o preditor proposto oferece previsões consideravelmente próximas da realidade.

No entanto, comparar os preditores utilizando as duas medidas apresentadas até o momento pode ser insuficiente para se chegar a uma conclusão sobre qual seria o melhor preditor para o contexto de grades oportunistas, ou quanto melhor seria um em relação ao outro. Por esta razão, avaliou-se também a efetividade média de cada preditor. A medida de efetividade permite identificar qual preditor acerta mais e com mais acurácia, ou seja, a efetividade média $\overline{Ef}$ agrupa em uma única medida a taxa de acertos e a acurácia de previsão, facilitando a comparação dos preditores.

A figura 5.15 apresenta a efetividade média coletiva de cada preditor representada pela medida $\overline{Ef}$.

Como pode ser observado, o AvSchedP apresentou maior efetividade que o preditor

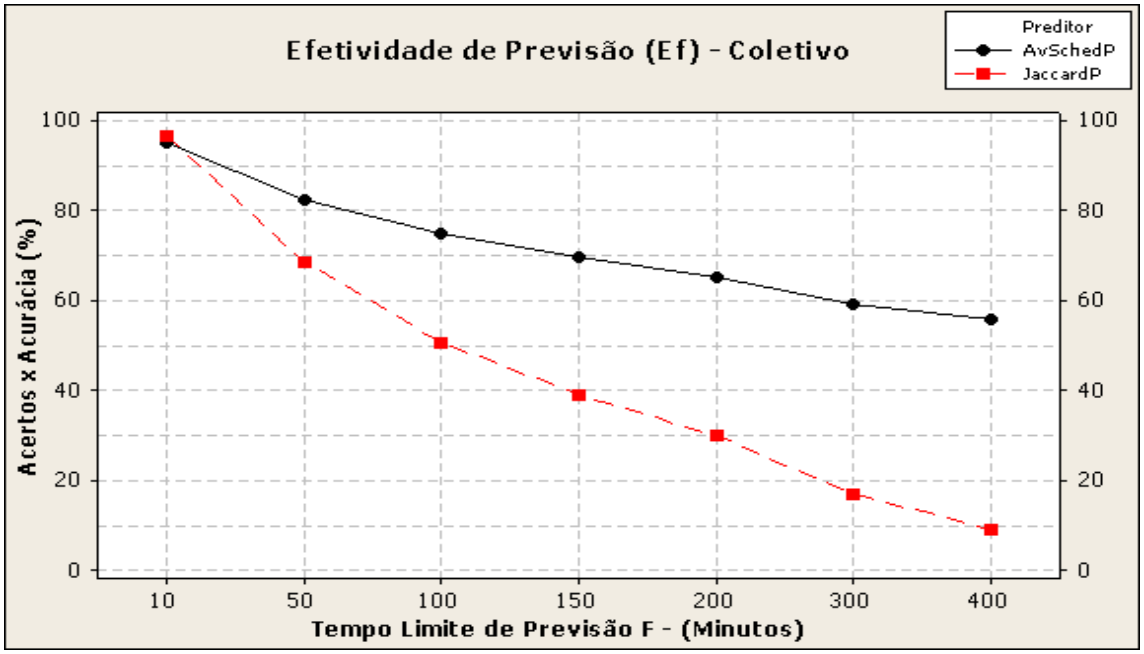

Figura 5.15: Efetividade de Previsão $(\overline{Ef})$ - Coletivo

Tabela 5.9: Estatística Descritiva da Efetividade do Preditor $(\overline{Ef})$ - Coletivo

Preditor	$\overline{x}$	σ	C_v
AvSchedP	71,79	16,04	22,35
JaccardP	44,51	34,81	78,23

JaccardP para todos os valores do parâmetro f, exceto com o valor $f = 10$, destacando-se principalmente quando o valor deste parâmetro é maior. Com o parâmetro $f = 150$ o AvSchedP foi aproximadamente 30% mais efetivo que o preditor Jaccard. Já com o parâmetro $f = 400$, o preditor AvSchedP foi aproximadamente 40% mais efetivo que o preditor JaccardP. Isso demonstra a tendência do preditor AvSchedP de produzir resultados com melhor qualidade que o JaccardP, quando as durações de disponibilidade a serem previstas forem melhores.

Já o preditor JaccardP apresentou superioridade ao AvSchedP quando o parâmetro f estava ajustado com o valor $f = 10$. Apesar de não apresentado nos gráficos e tabelas apresentadas, o JaccardP superou o AvSchedP na medida de efetividade nos valores de f menores que 10 minutos. Isso demonstra a existência de uma deficiência no preditor AvSchedP no que diz respeito à previsão de disponibilidade com pequenas durações.

A tabela 5.9 apresenta a estatística descritiva da medida de efetividade obtida para os preditores AvSchedP e JaccardP, como complemento ao gráfico da figura 5.15.

Assim como realizado nas medidas de taxa de acertos e acurácia, a figura 5.16 apresenta os gráficos de efetividade dos preditores AvSchedP e JaccardP, por ambiente experimentado. Já a tabela 5.10 apresenta a estatística descritiva dos resultados obtidos com a medida de efetividade.

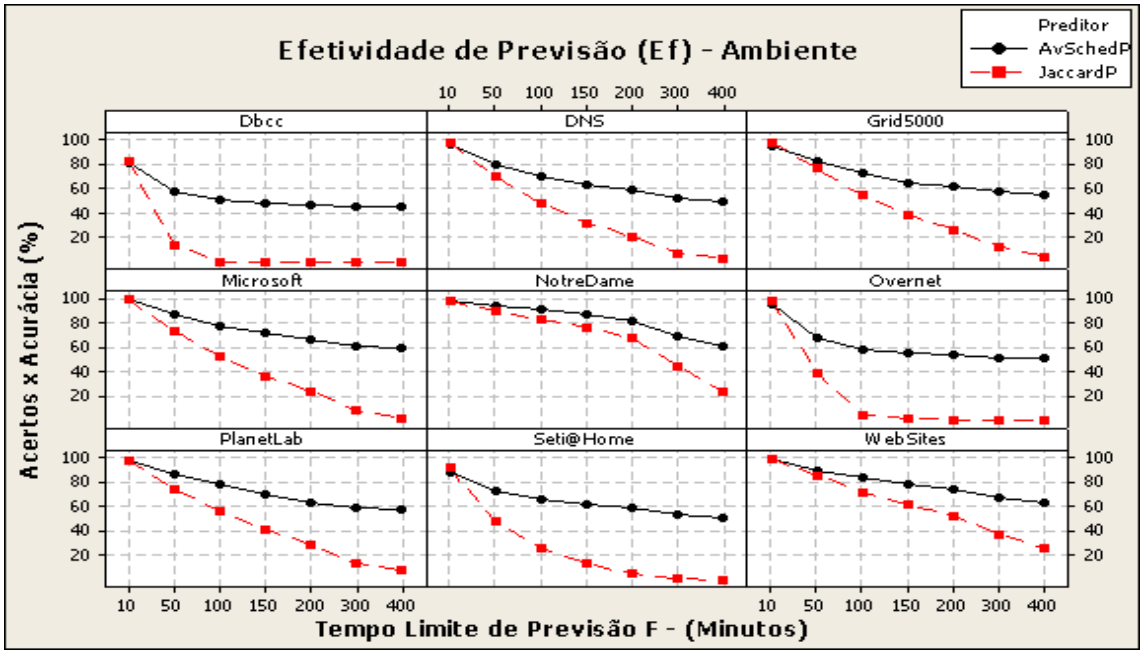

Figura 5.16: Efetividade de Previsão $(\overline{Ef})$ - Por Ambiente

Tabela 5.10: Estatística Descritiva da Efetividade do Preditor $(\overline{Ef})$ - Ambiente

	AvSchedP			**JaccardP**		
Ambiente	$\overline{x}$	σ	C_v	$\overline{x}$	σ	C_v
Dbcc	53,37	13,91	26,06	13,74	28,85	210,04
DNS	66,91	16,64	24,87	39,57	32,91	83,18
Grid5000	69,94	13,72	19,62	44,32	31,90	71,98
Microsoft	75,29	14,07	18,69	42,27	34,18	80,86
NotreDame	83,59	13,03	15,60	69,24	25,15	36,33
Overnet	62,28	15,04	24,15	20,44	34,87	170,63
PlanetLab	73,07	14,76	20,20	45,61	34,08	74,71
Seti@Home	64,54	13,57	21,02	26,15	31,69	121,17
WebSites	79,49	11,88	14,95	61,91	25,34	40,93

Com os resultados obtidos pode-se observar que o AvSchedP superou as previsões realizadas pelo preditor JaccardP, inclusive quando considerada a medida de efetividade apresentada. Neste ponto, faz-se necessário destacar que a superioridade do AvSchedP sobre o JaccardP sugere a superioridade do preditor AvSchedP sobre os demais preditores, preditores estes comparados com o JaccardP por Rahman, Hassan e Buyya (2010).

5.3.3 AvSchedP Integrado ao XtremWeb

Após a avaliação dos parâmetros do modelo e da qualidade das previsões realizadas pelo AvSchedP, avaliou-se a integração do protótipo do modelo AvSchedP (XtremWebAv) integrado ao sistema de grade oportunista *XtremWeb*.

A avaliação foi realizada através da comparação de resultados entre o protótipo *XtremWebAv* e o *XtremWeb*. Para realizar a comparação foram utilizados os seguintes critérios:

- Número de tarefas executadas: quantifica o número de tarefas executadas com sucesso pelo escalonador de tarefas. Este critério permite comparar qual dos escalonadores executou mais tarefas em um mesmo espaço de tempo.

- Número de tarefas canceladas: quantifica o número de tarefas canceladas pelo escalonador de tarefas. Este critério permite comparar qual dos escalonadores cancelou mais tarefas em um mesmo espaço de tempo.

- Tempo gasto em tarefas executadas: quantifica o tempo de execução gasto na execução de tarefas que completaram com sucesso. Este critério permite comparar qual dos escalonadores conseguiu aproveitar melhor o tempo disponível para execução de tarefas.

- Tempo gasto em tarefas canceladas: quantifica o tempo de execução gasto na execução de tarefas canceladas. Este critério permite comparar qual dos escalonadores mais desperdiçou o tempo disponível para execução de tarefas. Este critério também é conhecido na literatura como *Wasted Time* e é frequentemente utilizado para avaliar escalonadores no que diz respeito a tempo de processamento desperdiçado. Nadeem (2008b) e Ballier (2005) utilizaram o tempo desperdiçado (*Wasted Time*) como critério de avaliação de escalonadores de grades computacionais e oportunistas.

Com a utilização destes critérios tem-se como objetivo, avaliar as possíveis perdas e ganhos em se utilizar o modelo AvSchedP integrado a um sistema de grade oportunista, como o *XtremWeb*.

Além disso, a avaliação do tempo gasto com a execução de tarefas executadas e canceladas possibilita verificar se a política de escalonamento adotada no protótipo, atinge o seu objetivo[7].

Para avaliar estes critérios, foram conduzidos experimentos em um ambiente de grade oportunista controlado, onde o servidor foi inicializado com cinco aplicações. Cada aplicação computa os primeiros n números primos e os escreve em um arquivo de resultados. As cinco aplicações instaladas no servidor foram chamadas de **App1**, **App2**, **App3**, **App4** e **App5**. Estas aplicações geram os primeiros 33000, 40000, 60000, 75000 e 95000 números primos, respectivamente.

Para cada aplicação foram geradas 1000 tarefas, totalizando 5000 tarefas para execução no servidor. O tempo médio de execução das tarefas de cada aplicação foi inicializado com 8 minutos para a aplicação **App1**, 9 minutos para a aplicação **App2**, 16 minutos para a aplicação **App3**, 22 minutos para a aplicação **App4** e 32 minutos para a aplicação **App5**.

Como descrito na apresentação da política de escalonamento implementada no protótipo *XtremWebAv*, o tempo necessário para execução de uma tarefa deve ser conhecido pelo escalonador para possibilitar a tomada de decisão de escalonamento. Para obter o tempo médio de execução de cada aplicação, realizou-se a execução de uma rodada do próprio experimento. O tempo médio obtido para cada aplicação ao final do experimento, foi utilizado como valor inicial para o tempo de execução de tarefas daquela aplicação. É importante citar que os resultados obtidos na execução deste experimento foram desconsiderados.

Os recursos utilizados no experimento são heterogêneos, como apresentado na tabela 5.11 abaixo, e a utilização de uma média por aplicação pode levar à subestimação ou

[7]Como apresentado na Seção 5.2 do Capítulo 5, o objetivo de escalonamento da política proposta é utilizar os recursos de forma mais eficiente, através da redução do tempo desperdiçado em tarefas canceladas.

superestimação do tempo necessário para o recurso executar uma tarefa disponível. Para tratar com a heterogeneidade dos recursos, após um *Worker* executar alguma tarefa de uma aplicação, o tempo médio de execução de tarefas daquela aplicação para aquele *Worker* é ajustado. Esta situação foi tratada anteriormente na descrição do protótipo. Para os próximos escalonamentos de tarefas daquela aplicação para aquele *Worker*, o novo tempo obtido passa a ser considerado pelo escalonador nas tomadas de decisões.

Para facilitar a análise dos resultados obtidos, alguns cuidados foram tomados de forma a manter o ambiente de execução o mais homogêneo possível entre as diversas execuções deste experimento.

Ao todo foram realizadas trinta e oito (38) execuções deste experimento, sendo dezenove (19) execuções do experimento com o XtremWeb e outras dezenove execuções com o XtremWebAv.

Os cuidados tomados têm por objetivo reduzir ao máximo a variância entre os resultados obtidos de modo a permitir a utilização de média aritmética na consolidação dos resultados de todos os experimentos realizados.

Um dos primeiros cuidados tomados foi o de inicializar cada experimento com todas as tarefas já carregadas no banco de dados do servidor, evitando-se assim a ocorrência de requisições por tarefa não atendidas, devido a um possível gargalo na submissão de tarefas para o servidor. Nos experimentos iniciais utilizou-se dois recursos além dos utilizados neste experimento para servirem de recursos de submissão. De fato, o gargalo ocorreu gerando variações nos resultados devido à indisponibilidade de tarefas para execução. Ao carregar o sistema com todas as tarefas antes de inicializar a execução do experimento, garantiu-se que nenhum *Worker* ficaria sem receber tarefas quando realiza-se uma requisição ao servidor. Assim, os resultados obtidos entre as diversas execuções tornaram-se mais homogêneos.

Outro cuidado tomado foi o de inicializar os recursos sempre na mesma ordem. Devido à heterogeneidade dos recursos envolvidos, a inicialização dos recursos em ordem aleatória leva a resultados diferentes ao final de cada experimento. Também durante os experimentos iniciais observou-se variância nos dados devido à ordem de entrada dos recursos no experimento. O experimento foi então alterado para garantir uma única ordem de entrada dos recursos no sistema.

Para existir a ocorrência de períodos de disponibilidade e indisponibilidade nos recursos, ou seja, para simular a presença de usuários nos recursos, foram utilizados os arquivos de comportamento obtidos com o monitoramento dos recursos pertencentes ao ambiente Dbcc, arquivos estes utilizados no Capítulo 3.

Cada *Worker* recebeu o conjunto de arquivos de comportamento de um dos recursos monitorados, não existindo a repetição de comportamentos entre os *Workers*. Tomou-se o cuidado de não repetir os comportamentos entre os recursos para evitar a inclusão de tendências nos resultados.

Ao iniciar a execução do *Worker* é selecionado o arquivo mais atual entre os arquivos de comportamento disponíveis para simular a presença do usuário no recurso, ou seja, as ocorrências de períodos de disponibilidade e indisponibilidade seguem o que ocorreu no arquivo selecionado. Este procedimento foi realizado tanto no *XtremWebAv* quanto no *XtremWeb*, permitindo assim a comparação entre os resultados obtidos.

Para realizar a previsão de disponibilidade, o protótipo *XtremWebAv* utilizou os demais arquivos pertencentes ao conjunto de arquivos instalado no recurso.

Pelo fato de utilizar recursos heterogêneos, outro cuidado tomado foi o da especificação do conjunto de arquivos para recursos. Cada recurso foi inicializado sempre com o

mesmo conjunto de arquivos em todos os experimentos realizados. Este cuidado é importante nesta situação, visto que ao atribuir comportamentos diferentes a recursos heterogêneos poderia-se elevar a variância nos resultados obtidos, devido aos diferentes intervalos de disponibilidade em recursos com diferentes capacidades computacionais.

Cada execução do experimento foi programada para durar 6 horas. Um dos pontos destacados na análise de disponibilidade realizada neste trabalho e em análises realizadas por outros autores, como relatado na Seção 3.3 do Capítulo 3, a distribuição de frequência das durações de disponibilidade é diferente conforme a hora do dia. Por esta razão, mais outro cuidado foi tomado na realização dos experimentos, o de manter o horário de início e término de cada experimento iniciando às 09:00 horas e finalizando às 15:00 horas.

No que diz respeito ao ambiente de execução dos experimentos, foram utilizados dois *clusters* que totalizaram 13 nodos com a funcionalidade de *worker* e 1 nodo com a funcionalidade de *Server*. Com a utilização deste ambiente, retira-se possíveis variações de banda existentes na rede e outros fatores que poderiam elevar a variância dos resultados, visto que este ambiente foi utilizado de forma exclusiva durante a condução dos experimentos.

A tabela 5.11 apresenta um resumo com a configuração de cada recurso utilizado nos experimentos.

Alguns recursos do tipo *Worker* são biprocessados. O sistema *XtremWeb* realiza o escalonamento de uma tarefa para cada processador existente em um *Worker*. Assim, no caso de recursos com (n) processadores, serão enviadas pelo *Worker* n requisições por tarefas, o escalonador realizará n escalonamentos e as n tarefas entregues serão executadas concorrentemente no *Worker*. Este comportamento foi mantido no protótipo *XtremWebAv*.

Tabela 5.11: Configuração dos recursos utilizados nos experimentos

Recurso	**RAM(Mb)**	**Processador**	**CPUs**
integridade, c1-0, c1-1 e c1-2	512	Pentium 4Ghz	2
gradep	512	Pentium 2Ghz	1
c0-1, c0-2, c0-3, c0-4 e c0-8	512	Pentium 1.8Ghz	1
c0-6	512	Pentium 1Ghz	1
c0-9 e c0-10	512	Pentium 2.3Ghz	1

Por fim, como descrito no modelo proposto, o algoritmo de previsão de disponibilidade do AvSchedP utiliza 3 parâmetros (c, m e f) que podem ser ajustados para obter melhores resultados. Estes parâmetros foram calibrados com os valores $c = 90\%$, $m = 10$ e $f = 50$. O parâmetro f foi ajustado com o valor 50, por este valor ser maior e próximo ao tempo necessário para executar as maiores tarefas (35 minutos) existentes na fila de tarefas do servidor.

Inicialmente, alguns experimentos foram realizados em um ambiente não tão controlado. Estes experimentos foram conduzidos na grade francesa **Grid5000**. No entanto, os resultados obtidos foram inutilizados devido a uma série de problemas ocorridos que acabaram por impossibilitar a comparação dos resultados obtidos com o XtremWeb e o XtremWebAv. Alguns dos problemas encontrados foram:

- **Alocação de Recursos:** por muitas vezes tornava-se difícil obter o mesmo número de recursos para realizar as diversas execuções dos experimentos. Além disso, os recursos alocados para cada rodada do experimento eram diferentes, o que acabava

por gerar resultados diferentes. Outro problema encontrado era a ocorrência do desligamento de recursos que estavam em uso nos experimentos, causando a redução do número de tarefas executadas;

- **Tempo de Alocação:** o tempo necessário para a execução dos experimentos era muito grande (6 horas). O tempo de utilização para os recursos da Grade5000 é limitado, sendo estas seis horas o tempo total do experimento, incluindo a preparação do ambiente;

- **Tempo de Preparação:** a preparação do ambiente para execução de cada experimento levava em torno de uma hora e trinta minutos. Isto porque cada reserva de equipamentos da grade exigia a reconfiguração de todo ambiente.

Por estes e outros problemas encontrados, optou-se por utilizar um ambiente controlado para a realização destes experimentos.

5.3.3.1 Resultados

Para demonstrar a homogeneidade obtida no ambiente de execução configurado, a figura 5.17 apresenta o gráfico de tarefas executadas e canceladas para cada escalonador em cada experimento realizado.

É possível observar a estabilidade dos resultados sob o aspecto de número de tarefas executadas e canceladas durante às 19 execuções de experimentos realizadas para cada ambiente. No XtremWebAv, por exemplo, observa-se que o número de tarefas executadas variou pouco ficando sempre próximo de 120 tarefas executadas em cada experimento. Já no XtremWeb, por exemplo, verifica-se que o número de tarefas canceladas ficou sempre próximo de 130 tarefas.

Com este resultado, assume-se que a execução dos experimentos foi realizada em um ambiente homogêneo, com baixa variabilidade, permitindo a realização da comparação dos resultados obtidos utilizando-se medidas de tendência central, como a média aritmética.

A primeira avaliação a ser apresentada é a do número de tarefas executadas e canceladas por escalonador de tarefas. Inicia-se esta avaliação com a apresentação do gráfico da figura 5.18, contendo o número acumulado de tarefas executadas e canceladas por escalonador e hora do experimento.

Observa-se que ao longo de todo o tempo de realização do experimento, o *XtremWebAv* manteve o número médio de tarefas executadas superior a número obtido com o *XtremWeb*. Na primeira hora de experimento o *XtremWebAv* executou em média 30 tarefas contra 18 tarefas executadas pelo *XtremWeb*, sendo em média 40% superior ao *XtremWeb* neste horário.

Esta superioridade do *XtremWebAv* sobre o *XtremWeb* se mantém durante a maior parte dos horários do experimento. No entanto, na quinta hora do experimento(13), observa-se que o *XtremWeb* aproximou-se consideravelmente do *XtremWebAv*, sendo o *XtremWebAv* apenas 8% superior ao *XtremWeb* neste horário.

Para melhor entender o que ocorreu neste horário, foi gerado o gráfico apresentado na figura 5.19. Este gráfico apresenta os resultados obtidos considerando o número de tarefas executadas por escalonador, hora e aplicação. Cada painel do gráfico apresenta o resultado do respectivo escalonador para a respectiva hora do experimento.

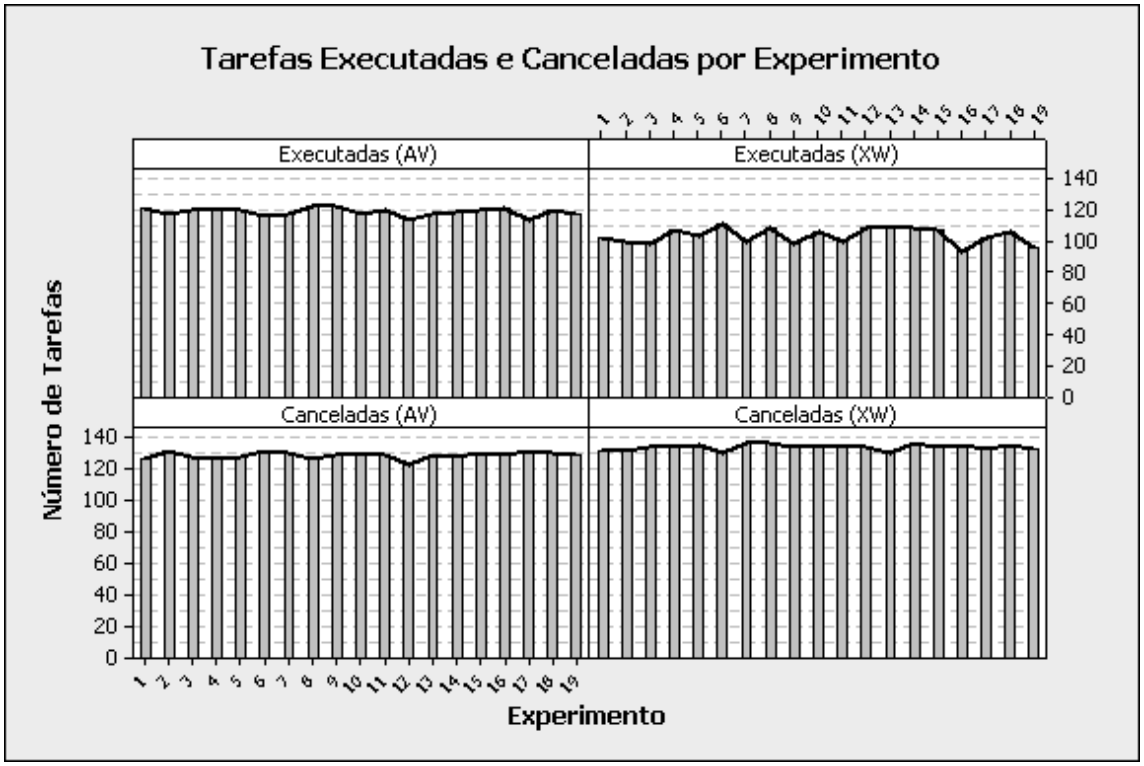

Figura 5.17: Número de tarefas executadas e canceladas por experimento

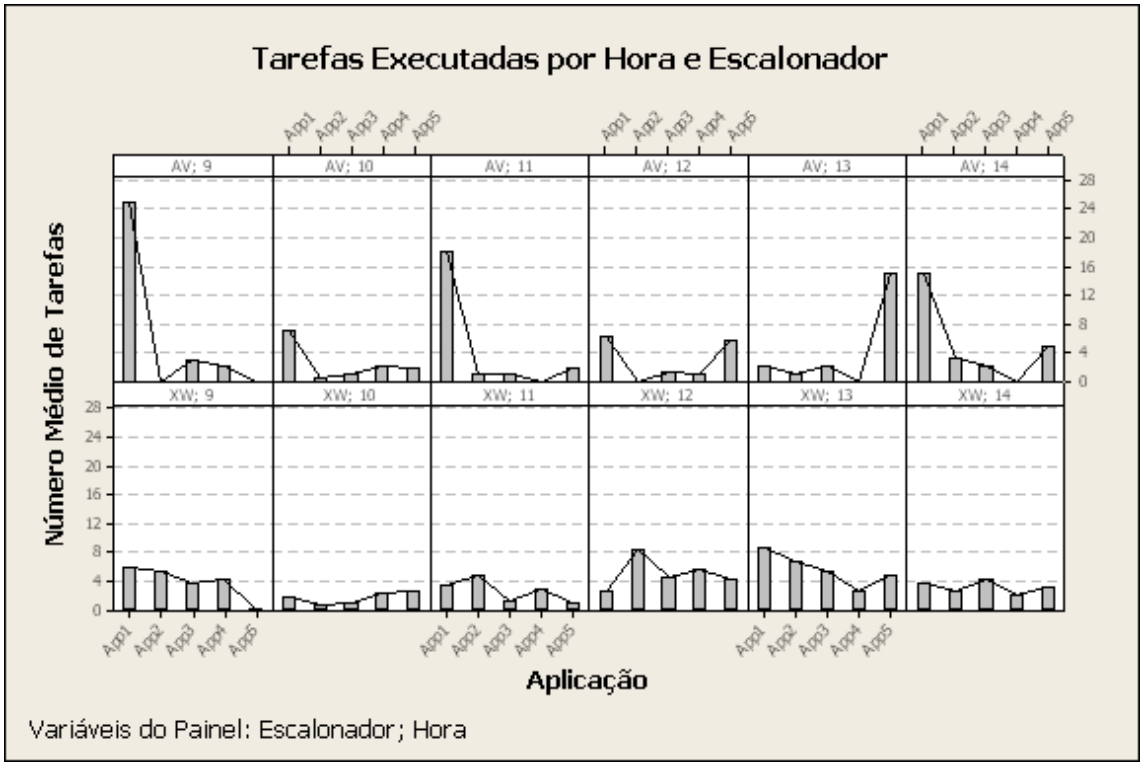

Figura 5.19: Número médio de tarefas executadas por escalonador, horário e aplicação

Ao observar o gráfico da figura 5.19, o primeiro aspecto que se observa é a homogeneidade no número de tarefas executadas por aplicação no *XtremWeb*. O escalonador *FCFS* do *XtremWeb* apresentou uma distribuição justa das tarefas executadas. Observa-se que

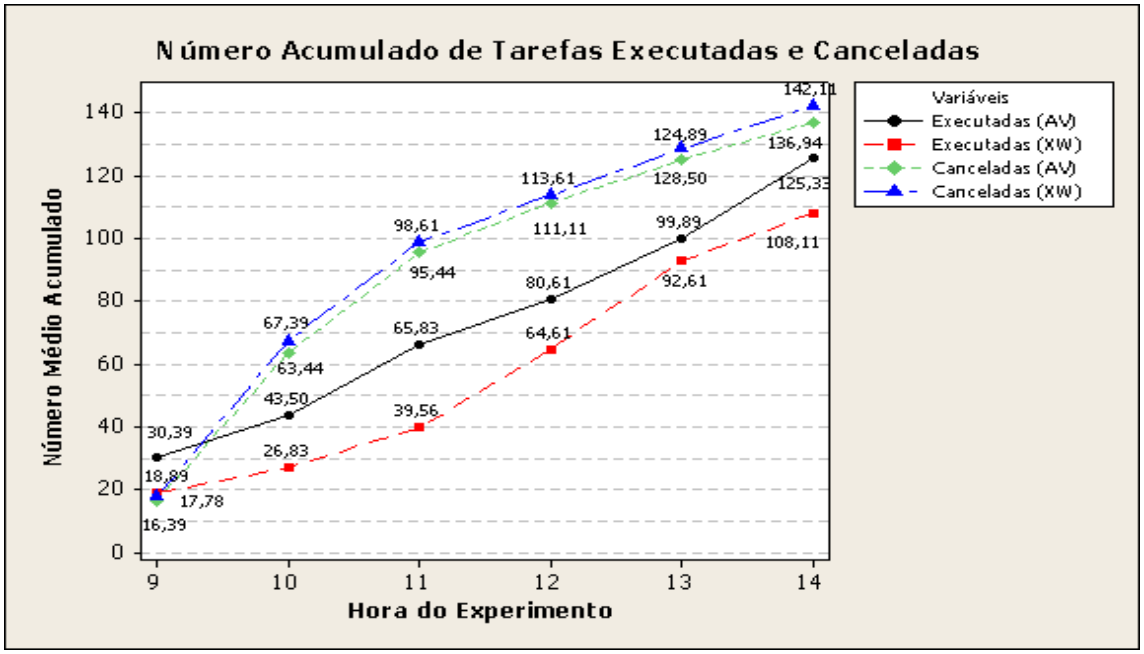

Figura 5.18: Número médio acumulado de tarefas executadas e canceladas por escalonador e horário do experimento

nos diversos horários do experimento, o número de tarefas executadas não foi o mesmo porém, apresentou-se mais uniforme que os resultados obtidos com o *XtremWebAv*.

Como se pode observar na quinta hora do experimento(13), a maioria das tarefas executadas pelo *XtremWebAv* são da aplicação App5, ou seja, a maioria das tarefas executadas pertence a maior aplicação contida no servidor. Devido aos maiores tempos de disponibilidade deste ambiente estarem concentrados nos intervalos compreendidos entre às 12 e 14 horas e a política de escalonamento adotada no escalonador *XtremWebAv* selecionar tarefas maiores para períodos de disponibilidade maiores, foram executadas mais tarefas da aplicação App5. Como estas tarefas levam mais tempo para serem executadas, é natural que o número de tarefas executadas naquela hora do experimento seja menor. Por esta razão ocorre a aproximação do número de tarefas executadas entre os escalonadores.

A política de escalonamento do *XtremWebAv* prioriza a execução de tarefas com maior tempo de duração quando é fornecida uma previsão com tempo de duração de disponibilidade também maior. Isso mostra que quando foram previstos intervalos de disponibilidade maiores que 32 minutos, tempo médio necessário para executar uma tarefa da aplicação App5, o escalonador realmente escolheu a maior tarefa para entregar.

Observa-se também que o número de tarefas executadas da aplicação App1 (aplicação com tarefas de menor tempo de execução) pelo escalonador *XtremWebAv* foi maior em quase todos os horários do experimento. As tarefas da aplicação App1 foram executadas em maior número por dois motivos principais:

1. Intervalos de disponibilidade com pequenas durações são mais frequentes que intervalos com maiores durações, como relatado na Seção 3.3 de análise de disponibilidade deste trabalho. Essa situação associada à política que seleciona tarefas com tempo de execução compatível com a duração prevista, faz com que o escalonador selecione tarefas que requerem pouco tempo de execução;

2. Pelo fato do escalonador sempre selecionar a menor tarefa da fila quando o tempo

de disponibilidade previsto é menor que o tempo necessário para executar qualquer tarefa existente na fila[8]. Este mesmo motivo também serve como justificativa para o elevado número de tarefas canceladas da aplicação App1.

Com relação ao número de tarefas canceladas por escalonador, o *XtremWebAv* também apresentou-se superior ao *XtremWeb*, cancelando menos tarefas durante realização do experimento. Diferentemente do número de tarefas executadas, constata-se que a diferença do número de tarefas canceladas pelos escalonadores manteve-se muito semelhante durante todos os horários do experimento.

No entanto, considerar apenas o número de tarefas executadas e canceladas fornece apenas uma visão parcial dos resultados obtidos com os escalonadores avaliados. Executar mais tarefas não significa que o escalonador tomou as melhores decisões para o ambiente em que está inserido, tampouco significa que o escalonador utilizou melhor os recursos enquanto estavam disponíveis.

Para exemplificar a observação realizada no parágrafo anterior, pode-se considerar dois escalonadores, escalonador A e escalonador B. Se o escalonador A executou 50 tarefas e o escalonador B executou 100 tarefas no mesmo espaço de tempo, pode-se dizer que o escalonador B teve maior *throughput* que o escalonador A. Assim, pode-se constatar que o escalonador B seria melhor que o escalonador A.

No entanto, esta constatação nada assume sobre as características das tarefas executadas. Ao considerar a característica de tempo necessário para executar a tarefa, por exemplo, esta constatação poderia se modificar. Se o escalonador A executou 50 tarefas de 30 minutos cada e o escalonador B executou 100 de 2 minutos cada, pode-se constatar que o escalonador A foi melhor que o escalonador B, pois em um mesmo espaço de tempo conseguiu utilizar melhor os recursos, obtendo um maior tempo de execução.

De fato, dizer que o escalonador A foi melhor que o escalonador B depende da política de escalonamento adotada para o escalonador.

Já se sabe pelos resultados apresentados anteriormente que o *XtremWebAv* superou o *XtremWeb* no que diz respeito ao número de tarefas executadas e canceladas.

Parte-se agora para a avaliação do objetivo de escalonamento proposto para o protótipo *XtremWebAv*.

5.3.4 Avaliação do Objetivo de Escalonamento

O objetivo de escalonamento adotado para o protótipo *XtremWebAv* trata-se da melhor utilização dos recursos disponíveis através da redução dos tempos de execução perdidos, desperdiçados, devido ao cancelamento de tarefas.

Para avaliar o *XtremWeb* e o *XtremWebAv* sobre este aspecto, mensurou-se o tempo médio efetivo de tarefas executadas e o tempo médio desperdiçado em todos os experimentos realizados.

Para obter o tempo médio efetivo de tarefas executadas, foi calculada a média do somatório dos tempos de execução de todas as tarefas completamente executadas, por experimento.

Para obter o tempo desperdiçado em execuções falhas, calculou-se a média do somatório dos tempos de execução perdidos devido ao cancelamento das tarefas, por experimento.

[8] O *AvSched* sempre entrega a menor tarefa existente, devido a chance de execução desta ser maior.

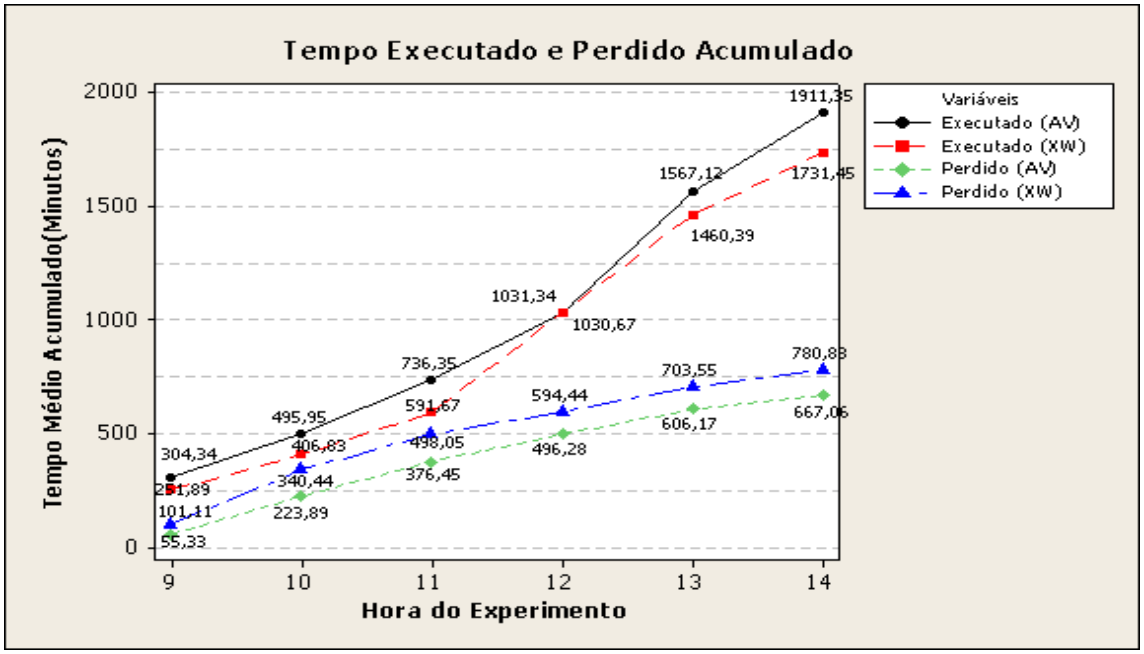

Figura 5.20: Tempos de execução efetivo e perdido acumulado

O gráfico da figura 5.20 apresenta os tempos de execução efetivo e perdido acumulado por escalonador. Cada hora do experimento apresentada no gráfico agrupa a média de tempo de tarefas executadas ou canceladas naquela hora do experimento. Por exemplo, na primeira hora de experimento (9) o tempo de execução efetivo foi de 304,34 minutos, que corresponde a média de tempo de tarefas concluídas entre às 09:00 e 09:59 horas.

Considerando-se o tempo efetivo de execução, observa-se que o *XtremWebAv* manteve o tempo de execução efetivo acumulado superior durante todos os horários dos experimentos. É possível observar que na maioria dos horários do experimento, o *XtremWebAv* utilizou mais eficientemente os recursos ao apresentar tempos de execução efetivos, maiores que os apresentados nos mesmos horários pelo *XtremWeb*.

Apenas na quarta hora do experimento (12), o *XtremWeb* aproximou-se do *XtremWebAv*. Esta aproximação ocorreu devido ao *XtremWebAv* dedicar-se a executar tarefas maiores neste horário. Observando os resultados obtidos na quinta e sexta horas do experimento (13 e 14) é possível identificar a rápida recuperação do *XtremWebAv* sobre o *XtremWeb*. Esta recuperação se deve ao término de execução de tarefas com tempo de execução maior naquele horário, como foi demonstrado no gráfico da figura 5.19, onde na quinta hora do experimento foram concluídas 15 tarefas da aplicação App5.

O resultado aqui exposto é importante pois demonstra a capacidade do escalonador proposto em apresentar um maior tempo de execução efetiva de tarefas, em um mesmo intervalo de tempo, quando comparado com o escalonador do XtremWeb.

Ao final da execução, apresentado na sexta hora do experimento (14), o *XtremWebAv* acumulou um tempo médio de execução efetivo de 1.911 minutos contra os 1.731 minutos de execução efetiva acumulados pelo *XtremWeb* no mesmo período.

Apesar da diferença ser pequena chama-se a atenção para a unidade utilizada que é minuto. Assim, com o resultado obtido o *XtremWebAv* apresentou um ganho de aproximadamente 3 horas de tempo efetivo de execução sobre o *XtremWeb* durante as 6 de execução do experimento.

Isto significa que o *XtremWebAv* seria capaz de executar aproximadamente 6 tarefas a mais da aplicação App5, ou até mesmo, 36 tarefas a mais da App1 que o escalonador

XtremWeb.

Considerando-se o tempo desperdiçado, o *XtremWeb* mostrou-se constantemente menos eficiente que o *XtremWebAv*, tendo o tempo desperdiçado acumulado superior em todos os horários dos experimentos realizados.

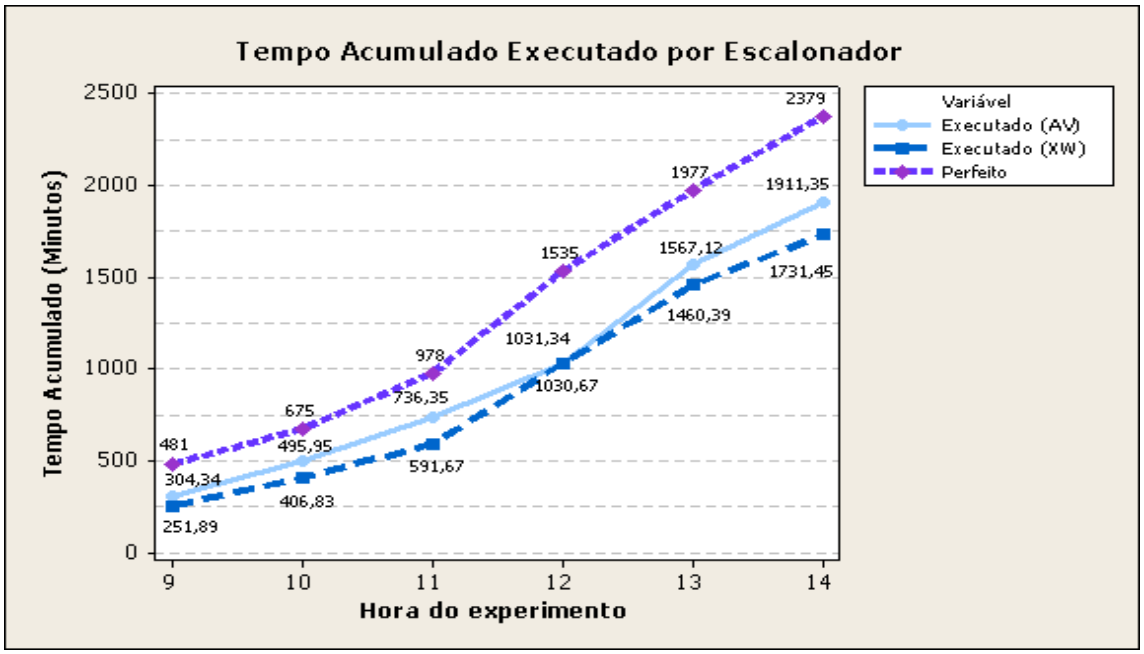

Figura 5.21: Tempo de execução efetivo acumulado X perfeito

Para finalizar a análise dos resultados obtidos, apresenta-se a figura 5.21. Nela encontra-se o gráfico de tempo efetivo acumulado por escalonador, incluindo-se uma terceira variável que apresenta o tempo efetivo acumulado para um escalonador perfeito, ou seja, um escalonador capaz de aproveitar ao máximo o tempo de disponibilidade dos recursos, considerando as tarefas existentes na fila.

Para gerar os dados desta variável, a cada intervalo de disponibilidade existente em um recurso, verificava-se qual tarefa melhor ocuparia o intervalo de disponibilidade encontrado. O tamanho desta tarefa era então computado como tempo de execução. Os tempos de execução forma agrupados e somados por hora do experimento.

Este gráfico foi gerado com o propósito de demonstrar o quão próximo cada escalonador ficou em relação ao escalonador perfeito.

Tanto o *XtremWeb*, quanto o *XtremWebAv* apresentaram a mesma tendência que o escalonador perfeito. Percebe-se, no entanto, que o escalonador *XtremWebAv* sempre se manteve mais próximo do escalonador perfeito. Ao final do experimento, com o escalonador perfeito acumulou-se um total de aproximadamente 2379 minutos de tempo efetivo de execução de tarefas. Com o escalonador *XtremWeb* foram acumulados aproximadamente 1731 minutos de tempo efetivo de execução de tarefas, ou seja, o *XtremWeb* perdeu aproximadamente 648 minutos em tempo de execução de tarefas. Já com o escalonador *XtremWebAv*, o tempo de execução efetivo de tarefas saltou para aproximadamente 1911 minutos, perdendo aproximadamente 468 minutos de tempo de execução.

Apesar da maior eficiência apresentada pelo escalonador *XtremWebAv*, nota-se que mesmo assim ainda existe um grande tempo desperdiçado. Os 468 minutos perdidos pelo *XtremWebAv* totalizam 7.8 horas de tempo de execução perdidos, para o mesmo intervalo de 6 horas de experimento. Com isso, é possível concluir que apesar de apresentar bons

resultados, o escalonador *XtremWebAv* deve ser estudado e novas propostas de escalonamento avaliadas, na tentativa de se obter um aproveitamento ainda melhor do tempo de disponibilidade dos recursos.

5.4 Considerações Finais

No que diz respeito aos experimentos sobre a qualidade do preditor AvSchedP, foi possível identificar que nas três medidas de qualidade adotadas ($\overline{TxA}$, $\overline{Acc}$ e $\overline{Ef}$), o preditor apresentou resultados significativos demonstrando a boa qualidade de previsão do modelo proposto.

A qualidade de previsão do AvSchedP foi avaliada de forma coletiva e pro ambiente experimentado. Como foi observado, em todos os ambientes experimentados, o AvSchedP superou o preditor com o qual foi comparado, o preditor JaccardP.

No que diz respeito a integração do AvSchedP ao sistema de grades oportunistas *XtremWeb*, o *XtremWebAv* demonstrou superioridade sobre o escalonador do *XtremWeb* em todos os experimentos realizados. Os resultados obtidos demonstram que considerar a disponibilidade dos recursos durante o escalonamento de tarefas pode sim proporcionar melhores resultados.

No entanto, percebeu-se que o escalonador do *XtremWebAv* apresentou uma deficiência grave que deve ser melhor estudada. Os resultados demonstram que o escalonador do *XtremWebAv* não seleciona tarefas para execução de um modo justo, ou seja, enquanto algumas aplicações têm tarefas executadas em grandes quantidades, como no caso das aplicações App1 e App5, outras não têm tarefas executadas. Este é um ponto preocupante pois pode causar a postergação indefinida de tarefas.

A falta de seleção de tarefas das outras aplicações para execução atribui-se a baixa frequência ou ausência de intervalos de disponibilidade compatíveis com o tempo de execução destas aplicações. Outra possibilidade poderia ser a dificuldade de prever estes intervalos com o algoritmo de previsão adotado.

Apesar de serem possíveis causas para a seleção injusta de tarefas, o escalonador para um ambiente como o experimentado deveria sofrer modificações para evitar a ocorrência deste tipo de problema. Neste trabalho não se realizou estudos para avaliar possíveis melhorias no escalonador. De fato, o escalonador de tarefas não era o escopo deste trabalho, sendo este escopo restrito ao modelo de previsão de disponibilidade.

No entanto, apesar da injustiça de escalonamento ser um fator indesejado para um escalonador, o fato de executar mais tarefas da aplicação App5 é um resultado importante, visto que tarefas maiores apresentam menores chances de serem executadas por completo. Além disso, os resultados obtidos demonstraram que o AvSchedP pode sim ser utilizado com escalonadores de tarefas em sistemas de grade oportunista e apresentar bons resultados.

6 CONCLUSÃO

Grades oportunistas são frequentemente utilizadas em projetos de pesquisa científica, principalmente onde não se dispõem de grandes recursos financeiros para a aquisição de supercomputadores. Nestes ambientes, frequentemente são utilizados computadores emprestados, que cedem uma fatia de tempo disponível para execução de tarefas computacionais geradas pelo projeto.

Um dos principais problemas deste ambiente origina-se justamente no fato que estimula a existência do mesmo: o uso compartilhado de recursos.

A utilização de recursos não dedicados permite o cancelamento de tarefas em execução a qualquer instante e sem a necessidade de aviso prévio. Esta liberdade é a principal fonte (causa) de problemas de escalonamento de tarefas (consequência) em ambientes como os de grades oportunistas. O cancelamento de uma tarefa leva ao desperdício de tempo de execução e a consequente necessidade de realizar um novo escalonamento desta tarefa.

Como não se pode eliminar a causa dos problemas de escalonamento destes ambientes, pode-se então, reduzir as consequências geradas utilizando-se algum mecanismo. Neste contexto, um mecanismo de previsão de disponibilidade de recursos.

A previsão de disponibilidade é importante em diversos contextos, sendo estes computacionais ou não. Em grades oportunistas, a previsão de disponibilidade de recursos computacionais pode ser utilizada com o objetivo de reduzir o número de erros de escalonamento e o desperdício de tempo de processamento (*wasted time*). A previsão de disponibilidade é então utilizada como informação auxiliar nas tomadas de decisões do escalonador de grades oportunistas.

Artigos relacionados, apresentados neste trabalho, já demonstraram a importância em se utilizar tal mecanismo com escalonadores de grades oportunistas. No entanto, neste trabalho destacou-se a carência de preditores de disponibilidade para diferentes tipos de dados históricos de utilização mantidos principalmente, quando os dados históricos disponíveis são binários.

O AvSchedP surgiu como uma nova proposta de previsão de duração de disponibilidade para grades oportunistas. A análise de disponibilidade realizada na Seção 3.3 do Capítulo 3 deste trabalho, e os trabalhos relacionados apresentados no mesmo Capítulo, demonstraram a necessidade de se propor um modelo de previsão de disponibilidade capaz de realizar a previsão de disponibilidade de forma dinâmica, individual e possível de ser realizada localmente pelos recursos participantes de uma Grade Oportunista. O AvSchedP se propõe a realizar a previsão de disponibilidade, baseando-se nestes princípios.

O principal componente deste modelo, chamado de Preditor, realiza a previsão de disponibilidade de um recurso, baseando-se no histórico de utilização do mesmo. Este

componente utiliza um novo algoritmo de previsão de disponibilidade, também proposto neste trabalho, que quantifica a similaridade existente entre a utilização corrente do recurso e uma coleção de dados históricos de utilização deste mesmo recurso, com o objetivo de encontrar um comportamento no passado consideravelmente similar ao que está ocorrendo, podendo este ser utilizado na previsão do futuro. Para medir esta similaridade, o algoritmo proposto empregou a equação utilizada no cálculo do coeficiente de Hamann, equação esta que mede a similaridade entre duas sequências de dados binários.

O AVSchedP é então, uma proposta de previsão de disponibilidade que pode ser aplicada a dados binários. Devido a esta característica, o AvSchedP pode ser utilizado com os mais diversos tipos de dados inclusive os provenientes de medidas continuas. Para tanto, basta existir a possibilidade de realizar a classificação binária da medida em questão.

O AvSchedP apresentou-se como uma alternativa interessante para a realização de previsão de disponibilidade em ambientes altamente voláteis, como o existente em grades oportunistas. Os experimentos realizados demonstraram desde a qualidade das previsões de disponibilidade realizadas pelo componente Preditor, até a aplicabilidade do modelo proposto em uma Grade Oportunista experimental, utilizando o sistema XtremWeb.

Para medir a qualidade das previsões obtidas, foram definidas e utilizadas três medidas diferentes: (a) a taxa de acertos de previsão; (b) a acurácia de previsão; e (c) a efetividade de previsão. Os resultados obtidos com estas medidas foram apresentados de forma coletiva (resultados obtidos com a união dos resultados obtidos em cada ambiente) e por ambiente (resultados por ambiente experimentado).

Nos resultados obtidos para a taxa de acertos de previsão ($\overline{TxA}$) coletiva, a superioridade do AvSchedP foi demonstrada ao obter a média de 74,7% de acertos contra os 57,6 % de acertos obtidos com o preditor aqui chamado de JaccardP. A diferença destas medidas indica que uma grade oportunista utilizando o AvSchedP receberia 17% a mais de previsões de tempo de disponibilidade corretas uma grade oportunista utilizando o JaccardP. Neste caso, previsões corretas remetem a tempos de disponibilidade que seriam previstos e que realmente ocorreriam no recurso com duração maior ou igual ao previsto.

Nos resultados obtidos para a acurácia de previsão ($\overline{Acc}$) coletiva, o AvSchedP demonstrou-se superior novamente ao apresentar a média de acurácia de aproximadamente 95% contra aproximadamente 60% obtido com o JaccardP. A diferença entre estas medidas indica que o AvSchedP forneceria previsões aproximadamente 35% mais confiáveis do que as previsões fornecidas pelo JaccardP. Neste caso, previsões mais confiáveis remetem ao fato dos tempos de duração de disponibilidade que seriam previstos pelo AvSchedP seriam mais próximos dos tempos que realmente viriam a ocorrer nos recursos.

Nos resultados obtidos com a efetividade de previsão ($\overline{Ef}$) coletiva, O AvSchedP definitivamente mostrou sua superioridade ao apresentar 71,7% de efetividade contra os 44,5% de efetividade obtido com o JaccardP. A diferença entre estas medidas indica o quanto superior o AvSchedP foi em relação ao JaccardP, sendo esta diferença de 27,2%. Neste caso, esta medida indica a superioridade por considerar a relação entre taxa de acertos ($\overline{TxA}$) e acurácia ($\overline{Acc}$), indicando o quanto efetiva é a previsão de disponibilidade do preditor analisado. A acurácia do AvSchedP superou o preditor baseado no coeficiente de Jaccard, preditor este utilizado na comparação dos resultados, principalmente quando o tamanho do intervalo de disponibilidade a ser previsto for maior.

Já nos experimentos realizados para avaliar a integração do modelo AvSchedP a um sistema de grade oportunista, os resultados obtidos com a utilização de uma política de escalonamento que utiliza a informação de duração de disponibilidade nas decisões de escalonamento, demonstraram a possibilidade de ser obter ganhos efetivos com a utilização

do modelo.

Conforme apresentado nos experimentos realizados, o XtremWebAv superou os resultados obtidos com o XtremWeb em sua versão original. O XtremWebAv superou o XtremWeb tanto em número de tarefas executadas, quanto no tempo de execução efetivo obtido durante todo o tempo do experimento.

O número de tarefas canceladas demonstraram que nas 6 horas de duração do experimento, foram canceladas em média 142 tarefas pelo escalonador original do XtremWeb contra 136 tarefas canceladas pelo escalonador utilizando o AvSchedP. Neste mesmo tempo de experimento, XtremWeb conseguiu concluir a execução de 108 tarefas contra às 125 executadas pelo escalonador utilizando o AvSchedP.

No que diz respeito ao tempo gasto na execução de tarefas que realmente foram concluídas, o escalonador original do XtremWeb conseguiu atingir em média 1731 minutos durante às 6 horas de execução do experimento. Já o escalonador utilizando o AvSchedP conseguiu atingir em média 1911 minutos de tempo gasto na execução de tarefas que realmente foram concluídas. Este resultado foi importante, pois demonstrou a possibilidade de se obter aproximadamente 3 horas de tarefas executadas a mais ao se utilizar as informações de previsão fornecidas pelo AvSchedP durante o escalonamento de tarefas.

O escalonador utilizando o AvSchedP também foi superior ao escalonador original do XtremWeb no que diz respeito ao tempo gasto na execução de tarefas que foram canceladas (*wasted time*). O escalonador original do XtremWeb desperdiçou 780 minutos em média, no processamento de tarefas que foram canceladas, contra os 667 minutos desperdiçados pelo escalonador utilizando o AvSchedP.

No entanto, o AvSchedP ainda possui pontos que podem evoluir, principalmente no que diz respeito a taxa de acertos de previsão. Apesar da previsão de disponibilidade apresentar bons resultados, a taxa de acertos reduz consideravelmente com o aumento do tempo de duração de disponibilidade a ser previsto.

Neste sentido, acredita-se que a pesquisa sobre este assunto deva prosseguir, na tentativa de aperfeiçoar o modelo gerando-se previsões de disponibilidade capazes de se atingir taxas de acertos ainda maiores.

Além desta evolução, diversos trabalhos futuros foram identificados durante a realização da pesquisa. Tem-se portanto como trabalhos futuros, um conjunto de atividades que encontram-se enumeradas abaixo:

1. Monitoramento de recursos pertencentes a outros ambientes com o objetivo de obter diferentes características de utilização de recursos;

2. Identificação de outras políticas de escalonamento baseadas na disponibilidade de recursos para experimentação com o modelo AvSchedP ;

3. Realizar experimentos com o ***AvSchedP*** em ambientes de maior escala;

4. Integrar o modelo de previsão proposto com projetos pertencentes a outros contextos como o JavaRMS (GOMES, 2008), GRAND (MANGAN, 2006) e FreeMMG 2 (CECIN et al., 2004).

5. Implementação do modelo AvSchedP em simuladores de grades oportunistas com o objetivo de avaliar novas políticas de escalonamento utilizando a disponibilidade de recursos.

De fato, este trabalho já encontra-se em evolução. Alguns dos trabalhos futuros já encontram-se em pesquisa. No Centro Universitário La Salle, encontra-se em desenvolvimento um trabalho de conclusão que tem como objetivo, propor e realizar experimentos com diferentes políticas de escalonamento para o AvSchedP. Nesta mesma instituição, encontra-se em desenvolvimento outro trabalho de conclusão que tem por objetivo, adaptar e integrar o modelo AvSchedP ao modelo de escalonamento do GRAND (MANGAN, 2006), consideravelmente diferente do modelo de grades oportunistas.

REFERÊNCIAS

ABDENNADHER, N.; BOESCH, R. A scheduling algorithm for high performance peer-to-peer platform. In: COREGRID 2006, UNICORE SUMMIT 2006, PETASCALE COMPUTATIONAL BIOLOGY AND BIOINFORMATICS CONFERENCE ON PARALLEL PROCESSING, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2007. p.126–137.

ANDERSON, D. P. BOINC: a system for public-resource computing and storage. In: TH PROCEEDINGS OF THE IEEE/ACM INTERNATIONAL WORKSHOP ON GRID COMPUTING, 5., Washington, DC, USA. **Anais...** IEEE Computer Society, 2004. p.4–10.

ANDERSON, D. P.; MCLEOD, J. Local Scheduling for Volunteer Computing. **Parallel and Distributed Processing Symposium, International**, Los Alamitos, CA, USA, v.0, p.477, 2007.

ANURAG, A.; GUY, E.; JOEL, S. The utility of exploiting idle workstations for parallel computation. In: ACM SIGMETRICS INTERNATIONAL CONFERENCE ON MEASUREMENT AND MODELING OF COMPUTER SYSTEMS, 1997., New York, NY, USA. **Proceedings...** ACM, 1997. p.225–234.

BALLIER, A. et al. Simulating Grid Schedulers with Deadlines and Co-Allocation. In: CoreGRID integration workshop, Pisa, Italy. **Anais...** [S.l.: s.n.], 2005.

BASNEY, J.; LIVNY, M. Deploying a High Throughput Computing Cluster. In: BUYYA, R. (Ed.). **High Performance Cluster Computing**: architectures and systems, volume 1. [S.l.]: Prentice Hall PTR, 1999.

BEGOLE, J. B. et al. Work rhythms: analyzing visualizations of awareness histories of distributed groups. In: ACM CONFERENCE ON COMPUTER SUPPORTED COOPERATIVE WORK, 2002., New York, NY, USA. **Proceedings...** ACM, 2002. p.334–343.

BEGOLE, J. B.; TANG, J. C.; HILL, R. Rhythm modeling, visualizations and applications. In: TH PROCEEDINGS OF THE ANNUAL ACM SYMPOSIUM ON USER INTERFACE SOFTWARE AND TECHNOLOGY, 16., New York, NY, USA. **Anais...** ACM, 2003. p.11–20.

BEHSAZ, B.; JAFERIAN, P.; MEYBODI, M. R. Comparison of Global Computing with Grid Computing. **Parallel and Distributed Computing Applications and Technologies, International Conference on**, Los Alamitos, CA, USA, v.0, p.531–534, 2006.

BREVIK, J.; NURMI, D.; WOLSKI, R. Automatic methods for predicting machine availability in desktop Grid and peer-to-peer systems. In: IEEE INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, 2004., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2004. p.190–199.

BREVIK, J.; WOLSKI, R.; NURMI, D. Quantifying Machine Availability in Networked and Desktop Grid Systems. In: IN PROCEEDINGS OF CCGRID04. **Anais...** [S.l.: s.n.], 2003.

BYUN, E. et al. Scheduling Scheme based on Dedication Rate in Volunteer Computing Environment. In: TH PROCEEDINGS OF THE THE INTERNATIONAL SYMPOSIUM ON PARALLEL AND DISTRIBUTED COMPUTING, 4., Washington, DC, USA. **Anais...** IEEE Computer Society, 2005. p.234–241.

BYUN, E. et al. Advanced Stochastic Host State Modeling to Reliable Computation in Global Computing Environment. In: EUC. **Anais...** [S.l.: s.n.], 2006. p.183–192.

CALDER, B. et al. The Entropia Virtual Machine for Desktop Grids. In: ACM/USENIX INTERNATIONAL CONFERENCE ON VIRTUAL EXECUTION ENVIRONMENTS. **Proceedings...** [S.l.: s.n.], 2005. p.186–196.

CALDWELL, C. **Great Internet Mersenne Prime Search - GIMPS**: finding world record primes since 1996. Disponível em: <http://www.mersenne.org>. Acessado em: junho 2010.

CAMENISCH, J. et al. **A Statistical Approach to Predicting Workstation Idle Times**. 1999.

CAPPELLO, F. et al. Computing on large-scale distributed systems: xtremweb architecture, programming models, security, tests and convergence with grid. **Future Generation Comp. Syst.**, [S.l.], v.21, n.3, p.417–437, 2005.

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. **IEEE Trans. Softw. Eng.**, Piscataway, NJ, USA, v.14, n.2, p.141–154, 1988.

CECIN, F. R. et al. FreeMMG: a scalable and cheatresistant distribution model for internet games. dsrt. In: PROC. OF INTERNATIONAL SYMPOSIUM ON DISTRIBUTED SIMULATION AND REAL-TIME APPLICATIONS. **Anais...** [S.l.: s.n.], 2004. p.83–90.

CHIEN, A. A. **Architecture of a Commercial Enterprise Desktop Grid**: the entropia system. [S.l.]: John Wiley & Sons, Ltd, 2003. 337–350p.

CHIEN, A. et al. Entropia: architecture and performance of an enterprise desktop grid system. **Journal of Parallel and Distributed Computing**, [S.l.], v.63, p.597–610, 2003.

CHOI, S. et al. **A Taxonomy of Desktop Grid Systems Focusing on Scheduling**. [S.l.]: Department of Computer Science and Engeering, Korea University, 2006. Tech. Rep. KU-CSE-2006-1120-01. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.75.5256>. Acessado em: agosto 2007.

CHOI, S. et al. **A Taxonomy of Desktop Grids and its Mapping to State-of-the-Art Systems**. [S.l.]: Grid Computing and Distributed Systems Laboratory, The University of Melbourne, Australia, 2008. Technical Report, GRIDS-TR-2008-3. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.146.4696>. Acessado em: janeiro 2009.

CHOOSING BOINC projects. Disponível em: <http://boinc.berkeley.edu/projects.php>. Acessado em: outubro 2010.

CLIMATEPREDICTION.NET. Disponível em: <http://climateprediction.net>. Acessado em: junho 2010.

CONSTANTINESCU-FULØP, Z. **A Desktop Grid Computing Approach for Scientific Computing and Visualization**. 2008. 246p. Tese (Doutorado em Ciência da Computação) — Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering. (2008:153).

DOMINGUES, P.; MARQUES, P.; SILVA, L. DGSchedSim: a trace-driven simulator to evaluate scheduling algorithms for desktop grid environments. In: TH PROCEEDINGS OF THE EUROMICRO INTERNATIONAL CONFERENCE ON PARALLEL, DISTRIBUTED, AND NETWORK-BASED PROCESSING, 14., Washington, DC, USA. **Anais...** IEEE Computer Society, 2006. p.83–90.

FEDAK, G. et al. XtremWeb: a generic global computing system. In: INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2001. p.582.

FOSTER, I. The Anatomy of the Grid: enabling scalable virtual organizations. **INTERNATIONAL JOURNAL OF SUPERCOMPUTER APPLICATIONS**, [S.l.], v.15, n.3, p.2001, 2001.

FRANK, I.; TODESCHINI, R. **The Data Analysis Handbook**. Amsterdam, The Netherlands: Elsevier Science, 1994.

GOMES, D. S. **JavaRMS**: um sistema de gerência de dados para grades baseado num modelo par-a-par. 2008. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Ufrgs, Porto Alegre.

HAN, J.; KAMBER, M. **Data Mining**: concepts and techniques. 2.ed. [S.l.]: Morgan Kaufmann Publishers, 2006. (The Morgan Kaufmann Series).

HUANG et al. Availability-based Task Monitoring and Adaptation Mechanism in Desktop Grid System. In: TH INTERNATIONAL CONFERENCE ON GRID AND COOPERATIVE COMPUTING, 6., Washington, DC, USA. **Anais...** IEEE Computer Society, 2007. p.444–450.

IOSUP, A.; OZAN SONMEZ, M. J. ans; EPEMA, D. H. J. On the Dynamic Resource Availability in Grids. In: IEEE/ACM INTERNATIONAL CONFERENCE ON GRID COMPUTING, 8. **Anais...** [S.l.: s.n.], 2007.

JAVADI, B. et al. Mining for Statistical Availability Models in Large-Scale Distributed Systems: an empirical study of seti@home. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON MODELLING, ANALYSIS AND SIMULATION OF COMPUTER AND TELECOMMUNICATION SYSTEMS (MASCOTS), 17. **Anais...** [S.l.: s.n.], 2009.

KONDO, D. **Scheduling task parallel applications for rapid turnaround on desktop grids**. 2005. Tese (Doutorado em Ciência da Computação) — , La Jolla, CA, USA. Chair-Casanova, Henri and Chair-Chien, Andrew A.

KONDO, D.; ANDRZEJAK, A.; ANDERSON, D. P. On Correlated Availability in Internet Distributed Systems. In: IEEE/ACM INTERNATIONAL CONFERENCE ON GRID COMPUTING (GRID), Tsukuba,Japan. **Anais...** [S.l.: s.n.], 2008.

KONDO, D.; CHIEN, A. A.; CASANOVA, H. Scheduling Task Parallel Applications for Rapid Turnaround on Enterprise Desktop Grids. **J. Grid Comput.**, [S.l.], v.5, n.4, p.379–405, 2007.

KONDO, D. et al. Characterizing and Evaluating Desktop Grids: an empirical study. In: IN PROCEEDINGS OF THE INTERNATIONAL PARALLEL AND DISTRIBUTED PROCESSING SYMPOSIUM. **Anais...** [S.l.: s.n.], 2004.

KONDO, D. et al. **Resource Availability in Enterprise Desktop Grids**. [S.l.]: Dept of Computer Science INRIA, 2006. Technical Report, Disponível em: <http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.136.2505>. Acessado em: setembro 2007.

KONDO, D. et al. Characterizing resource availability in enterprise desktop grids. **Future Gener. Comput. Syst.**, Amsterdam, The Netherlands, v.23, n.7, p.888–903, 2007.

KONDO, D. et al. The Failure Trace Archive: enabling comparative analysis of failures in diverse distributed systems. **Cluster Computing and the Grid, IEEE International Symposium on**, Los Alamitos, CA, USA, v.0, p.398–407, 2010.

KUMAR, V. et al. **Introduction to parallel computing**: design and analysis of algorithms. Redwood City, CA, USA: Benjamin-Cummings Publishing Co., Inc., 1994.

LETRAS, A. B. de (Ed.). **Dicionário escolar da língua portuguesa**. [S.l.]: Ibep, 2008.

LHC@HOME. Disponível em: <http://lhcathome.cern.ch/lhcathome/>. Acessado em: junho 2010.

LILJA, D. J. **Measuring Computer Performance**: a practitionerś guide. 1st.ed. [S.l.]: Cambridge University Press, 2000.

LITZKOW, M.; LIVNY, M.; MUTKA, M. Condor - A Hunter of Idle Workstations. In: INTERNATIONAL CONFERENCE OF DISTRIBUTED COMPUTING SYSTEMS, 8. **Proceedings...** [S.l.: s.n.], 1988.

LODYGENSKY, O. et al. XtremWeb & Condor sharing resources between Internet connected Condor pools. In: INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, 3., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2003. p.382.

MALARIACONTROL.NET. Disponível em: <http://www.malariacontrol.net/>. Acessado em: junho 2010.

MANGAN, P. K. V. **GRAND**: um modelo de gerenciamento hierárquico de aplicações em ambiente de computação em grade. 2006. Tese (Doutorado em Ciência da Computação) — Departamento de Engenharia de Sistemas e Computação, COPPE/UFRJ.

MICKENS, J. W.; NOBLE, B. D. Exploiting Availability Prediction in Distributed Systems. In: NETWORKED SYSTEMS DESIGN & IMPLEMENTATION, 3. **Proceedings...** [S.l.: s.n.], 2006.

MORETTIN, P. A.; BUSSAB, W. D. O. **Estatística Básica**. 6.ed. [S.l.]: Saraiva, 2010. 540p.

MORETTIN, P. A.; TOLOI, C. C. **Análise de Séries Temporais**. [S.l.]: Edgard Blücher, 2006. 544p.

NABRZYSKI, J.; SCHOPF, J. M.; WEGLARZ, J. **GRID RESOURCE MANAGEMENT – State of the art and future trends**. 2003.

NADEEM, F. et al. **Availability-based Resource Selection Risk Analysis in the Grid**. [S.l.]: Institute on Grid Information, Resource and Workflow Monitoring Services , CoreGRID - Network of Excellence, 2008. (TR-0169).

NADEEM, F. et al. **An Evaluation of Availability Comparison and Prediction for Optimized Resource Selection in the Grid**. [S.l.]: Springer, 2008. 63-76p. (CoreGRID).

NADEEM, F.; PRODAN, R.; FAHRINGER, T. Characterizing, Modeling and Predicting Dynamic Resource Availability in a Large Scale Multi-purpose Grid. In: CCGRID. **Anais...** [S.l.: s.n.], 2008. p.348–357.

PENTLAND, A.; LIN, A. Modeling and Prediction of Human Behavior. **Neural Computation**, [S.l.], v.11, p.229–242, 1995.

PEREIRA, J. C. R. **Análise de Dados Qualitativos**: estratégias metodológicas para as ciências da saúde, humanas e sociais. [S.l.]: Editora Universidade de São Paulo, 2004.

RAHMAN, M.; HASSAN, M. R.; BUYYA, R. Jaccard Index based Availability Prediction in Enterprise Grids. In: INTERNATIONAL CONFERENCE ON COMPUTATIONAL SCIENCE, 10., Amsterdam. **Proceedings...** Elsevier - Procedia Computer Science, 2010.

REAL, R. et al. Tratamento da incerteza no escalonamento de recursos em Pervasive Computing. In: IADIS INTL. CONFERENCE WWW/INTERNET. **Anais...** [S.l.: s.n.], 2003.

REN, X.; EIGENMANN, R. Empirical Studies on the Behavior of Resource Availability in Fine-Grained Cycle Sharing Systems. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 2006., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2006. p.3–11.

REN, X. et al. Resource Availability Prediction in Fine-Grained Cycle Sharing Systems. **High-Performance Distributed Computing, International Symposium on**, Los Alamitos, CA, USA, v.0, p.93–104, 2006.

REN, X. et al. Prediction of Resource Availability in Fine-Grained Cycle Sharing Systems Empirical Evaluation. **J. Grid Comput.**, [S.l.], v.5, n.2, p.173–195, 2007.

ROMESBURG, H. **Cluster Analysis for Researchers**. Belmont, California: Lifetime Learning Publications, 1984.

ROOD, B.; LEWIS, M. J. Multi-state grid resource availability characterization. In: IEEE/ACM INTERNATIONAL CONFERENCE ON GRID COMPUTING, 8., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2007. p.42–49. (GRID '07).

ROOD, B.; LEWIS, M. J. Resource Availability Prediction for Improved Grid Scheduling. In: IEEE INTERNATIONAL CONFERENCE ON ESCIENCE, 2008., Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2008. p.711–718.

SARMENTA, L. F. G. **Volunteer Computing**. 2001. Tese (Doutorado em Ciência da Computação) — Massachusetts Institute of Technology.

SILVA, D. P. da; CIRNE, W.; BRASILEIRO, F. V. **Robust Scheduling for Bag-of-Tasks Applications Through Task Replication**. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.18.1920>. Acessado em: março 2008.

SULLIVAN WOODRUFF T., I. et al. **A new major SETI project based on Project SERENDIP data and 100,000 personal computers**. [S.l.]: Bologna, Italy, 1997. 729p.

SUPERLINK@TECHNION. Disponível em: <http://cbl-link02.cs.technion.ac.il/superlinkattechnion/>. Acessado em: junho 2010.

TAUFER, M. et al. Metrics for Effective Resource Management in Global Computing Environments. In: INTERNATIONAL CONFERENCE ON E-SCIENCE AND GRID COMPUTING, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2005. p.204–211.

THAIN, D.; TANNENBAUM, T.; LIVNY, M. **Distributed Computing in Practice**: the condor experience. [S.l.]: John Wiley & Sons, 2004.

TOTH, D.; FINKEL, D. Characterizing resource availability for volunteer computing and its impact on task distribution methods. In: SEPADS'07: PROCEEDINGS OF THE 6TH WSEAS INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, PARALLEL AND DISTRIBUTED SYSTEMS, Stevens Point, Wisconsin, USA. **Anais...** World Scientific and Engineering Academy and Society (WSEAS), 2007. p.107–114.

ULRIK, C.; JAKOB, S.; PEDERSEN, G. **Developing Distributed Computing Solutions Combining Grid Computing and Public Computing M.Sc. Thesis**. 2005.

VIDYARTHI, D. P. et al. **Scheduling in Distributed Computing Systems**: analysis, design and models. [S.l.]: Springer Publishing Company, Incorporated, 2008.

WEICKER, R. P. An Overview of Common Benchmarks. **Computer**, Los Alamitos, CA, USA, v.23, p.65–75, 1990.

WILLIAMS, C. Applied Multivariate Data Analysis (2nd Edition). **The American Statistician**, [S.l.], v.56, p.248–249, 2002.

WRIGHT, D. **Cheap cycles from the desktop to the dedicated cluster**: combining opportunistic and dedicated scheduling with condor. 2001.

WU, J. **Distributed System Design**. 1st.ed. Boca Raton, FL, USA: CRC Press, Inc., 1998. 496p.

YORK, R. **Benchmarking in context**: dhrystone. ARM Ltd. white paper. Disponível em: <http://www.arm.com/pdfs/Dhrystone.pdf>. Acessado em: maio 2010.

APÊNDICE A SIMILARIDADE ENTRE VARIÁVEIS BINÁRIAS

O comportamento de um recurso é dado pelo conjunto de transições de estado de disponibilidade ao longo do tempo. Como apresentado anteriormente, um recurso pode apresentar dois estados de disponibilidade representados por (0) quando o recurso encontra-se disponível e (1) quando o recurso encontra-se indisponível. Consequentemente, o comportamento de um recurso ao longo do tempo pode ser modelado por uma variável qualitativa.

Variáveis qualitativas são mensuradas em escala nominal, sendo a mais simples delas, a variável binária (ROMESBURG, 1984). As variáveis binárias apresentam apenas dois estados que podem ser vistos como sim ou não, homem ou mulher, ou até mesmo como disponível e indisponível.

Em algumas áreas de pesquisa, como na medicina, biologia entre outras, a comparação de dados qualitativos é muito utilizada. Com esta necessidade, diversas equações foram criadas por pesquisadores para possibilitar a comparação destes dados. Uma das formas de comparação destes dados é a identificação da similaridade entre dois conjuntos de dados, ou seja, a similaridade entre os dados gerados por duas variáveis binárias.

A.1 Coeficientes de Similaridade

Para determinar a similaridade entre duas variáveis binárias, foram propostos diversos coeficientes de similaridade, cada um com seus próprios objetivos. Os coeficientes de similaridade de variáveis binárias são aplicados aos dados condensados em uma tabela de contingência ou, matriz de dados. A figura A.1 apresenta a variável ca, que representa o comportamento atual de um recurso [1] e a variável ch_i que representa o histórico i de um recurso [2]. Estas variáveis podem ser vistas como vetores, onde cada elemento do vetor representa o estado de disponibilidade do recurso em um instante de tempo j, onde j corresponde ao índice deste vetor.

A geração da tabela de contingência é realizada com base nos valores contidos nas variáveis ca e ch_i. A célula A da tabela corresponde a contagem das ocorrências onde a variável ca em um instante j e a variável ch_i no mesmo instante j apresentavam o valor 0. A célula B é formada pelas ocorrências onde a variável ca em um instante j apresentava o valor 0 e a variável ch_i no mesmo instante j apresentava o valor 1. A célula C é formada pelas ocorrências onde a variável ca em um instante j apresentava o valor 1 e a variável ch_i no mesmo instante j apresentava o valor 0. E finalmente, a célula D é formada pelas

[1] A obtenção do comportamento atual encontra-se detalhada no modelo proposto.

[2] A obtenção do comportamento histórico encontra-se detalhada no modelo proposto.

ocorrências onde a variável ca e a variável ch_i no mesmo instante j apresentavam o valor 1.

Após a geração da tabela de contingência, aplica-se a equação correspondente ao coeficiente selecionado sobre os dados da tabela, obtendo-se assim a similaridade entre as variáveis comparadas.

Cada coeficiente de similaridade tem um objetivo diferente, devendo ser selecionado conforme o objetivo do pesquisador. Alguns destes coeficientes encontram-se descritos abaixo. Para facilitar o entendimento, as variáveis de cada equação correspondem ao nome das variáveis apresentadas na figura A.1. Será utilizada a letra $Ccach_i$ para representar o coeficiente de similaridade, para fins de generalização. No entanto, a figura A.1 apresenta apenas o cálculo do coeficiente de Hamann, detalhado abaixo.

- **Coeficiente Jaccard** - O coeficiente de Jaccard ignora a ocorrência de pares 0-0(D). A similaridade total, representada por $Ccach_i = 1.0$, ocorre quando não ocorreram pares 1-0 e 0-1($A = B = 0.0$). A dissimilaridade total, representada por $Ccach_i = 0.0$, ocorre quando não ocorreram pares 1-1, ou seja, ($A = 0$).

$$Ccach_i = \frac{A}{A + B + C} \tag{A.1}$$

- **Coeficiente Hamann** - O coeficiente de Hamann considera todos os pares possíveis (A, B, C e D) em seu numerador e denominador. Este coeficiente considera os pares (1-1) e (0-0) como casos de sucesso e os pares (1-0) e (0-1) como casos de insucesso. O coeficiente mede a proporção de casos de sucesso, subtraindo-se os casos de insucesso, em relação ao número total de ocorrências e o seu resultado pode variar entre -1.0 (dissimilaridade total) e 1.0 (similaridade total) sendo $Ccach_i = -1.0$, quando não ocorre pares 1-1 nem pares 0-0 ($A = D = 0$) e $Ccach_i = 1.0$, quando não ocorrerem pares 1-0(b) e 0-1(c) ou seja, ($B = C = 0$).

$$Ccach_i = \frac{(A + D) - (B + C)}{(A + D) + (B + C)} \tag{A.2}$$

- **Coeficiente Sorenson** - O coeficiente de Sorenson dobra o peso da ocorrência de pares 1-1(A) e ignora a ocorrência de pares 0-0(D). Este coeficiente pode variar de 0.0 (dissimilaridade total) a 1.0 (similaridade total) sendo $Ccach_i = 0.0$, quando não ocorre pares 1-1 ($A = 0$) e $Ccach_i = 1.0$, quando não ocorrerem pares 1-0(b) e 0-1(c) ou seja, ($B = C = 0$).

$$Ccach_i = \frac{2A}{2A + B + C} \tag{A.3}$$

- **Coeficiente Russel e Rao** - O coeficiente de Russel e Rao representa a proporção de ocorrências de pares 1-1 entre todos os pares obtidos. Este coeficiente pode variar de 0.0 (dissimilaridade total) a 1.0 (similaridade total) sendo $Ccach_i = 0$, quando não ocorre pares 1-1 ($A = 0$) e $Ccach_i = 1$, quando todos os pares forem 1-1 ($B = C = D = 0$).

$$Ccach_i = \frac{A}{A + B + C + D} \tag{A.4}$$

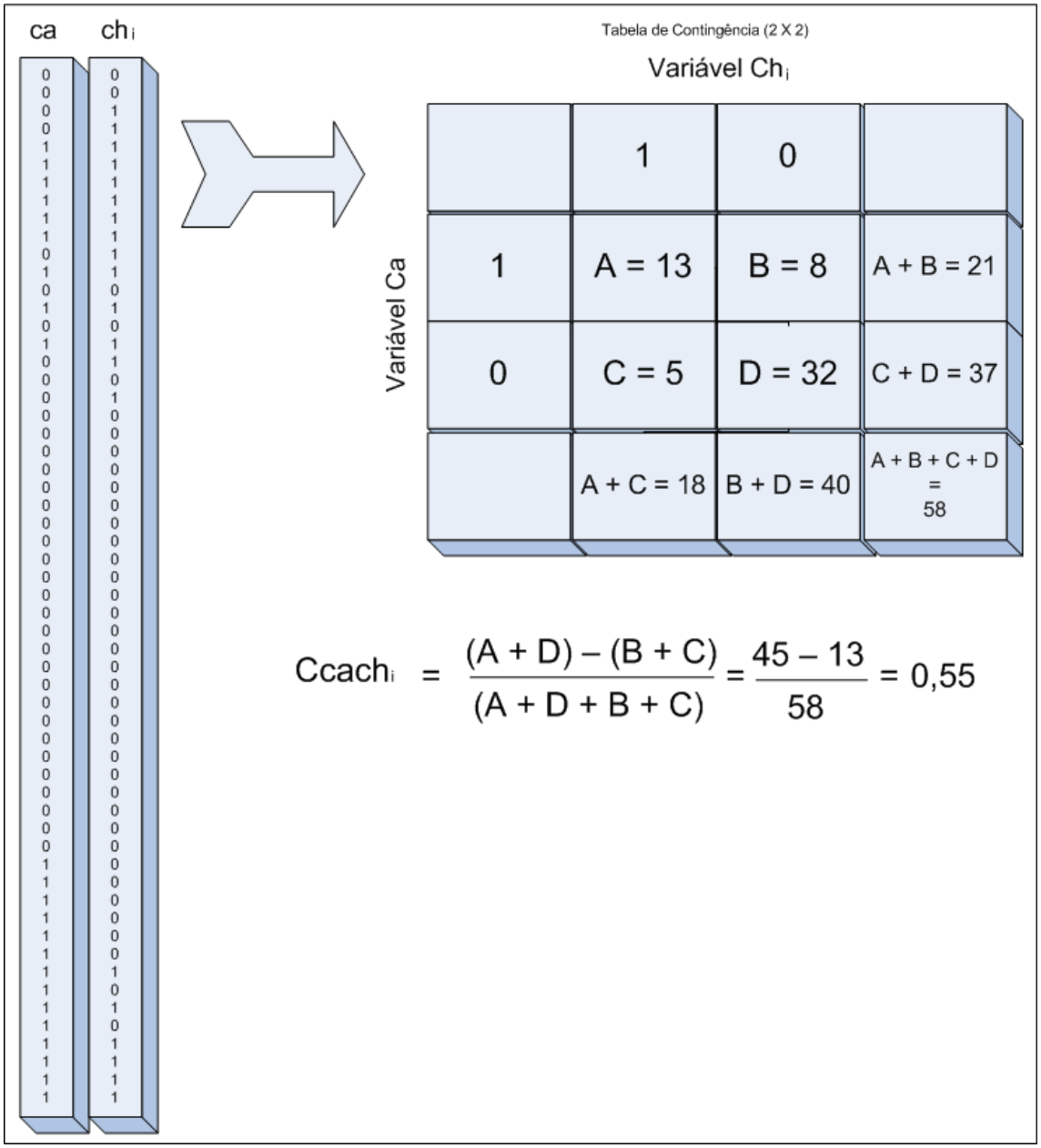

Figura A.1: Cálculo do coeficiente de Hamann

Existem muitos outros coeficientes na literatura, como os coeficientes de Yule, Rogers e Tanimoto, Sokal e Sneath, entre outros. Por tratar-se de uma descrição sucinta, estes coeficientes não foram apresentados. Um maior detalhamento sobre os coeficientes de similaridade entre variáveis binárias foi apresentado por Romesburg (1984) , Frank e Todeschini (1994) e Williams (2002), podendo servir como referencial teórico para estudo.

A.2 A Seleção do Coeficiente de Similaridade

A seleção da equação de similaridade, baseou-se no que se entende por similaridade para o modelo proposto. Como apresentado nas equações acima, algumas desconsideram a ocorrência de pares 0-0 ou atribuem pesos diferenciados para um determinado par, enquanto outras equações não. Isto mostra que a equação deve ser selecionada conforme o

objetivo, ou entendimento de similaridade, para o problema estudado.

É possível observar que quando uma equação desconsidera a ocorrência de pares 0-0, ela considera o valor 0 como ausência de uma determinada característica nas variáveis comparadas, sendo que esta ausência não deve influenciar no quanto similar são as duas variáveis comparadas. Notadamente, este não seria o objetivo do problema de estudo deste trabalho.

Similaridade para o problema de estudo deste trabalho deve incluir todos os pares. O entendimento de similaridade aqui é que uma variável é totalmente similar a outra se, e somente se, a soma da ocorrência dos pares 1-1 e 0-0 corresponder ao total de ocorrências de todos os pares, ou seja, $Ccach_i = 1.0$ se $A + D = N$, onde N corresponde ao total de ocorrências consideradas.

A equação que atende ao objetivo deste trabalho é a apresentada pelo coeficiente de Hamann e por esta razão, optou-se pela seleção deste coeficiente para avaliar a similaridade entre comportamentos, conforme o modelo proposto neste trabalho.