UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Campus de São José do Rio Preto

Diogo Tavares da Silva

Implementação de um Banco de *Traces* de Cargas de Trabalho para o iSPD

São José do Rio Preto
2012

Diogo Tavares da Silva

Implementação de um Banco de *Traces* de Cargas de Trabalho para o iSPD

Monografia apresentada ao Departamento de Ciências de Computação e Estatística do Instituto de Biociências, Letras e Ciências Exatas da Universidade Estadual Paulista "Júlio de Mesquita Filho", como parte dos requisitos necessários para aprovação na disciplina Projeto Final.

Aleardo Manacero Jr.

Diogo Tavares da Silva

Banca Avaliadora:

Prof. Dr. Adriano Mauro Cansian

Prof. Dr. Norian Marranghello

São José do Rio Preto
2012

Aos meus amados pais Pedro e Benedita

À minha querida avó Carmem e tia Aparecida

Ao meu irmão Daniel e primo José Luiz

À Tatiana

"Tão inevitável quanto a morte é a vida."

-Charles Chaplin

Agradecimentos

Agradeço primeiramente a Deus por estar sempre presente em minha vida e a Nossa Senhora Aparecida da qual sou devoto e sempre pedi intercessão.

Agradeço ao meu pai Pedro Tavares, pelo exemplo de honestidade, ética e respeito, e por não me dar tudo que eu quis e sim tudo que eu precisava e à minha mãe Benedita Chiampezam, por sempre ser minha amiga, conselheira e me recolocar no foco dos meus objetivos sempre que penso em perder a fé.

Agradeço à Tatiana, minha companheira, pelo apoio, amor e carinho, compreendendo minha ausência quando necessário.

Agradeço à minha avó Carmem Chiampezam pelo amor dedicado e valiosas histórias e conselhos e minha tia Aparecida Chiampezam pois nem todos têm a sorte de ter duas mães nessa vida, mas eu tenho. Obrigado a meu irmão Daniel Tavares que não me deixa perder a essência de criança e José Luiz Amate, irmão que a vida me deu, pela grande amizade e parceria. Também agradeço à toda equipe "José & Diogo - Explosões Sertanejas" (Jô, Vagno e Gil) pela amizade e companheirismo e a meus tios Antônio e Vania, Nardo e Ângela, Vera e Tatá e a todos os outros por todo apoio durante esses anos.

Gostaria de agradecer também, todos que direta e indiretamente tornaram a realização deste trabalho possível, especialmente meus ex-professores da Escola José Marcelino de Almeida pelo suporte intelectual e financeiro no início desta empreitada, aos docentes do BCC pelo conhecimento e atenção cedidos, à UNESP pelo apoio concedido através de bolsa BAAE, Moradia Estudantil e pelo financiamento juntamente com o CNPq de toda minha iniciação científica através de bolsas PIBIC/Reitoria que mantive desde 2010 até o presente momento e por fim à FAPESP pelo auxílio com equipamentos ao laboratório GSPD e recente aprovação de financiamento ao projeto no qual este trabalho está inserido.

Gostaria de agradecer também ao meu orientador Prof. Aleardo Manacero Jr. pela amizade, apoio, confiança e orientação sempre que precisei. Da mesma forma agradeço a Prof.ª Renata Spolon Lobato pela coorientação e suporte sempre que meu orientador não pode estar presente e ao meu grande amigo e parceiro de projeto Denison Menezes que contribuiu com inúmeras críticas, sugestões, ideias e suporte para que este trabalho fosse realizado.

Por fim não posso deixar de agradecer a todos as amizades que construi durante a graduação, listadas a seguir: MORADIA=[Pedro, Marquim, Kapial, Kleber, Felipe, Guilherme, Luciano, Driely e Dú Munari]; ANOS-ANTERIORES=[Luanzinho(14), Leandrinho, João, Rodrigo, Budha, Sprite, Andrés, Thiago(javaboy) e Júlio]; 2009=[Vitinho, Daniel, Fernando Carvalho, Chimbinha, Luga, Régis, Thunder, Zani, Louison, Adriano, Gui(Alf), Bárbara e Letícia]; 2010=[ViniVini(Galhardi e Mano), Heitor, Mariana, João Marcos(16), William e Gustavo]; 2011=[João Matheus(17), Guilherme]; e por fim, GSPD=[Paulo, Marco, Aldo, Victor, Tiago Brait, Renato(Tim), Hector, Daniel, Igor, Gabriel Saraiva(Muringa), Gabriel Covello(terminando a gramática...), Rafael(Pai), Thiago(Japa), Cássio, Matheus (number one), Danilo (Gordo Randômico), Leandro & Leonardo].

Resumo

Várias aplicações comerciais e científicas necessitam de cada vez mais poder computacional, poder este que se tornou possível através do barateamento de computação de alto desempenho. Neste cenário pode se destacar um aumento no uso de sistemas como grades computacionais (*grid computing*), incluindo usuários sem experiência em programação ou análise de desempenho. Para auxiliar o processo de avaliação do desempenho de grades e de seus escalonadores surgiram várias ferramentas para simulação de tais sistemas, dentre quais o iSPD (*iconic Simulator of Parallel and Distributed Systems*) se destaca pela facilidade de geração de modelos do sistema sem necessidade de programação. No iSPD a modelagem e simulação de grades ocorre através de uma interface icônica, de fácil uso. Entretanto, em sua primeira versão o iSPD tratava cargas de trabalho apenas por geração aleatória ou pela definição de tarefas alocadas em escalonadores específicos. Este trabalho acrescenta a capacidade de tratar cargas de trabalho a partir de arquivos de *trace* de execução de sistemas de alto desempenho. Essa capacidade é acrescentada através de uma estrutura para manipulação de um banco próprio de arquivos de trace, que permitirá que sejam simuladas grades executando cargas de trabalho mais realistas. Com isso também será possível comparar mais facilmente diferentes configurações de grade ou escalonadores, bastando que a carga aplicada ao modelo venha do mesmo arquivo de *trace*. Os resultados obtidos com a implementação dessa nova capacidade no iSPD são excelentes, uma vez que o banco próprio de *traces* é de fácil manipulação e propicia melhorias no processo de simulação.

Palavras-chave: Simulação, Cargas de trabalho, Grades Computacionais

Abstract

Several scientific and commercial applications require ever more computing power, which has been made possible through the cost reduction of high performance computing systems. In this scenario on notable event is the increase in the use of distributed systems such as Grids, allowing for the appearance of users with no experience in programming or performance evaluation. To help the process of performance evaluation of grids and their schedulers several several simulation tools have been proposed, among them iSPD (*iconic Simulator of Parallel and Distributed systems*) comes ahead because it is easy to model the systems without having to program. With iSPD the grid modeling and simulationis is performed through an easy-to-use iconic interface. However, in its first version, iSPD managed only workloads defined randomly or through task allocation over specific schedulers. This work adds the capability of managing workloads extracted from trace files coming from the execution of real HPC systems. This functionality is added through a structure to manipulate a local traces database, allowing that grids can be simulated using workloads more realistic. It will also be possible to compare different grid configurations or different schedulers more easily, through the application of the same trace file as workload. Results achieved with the inclusion of this new feature into iSPD are excellent, since the local trace database is easily managed and allows an improved simulation process.

Keywords: Simulation, workloads, grid computing

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Existe atualmente uma demanda crescente pelo uso de computação de alto desempenho, tanto aplicada no meio científico (processamento genômico, física de partículas, simulações de física e engenharia, entre outros) quanto na indústria (prospecção petrolífera, animação gráfica ou *e-commerce*, por exemplo). Isto ocorre principalmente pela necessidade de resolução de problemas cada vez mais complexos, que manipulam quantidades de dados cada vez maiores, devendo apresentar resultados em tempos factíveis. Com isso, têm se investido cada vez mais em sistemas de computação baseados em arquiteturas paralelas e distribuídas, como supercomputadores, grades computacionais e mais recentemente *cloud computing*, para se obter alto desempenho (Branco, 2004).

Entretanto, o insvestimento em tais sistemas também possui algumas desvantagens. No caso de supercomputadores, podemos destacar o fato de que estes são extremamente caros e tornam-se obsoletos com o passar do tempo. Como exemplo, (Meuer and Gietl, 2012) destaca o fato de que em 1986, o supercomputador mais potente no mundo era o Cray-2, estimado em 22 milhões de dólares em valores da época. A Apple Computer lançou recentemente o iPad 2 (*tablet pc*) que apresenta o equivalente a dois terços do desempenho do Cray-2, porém custando apenas 500 dólares. Desta forma pode-se antever que os supercomputadores atuais serão os computadores pessoais em um futuro relativamente próximo.

É neste contexto que as grades computacionais surgem, com o intuito de prover alto desempenho de forma mais econômica, através da aglomeração e compartilhamento de recursos (Foster et al., 2001). Dentro deste conceito, pode-se citar projetos como o I-WAY (Foster et al., 1996), criando um sistema que interliga supercomputadores de diversas organizações através de redes de alta velocidade, o BOINC (BOINC, 2012), que se utiliza de computação voluntária de recursos ociosos em todo o mundo e o Condor (Litzkow et al., 1988), desenvolvido com o intuito de utilizar recursos subutilizados de uma instituição.

Com o aumento da utilização de grades computacionais houve a formação de um espectro de usuários que não são especialistas no uso de computação de alto desempenho, os quais têm como objetivo o simples uso da grade para a execução de suas aplicações. Por consequência, criam-se demandas por ferramentas de apoio a estes

usuários, incluindo-se ferramentas para avaliação de desempenho de tais sistemas. Nesse sentido, simuladores se tornam uma ferramenta importante pois permitem identificar como aplicações podem ser ajustadas ao ambiente de grade, incluindo-se o desempenho de diferentes políticas de escalonamento para a execução das aplicações no sistema frente a uma ou mais configurações de grade.

1.1 Motivação

Existem atualmente vários simuladores de grades computacionais, como por exemplo o SimGrid (Casanova et al., 2008), GridSim (Buyya and Murshed, 2002), GangSim (GangSim, 2012), OptorSim (OptorSim, 2012) e Bricks (Takefusa and Matsuoka, 2000). No entanto, apesar da quantidade de ferramentas observa-se que estas apresentam uma série de obstáculos à utilização por usuários não especialistas, como interface pouco intuitiva para a criação de modelos de grade, escalonadores e cargas de trabalho. Além disso tais simuladores também apresentam problemas de portabilidade, tanto da ferramenta propriamente dita, desenvolvida especificamente para uma determinada plataforma, quanto a portabilidade dos modelos gerados, que não podem ser reutilizados em outras ferramentas.

Com o intuito de resolver os problemas descritos anteriormente, vem sendo desenvolvido pelo Grupo de Sistemas Pararelos e Distribuídos (GSPD, 2012), o iSPD (***i**conic **S**imulator of **P**arallel and **D**istributed systems*) (Manacero et al., 2012). O iSPD busca fornecer facilidades na construção de modelos de grades, escalonadores e cargas de trabalho. Secundariamente, também se busca fornecer compatibilidade com modelos de grade gerados pelo SimGrid e GridSim.

Dentro desse cenário uma característica importante é a possibilidade de execução de cargas de trabalho realistas, derivadas de *traces* [1] de aplicações reais. Como o iSPD não oferecia inicialmente esta possibilidade, o trabalho descrito nesta monografia foi desenvolvido de forma a suprir essa funcionalidade, acrescentando ainda a capacidade de se criar um banco de cargas de trabalho próprio.

1.2 Objetivo

OO objetivo deste trabalho foi o desenvolvimento de uma estrutura para criação, uso e manipulação de um banco próprio de cargas de trabalho realistas para a ferramenta iSPD, permitindo a inclusão de modelos de cargas de trabalho extraídos de *traces* de aplicações reais de alto desempenho neste banco, mantendo-se para tal a facilidade de uso para o usuário.

[1] Arquivo que registra valores de parâmetros que caracterizam a execução de uma carga de trabalho em um determinado sistema

1.3 Organização do texto

Após essa introdução os capítulos 2 e 3 apresentam uma fundamentação teórica sobre o trabalho desenvolvido. O capítulo 2 apresenta uma visão geral da plataforma iSPD e o capítulo 3 uma introdução sobre caracterização de cargas de trabalho e formatos de *traces* de aplicações de alto desempenho. A metodologia e desenvolvimento deste trabalho são apresentados no capítulo 4, que aborda o trabalho de implementação do banco próprio de cargas de trabalho realistas e das estruturas necessárias para torná-lo operacional. Os testes e validação da ferramenta desenvolvida são apresentados no capítulo 5. Por fim, o capítulo 6 apresenta as conclusões sobre este trabalho, incluindo conclusões e direções futuras.

Capítulo 2

O simulador de grades computacionais iSPD

Este capítulo apresenta a plataforma de simulação de grades computacionais iSPD, que é o objeto de trabalho deste projeto, exibindo uma visão geral da mesma, com destaque para a descrição do tratamento de criação e gerenciamento de cargas de trabalho nesta ferramenta.

2.1 Visão geral da plataforma de simulação de grades computacionais iSPD

O iSPD (***i**conic **S**imulator of **P**arallel and **D**istributed systems*) (Manacero et al., 2012) é uma plataforma de simulação de grades computacionais desenvolvida pelo GSPD (**G**rupo de **S**istemas **P**aralelos e **D**istribuídos) da UNESP (GSPD, 2012). Esta ferramenta é desenvolvida em Java e possui como principal característica a possibilidade de modelagem de grades computacionais através de uma interface icônica, como o próprio nome busca salientar. Em sua versão atual é possível executar tarefas do tipo BoT (*Bag of Tasks*) executadas em ambientes mestre-escravo.

O iSPD surgiu para contornar alguns problemas encontrados na utilização de outras plataformas. Pode-se destacar que a maioria dos simuladores disponíveis exigem de seus usuários um conhecimento avançado de programação, o que nem sempre é visto entre os usuários de grades. Além disto, observa-se o fato de que as principais ferramentas disponíveis enfrentam problemas quanto a portabilidade dos modelos de grade gerados, de forma que os mesmos são incompatíveis com outros simuladores.

Para contornar tais problemas, a plataforma iSPD é constuída de forma que cada um de seus componentes busca tornar o processo de simulação de grades computacionais mais intuitivo e simplificado. A figura 2.1 apresenta o diagrama conceitual da ferramenta iSPD, destacando como ocorre a iteração entre os seus componentes, descritos mais detalhadamente a seguir:

- **Interface Icônica:** Este módulo é responsável pela modelagem do ambiente de

grade e da carga de trabalho aplicada ao mesmo, realizando a costrução do modelo através de interface gráfica baseada em ícones (Guerra et al., 2010);

- **Interpretador de Modelos:** Este componente é responsável por interpretar o modelo icônico, convertendo-o para o modelo simulável usado no motor de simulação, ou interpretar modelos externos, convertendo-os para o modelo icônico (e vice-versa) (Aoqui et al., 2010);
- **Motor de Simulação:** Responsável por realizar a simulação de fato a partir de um modelo simulável, além de apresentar os resultados e métricas da simulação realizada (Oliveira et al., 2010);
- **Gerador e Gerenciador de Escalonadores:** Estes dois módulos em conjunto são responsáveis por gerenciar os escalonadores disponíveis para a simulação, além de gerar novas políticas de escalonamento de maneira facilitada, através de formulações matemáticas simples (Menezes et al., 2012).

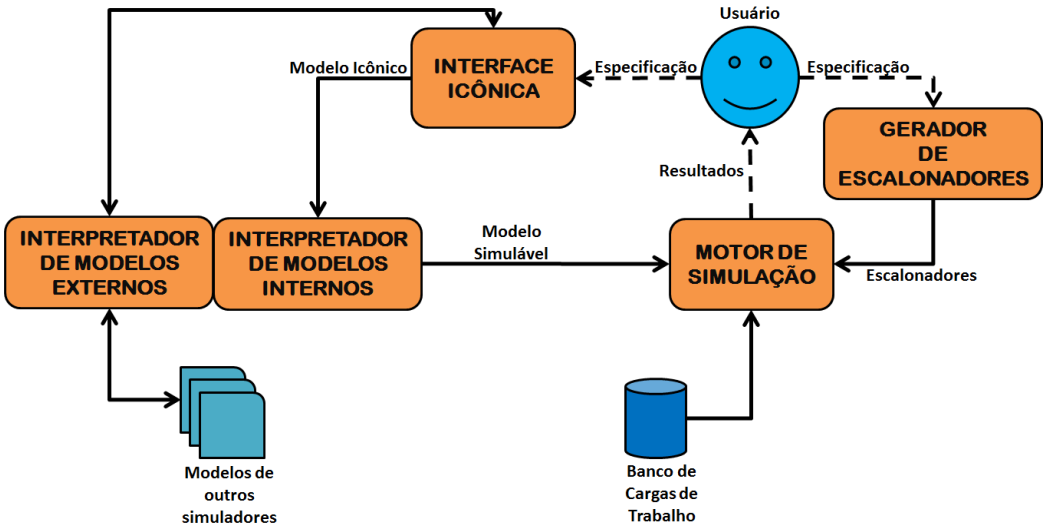

Figura 2.1: Diagrama conceitual do iSPD

2.2 Interface icônica

A interface icônica apresenta-se como o principal componente de iteração do usuário com a ferramenta, de maneira que qualquer operação e acesso a qualquer outro componente no iSPD ocorre por meio da mesma. Entretanto, dentro desta interface principal podemos destacar principalmente o suporte ao processo de construção do modelo icônico, que posteriormente se tornará o modelo simulável, além da configuração da carga de

trabalho, onde este trabalho se insere pela opção da configuração da carga de trabalho a partir de *traces*.

A figura 2.2 apresenta uma visão da interface gráfica principal, com um modelo já carregado. Observa-se a representação icônica do modelo na área de desenho central, logo acima estão os ícones utilizados para a modelagem juntamente com os botões de configuração de carga de trabalho, adição de usuários e início de simulação. Ao lado esquerdo da área de desenho, está a área de configurações do ícone selecionado e na parte inferior a área de notificações.

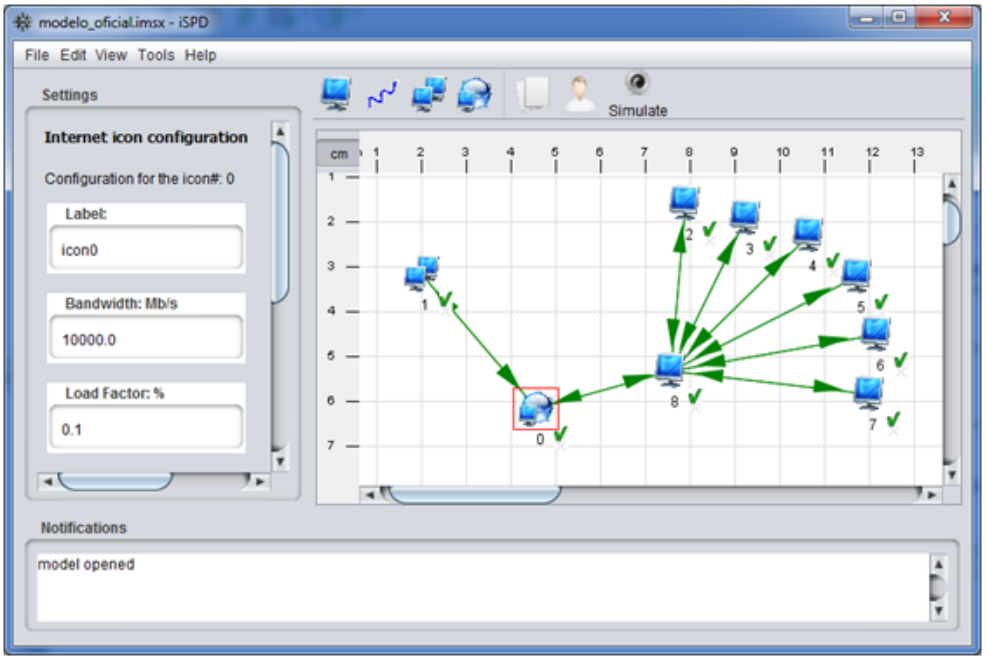

Figura 2.2: Interface principal do iSPD (Menezes, 2012)

2.2.1 Especificação do ambiente de simulação

Visando manter o processo intuitivo, a interface de modelagem segue um modelo icônico, no qual o usuário modela uma arquitetura de grade computacional adicionando ícones que representam os recursos da grade, interligados por meio de conexões de rede. Como pode ser visto na figura 2.2 e na figura 2.3 a seguir, temos os seguintes ícones para a modelagem da infraestrutura da grade:

- O ícone de monitor único, que representa uma máquina individual;
- O ícone de dois monitores aninhados, que representa um *cluster*;
- O ícone com monitor frente ao globo, que representa uma rede complexa (como a internet);

- O ícone seta, que representa enlaces ponto-a-ponto entre os demais elementos.

Ao adicionar um ícone no campo de desenho, apresentado na figura 2.2 pela área quadriculada abaixo da lista de ícones, o mesmo encontra-se desconfigurado, o que é indicado por um `x` nos ícones para máquinas, *cluster* e conexão com internet e pela cor vermelha nos ícones para enlaces ponto-a-ponto, conforme pode ser observado na figura 2.3. Após a configuração de um elemento, que ocorre através da área *Settings* exibida na figura 2.2 ou através de um duplo clique sobre o ícone selecionado, o `x` é substituído por uma marca de verificação e o enlace apresenta-se na cor verde. Para identificar a seleção de um ou mais elementos, ícones como máquinas exibem uma borda quadrada em destaque, enquanto a conexão de rede passar a ser exibida em tom preto.

Figura 2.3: Ambiente parcialmente configurado (Menezes, 2012)

Após realizar a configuração de todos os ícones, para concluir a construção de um modelo ainda é necessário configurar as cargas de trabalho. No estado do simulador anterior a este trabalho, a configuração de tarefas permitia a criação de cargas de duas maneiras:

- **Carga randômica:** Nesta opção é gerada uma carga de trabalho completamente randômica, através da passagem dos parâmetros: número de tarefas, tamanho de computação médio, máximo e mínimo, tamanho de comunicação médio, máximo e mínimo, além dos parâmetros utilizados pelo gerador de números aleatórios, como parâmetro da distribuição exponencial, que gera o tempo de chegada das tarefas e as probabilidades de computação e comunicação, utilizados para gerar os tamanhos das tarefas através da distribuição uniforme de dois estágios (*two-stage uniform*) indicada em (Lublin and Feitelson, 2001) para modelagem de tarefas paralelas. A figura 2.4 apresenta a janela de configuração de cargas randômicas, onde é possível ver primeiramente o campo de número de tarefas, seguido dos campos de configuração de tamanho computacional e tamanho de comunicação e por último o campo de entrada do parâmetro de tempo de chegada.

- **Carga por nó:** Nesta opção uma carga é aplicada específicamente a um nó mestre, e pode ser distribuída entre diversos usuários, configurando os parâmetros de números de tarefas e tamanhos máximo e mínimo de computação. A figura 2.5 apresenta a janela de configuração de cargas por nó computacional, onde seleciona-se, a partir do campos, o nó mestre a qual o conjunto de tarefas é aplicado, o número de tarefas a serem criadas e o usuário proprietário das mesmas, juntamente com os campos de entrada de tamanho computacional e tamanho de comunicação.

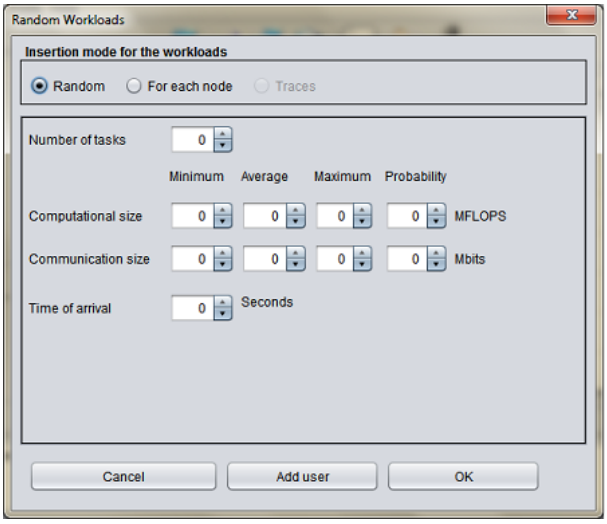

Figura 2.4: Configuração de cargas de trabalho randômicas

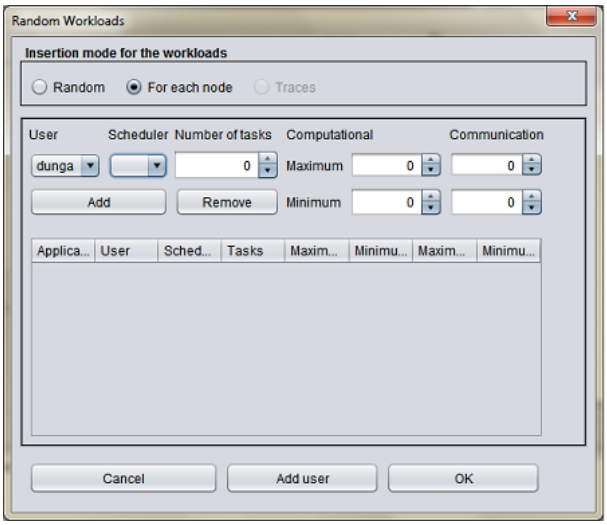

Figura 2.5: Configuração de carga por nó computacional

2.3 Interpretador de modelos

Este componente é o responsável pelas operações de conversão entre modelos externos de outros simuladores e o modelo icônico da própria ferramenta iSPD para um modelo simulável. Uma descrição detalhada deste módulo pode ser vista em (Aoqui et al., 2010). Este componente é dividido em dois módulos:

- **Interpretador de modelos internos:** A ferramenta iSPD utiliza dois formatos de modelos internamente para realizar a simulação, o modelo icônico, construído pelo usuário por meio da interface gráfica, e o modelo de redes de filas, que é

utilizado pelo motor de simulação. Ambos os modelos são representados por gramáticas específicas, portanto possuem interpretadores próprios. Durante o processo de simulação ocorre uma análise do modelo icônico, sendo este convertido para um modelo simulável, que é interpretado e utilizado pelo motor de simulação.

- **Interpretador de modelos externos:** Responsável pela conversão de formatos de modelos de outros simuladores para o modelo icônico do iSPD. Atualmente é possível converter modelos escritos para o SimGrid e existe um projeto em andamento para o desenvolvimento um novo módulo para interpretação de modelos do GridSim e conversão de modelos icônicos para os modelos do GridSim.

2.4 Motor de simulação

Este componente é responsável pelo processo efetivo de simulação e exibição de resultados e métricas ao final da simulação (Oliveira et al., 2010). O motor baseia-se na simulação de eventos discretos, utilizando-se dos conceitos de modelagem orientada a eventos e teoria das filas. Uma descrição detalhada do funcionamento e implementação do motor de simulação pode ser vista em (Oliveira et al., 2010) e (Menezes, 2012). A figura 2.6 apresenta o fluxograma de funcionamento do motor de simulação, destacando como os eventos criados são atendidos durante o processo de simulação.

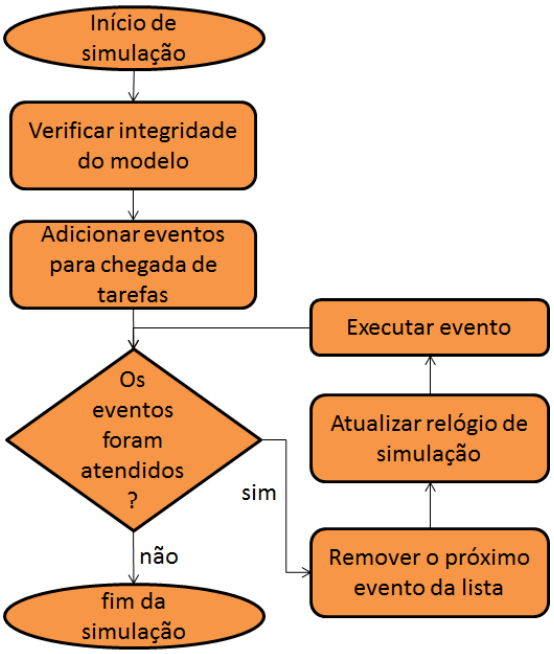

Figura 2.6: Fluxograma de funcionamento do motor de simulação (Manacero et al., 2012)

2.5 Gerador de políticas de escalonamento

Este componente tem como finalidade a geração e gerenciamento de políticas de escalonamento para a plataforma iSPD, permitindo ao usuário a criação de políticas de escalonamento de forma intuitiva (Menezes et al., 2012). Este componente está dividido em dois módulos:

- **Gerador de políticas de escalonamento:** que permite ao usuário a criação de políticas de escalonamento por meio de formulações matemáticas simples, através da interface de gerador avançado ou mesmo pela opção de políticas simples em uma lista através da interface de gerador simples. A figura 2.7 apresenta a interface do gerador simples de políticas, onde o usuário realiza a seleção de políticas simples para os escalonadores de tarefas e recursos apartir de listas de políticas pré-definidas. Já a figura 2.8 apresenta o gerador avançado, através de uma interface de "calculadora", onde o usuário elabora uma política de escalonamento a partir de uma formulação matemática simples gerada pela seleção de parâmetros de tarefas e recursos.

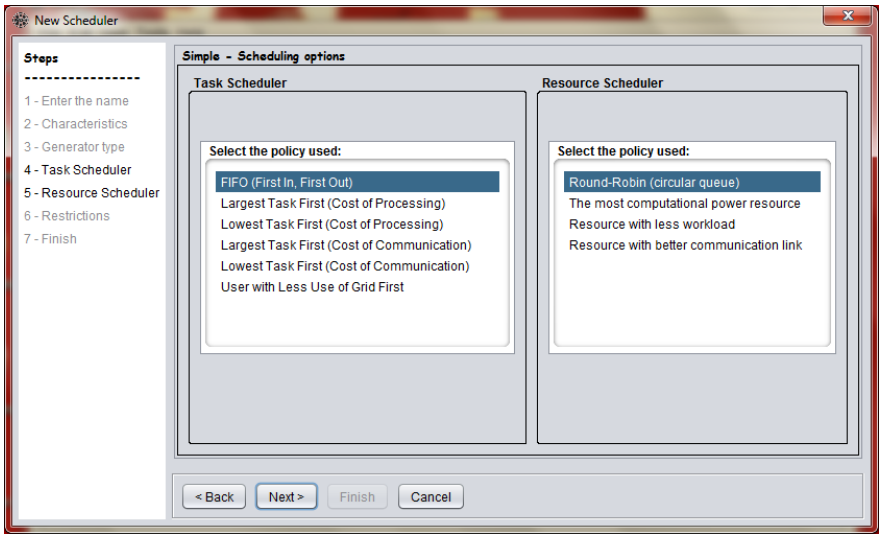

Figura 2.7: Gerador simples de políticas de escalonamento

- **Gerenciador de escalonadores:** Este módulo tem como intuito criar e editar escalonadores através de programação em Java, a partir de métodos pré-definidos, possibilitando até mesmo a edição das políticas geradas pelo componente gerador de políticas citado no item anterior. A importância deste componente está em criar classes de escalonadores mais avançadas que as construídas pelo gerador.

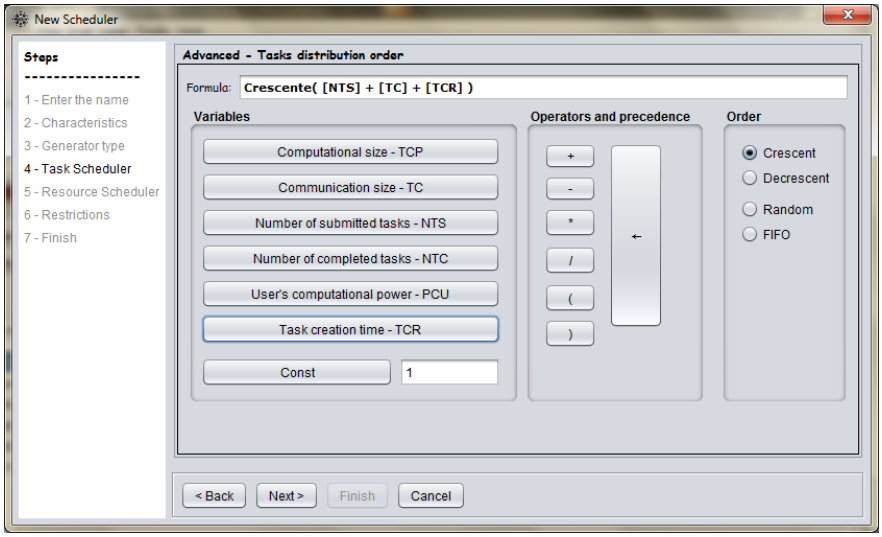

Figura 2.8: Gerador avançado de políticas de escalonamento

2.6 Considerações finais

Neste capítulo apresentou-se a ferramenta sobre a qual este trabalho se apoia, mostrando seus aspectos principais, e examinando os componentes para tratamento de cargas, nos quais a especificação do banco de cargas de trabalho será estruturada. O próximo capítulo apresenta a fundamentação teórica necessária sobre cargas de trabalho.

Capítulo 3

Cargas de trabalho e *traces* de aplicações de alto desempenho

Este capítulo trás uma introdução sobre cargas de trabalho e sua importância, com foco maior em cargas de trabalho realistas, extraídas de execuções de aplicações reais. Essas cargas são normalmente apresentadas através de arquivos de *traces*, com formatos específicos de acordo com a fonte de sua geração.

3.1 Cargas de trabalho

Inicialmente, pode-se definir carga de trabalho de um sistema como a quantidade de trabalho a ser realizada, dentro de um intervalo temporal, pelo mesmo. Dentro do contexto de análise de desempenho, o conceito de carga de trabalho é muito importante, sendo que Feitelson (Feitelson, 2011) afirma que o desempenho de um sistema computacional é afetada diretamente por três fatores, como representado na figura 3.1: projeto, forma de implementação e carga computacional, sendo que este talvez seja o mais difícil de lidar. Desta maneira, é impossivel estimar corretamente o desempenho de um sistema sem o conhecimento da carga de trabalho à qual este se encontra submetido (Calzarossa et al., 2000).

Ao se avaliar o desempenho de um sistema aplicando-lhe uma carga de trabalho inadequada, a avaliação acabará por obter resultados equivocados. Isto pode levar a decisões de projeto equivocadas, fazendo com que o sistema acabe por realizar uma má utilização de seus recursos ou mesmo não atingir suas metas. Decorrente desta situação, é pertinente que se busquem modelos de carga adequados para cada tipo de sistema em estudo, motivando o desenvolvimento da caracterização de cargas de trabalho.

3.1.1 Caracterização de cargas de trabalho

Calzarossa (Calzarossa et al., 2000) define caracterização de cargas de trabalho como a "descrição da carga de trabalho por meio de funções e parâmetros quantitativos,

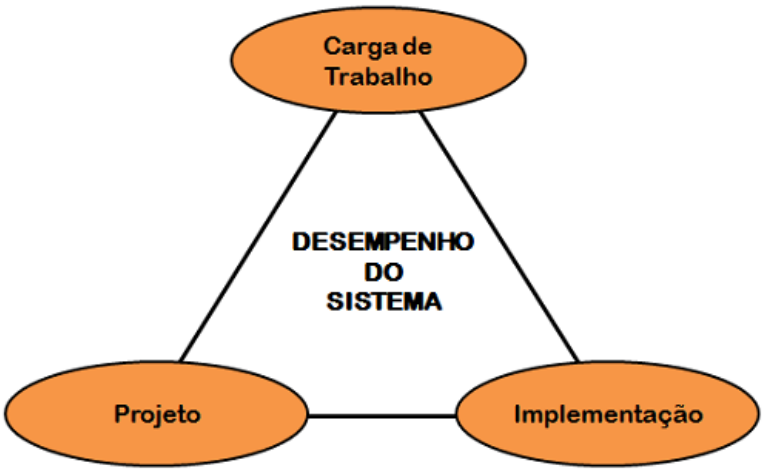

Figura 3.1: Os três pilares que influenciam diretamente o desempenho de um sistema, segundo (Feitelson, 2011)

com o objetivo de derivar um modelo que permita exibir, capturar e reproduzir o comportamento da carga e de suas características mais importantes".

Pode-se observar desta definição que os parâmetros obtidos de tal caracterização podem ser classificados entre **parâmetros estáticos**, referentes aos consumos de *hardware* e *software*, e **parâmetros dinâmicos**, que se referem de fato ao comportamento das requisões que compõe uma carga de trabalho.

A caracterização de cargas de trabalho geralmente é um processo experiemental, onde busca-se obter um compromisso entre a precisão do que está sendo medido e o volume de dados coletados para o mesmo, de modo que o modelo de carga derivado possua representatividade da carga real ao qual o sistema é submetido. Existem diversas técnicas para realizar a caracterização de cargas de trabalho, que podem ser classificadas em duas categorias (Calzarossa et al., 2000):

- **Técnicas de análise exploratória de dados:** que são fundamentais para a descoberta de características essenciais de medidas quantitativas e suas relações;
- **Técnicas de análise numérica e processos estocásticos:** que são úteis para a reprodução do comportamento dinâmico de uma carga de trabalho.

Estudos mais completos sobre caracterização de cargas de trabalho podem ser encontrados em trabalhos como (Feitelson, 2011), (Calzarossa and Serazzi, 1993) e (Calzarossa et al., 2000).

3.2 *Traces* de aplicações de alto desempenho

Um arquivo de *trace* (ou *log*) registra determinados valores de parâmetros que caracterizam a execução de uma instância de uma determinada aplicação, ou conjunto de

aplicações, em um determinado ambiente, por meio de sensores e rotinas de monitoramento. Logo, um *trace* consiste na caracterização de uma carga de trabalho pela observação de uma execução real da mesma. A partir desta definição, pode-se observar que os parâmetros variam de acordo com a aplicação e o motivos de investigação.

Em (Zhao et al., 2008), o autor afirma a importância de realizar o processo de simulação de grades utilizando-se da carga de *traces* de grades reais, o que leva a uma maneira mais precisa e controlada de experimentar novos métodos e idéias aplicadas a avaliação de desempenho e ao escalonamento de tarefas.

Para o iSPD estava previsto, porém não implementado, um banco de *traces* de cargas de trabalho realistas. Como se trata de um simulador de grades é importante que esses *traces* sejam originados por aplicações de alto desempenho executando em grades. Com isso é necessário que o mesmo seja capaz de converter *traces* contidos em bancos externos para o formato de carga simulável por ele. Esse trabalho, que é aqui apresentado, envolve a identificação de fontes de *traces*, seus formatos e formas de conversão.

Entretanto, observa-se uma certa dificuldade em encontrar formatos de *traces* de alto desempenho disponíveis, com boa documentação e em boa quantidade. Neste contexto, as próximas subseções apresentam dois destes formatos, o SWF (*Standard Workload Format*) e o GWF (*Grid Workload Format*), ambos disponíveis publicamente na internet, apresentando boa documentação e quantidade de amostras de seus *traces*, além de facilidade de obtenção dos mesmos.

3.2.1 *Standard Workload Format* (SWF)

O formato de *traces* SWF (PWA, 2012g) é mantido pelo PWA (*Parallel Workloads Archive*) (PWA, 2012c), sob coordenação do Prof. Dr. Dror Feitelson. O principal objetivo do SWF é prover facilidade de uso, de modo que ferramentas que realizam análise de cargas de trabalho e simulação de escalonamento de sistemas necessitem apenas realizar o *parsing* em um único formato. Dentre as características do arquivo pode-se destacar:

- Cada carga é descrita em um arquivo ASCII único;
- Cada linha descreve uma tarefa (*job*);
- Cada linha apresenta 18 campos separados por espaços, em que campos irrelevantes para determinado registro são demarcados com -1;
- Arquivos esperam comentários de cabeçalho, que determinam o ambiente do *trace* em particular;
- Comentários são identificados por linhas que iniciam com " ;".

Campos de dados

Uma descrição completa de todos os campos de dados pode ser encontrada em (PWA, 2012g), aqui serão apresentados os campos de dados úteis ao desenvolvimento da especificação do banco de *traces* para o iSPD, que são:

- **Número do *Job*:** Um contador identificando a tarefa, pode ser utilizado como Identificador da tarefa;
- **Tempo de Submissão:** dado em segundos, apresenta o tempo a partir da submissão da primeira tarefa em que uma tarefa é submetida. As tarefas são ordenadas no *trace* por este valor;
- **Tempo de Espera:** Diferença, em segundos, entre o tempo de submissão da tarefa e o tempo que a mesma começa a ser efetivamente executada. Valor interessante para fins de validação e avaliação de métricas;
- **Tempo de Execução:** Tempo em que uma tarefa permanece sendo executada. Caso este campo apresente o valor 0, significa que a tarefa foi executada em menos de 0,5 segundos. Este campo pode ser utilizado para estimar o tamanho computacional de uma tarefa;
- *Status* **da Tarefa:** representado por um inteiro de 0 a 5. O valor 0 indica falha na execução da tarefa, o valor 1 significa que a execução da tarefa foi completada, o valor 2 significa uma execução parcial da tarefa que será continuada, o valor 3 representa a última execução parcial, com a tarefa sendo completada, o valor 4 a última execução parcial da tarefa, entretanto ocorrendo falha da mesma e por fim o valor de *status* 5 significa que a tarefa foi cancelada por alguma razão;
- **ID do usuário:** Representado por um número natural, distinguindo os diversos usuários.

3.2.2 *Grid Workload Format* (GWF)

O formato de arquivo de *trace* GWF (GWA, 2012c) é mantido pelo GWA (*Grid Workload Archive*) (GWA, 2012d), sendo uma extenção ao formato SWF descrito na seção 3.2.1 para o caso específico de *traces* provenientes de ambientes de grades computacionais.

Dado que o formato é uma extensão ao SWF, as características principais se mantém, sendo seus principais campos descritos a seguir.

Campos de Dados

- **Id da Tarefa:** Um contador, identificando o a tarefa, correspondente ao número da tarefa do padrão SWF;
- **Tempo de Submissão:** Este campo é dado em segundos, e apresenta o tempo a partir de um relógio global em que as tarefas são submetidas, as tarefas são ordenadas no *trace* por este valor;

- **Tempo de Espera:** Diferença, em segundos, entre o tempo de submissão da tarefa e o tempo que a mesma começa a ser efetivamente executada. Valor interessante para fins de validação. Idem ao padrão SWF;
- **Tempo de Execução:** Tempo decorrido, em que uma tarefa permanece sendo executada. Caso este campo apresente o valor 0, isto significa que a tarefa foi executada em menos de 0,5 segundos. Este campo pode ser utilizado para estimar o tamanho computacional de uma tarefa. Idem ao padrão SWF;
- ***Status* da Tarefa:** idem ao padrão SWF;
- **ID do usuário:** Representado por uma *string*, identificando cada um usuários. Corresponde ao campo de mesmo nome do padrão SWF, que utiliza, no entanto, um número natural.
- **Rede Utilizada:** Medida em kilobytes/seg, apresenta uma média por processador.

3.2.3 Outros formatos

Além dos padrões SWF e GWF foram identificados os seguintes padrões, os quais ainda não estão incorporados ao iSPD pelos motivos indicados em suas descrições:

- *Google Cluster Data (Google, 2012a) :* Apresenta uma série de *traces* de cargas de trabalho extraídas de aplicações executadas nas células de computação da empresa Google. Uma descrição detalhada deste tipo de *trace* pode ser encontrada em (Reiss et al., 2011). O grande problema na utilização deste tipo de padrão está na obtenção do mesmo, por usuários do iSPD dado que é exigido um cadastro sob aprovação do administrador, e conhecimentos específicos para utilizar a ferramenta *gsutil*(Google, 2012b), com a qual os *traces* precisam ser obtidos.
- *Grid Observatory (CERN, 2012):* Apresenta dois tipos de *traces* obtidos através do uso da ferramenta glite (E. Laure et al., 2006), *traces* RTM (*Real Time Monitor*), que apresenta geração de *traces* através do monitoramento de aplicações executando em tempo real, e também através de *traces* de utilização, consumo de energia, entre outras características obtidas pelo uso do glite. O problema também se encontra na obtenção destes *traces* por usuários do iSPD, pela exigência de cadastro sob aprovação do administrador e necessidade de conhecimentos sobre a ferramenta glite utilizada para tal finalidade.

3.3 Considerações finais

Neste capítulo apresentou-se a fundamentação teórica sobre cargas de trabalho e formatos de *traces* computacionais, para que a partir do capítulo seguinte possa ser apresentado o desenvolvimento dos procedimentos de criação do banco de *traces* de aplicações de alto desempenho próprio para o iSPD.

Capítulo 4

Inserção de cargas de trabalho reais no iSPD

Após a fundamentação teórica apresentada nos capítulos 2 e 3, este capítulo tem como objetivo descrever o desenvolvimento deste trabalho, apresentando toda estrutura desenvolvida para a inclusão do banco próprio de cargas de trabalho reais. Desta forma, este capítulo aborda desde o processo de identificação e especificação de um padrão próprio para o iSPD e de conversão de padrões de *traces* externos para o padrão iSPD, até a identificação do processo de inserção de cargas no iSPD, desenvolvimento do processo de simulação de carga obtida de *traces* e por fim o processo de extração de *traces* da simulações realizadas.

A figura 4.1 apresenta um diagrama de casos de uso do iSPD, no que diz respeito ao tratamento de cargas de trabalho por *traces*. Esses casos de uso guiaram todo projeto de inclusão de cargas reais e dos itens a serem descritos nesse capítulo.

4.1 Especificação do padrão de *trace* próprio para o iSPD

Após a apresentação de padrões de *traces* na seção 3.2, a próxima etapa do projeto buscou o estabelecimento de um padrão próprio de *trace* para o iSPD. Para que esta representação fosse possível, foram escolhidas as características essenciais para representação uma carga de trabalho proveniente de *traces* de execução de aplicações de alto desempenho, sendo importante que tais características estejam disponíveis nos arquivos *traces*.

Para realizar o tratamento de *traces* vindos de diferentes fontes é preciso definir primeiramente o formato de cada origem. Portanto os *traces* do iSPD devem, antes de mais nada, conter a informação sobre o formato de arquivo do qual foram extraídos. No momento os formatos selecionados foram o SWF (PWA, 2012g) e GWF (GWA, 2012c), pela maior facilidade para o usuário do iSPD em obter amostras destes formatos, entretanto, deve-se permitir a expansão para outros possíveis formatos disponíveis.

Desse modo o primeiro passo é estabelecer como cada informação é armazenada nos arquivos de *traces* e, a partir disso, definir os procedimentos para recuperar esses dados

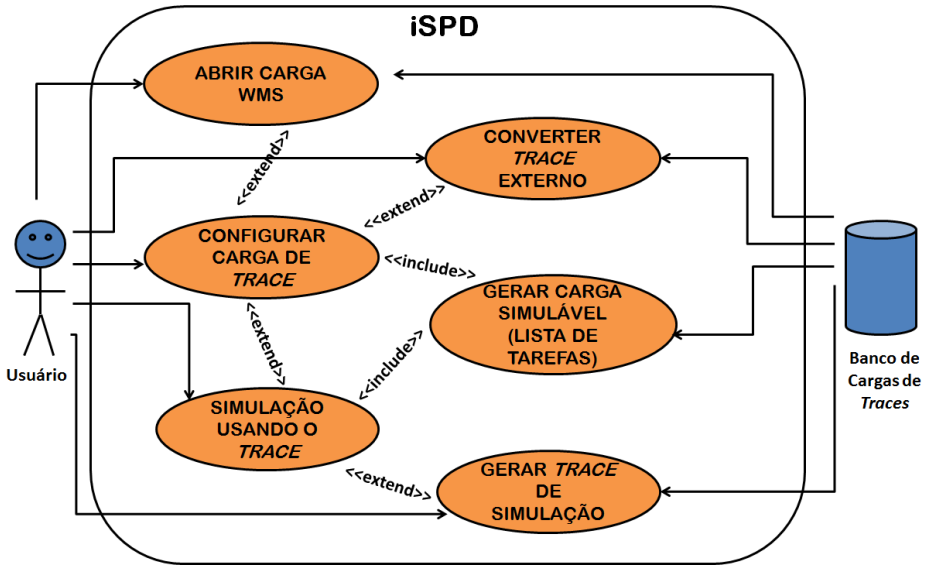

Figura 4.1: Diagrama de caso de uso do tratamento de cargas realistas pelo iSPD

e transformá-los no formato específico para o iSPD. Desta forma, após a observação das informações que constituem os padrões de *traces* utilizados e das informações necessárias para a simulação de cargas de trabalho para a ferramenta iSPD, ficou definido que para o padrão criado, cada tarefa apresenta as seguintes características:

- **Id da Tarefa:** Campo de identificador da tarefa, representado por um número natural;
- ***Status* da Tarefa:** Indica o *status* de execução da tarefa, seguindo o padrão dos formatos SWF e GWF, no atual estado de implementação do motor de simulação do iSPD, apresenta o valor `"1"` indicando a execução completa da tarefa;
- **Tamanho Computacional:** Armazena o valor relativo à quantidade de computação consumida pela tarefa, representado por um número real positivo e medido em Mflop;
- **Tamanho de comunicação:** Armazena o valor relativo ao consumo de recursos de rede pela tarefa, representado por um número real positivo e medido em MBytes;
- **Proprietário da tarefa:** Campo referente ao usuário proprietário da tarefa, indicado por uma *string* de caracteres;
- **Tempo de chegada:** Informa o instante de submissão da tarefa no sistema, medido em segundos e relativo à chega da primeira tarefa a ser executada.

Dada a descrição das características necessárias ao arquivo de *trace*, definiu-se que o arquivo utilizado para a composição do banco de cargas de trabalho realistas, seguiria um padrão de documento construído por XML, seguindo o mesmo conceito de construção do modelo IMS (*Iconic Model of Simulation*) utilizado para a modelagem da grade através da interface icônica descrita na seção 2.2. Para tal, criou-se o arquivo `"iSPDcarga.dtd"`, um arquivo DTD (*Document Type Definition*) de definição de formato de arquivo XML e por fim, definiu-se o nome de formato de arquivo de carga de *trace* do iSPD como WMS (*Workload Model of Simulation*), utilizando a extensão `".wmsx"`.

A figura 4.2 apresenta o trecho de verificação de conformidade do padrão WMS do arquivo `"iSPDcarga.dtd"`, para cargas de *trace*. Pode-se observar que uma carga de *trace* do iSPD é constituída de dois elementos, o formato de origem e as tarefas contidas no *trace*. Para o elemento `"format"`, o atributo `"kind"` indica o padrão de origem do qual o *trace* foi extraído, já para os elementos `"task"`, o atributo `"id"` indica o identificador da tarefa, o atributo `"arr"` o tempo de chegada no sistema, o atributo `"sts"` o estado de execução da tarefa, os atributos `"cpsz"` e `"cmsz"` indicam os custos de computação e comunicação da tarefa e por fim, o atributo `"usr"` indica o usuário proprietário da tarefa.

```
<!ELEMENT trace (format,task+)>
<!ELEMENT format EMPTY>
<!ATTLIST format kind CDATA "iSPD" >
<!ELEMENT task EMPTY>
<!ATTLIST task id CDATA "task1">
<!ATTLIST task arr CDATA "0.0">
<!ATTLIST task sts CDATA "1">
<!ATTLIST task cpsz CDATA "-1">
<!ATTLIST task cmsz CDATA "-1">
<!ATTLIST task usr CDATA "user1">
```

Figura 4.2: DTD referente a verificação de cargas de *traces*

4.2 Construção de componente conversor de arquivos de *trace*

Após a definição do padrão próprio de *trace* para o iSPD, é necessário que haja um componente responsável pela conversão de *traces* externos para o formato específico do iSPD. No iSPD, o componente responsável por esta tarefa é a classe *Interpretador.java* contida no pacote *ispd.arquivo.interpretador.carga*. Este pacote contém todos os métodos responsáveis por leitura, conversão e geração de cargas de trabalho contidas em *traces* externos para o padrão WMS. A figura 4.3 apresenta o diagrama conceitual deste componente. Pode-se ver que o funcionamento é simples e consiste basicamente

em entrar com arquivos de *traces* de formatos externos e gerar um arquivo WMS a partir do mesmo.

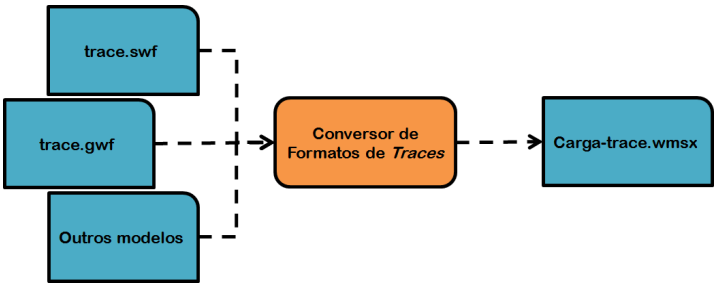

Figura 4.3: Diagrama conceitual do componente conversor de arquivos de *traces*

A partir desta estrutura, a classe *Interpretador.java* utiliza o método *convert()* para realizar a conversão do padrão estabelecido para o formato WMS, recebendo como parâmetro do construtor da classe o caminho absoluto do arquivo a ser convertido e retornando o arquivo gerado no mesmo diretório do arquivo de origem. A partir deste ponto ocorre o tratamento adequado de cada formato de origem, descrito nas próximas seções. A figura 4.4 apresenta um diagrama do processo de conversão, dando enfoque ao funcionamento do método convert() da classe Interpretador.java, que é o responsável real pela conversão dos formatos externos para o padrão WMS, que ao final do processo são incluídos no banco de cargas reais, enquanto a figura 4.5 apresenta o fluxograma de execução deste método. Nela pode ser destacado o processo repetitivo de conversão linha a linha, ou seja, por tarefa, adotado para cada arquivo."

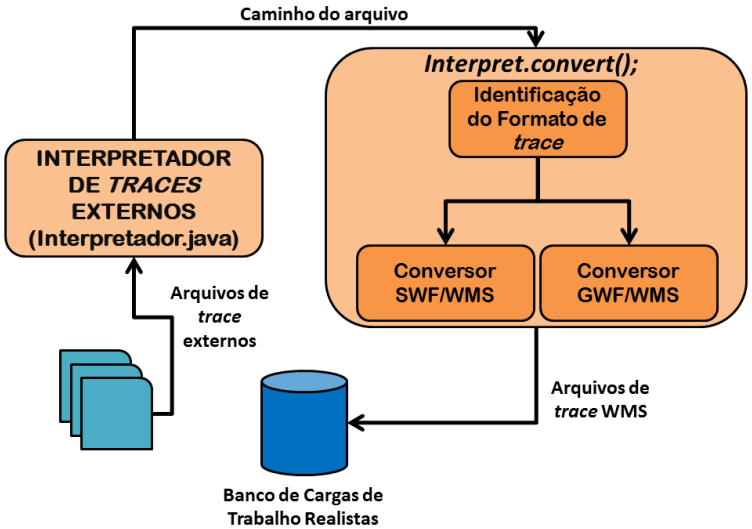

Figura 4.4: Diagrama de funcionamento do método *convert()* na classe *Interpretador.java*

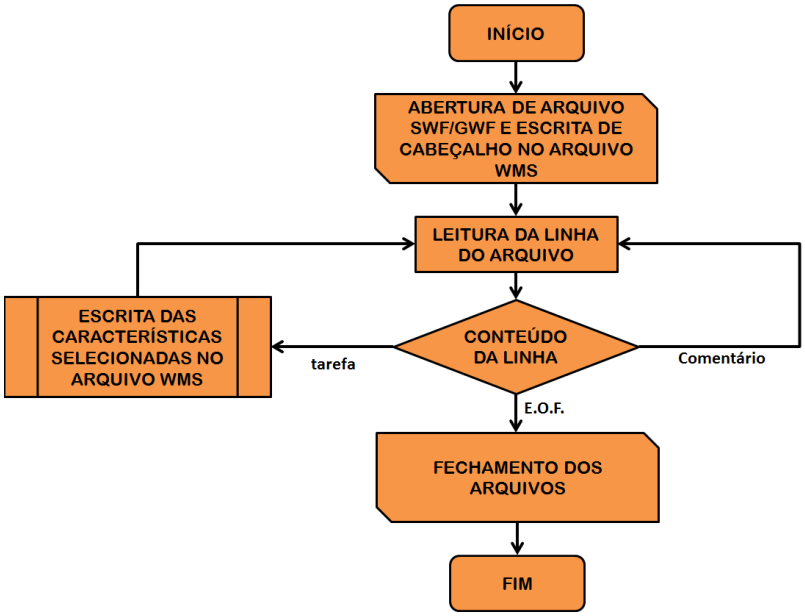

Figura 4.5: Fluxograma de execução do método de conversão de *traces* externos

4.2.1 Conversão de arquivos no padrão SWF

Como descrito na seção 3.2.1, um arquivo no padrão SWF é constituído de linhas de comentários, iniciadas por ";" e linhas que descrevem as tarefas através de um conjunto de características. A figura 4.6 apresenta um pequeno exemplo de um arquivo sob o padrão SWF, destacando-se os campos utilizados durante a conversão. Desta forma os valores da coluna em azul apresentam os identificadores das tarefas, a coluna amarela os tempos de submissões, a coluna em vermelho o tempo de execução das tarefas e por fim, as colunas verde e lilás apresentam os valores de *status* da tarefa e usuário proprietário.

```
; Comentários sobre o trace
;mais comentários
   1      0    -1   1451  128    -1   -1   -1   -1   -1 -1    1   1  -1  1 -1 -1 -1
   2   1460    -1   3726  128    -1   -1   -1   -1   -1 -1    1   1  -1  1 -1 -1 -1
   3   5198    -1   1067  128    -1   -1   -1   -1   -1 -1    1   1  -1  1 -1 -1 -1
;comentários
   4   6269    -1  10927  128    -1   -1   -1   -1   -1 -1    2   1  -1  1 -1 -1 -1
   5  17201    -1   2927  128    -1   -1   -1   -1   -1 -1    1   1  -1  1 -1 -1 -1
;final
```

Figura 4.6: Exemplo de arquivo de *trace* no padrão SWF

O primeiro fator a ser considerado é como utilizar os campos disponíveis no *trace* SWF para geração dos campos existentes no padrão WMS. Desta forma a partir da definição dos campos de dados do padrão SWF descritos na seção 3.2.1, os campos **Nº do Job** e **ID da tarefa** são equivalentes, assim como os campos **Tempo de submissão**

e **Tempo de chegada**. Da mesma forma, os campos **Usuário proprietário** e ***Status* da tarefa** estão presentes nos dois padrões. Para estes campos, o processo de conversão é imediato, apenas extraindo os valores do arquivo SWF e inserindo-os no arquivo WMS, segundo o padrão definido no DTD `"iSPDcarga.dtd"`.

Entretanto dois campos são problemáticos para a conversão entre estes formatos. A informação sobre tamanhos de computação e comunicação não está disponível no padrão SWF, sendo que para estes campos o padrão WMS armazena o **tempo de execução** da tarefa presente no padrão SWF em seu campo de **Tamanho de computação**, e o campo **Tamanho de Comunicação** é simplesmente preenchido com o valor `"-1"`. Para estes campos o tratamento adequado é dado posteriormente durante a leitura do *trace* para a geração das tarefas a serem simuladas pelo iSPD.

Por fim, ainda é necessário dar um tratamento às linhas de comentários presentes no padrão SWF, que no padrão WMS são simplesmente excluídas. Desta forma, após o processo de conversão descrito nesta seção, o arquivo gerado a partir do exemplo exibido na figura 4.6 é apresentado na figura 4.7. Destacam-se os campos utilizados no exemplo anterior, exibidos neste exemplo pelo mesmo padrão de cores.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE system SYSTEM "iSPDcarga.dtd">
<system>
<trace>
<format kind="SWF" />
<task id="1" arr="0" sts="-1" cpsz ="1451" cmsz="-1" usr="user1" />
<task id="2" arr="1460" sts="-1" cpsz ="3726" cmsz="-1" usr="user1" />
<task id="3" arr="5198" sts="-1" cpsz ="1067" cmsz="-1" usr="user1" />
<task id="4" arr="6269" sts="-1" cpsz ="10927" cmsz="-1" usr="user2" />
<task id="5" arr="17201" sts="-1" cpsz ="2927" cmsz="-1" usr="user1" />
</trace>
</system>
```

Figura 4.7: Arquivo de *trace* no padrão WMS obtido a partir do exemplo da figura 4.6

4.2.2 Conversão de arquivos no padrão GWF

Para o caso de conversão de arquivos GWF para o padrão WMS, o processo é análogo à conversão de arquivos SWF, dado que esse padrão de arquivo de *trace* é uma extensão ao padrão SWF. Entretanto, existem algumas pequenas diferenças, principalmente no acréscimo de campos de dados, na maneira como o tempo de chegada da tarefa é tratado e nas linhas de comentários iniciadas com `"#"`. Um exemplo parcial dearquivo GWF, pode ser visto na figura 4.8. Assim como para o padrão anterior, estão destacados os campos utilizados para a conversão, desta forma os valores da coluna em azul apresentam os identificadores das tarefas, a coluna amarela os tempos de submissões, a coluna em vermelho o tempo de execução das tarefas e por fim, as colunas verde e lilás apresentam os valores de *status* da tarefa e usuário proprietário.

A partir da descrição dos campos do padrão GWF feita na seção 3.2.2, observa-se que a equivalência entre seus campos e os do padrão WMS é praticamente a mesma que

```
#Comentários sobre o trace
#mais comentários
0  1083658801  1  0   4   -1  -1  4   3600    -1  1  user386  group4  app34  queue0  -1
1  1083658849  1  19  1   -1  -1  1   3600    -1  1  user112  group6  app0   queue0  -1
2  1083658875  2  10  5   -1  -1  5   3600    -1  1  user112  group6  app0   queue0  -1
3  1083658891  5  8   90  -1  -1  90  3600    -1  1  user112  group6  app0   queue0  -1
#final
```

Figura 4.8: Exemplo de arquivo de *trace* no padrão GWF

a deste com o padrão SWF. Os campos **Nº do Job** e **ID da tarefa** são equivalentes, assim como os campos **Tempo de submissão** e **Tempo de chegada**. Os campos **Usuário proprietário** e **Status da tarefa** também estão presentes nos dois padrões. A principal diferença para o caso específico do GWF é a maneira como o instante de chegada da tarefa é medido. Enquanto no padrão SWF o tempo de chegada é dado em função da chegada da primeira tarefa, para o padrão GWF o tempo de chegada é dado em função de um relógio global. Para resolver esta diferença, para o caso de *trace* extraído do padrão GWF, o valor do tempo de chegada é recalculado em relação à primeira tarefa.

Os valores de **Tamanho de computação** e **Tamanho de comunicação**, também são problemáticos no processo de conversão para o padrão GWF. Embora esteja previsto para o padrão GWF que o mesmo apresente informação sobre a quantidade de rede consumida pela tarefa, observou-se que este campo apresenta-se inválido em todas as amostras de *traces* obtidas. Portanto, o procedimento para estes casos é análogo ao tomado para o padrão SWF, descrito na seção anterior.

Para o tratamento de comentários do padrão GWF, o processo de conversão para o padrão WMS simplesmente ignora tal informação, sendo que para este caso em específico também devem ser tratadas linhas em branco, que podem existir neste tipo de arquivo de *trace*. A figura 4.9 apresenta o arquivo WMS gerado a partir do exemplo descrito na figura 4.8, onde os valores dos campos extraídos estão destacados pelo mesmo padrão de cores.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<!DOCTYPE system SYSTEM "iSPDcarga.dtd">
<system>
<trace>
<format kind="GWF" />
<task id="0" arr="0" sts="1" cpsz ="0" cmsz="-1" usr="user386" />
<task id="1" arr="48" sts="1" cpsz ="19" cmsz="-1" usr="user112" />
<task id="2" arr="74" sts="1" cpsz ="10" cmsz="-1" usr="user112" />
<task id="3" arr="90" sts="1" cpsz ="8" cmsz="-1" usr="user112" />
</trace>
</system>
```

Figura 4.9: Arquivo de *trace* no padrão WMS obtido apartir do exemplo da figura 4.8

4.3 Adaptação do iSPD para utilização de cargas de trabalho obtidas de *traces*

Após o processo de especificação do padrão de *trace* WMS e desenvolvimento do componente de conversão de arquivos de *traces* externos para o padrão da ferramenta, é necessário que possam haver meios de que essa carga seja utilizada pela ferramenta. Desta forma, é necessário promover uma adaptação do iSPD, incluindo os processos de abertura e simulação das cargas contidas no banco próprio de cargas de *traces* sob formato WMS. Para uma melhor compreensão, antes de apresentar o processo desenvolvido para a inclusão de cargas de *trace*, apresenta-se uma breve descrição do processo de abertura e salvamento de modelos de ambiente para o iSPD.

4.3.1 iSPD e cargas de trabalho

Como discutido na seção 2.2, a atual implementação do iSPD gera um modelo icônico de especificação do ambiente de simulação, ao qual estão vinculados o ambiente computacional da grade juntamente com a carga de trabalho atribuída para a mesma, que é salvo em um arquivo XML, sob o formato IMS.

Durante a execução da plataforma iSPD, dada a existência de um arquivo IMS, a abertura do mesmo resume-se primeiramente em verificar a integridade do arquivo, através de um arquivo DTD chamado `"iSPD.dtd"`, que contém as regras de construção de um arquivo IMS válido. Após a verificação pelo DTD, o arquivo é interpretado e são carregados para a interface principal, tanto a configuração de ambiente quanto as configurações de carga de trabalho. A figura 4.10 apresenta um diagrama deste processo. Da mesma forma, ao encerrar a execução de uma instância do iSPD, ou por opção do usuário, o processo de salvar um arquivo, se resume a construir um modelo IMS contendo a informação do ambiente e carga de trabalho.

Esta abordagem para a formatação de arquivos IMS é obviamente intratável para conter *traces* de cargas de trabalho. Se isso fosse aplicado, tem-se primeiro a geração de arquivos de modelos muito grandes e depois uma amarração desnecessária entre modelo e carga. Assim, optou-se pela separação entre modelo da grade e modelo de carga em dois arquivos.

4.3.2 Incluindo cargas de trabalho provenientes de *traces*

Para que possam ser incluídas cargas reais contidas no padrão WMS, primeiramente houve uma alteração nos processos de abertura, configuração e salvamento de modelos. Desta forma, ao se abrir um modelo IMS, o arquivo `"iSPD.dtd"` verifica a integridade do mesmo e a rotina de interpretação irá carregá-lo na interface icônica, sendo que este pode ou não conter uma carga computacional vinculada. O modelo de carga apenas pode aparecer no modelo IMS para cargas definidas randomicamente ou por nó. Logo após a abertura do modelo IMS o usuário pode configurar uma carga utilizando a interface padrão do iSPD, ou selecionar um modelo de carga de *trace* WMS, tanto

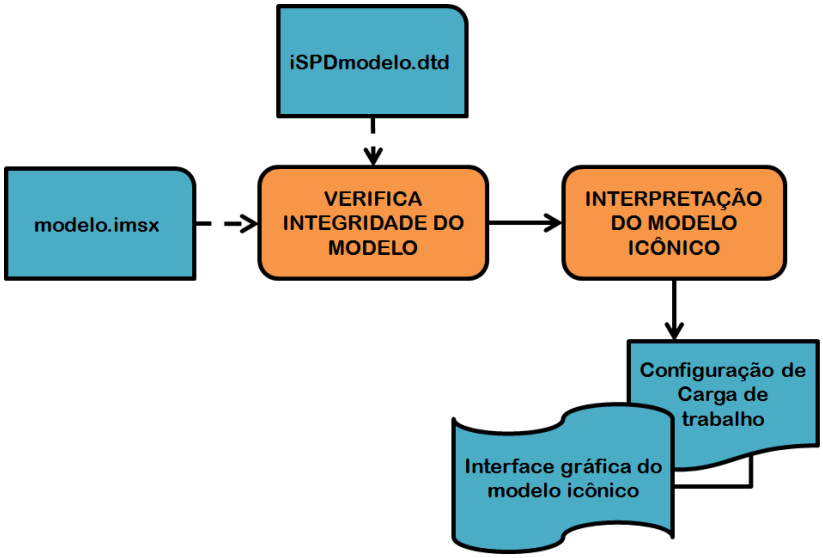

Figura 4.10: Diagrama de abertura de um modelo IMS

através da conversão de um formato de *trace* externo como pela abertura de um arquivo WMS anteriormente gerado.

A partir deste ponto um objeto *Interpretador* é construído, passando como parâmetro ao construtor da classe o caminho absoluto do arquivo selecionado e duas operações distintas podem ocorrer, dependendo do padrão de arquivo selecionado:

- **Conversão de arquivo externo:** se o padrão de arquivo selecionado for externo, o objeto *Interpretador* realiza uma chamada ao método *convert()*, que retornará um arquivo de carga WMS extraído do arquivo de *trace* original, além da informação sobre a origem (formato) e número de tarefas contidas em tal arquivo.
- **Leitura de arquivo WMS previamente existente:** Neste caso, o objeto *Interpretador* realiza uma chamada ao método *LerCargaWMS()* que irá verificar a integridade do arquivo WMS, retornando o arquivo selecionado, sua origem (formato) e número de tarefas contidas.

Ao fim do processo de leitura, com a confirmação do usuário, a ferramenta está configurada para gerar a carga a ser simulada a partir do modelo de carga de *trace* gerado. Desta forma, o esquema de abertura e configuração de arquivo de carga provenientes do banco de arquivos de *traces* está representado na figura 4.11.

4.3.3 Salvando *traces* da simulação realizada

Com a inclusão da possibilidade de uso de arquivos de *traces* como modelos de cargas de trabalho, é interessante que o iSPD possa também gerar *traces*. Assim, ao final de uma execução do iSPD a carga de trabalho simulada pode ser salva em um arquivo

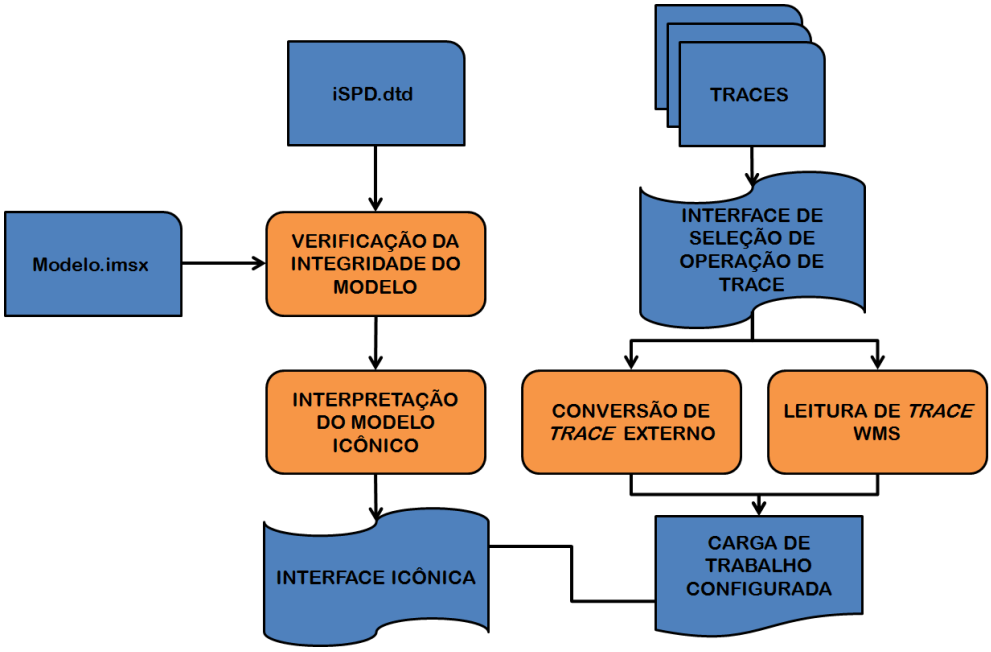

Figura 4.11: Diagrama de abertura de modelo com configuração de carga a partir de *trace*

WMS que será incluso no banco local de cargas de trabalho sob o padrão WMS. Para que tal processo ocorra, dado que o usuário deseja salvar o *trace* da simulação, o objeto *Interpretador* realiza uma chamada ao método *geraTraceSim()*, que recebe como parâmetro a lista de tarefas que foram simuladas e salva o arquivo WMS a partir das mesmas, retornando além do arquivo gerado a informação sobre o formato de *trace* definido como `"iSPD"`, além do número de tarefas contidas no *trace*.

A importância da geração de um *trace* da simulação é que este processo permite a simulação do mesmo conjunto de tarefas, contidas nesta carga em ambientes de grade diferentes ou aplicando políticas de escalonamento distintas. Isto é essencial para o estudo de características de ambiente simulado e de estratégias de escalonamento da tarefas em ambiente de grade computacional.

4.4 Simulando cargas de trabalho reais

Por fim, é preciso descrever o processo mais importante para este trabalho, que é permitir que as cargas de trabalho provenientes de *traces* sejam interpretadas e aplicadas aos modelos que serão simulados, ou seja, permitir a simulação do conjunto de tarefas contidas em arquivos de carga WMS. Primeiramente, se houver a seleção de modelo de *trace* é necessário que as tarefas a serem simuladas sejam criadas e armazenadas em uma estrutura de dados, a partir da descrição das mesmas no arquivo WMS selecionado. Este processo é realizado pela classe *CargaTrace.java* que implementa a classe abstrata

GerarCarga.java, presentes no pacote *ispd.motor.carga*. O construtor dessa classe recebe como parâmetros a carga WMS selecionada e as informações sobre tipo e número de tarefas de tais cargas e ao fim do processo retorna uma lista de tarefas que serão simuladas posteriormente. As subseções seguintes discutem o processo de geração de carga a ser simulada a partir de cada formato de *trace*.

4.4.1 Geração de carga a partir do formato SWF e GWF

Dado que o formato de *trace* selecionado para a geração de carga seja SWF ou GWF, as tarefas contidas no arquivo são divididas entre os mestres (escalonadores) presentes no ambiente de grade modelado e, a partir de cada linha de descrição no arquivo, um objeto Tarefa é construído a partir da classe *Tarefa.java* contida no pacote *ispd.motor.filas* e adicionado à lista de tarefas a serem simuladas. Para o caso de cargas geradas por SWF e GWF, o construtor da classe Tarefa recebe os parâmetros descritos a seguir:

- **ID da tarefa:** recebendo como parâmetro o valor do atributo `"id"` da tarefa lida no arquivo WMS;
- **Usuário proprietário:** recebendo como parâmetro o valor do atributo `"usr"` da tarefa lida no arquivo WMS;
- **Tamanho de comunicação:** Como descrito na seção 4.2, os formatos SWF e GWF não contém informação sobre quantidade de rede consumida pela tarefa. Portanto, este valor é gerado através de um gerador de números aleatório, utilizando a distribuição *two-stage uniform*(Lublin and Feitelson, 2001).
- **Tamanho de computação:** Como os padrões SWF e GWF armazenam o valor referente ao tempo de execução da tarefa em segundos e não em MFlops como usado pelo iSPD, para que possa ser obtido um valor para o tamanho computacional da tarefa, foi desenvolvido dentro da classe *CargaTrace.java* o método *MediaCapProcGrade()* que retorna a média da capacidade de computação da grade. Desta forma, para este parâmetro, o valor passado é a multiplicação dos valores de tempo de execução da tarefa e a média de capacidade de computação da grade.
- **Tempo de chegada da tarefa:** recebendo como parâmetro o valor do atributo `"arr"` da tarefa lida no arquivo WMS;

Ao fim desse processo, a lista de tarefas é construída e o processo de simulação ocorre normalmente, sendo possível a geração de um *trace* da simulação do *trace* selecionado e simulado.

4.4.2 Geração de carga a partir do formato iSPD

Como o *trace* no formato iSPD é resultante de uma simulação executada anteriormente, os valores nele contidos já são do mesmo tipo usado no simulador. Assim para o caso de carga WMS proveniente do formato iSPD, o construtor da classe *Tarefa* recebe os seguintes parâmetros:

- **ID da tarefa:** recebendo como parâmetro o valor do atributo `"id"` da tarefa lida no arquivo WMS;
- **Usuário proprietário:** recebendo como parâmetro o valor do atributo `"usr"` da tarefa lida no arquivo WMS;
- **Tamanho de comunicação:** recebendo como parâmetro o valor do atributo `"cpsz"` da tarefa lida no arquivo WMS;
- **Tamanho de computação:** recebendo como parâmetro o valor do atributo `"cmsz"` da tarefa lida no arquivo WMS;
- **Tempo de chegada da tarefa:** recebendo como parâmetro o valor do atributo `"arr"` da tarefa lida no arquivo WMS;

Após a geração da lista de tarefas para o padrão iSPD, o processo de simulação ocorre normalmente, sendo exibidos os resultados da simulação ao final.

4.5 Especificação da interface usuário-aplicação para uso do banco de cargas

Para que a estrutura descrita nas seções anteriores torne-se operacional, é necessário ainda a criação de interfaces entre o usuário e a ferramenta. Esta seção apresenta as principais interfaces responsáveis pela integração do banco de cargas desenvolvido pelos arquivos WMS à ferramenta iSPD.

Primeiramente, dentro da janela responsável pela configuração das cargas, além das opções de carga randômica e aplicada por nó, deve-se incluir também a configuração de carga de *trace*. A figura 4.12 apresenta a interface de configuração de cargas de trabalho, dado que a opção de carga de *trace* for selecionada, apresenta-se um painel requisitando ao usuário a seleção entre abrir um modelo de carga de um arquivo WMS previamente gerado ou realizar a conversão de um formato de *trace* externo.

A partir desta interface, conforme a opção selecionada são exibidas as seguintes interfaces:

- **Abertura de Modelo WMS:** Esta opção deve permitir o carregamento de um modelo WMS constituinte do banco próprio de cargas que tenha sido salvo previamente. A figura 4.13 apresenta a interface utilizada para este caso. Ao selecionar o botão `"Open"`, este abrirá uma janela de seleção de arquivo, onde é selecionado um modelo de cargas entre as disponíveis no banco próprio de cargas de trabalho sob a extensão `".wmsx"`. A partir deste ponto, o campo retangular ao lado do botão de abertura, exibe o caminho absoluto do arquivo selecionado e uma chamada ao método *LerCargaWMS()* do objeto *Interpretador* é realizada, retornando informações sobre nome, formato e número de tarefas que são exibidas no campo de notificações.

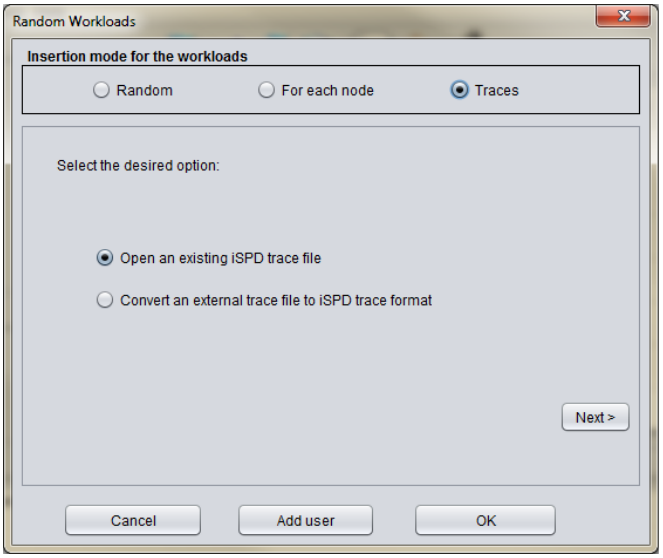

Figura 4.12: Interface de seleção de configuração de carga de *traces*

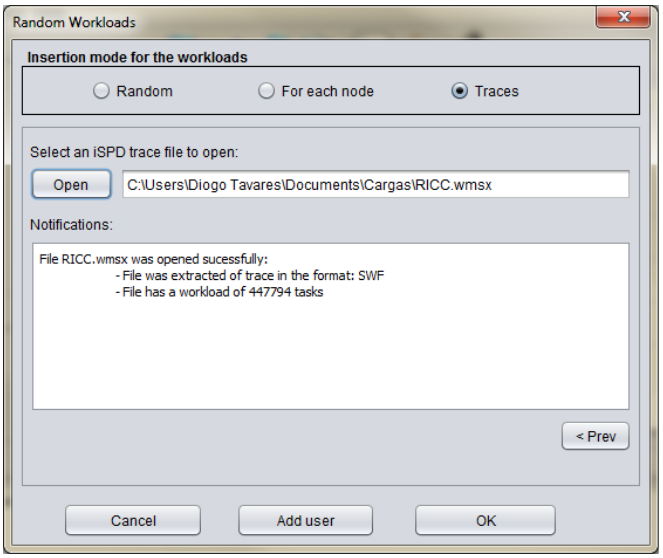

Figura 4.13: Interface de abertura de modelo de carga de trabalho WMS

- **Opção de conversão de carga extraída de *trace*:** Nesta opção, dado que o usuário acione o botão `"Open"`, o caminho do arquivo selecionado é exibido no campo retangular ao lado do botão. Após a seleção do arquivo, o botão `"convert"` inicia o processo de conversão para um modelo de carga de trabalho WMS através do método *convert()* do objeto *Interpretador*, exibindo ao final do procedimento as informações sobre o nome, tipo e número de tarefas contidas do arquivo no campo de notificações. A figura 4.14 apresenta a interface desta funcionalidade.

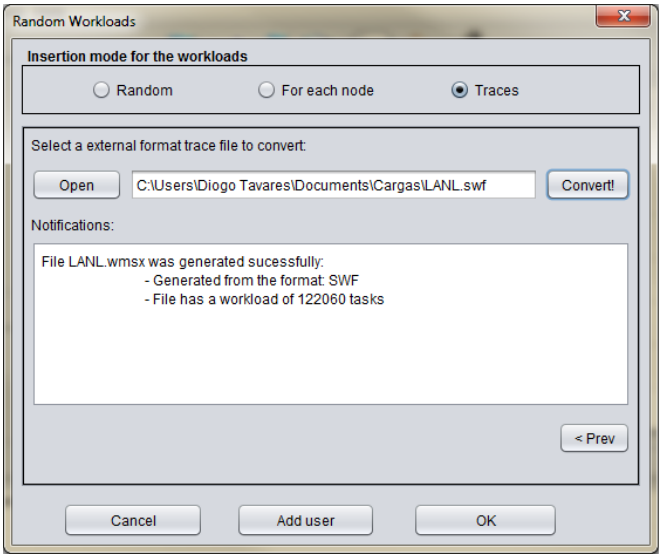

Figura 4.14: Interface do conversor de arquivos de *traces*

Foi desenvolvida ainda, a interface criada para permitir que o usuário escolha também se deseja salvar um *trace* da simulação, logo após o fim de um processo de simulação. Se a escolha foi afirmativa, o usuário configura o diretório e nome do arquivo a ser salvo em uma interface de seleção de arquivo, e a chamada ao método *geraTraceSim()* do objeto *Interpretador* é realizada, exibindo informações sobre o nome, formato e número de tarefas do arquivo gerado no campo de notificações da janela principal do iSPD. A figura 4.15 apresenta a janela de escolha da opção de geração do arquivo de *trace* WMS da simulação.

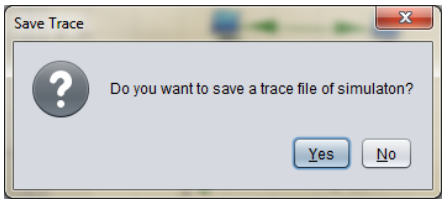

Figura 4.15: Interface do seletor para a geração de arquivo de *trace* da simulação realizada

4.6 Considerações finais

Neste capítulo apresentou-se o desenvolvimento da infraestrutura necessária para a construção de um banco próprio de cargas de trabalho para a plataforma iSPD, incluindo a apresentação das interfaces necessárias para seu funcionamento. O próximo capítulo apresenta os testes para a validação da ferramenta desenvolvida neste trabalho.

Capítulo 5

Testes e validação do banco de cargas de *trace* para o iSPD

Este capítulo apresenta uma descrição dos testes e resultados obtidos com o intuito de validar o trabalho desenvolvido neste projeto. Um primeiro teste apresenta um estudo de custo temporal e espacial para conversão de *traces* externos para o padrão WMS. Um segundo aplica o banco de cargas de trabalho para o estudo de políticas de escalonamento.

Não serão apresentados aqui testes de verificação da correção do processo de conversão. Considera-se que estes testes são úteis apenas no desenvolvimento da aplicação, a qual apenas tem utilidade se fizer a conversão de forma correta.

5.1 Teste de custo de conversão de *trace*

Neste experimento mediram-se os custos temporal (tempo decorrido para conversão para o formato WMS) e espacial (tamanho do arquivo WMS gerado) decorrentes da execução do processo de conversão de arquivos de *traces* externos para o padrão WMS. Inicialmente, foram selecionadas amostras de arquivos de *trace* externos de diferentes tamanhos e com diferentes quantidades de tarefas para os padrões SWF e GWF. A partir da execução do método *convert()*, foram obtidos valores de tempo de conversão e tamanho de arquivo gerado para o padrão WMS. Para a realização de tais testes foi utilizada uma máquina com as seguintes configurações:

- Processador Intel i5;
- Memória RAM de 4 GBytes;
- Sistema Operacional Windows 7 *Home Premium* (64 bits);
- *Java Virtual Machine* (JVM) utilizando Java 7 *Standard Edition* (32 bits);

As seções subsequentes apresentam os resultados obtidos para os formatos SWF e GWF.

5.1.1 Resultados obtidos para conversão de arquivos SWF

Para o padrão SWF, foram selecionadas 5 amostras dentre os 30 arquivos disponíveis em PWA (PWA, 2012c), sendo que tais amostras foram selecionadas pela variedade de tamanhos de arquivo e número de tarefas contidas. Uma descrição mais detalhada das amostras selecionadas é dada a seguir:

- `NASA.swf` (PWA, 2012b): Contendo informações sobre a execução de tarefas durante o período de outubro a dezembro de 1993 em um hipercubo iPSC/860 de 128 nós pertencente a NASA (*National Aeronautics and Space Administration*);
- `LANL.swf` (PWA, 2012a): Contendo informações sobre tarefas executadas em um Thinking Machines CM-5 de 1024 nós, durante o período de outubro de 1994 a setembro de 1996, pertencente ao LANL (*Los Alamos National Lab*).
- `SDSC.swf` (PWA, 2012e): Contendo a informação obtida da execução de tarefas em um IBM SP de 144 nós, durante o período de abril de 2000 a janeiro de 2003, pertencente ao SDSC (*San Diego Supercomputer Center*);
- `RICC.swf` (PWA, 2012d): Retirado da execução de tarefas em uma estrutura de *cluster* de *clusters*, pertencente ao projeto RICC (*RIKEN Integrated Cluster of Cluster*), localizado no Japão, durante o período de maio a setembro de 2010;
- `SHARCNET.swf` (PWA, 2012f): Retirado da execução de tarefas no grid SHARCNET, durante o período de dezembro de 2005 a janeiro de 2007, sendo formado pelo conjunto de 10 *clusters* pertencentes a diversas instituições de pesquisas em Ontário no Canadá.

A partir da aplicação do processo de conversão nas amostras selecionadas, foram obtidos os seguintes resultados apresentados na tabela 5.1:

Nome Arquivo	**Arquivo original(MB)**	**Arquivo gerado(MB)**	**Número de tarefas**	**Tempo execução(s)**
NASA	1,6	1,3	18239	0,5
LANL	10,8	9,1	122060	2,9
SDSC	21,4	18,3	243314	5,1
RICC	40,4	33,6	447794	9,4
SHARCNET	109,0	91,2	1195242	26,0

Tabela 5.1: Tabela com os valores de tempo de conversão e tamanho de arquivo WMS gerado para o padrão SWF

A figura 5.1 apresenta a relação entre o tamanho do arquivo SWF original e o tamanho de arquivo WMS gerado. Pode-se perceber que o arquivo WMS gerado apresenta aproximadamente 80% do tamanho do arquivo SWF original. Isto se deve ao fato

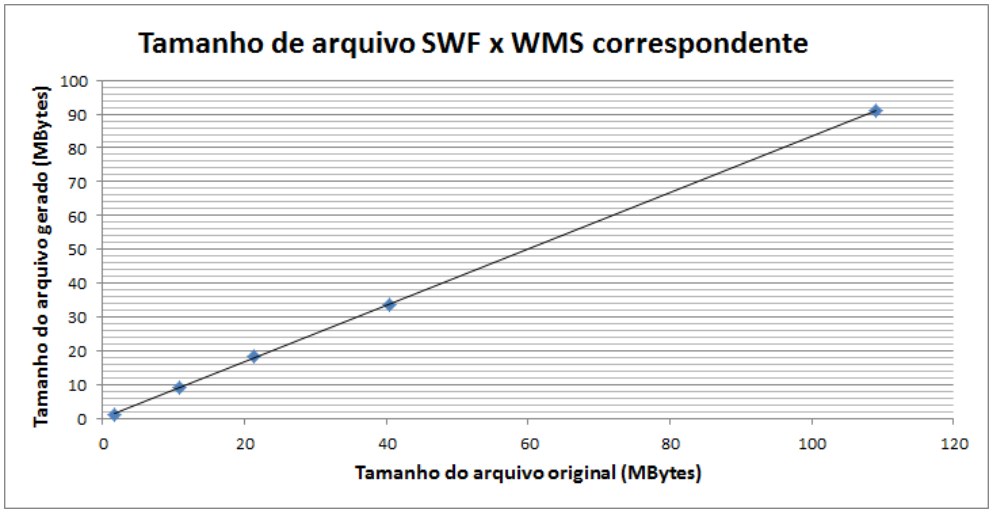

Figura 5.1: Relação entre tamanhos de arquivo de *trace* SWF e WMS gerado

de que o padrão WMS utiliza uma quantidade menor de caracteres por linha para a representação de cada tarefa do *trace*.

Quanto ao custo temporal de conversão, a figura 5.2 apresenta uma relação entre o número de tarefas constituinte do *trace* e o tempo gasto para a conversão do mesmo para o padrão WMS. Pode-se notar uma tendência de custo de conversão linear, o que é justificado pelo fato de que o método utilizado percorre o arquivo linearmente uma única vez, para que o processo de conversão seja realizado. Observa-se ainda que mesmo para o maior arquivo (109 MB e 1,2 milhões de tarefas) o tempo consumido foi bastante razoável (cerca de 26 segundos).

5.1.2 Resultados obtidos para conversão de arquivos GWF

De maneira análoga ao processo realizado para o padrão SWF, foram selecionadas três amostras de arquivos GWF dentre as cinco disponíveis em GWA (GWA, 2012d), tendo novamente como critério a variedade de tamanhos de arquivo e número de tarefas contidas. As amostras selecionadas para o padrão GWF são descritas a seguir:

- `AUVERGRID.gwf` (GWA, 2012a): Contendo informações sobre a execução de tarefas no ambiente de grid formado por cinco *clusters* distribuídos geograficamente na região de Auvergne, na França;
- `NORDUGRID.gwf` (GWA, 2012e): Contendo informações de execução de tarefas no ambiente de grade Nordugrid, situado no norte da Europa, e que reúne recursos de diversas organizações;
- `DAS-I.gwf` (GWA, 2012b): Contendo a informação obtida da execução de tarefas no ambiente de grade DAS-2, localizado na Holanda, que interliga recursos de cinco instituições acadêmicas.

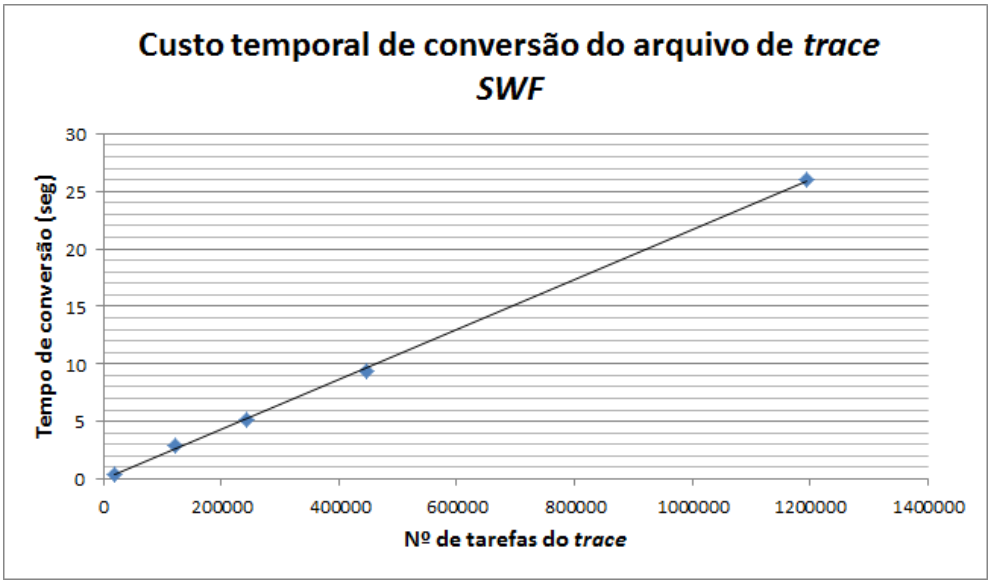

Figura 5.2: Relação entre o número de tarefas do *trace* SWF e o tempo decorrido para a conversão ao padrão WMS correspondente

A tabela 5.2 apresenta os resultados obtidos pela aplicação do processo de conversão às amostras de arquivos GWF selecionadas:

Nome Arquivo	**Arquivo original(MB)**	**Arquivo gerado(MB)**	**Número de tarefas**	**Tempo execução(s)**
AUVERGRID	47,1	26,0	404176	10,5
NORDUGRID	131,0	58,4	781370	21,7
DAS-I	205,0	78,7	1124772	34,8

Tabela 5.2: Tabela com os valores de tempo de conversão e tamanho de arquivo WMS gerado para o padrão GWF

A relação de tamanho de arquivo WMS gerado a partir do *trace* GWF está apresentada na figura 5.3. Observa-se que para o caso de conversão do arquivo GWF, o arquivo WMS gerado apresenta aproximadamente entre 40% e 50% do tamanho do arquivo GWF original. Esta diferença é maior para o padrão GWF em relação ao padrão SWF pois o arquivo GWF original armazena mais campos que o SWF, sendo que estes não são utilizados no padrão do iSPD.

Em relação ao custo temporal de conversão, a figura 5.4 apresenta uma relação entre o número de tarefas do *trace* GWF e o tempo gasto para a conversão do mesmo para o padrão WMS. Assim como para o caso anterior, pode-se observar uma tendência de custo de conversão linear. Entretanto, o tempo para que um arquivo GWF seja convertido é ligeiramente maior, devido ao fato de que existe o tratamento de uma quantidade maior de campos de informação para cada tarefa, além dos valores de tempo

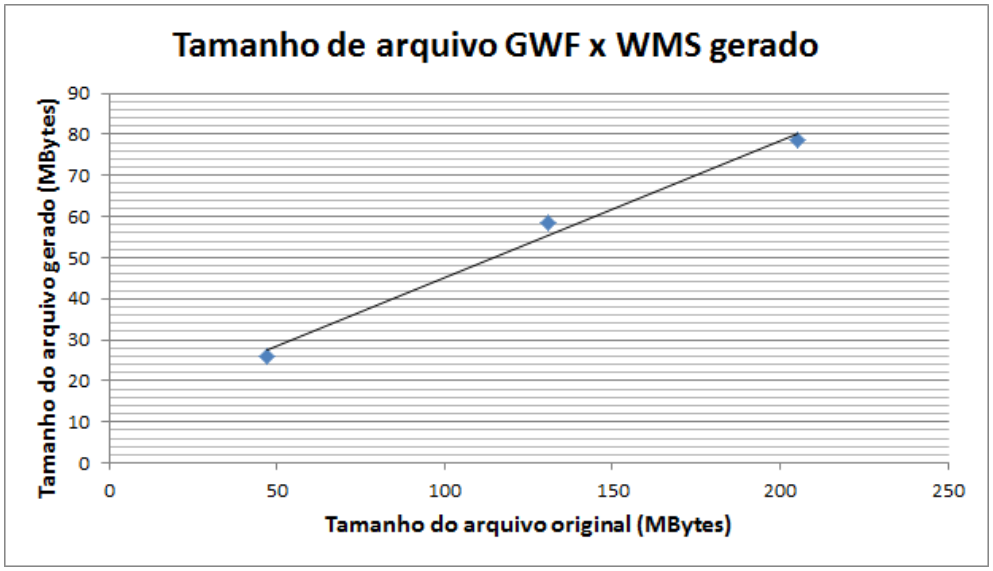

Figura 5.3: Relação entre tamanhos de arquivo de *trace* GWF e WMS gerado

de chegada da tarefa que devem ser tratados antes de escritos no arquivo WMS.

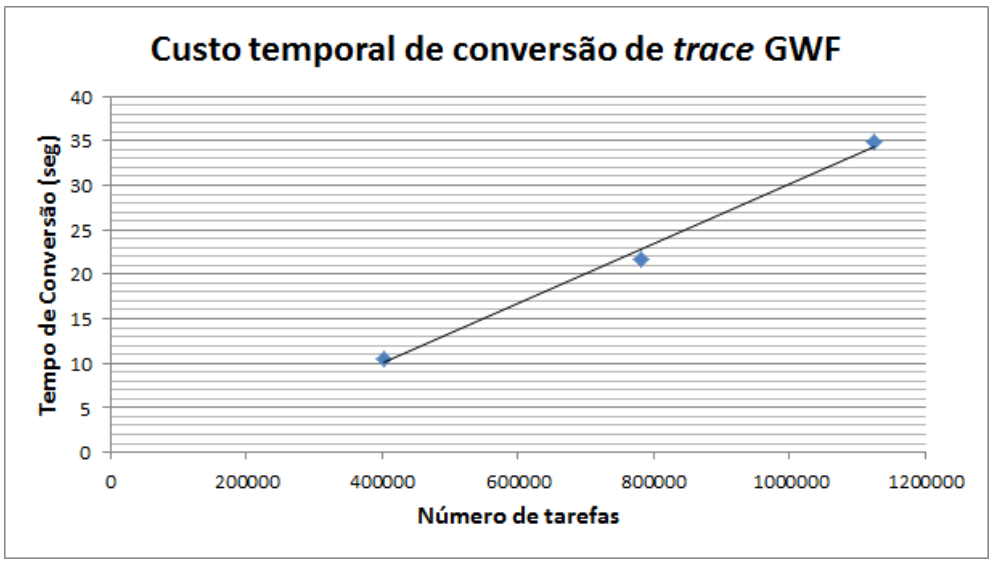

Figura 5.4: Relação entre o número de tarefas do *trace* GWF e o tempo decorrido para a conversão ao padrão WMS correspondente

5.2 Cargas de trabalho reais aplicadas ao estudo de políticas de escalonamento

Esta seção apresenta o desenvolvimento de testes de simulação de cargas de trabalho reais presentes no banco próprio de cargas, procurando observar as métricas obtidas pela simulação de um *trace* de alto desempenho, aplicando políticas de escalonamento variadas. Primeiramente foram definidas as configurações do ambiente simulado e da carga de *trace* aplicada ao mesmo e, a partir destas definições, foram aplicadas três políticas de escalonamento distintas, obtendo-se ao fim as métricas de simulação para a execução do conjunto de tarefas sob cada política de escalonamento aplicada. Estes testes são apresentados a seguir.

5.2.1 Ambiente experimental

O ambiente de grade a ser simulado é apresentado na figura 5.5. Para este modelo de grade observa-se que existe apenas um mestre, a máquina *icon0* no centro da imagem, que é responsável pelo escalonamento das tarefas, e para este mestre existem 9 escravos com capacidade de processamento de 100 MFlops cada um, com exceção das máquinas *icon7* e *icon8*, que apresentam capacidade de processamento igual a 300 MFlops. Quanto a comunicação, os enlaces de comunicação possuem capacidade de 100 Mbps e as máquinas *icon0, icon3, icon5, icon6* e *icon9* estão ligadas sob a internet pública, com velocidade de 100 Mbps.

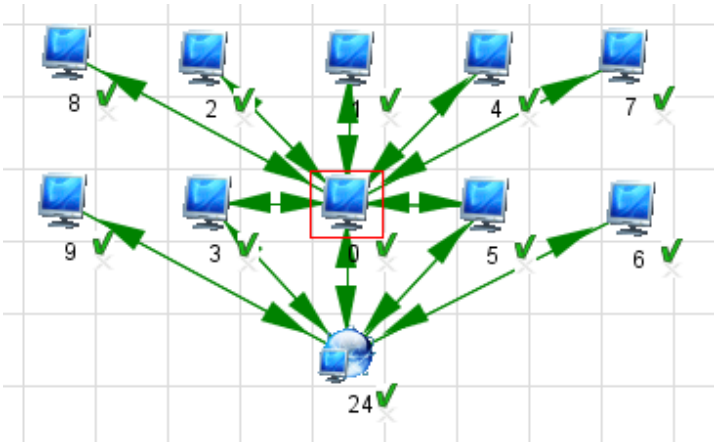

Figura 5.5: Modelo de grade utilizado para o ambiente experimental de testes

Foram selecionadas três políticas de escalonamento para a este conjunto de testes, descritas a seguir:

- *Round-Robin:* Algoritmo de escalonamento estático (não baseado em informações obtidas durante a simulação) que simplesmente atribui de maneira uniforme as tarefas a serem executadas para os recursos sob o qual o mestre tem controle. Também é conhecido como escalonamento circular;

- *Workqueue:* Trata-se de um algoritmo estático, assim como o *Round-Robin*, mas em que o escalonamento ocorre sob a política de fila simples, onde dado que uma máquina esteja apta a executar a tarefa, tal tarefa será atribuída para esta máquina;
- *Dynamic FPLTF (Fast Processor to Largest Task First):* Algoritmo dinâmico (realiza busca de informações sobre o ambiente durante a execução e faz o escalonamento segundo as informações recebidas) que tem como política a execução das tarefas maiores primeiro pelos processadores com maior capacidade de computação que estejam disponíveis.

Quanto à carga selecionada para este teste, optou-se pelo uso do *trace* do arquivo `"NASA.swf"` descrito anteriormente, pela quantidade significativa de tarefas e usuários.

5.2.2 Resultados obtidos para o teste de simulação de carga real e estudo de políticas de escalonamento

Os resultados obtidos com a simulação do modelo descrito, com os escalonadores indicados, são apresentados na tabela 5.3. Nela são apresentados os valores obtidos para as métricas de ociosidade de processamento e comunicação, bem como o tempo total simulado e a eficiência da simulação, que para o iSPD é estimada pela média entre os valores de eficiência para cada tarefa simulada, o que é obtido pela fórmula:

$$Eficiencia = CargaComputacional/(CapacidadeRecebida * TempodeExecucao)$$

Esta métrica é dada como "Boa" para valores superiores a 70%, "Média" entre 40% e 70%, e "Ruim", se inferior a 40%.

Política Escalonamento	**Tempo Simulado(s)**	**Ociosidade de Proces.(%)**	**Ociosidade de Comun.(%)**	**Eficiência**
Round-Robin	7,95E+06	76,34	98,98	Boa
Workqueue	7,95E+06	77,88	98,99	Boa
DynFPLTF	7,95E+06	86,02	97,41	Boa

Tabela 5.3: Tabela com os valores de métricas de simulação de carga real sob três políticas de escalonamento distintas

Os resultados obtidos para as métricas apresentadas mostram pouca diferença entre os algoritmos *Workqueue* e *Round-Robin*, o que era esperado pela pequena diferenciação entre estas políticas. Já o *DynFPLTF* apresenta mais ociosidade de processamento, o que também era esperado por usar mais informação que os outros dois. Entretanto esses números escondem o real comportamento do sistema, que pode ser melhor determinado examinando-se cada nó separadamente.

As figuras 5.6, 5.7 e 5.8 apresentam os gráficos de utilização dos recursos para cada política de escalonamento, destacando como cada nó do ambiente experimental

executa a carga contida na amostra utilizada. Esta representação demonstra com mais clareza as diferenças de comportamento de execução entre políticas de escalonamento distintas.

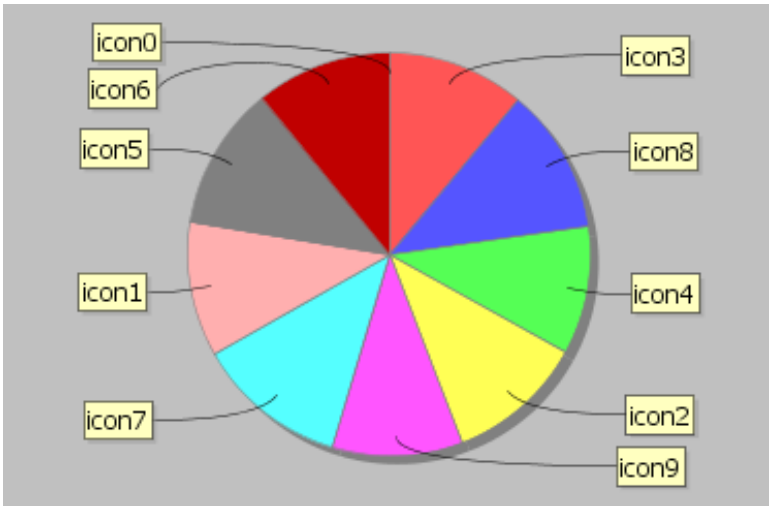

Figura 5.6: Gráfico de processamento por recurso do ambiente experimental para a política de escalonamento *Round-Robin*

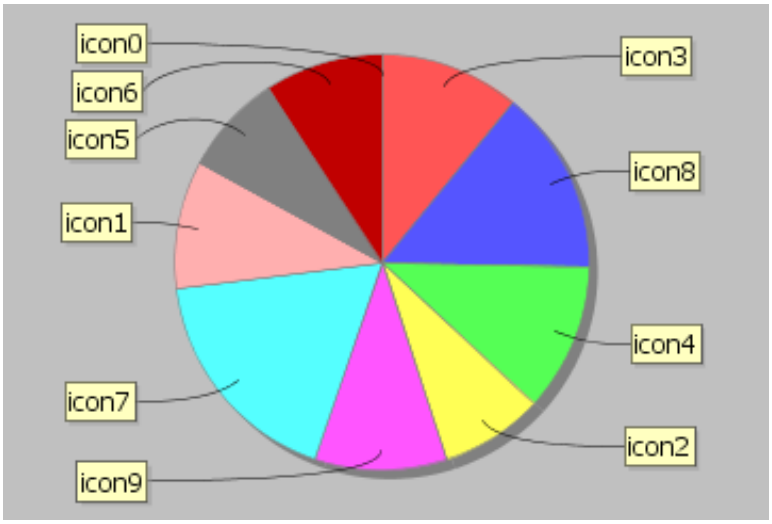

Figura 5.7: Gráfico de processamento por recurso do ambiente experimental para a política de escalonamento *Workqueue*

Pode-se observar que para o escalonamento *Round-Robin* as tarefas foram distribuídas igualmente entre os recursos. Já para o *workqueue*, os recursos *icon7* e *icon8* executam mais tarefas por possuírem maior capacidade de processamento, terminando mais rapidamente a execução das tarefas atribuídas e recebendo outras para que sejam

executadas. Por fim, é possível observar o comportamento diferenciado da política *Dynamic FPLTF*, onde as tarefas de maior tamanho computacional são executadas pelos recursos mais rápidos, *icon7* e *icon8*, o que fica evidente na figura 5.8.

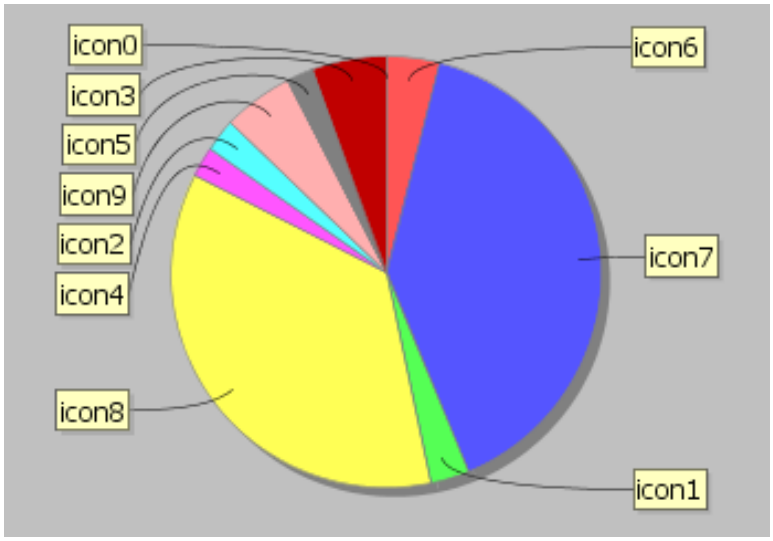

Figura 5.8: Gráfico de processamento por recurso do ambiente experimental para a política de escalonamento *Dynamic FPLTF*

5.3 Considerações finais

Os testes realizados e apresentados neste capítulo mostram que a utilização de arquivos de *traces* no iSPD é perfeitamente viável. Nó próximo capítulo são apresentados conclusões mais detalhadas bem como direções futuras para o iSPD e o tratamento de cargas de trabalho.

Capítulo 6

Conclusões

Este trabalho apresentou uma visão geral sobre conceitos fundamentais nos campos de pesquisa de avaliação de desempenho, simulação de eventos discretos e princiapalmente caracterização de cargas de trabalho. Além disto também ofereceu uma apresentação sobre a ferramenta iSPD, discutindo utilização e pontos de projeto. Por fim, este trabalho teve como maior contribuição o desenvolvimento e validação de tod estrutura necessária para a inclusão de um banco de cargas de trabalho flexível e realista, através da possibilidade de obtenção de cargas de trabalho extraídas de *traces* de aplicações de alto desempenho para esta plataforma de simulação, além da possibilidade de geração de cargas extraídas de traces de execução das simulações efetuadas pelo usuário.

6.1 Considerações finais

A plataforma de simulação de grades computacionais iSPD têm como intuito principal facilitar a iteração do usuário de ferramentas de avaliação de desempenho, buscando ser simples e intuitiva sem que isso prejudique sua precisão e estabilidade. Neste contexto, este trabalho contribuiu com o aumento das funcionalidades desta ferramenta. As principais conclusões que podem ser tiradas dele são:

- Facilita o processo de estudo de políticas de escalonamento, dado que simulação de cargas reais apresenta uma faixa de valores de tarefas e quantidade de usuários significativamente mais representativa que modelos aleatórios;
- Facilita o processo de comparação de diferentes configurações de grades. Sem arquivos de trace o usuário teria dificuldade em simular o modelo em condições equivalentes;
- Com a inclusão desse componente no iSPD é possível criar-se um banco de cargas de trabalho próprio, de maneira a permitir o armazenamento de cargas geradas em simulações realizadas pelo usuário, além de cargas reais extraídas apartir dos arquivos de *trace*;

- As modificações introduzidas apresentam-se com bom desempenho, tanto de processamento como de armazenamento, como os resultados obtidos nos testes apresentados no capítulo 5 puderam mostrar;
- O uso das cargas contidas no banco próprio de cargas de trabalho pode ser realizado de forma bastante simples, como é requisito de conceito do iSPD, oferecendo ao usuário facilidade de uso através de interfaces intuitivas.

6.2 Problemas encontrados

Dentre os problemas encontrados durante a realização deste trabalho destacam-se principalmente:

- Dificuldade de obtenção de *traces* de alto desempenho, principalmente devido ao pequeno número de padrões e amostras disponíveis.
- Dificuldade na obtenção arquivos ou bancos de traces que sejam de fácil acesso, o que é necessário para os possíveis usuários do iSPD, os quais não devem depender de ferramentas específicas de tais bases de dados, que nem sempre são tão simples de usar.
- Manutenção da facilidade de uso, mantendo o projeto simples e intuitivo, sem que isso interferisse no desempenho do componente desenvolvido.

6.3 Direções futuras

Dentre as direções futuras deste trabalho, destacam-se as seguintes atividades:

- Inclusão de formatos de traces de aplicações de alto desempenho provenientes de outros padrões encontrados, ampliando a quantidade de perfis de amostras no banco de cargas de trabalho.
- Paralelização do motor de simulação buscando melhorar o desempenho de execução deste componente, tornando o processo de simulação mais rápido;
- Simulação de erros e falhas buscando possibilitar a modelagem de comportamento mais realistas na simulação;
- Inclusão de *checkpointing* de execução permitindo simular a execução parcial das tarefas em algoritmos que usem preepção.
- Inclusão de dependência entre tarefas, permitindo a simulação de tarefas que interajam, como é o caso de programas paralelos.

6.4 Publicações

Este trabalho está incluso dentro do projeto mais amplo de desenvolvimento da plataforma iSPD e prestou contribuições direta ou indiretamente a uma série de publicações. Os trabalhos listados como 1, 2, 3 e 4 na lista a seguir são referentes ao projeto de iniciação científica "Plataforma de Simulação de Grades Computacionais: Geração de Algoritmos de Escalonamento", executado por este autor anteriormente a este trabalho, e que permitiu todo embasamento para sua realização. Já trabalho 5 envolve diretamente os resultados aqui apresentados.

1. *Interpretador de algoritmos de escalonamento para inserção de escalonadores em simulador de computação em grade.* Apresentado na II Escola Regional de Alto Desempenho de São Paulo (ERAD-SP 2011) (Menezes et al., 2011).

2. *iSPD: an iconic-based modeling simulator for distributed grids.* Apresentado no Annual Simulation Symposium (ANSS), Qualis B1, (Manacero et al., 2012)

3. *Scheduler simulation using iSPD, an iconic-based computer grid simulator.* Apresentado no IEEE Symposium in Computers and Communications (ISCC), Qualis A2, (Menezes et al., 2012).

4. *Plataforma de Simulação de grades Computacionais: Geração de Algoritmos de Escalonamento.* Apresentado no XXIII Congresso de Iniciação Científica da UNESP.

5. *Construção de Interpretador de Arquivos de Traces e Banco de Cargas de Trabalho para a ferramenta de simulação de grades iSPD.* Apresentado no XXIV Congresso de Iniciação Científica da UNESP.

6. Relatórios técnicos parciais e finais de atividades apresentados à Reitoria e ao CNPQ, em decorrência das atividades de iniciação científica sob financiamento por processo PIBIC/Reitoria nos períodos 2010-2011, 2011-2012 e 2012-2013.

Referências Bibliográficas

Aoqui, V., Guerra, A. I., Garcia, M. A. B. A., Oliveira, P. H. M. A., Lobato, R. S., and JR, A. M. (2010). Interpretador de modelos externos para simulador de grades computacionais. In *I Escola Regional de Alto Desempenho de São Paulo*, volume CD-ROM, pages 1–2, São Carlos. Universidade Presbiteriana Mackenzie.

BOINC (2012). Open-source software for volunteer computing and grid computing. Disponível em <http://boinc.berkeley.edu/index.php>.

Branco, K. R. L. J. C. (2004). *Índices de Carga e Desempenho em Ambientes Paralelos / Distribuídos - Modelagem e Métricas*. PhD thesis, Universidade de São Paulo (USP), São Carlos.

Buyya, R. and Murshed, M. (2002). Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. *Concurrency and Computation: Pract. and Exper.*, 14:1175–1220.

Calzarossa, M., Massari, L., and Tessera, D. (2000). Workload characterization issues and methodologies. In Haring, G., Lindemann, C., and Reiser, M., editors, *Performance Evaluation: Origins and Directions*, pages 459–482. Springer-Verlag. Lect. Notes Comput. Sci. vol. 1769.

Calzarossa, M. and Serazzi, G. (1993). Workload characterization: A survey. In *Proceedings of the IEEE*, pages 1136–1150.

Casanova, H., Legrand, A., and Quinson, M. (2008). SimGrid: a Generic Framework for Large-Scale Distributed Experiments. In *10th IEEE International Conference on Computer Modeling and Simulation*.

CERN (2012). Grid observatory. Disponível em <http://grid-observatory.org/index.php?id=74>.

E. Laure et al. (2006). Programming the Grid with gLite. Technical Report EGEE-TR-2006-001, CERN, Geneva.

Feitelson, D. G. (2011). *Workload Modeling for Computer Systems Performance Evaluation*. 0.34 edition. Disponível em http://www.cs.huji.ac.il/~feit/wlmod/.

Foster, I., Geisler, J., Nickless, B., Smith, W., and Tuecke, S. (1996). Software infrastructure for the i-way high-performance distributed computing experiment. In *High Performance Distributed Computing, 1996., Proceedings of 5th IEEE International Symposium on*, pages 562 –571.

Foster, I., Kesselman, C., and Tuecke, S. (2001). The anatomy of the grid: Enabling scalable virtual organizations. *The International Journal of Supercomputer Applications*, 15(3):200–222.

GangSim (2012). Gangsim: A simulator for grid scheduling studies with support for uslas. Disponível em <http://people.cs.uchicago.edu/ cldumitr/GangSim/>.

Google (2012a). Google cluster data. Disponível em <http://code.google.com/p/googleclusterdata/>.

Google (2012b). gsutil reference guide. Disponível em <https://developers.google.com/storage/docs/gsutil_reference_guide?hl=en>.

GSPD (2012). Gspd's homepage. Disponível em <http://www.dcce.ibilce.unesp.br/spd/>.

Guerra, A. I., Garcia, M. A. B. A., Oliveira, P. H. M. A., Aoqui, V., Lobato, R. S., and JR, A. M. (2010). Plataforma de simulação de grades computacionais: Interface icônica. In *I Escola Regional de Alto Desempenho de São Paulo*, volume CD-ROM, pages 1–2. Universidade Presbiteriana Mackenzie.

GWA, G. W. A. (2012a). Auvergrid trace analysis report. Disponível em <http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-4/trace_analysis_report.html>.

GWA, G. W. A. (2012b). Das-2 trace analysis report. Disponível em <http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-4/trace_analysis_report.html>.

GWA, G. W. A. (2012c). The grid workload format. Disponível em <http://gwa.ewi.tudelft.nl/TheGridWorkloadFormat_v001.pdf>.

GWA, G. W. A. (2012d). The grid worloads archive home page. Disponível em <http://gwa.ewi.tudelft.nl/pmwiki/pmwiki.php?n=Main.Home>.

GWA, G. W. A. (2012e). Nordugrid trace analysis report. Disponível em <http://gwa.ewi.tudelft.nl/pmwiki/reports/gwa-t-3/trace_analysis_report.html>.

Litzkow, M., Livny, M., and Mutka, M. (1988). Condor-a hunter of idle workstations. In *Distributed Computing Systems, 1988., 8th International Conference on*, pages 104 –111.

Lublin, U. and Feitelson, D. G. (2001). The workload on parallel supercomputers: Modeling the characteristics of rigid jobs. *Journal of Parallel and Distributed Computing*, 63:2003.

Manacero, A., Lobato, R., Guerra, A., Garcia, M., Oliveira, P., Aoqui, V., Menezes, D., and Silva, D. (2012). ispd: an iconic-based modeling simulator for distributed grids. In *Annals of 45th Annual Simulation Symposium*, volume CDROM of *ANSS12*, pages 1–8, Orlando, USA.

Menezes, D. (2012). Geração de algoritmos de escalonamento para simulação de grades computacionais. Master's thesis, Instituto de Biociências, Letras e Ciências Exatas (IBILCE), Universidade Estadual Paulista "Júlio de Mesquita Filho"(UNESP), Câmpus de São José do Rio Preto, São José do Rio Preto.

Menezes, D., Manacero, A., Lobato, R., da Silva, D., and Spolon, R. (2012). Scheduler simulation using ispd, an iconic-based computer grid simulator. In *Computers and Communications (ISCC), 2012 IEEE Symposium on*, pages 000637 –000642.

Menezes, D., Silva, D. T., and Manacero, A. (2011). Interpretador de algoritmos de escalonamento para inserção de escalonadores em simulador de computação em grade. In *II Escola Regional de Alto Desempenho de São Paulo*, volume CD-ROM, pages 1–4, São José dos Campos.

Meuer, H. W. and Gietl, H. (2012). Supercomputers - prestige objects or crucial tools for science and industry. Disponível em <http://http://www.top500.org/>.

Oliveira, P. H. M. A., Guerra, A. I., Garcia, M. A. B. A., Aoqui, V., JR, A. M., and Lobato, R. S. (2010). O motor de uma plataforma de simulação de grades computacionais. In *I Escola Regional de Alto Desempenho de São Paulo*, volume CD-ROM, pages 1–2. Universidade Presbiteriana Mackenzie.

OptorSim (2012). Simulating data access optimization algorithms - optorsim. Disponível em <http://optorsim.sourceforge.net/>.

PWA, P. W. A. (2012a). The los alamos national lab (lanl) cm-5 log. Disponível em <http://www.cs.huji.ac.il/labs/parallel/workload/l_lanl_cm5/index.html>.

PWA, P. W. A. (2012b). The nasa ames ipsc/860 log. Disponível em <http://www.cs.huji.ac.il/labs/parallel/workload/l_nasa_ipsc/index.html>.

PWA, P. W. A. (2012c). The parallel worloads archive home page. Disponível em <http://www.cs.huji.ac.il/labs/parallel/workload/>.

PWA, P. W. A. (2012d). The ricc log. Disponível em <http://www.cs.huji.ac.il/labs/parallel/workload/l_ricc/index.html>.

PWA, P. W. A. (2012e). The san diego supercomputer center (sdsc) blue horizon log. Disponível em <http://www.cs.huji.ac.il/labs/parallel/workload/l_sdsc_blue/index.html>.

PWA, P. W. A. (2012f). The sharcnet log. Disponível em <http://www.cs.huji.ac.il/labs/parallel/workload/l_ricc/index.html>.

PWA, P. W. A. (2012g). The standard workload format. Disponível em <http://www.cs.huji.ac.il/labs/parallel/workload/swf.html/>.

Reiss, C., Wilkes, J., and Hellerstein, J. L. (2011). Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA. Revised 2012.03.20. Posted at URL http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.

Takefusa, A. and Matsuoka, S. (2000). Performance issues in client-server global computing. *International Workshop on Global and Cluster Computing (WGCC'2000)*.

Zhao, Y., Shao, G., and Yang, G. (2008). A survey of methods and applications for trace analysis in grid systems. In *Proceedings of the The Third ChinaGrid Annual Conference (chinagrid 2008)*, CHINAGRID '08, pages 264–271, Washington, DC, USA. IEEE Computer Society.