Um Escalonador de Tarefas Dependentes Robusto às Incertezas nas Descrições das Aplicações em Grades

Daniel Macêdo Batista , André Costa Drummond , Nelson Luis Saldanha da Fonseca

[1]Instituto de Computação – Universidade Estadual de Campinas (Unicamp)
Avenida Albert Einstein, 1251 – 13084-971 – Campinas – SP

{batista,andred,nfonseca}@ic.unicamp.br

Abstract. *Scheduling in grids usually considers the state of the grid resources as well the applications demands. However, if these demands are not appropriately assessed, scheduling decision can lead to poor performance. Thus, it is of paramount importance to consider uncertainties on these demands. This paper introduces the PLITD-FUZZY, a scheduler for dealing with such uncertainties. The performance of the proposed scheduler was evaluated and it was shown that it is specially attractive when there is a high degree of uncertainty.*

Resumo. *O escalonamento de tarefas em grades é realizado considerando-se o estado corrente dos recursos da grade e a descrição dos requisitos das aplicações. Requisitos informados incorretamente podem levar à alocações que apresentam desempenhos piores do que os previstos, tornando necessária a implementação de escalonadores que sejam robustos frente à incertezas. Neste artigo, apresenta-se o escalonador PLITD-FUZZY, um escalonador de tarefas que utiliza técnicas de otimização fuzzy para produzir escalonamentos robustos frente às incertezas na descrição das aplicações. Experimentos de simulação comprovam a eficácia do PLITD-FUZZY nos casos em que há incerteza na descrição da utilização da rede pelas aplicações.*

1. Introdução

Um dos primeiros passos realizados no processo de execução de aplicações em grades é o escalonamento das tarefas que compõem essas aplicações. A fim de tirar proveito do paralelismo possibilitado pelos recursos distribuídos que formam as grades, as aplicações são decompostas em tarefas, cabendo ao escalonador definir o escalonamento, ou seja, qual o melhor recurso e qual o melhor instante de tempo em que cada tarefa deve ser executada para alcançar um objetivo específico. Quando o objetivo é a minimização do tempo de execução, mais conhecido como *makespan*, a busca pelo escalonamento torna-se um problema $\mathcal{NP}$ difícil [Papadimitriou and Steiglitz 1998]. A complexidade envolvida na alocação dos recursos justifica a adoção de heurísticas e métodos aproximados na implementação de escalonadores de tarefas para grades [Batista et al. 2006].

Mesmo com a utilização de escalonadores que executem rápido e que forneçam soluções próximas da ótima, o fato das grades serem ambientes distribuídos, dinâmicos, não dedicados e não sujeitos a um controlador central [Foster 2002] [Skillicorn 2002] aumenta a complexidade envolvida no processo de alocação de recursos. O escalonamento gerado em um dado instante de tempo t para uma aplicação submetida à grade pode tornar-se ineficiente caso, depois de um intervalo de tempo Δt $<$

$makespan$, os recursos previamente alocados diminuam a capacidade disponibilizada para a grade, falhem ou surjam recursos melhores do que aqueles disponíveis no instante t. As incertezas no estado dos recursos da grade costumam ser tratadas por *frameworks* baseados em medições, reescalonamentos e migrações de tarefas como em [Vadhiyar and Dongarra 2003], [Sun and Wu 2005] e [Batista et al. 2007].

Tratar somente as incertezas presentes nos recursos da grade não garante a robustez do escalonamento durante toda a execução da aplicação. Como os escalonamentos são fornecidos considerando a descrição da aplicação submetida pelo usuário, informações incorretas na descrição podem ocasionar escalonamentos que levam a um desempenho diferente do previsto. As incertezas na descrição das aplicações surgem por diversos motivos como desconhecimento do usuário, falhas na aplicação ou até mesmo por usuários maliciosos que descrevem as aplicações com requisitos diferentes a fim de obter maior prioridade no escalonamento. Esses problemas não são exclusivos das grades e já foram bastante estudados em sistemas paralelos convencionais [Chiang et al. 1994] [Chiang et al. 2002] [Lee et al. 2004]. O problema de reutilizar as soluções existentes para grades vem do fato delas serem voltadas para ambientes confinados em redes locais onde a rede não desempenha um papel importante e consequentemente pode ser ignorada, o que não é o caso das grades.

De forma semelhante ao que ocorre com as incertezas nos recursos da grade, as incertezas na descrição das aplicações são expressas através da Qualidade de Informação (QoI – *Quality of Information*) [Casanova et al. 2000] que designa a porcentagem de certeza que se tem sobre as demandas das aplicações. O valor de 100% corresponde a uma aplicação descrita precisamente.

Uma solução, a princípio simples, para diminuir o impacto causado pelas incertezas na descrição das aplicações seria aumentar a QoI através da implementação de um sistema que registrasse o histórico das execuções anteriores como proposto em [Prodan and Fahringer 2005]. Entretanto, essa solução é ineficiente para aplicações executadas poucas vezes e para aquelas que, mesmo sendo executadas diversas vezes, não apresentam um comportamento fácil de inferir.

Soluções para tratar as incertezas em grades têm sido propostas e baseiam-se tanto em soluções reativas [Casanova et al. 2000] [da Silva and Scherson 2000] [Sakellariou and Zhao 2004] quanto em soluções preventivas [Cirne 2001] [Oikonomakos et al. 2007] [Carole et al. 2007] que levam em consideração a QoI das aplicações e tentam aumentar o conhecimento sobre as aplicações através da análise do histórico das execuções passadas. Entretanto, as soluções propostas falham por não identificarem todas as incertezas presentes em aplicações formadas por tarefas dependentes, e, em especial, aquelas com incertezas sobre a quantidade de dados transferidos entre as tarefas. Uma outra fonte de erro consiste em não considerarem ambientes heterogêneos, característica marcante das grades reais.

Neste artigo, propõe-se um novo escalonador de tarefas em grades que fornece escalonamentos robustos frente às incertezas na descrição de aplicações formadas por tarefas dependentes e descritas por Grafos Acíclicos Orientados (DAGs – *Directed Acyclic Graphs*). Diferente de outras propostas apresentadas na literatura, esse escalonador trata incertezas nas demandas de comunicação das aplicações e não considera ne-

nhuma topologia específica formada pelos recursos da grade. O escalonador, batizado de PLITD-FUZZY, baseia-se em técnicas de otimização fuzzy e é comparado com o escalonador PLITD-MODIF, uma versão modificada do escalonador PLITD apresentado em [Batista et al. 2006]. Resultados obtidos através de simulações apontam que o escalonador PLITD-FUZZY apresenta um desempenho melhor do que o escalonador PLITD-MODIF em ambientes com QoI entre 20% e 33,33%. Além disso, o custo computacional do PLITD-FUZZY é menor do que o do escalonador PLITD-MODIF, tornando-o ideal mesmo quando a incerteza extrapola a esperada.

O restante deste artigo está organizado da seguinte forma: a Seção 2 fornece os argumentos que motivaram a pesquisa realizada. A Seção 3 resume o escalonador PLITD-MODIF e descreve o escalonador proposto PLITD-FUZZY. A Seção 4 avalia o desempenho do escalonador PLITD-FUZZY através de simulações e comparações com o escalonador PLITD-MODIF. A Seção 5 apresenta os trabalhos relacionados. Por último, a Seção 6 conclui o artigo e aponta os trabalhos futuros.

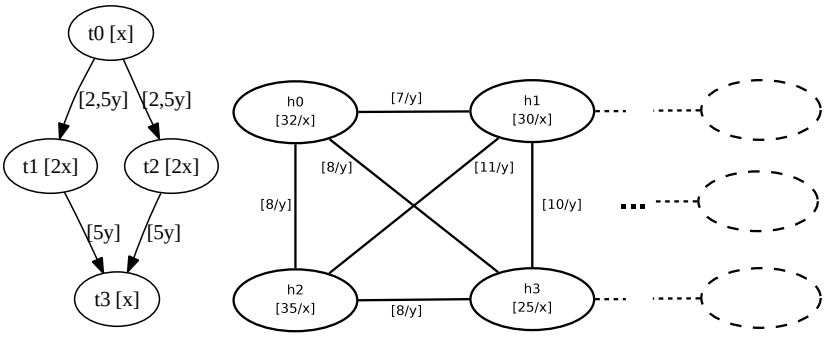
2. Motivação

Com a adoção do termo "grade", os cientistas e engenheiros da computação buscaram trazer várias características das grades do sistema elétrico para as grades computacionais [Foster and Kesselman 1999]. Dentre estas características, destaca-se a criação de um ambiente autônomo tanto para a grade quanto para os usuários. A fim de diminuir as intervenções dos usuários no processo de execução das aplicações, o escalonamento de tarefas deve ser automatizado e integrado à infraestrutura da grade, tirando do usuário a responsabilidade de escolher os recursos que serão utilizados. Apesar da integração do escalonador à grade, o usuário continua tendo um papel importante na descrição das demandas das aplicações e dos objetivos a serem alcançados com o escalonamento. O foco deste artigo são aplicações cujo objetivo é a minimização do tempo de execução, objetivo mais comum a ser alcançado em grades reais, e os escalonadores que escalonam um (1) DAG por vez. É importante destacar que o escalonamento de 1 único DAG por vez não limita a aplicação dos escalonadores, visto que 1 DAG pode descrever o agregado de várias aplicações. Para informações a respeito de composições de aplicações em 1 único DAG, e de escalonadores com outros objetivos diferentes da minimização do tempo de execução, recomenda-se que o leitor consulte [Zhao and Sakellariou 2006].

Pelo fato de haver a possibilidade das aplicações serem descritas de forma incorreta, não basta avaliar os escalonadores de tarefas pelo seu tempo de execução e pelo *makespan* previsto para a aplicação. Escalonadores de tarefas em grades devem ser robustos à incerteza na descrição das aplicações, ou seja, eles devem atender os objetivos definidos pelos usuários mesmo que as demandas das aplicações sejam informadas incorretamente.

O exemplo ilustrado na Figura 1 exemplifica a necessidade de se ter escalonadores robustos. Seja a aplicação descrita pelo DAG da Figura 1(a) a ser executada nos recursos representados pelo grafo da Figura 1(b) (Os itens tracejados na figura representam que a grade é formada por mais mais recursos, que não estão sendo exibidos para facilitar a visualização). As tarefas da aplicação são representadas pelos vértices do DAG e as dependências de dados entre duas tarefas são representadas pelos arcos. Cada arco no DAG interliga vértices que representam tarefas dependentes, de modo que, dado um arco (a, b), a tarefa a tem precedência sobre a tarefa b. O peso do vértice corresponde à quantidade de

instruções necessárias para a execução da tarefa relacionada. O peso do arco corresponde à quantidade de bits que devem ser transferidos para realizar a dependência de dados relacionada. O conhecimento do usuário influencia diretamente os pesos dos vértices e dos arcos do DAG. No DAG da Figura 1(a), cada vértice possui um rótulo e um peso, sendo este último representado entre colchetes. Para os arcos são exibidos somente os pesos entre colchetes.



(a) DAG. (b) Alguns *hosts* da grade.

Figura 1. Exemplo para ilustrar a motivação da pesquisa sobre incertezas na aplicação.

Neste artigo, considera-se que a grade fornece recursos de processamento (*hosts*) e recursos de comunicação (enlaces). De forma semelhante ao utilizado na descrição dos DAGs, a grade é descrita por um grafo. Os vértices representam os *hosts* e as arestas representam os enlaces. O peso de um vértice corresponde ao inverso da capacidade de processamento do *host* associado, em $(\frac{\text{instruções}}{\text{u.t.}})^{-1}$, onde u.t. representa a unidade de tempo. O peso de um enlace corresponde ao inverso da largura de banda disponível no enlace associado, em $(\frac{\text{bits}}{\text{u.t.}})^{-1}$.

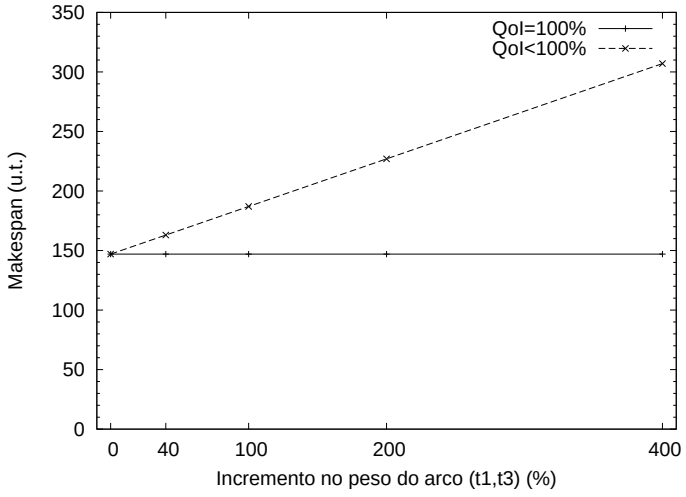


Figura 2. *Makespan* quando QoI=100% e quando QoI<100%.

A Figura 2 apresenta o gráfico do *makespan* da aplicação utilizando-se o escalonador PLITD-MODIF, um escalonador sem nenhuma técnica para suportar informações

incorretas das aplicações. Duas situações são exibidas no gráfico: (i) quando o peso do arco que interliga a tarefa $t1$ a $t3$ é descrito de forma incorreta pelo usuário (linha identificada por "QoI<100%"); e (ii) quando o peso do arco é descrito de forma correta, ou seja, uma situação em que o usuário soubesse de antemão o valor correto do peso e o informasse para o escalonador (linha identificada por "QoI=100%"). Quando QoI=100% o escalonamento fornecido faz a aplicação executar em 147u.t.. Em situações nas quais QoI<100%, o *makespan* alcançado pela aplicação sofre aumentos significativos. Quando o peso do arco $(t1, t3)$ na prática é 40, 100, 200 e 400% maior do que o valor informado pelo usuário, o escalonamento fornecido pelo PLITD-MODIF gera, respectivamente, *makespans* iguais a 11, 27, 54 e 109% maiores do que o esperado.

3. O escalonador proposto

Nesta seção é apresentada a principal contribuição do artigo, um escalonador de tarefas para grades que recebe como entrada, além do DAG com os pesos das tarefas, uma porcentagem de variação que os pesos podem sofrer; quanto maior a QoI, menor a porcentagem. O escalonador visa diminuir o impacto negativo causado pelas informações incorretas através de técnicas de otimização fuzzy.

O escalonador é denominado de PLITD-FUZZY (Programação Linear Inteira com Tempo Discreto - Fuzzy) e caracteriza-se por resolver um problema de programação inteira com o objetivo de minimizar o tempo de execução de aplicações formadas por tarefas dependentes. O termo "tempo discreto" no nome do escalonador deve-se à formulação do problema de programação inteira considerar que a linha do tempo de execução da aplicação é inteira ($\in \mathbb{Z}_+$). Apesar da discretização do tempo englobar aproximações que levam a soluções sub-ótimas, os ganhos no tempo de execução compensam eventuais perdas. O escalonamento é dado por variáveis que definem o host onde cada tarefa executará e o instante de tempo em que cada tarefa deve finalizar sua execução.

O desempenho do escalonador PLITD-FUZZY é avaliado através de comparações com o escalonador denominado de PLITD-MODIF. Tanto o escalonador PLITD-FUZZY, robusto frente às incertezas na descrição das aplicações e apresentado como contribuição neste artigo, quanto o escalonador PLITD-MODIF, utilizado na comparação dos ganhos do escalonador PLITD-FUZZY, tomam como base o escalonador PLITD proposto em [Batista et al. 2006].

O escalonador PLITD-FUZZY foi construído através da aplicação de técnicas fuzzy sobre o escalonador PLITD-MODIF. Como as comparações na análise de desempenho da Seção 4 são feitas entre o escalonador PLITD-FUZZY e o escalonador PLITD-MODIF, e a única diferença entre os dois é a aplicação das técnicas fuzzy, os ganhos obtidos são decorrentes única e exclusivamente pela aplicação de técnicas fuzzy. Apesar deste artigo focar no escalonador PLITD-FUZZY, o escalonador PLITD-MODIF é apresentado, dado que é utilizado na comparação com o escalonador PLITD-FUZZY.

As subseções 3.1 e 3.2 apresentam, respectivamente, os escalonadores PLITD-MODIF e PLITD-FUZZY, bem como as modificações implementadas sobre o escalonador PLITD-MODIF que resultam no escalonador PLITD-FUZZY. Ambos escalonadores escalonam aplicações formadas por tarefas dependentes em grades com o objetivo de minimizar o tempo de execução.

3.1. Escalonador PLITD-MODIF

O escalonador PLITD-MODIF resolve um problema de programação inteira que recebe dois grafos como entrada. O grafo $H = (V_H, A_H)$ representa a topologia formada pelos recursos da grade e o DAG $D = (V_D, A_D)$ representa as dependências entre as tarefas da aplicação a ser escalonada. V_H é o conjunto de m *hosts* da grade conectados pelo conjunto de enlaces A_H. Os *hosts* são identificados pelos rótulos $\{1, \ldots, m\}$. V_D é o conjunto de n tarefas da aplicação, com suas dependências representadas pelo conjunto de arcos A_D. As tarefas são identificadas pelos rótulos $\{1, \ldots, n\}$ de modo que a ordem crescente dos rótulos representa uma ordem topológica do DAG. Os DAGs aceitos pelo escalonador devem ter uma única tarefa de entrada e uma única tarefa de saída. DAGs que não atendam a essas condições devem ser modificados através da adição de tarefas artificiais com pesos zero.

As características do DAG da aplicação que precisam ser conhecidas são: I_i: quantidade de instruções da tarefa i ($I_i \in \mathbb{R}_+$); e $B_{i,j}$: quantidade de bits dos dados de dependência entre as tarefas i e j ($B_{i,j} \in \mathbb{R}_+$). Esses dois conjuntos de pesos são as informações que dependem do conhecimento do usuário. As características do grafo da grade que precisam ser conhecidas são: TI_k: intervalo de tempo que o *host* k leva para executar uma instrução ($TI_k \in \mathbb{R}_+$); $TB_{k,l}$: intervalo de tempo necessário para transferir um bit pelo enlace que conecta o *host* k ao *host* l ($TB_{k,l} \in \mathbb{R}_+$); e $\delta(k)$: conjunto de *hosts* com enlace para o *host* k, incluindo o próprio *host* k. O escalonador PLITD-MODIF gera como saída um diagrama de Gantt que fornece informações sobre os *hosts* alocados para cada tarefa, o instante de tempo de início da execução das tarefas e o instante de tempo que as transferências de dados devem acontecer.

O escalonamento encontrado pelo escalonador PLITD-MODIF é dado pelos valores das variáveis $x_{i,t,k}$ ($\in \{0, 1\}$) e f_i ($\in \mathbb{N}^*$). $x_{i,t,k}$ é uma variável binária que vale 1 se, e somente se, a tarefa i terminar sua execução no instante de tempo t e no *host* k; f_i é uma variável inteira que armazena o instante de tempo no qual a tarefa i termina sua execução ($f_i \in \mathbb{N}^*$).

O objetivo do escalonador é minimizar o tempo de execução da aplicação ($Minimize(f_n)$). Por conveniência, na formulação do problema resolvido pelo PLITD-MODIF exibida a seguir, utiliza-se a notação $\mathcal{T} = \{1, \ldots, T_{max}\}$, onde T_{max} é o intervalo de tempo no qual a aplicação levaria para executar caso todas suas tarefas executassem serialmente no *host* mais rápido da grade, i.e., $T_{max} = \min(\{ TI_{k|k \in V_H} \}) \times \sum_{i=1}^{n} I_i$. O PLITD-MODIF resolve o seguinte problema de programação inteira:

$$\text{Minimize} \sum_{t \in \mathcal{T}} \sum_{k \in \mathcal{H}} t \, x_{n,t,k}$$

sujeito a

$$\sum_{t \in \mathcal{T}} \sum_{k \in V_H} x_{j,t,k} = 1 \qquad \text{para } j \in V_D; \qquad \text{(D1)}$$

$$x_{j,t,k} = 0 \qquad \text{para } j \in V_D, \quad k \in V_H, \quad t \in \{1, \ldots, \lceil I_j \, TI_k \rceil\}; \qquad \text{(D2)}$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - I_j\, TI_l - B_{i,j}\, TB_{k,l} \rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \quad \text{para } j \in V_D, \quad ij \in A_D, \quad \text{para } l \in V_H, \quad t \in \mathcal{T}; \tag{D3}$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + I_j\, TI_k - 1 \rceil} x_{j,s,k} \leq 1 \quad \text{para } k \in V_H, \quad t \in \mathcal{T}, \quad t \leq \lceil T_{max} - I_j\, TI_k \rceil; \tag{D4}$$

$$x_{j,t,k} \in \{0,1\} \quad \text{para } j \in V_D, \quad l \in V_H, \quad t \in \mathcal{T}. \tag{D5}$$

As restrições (D1) garantem que cada tarefa do DAG só pode executar em um único host e finalizar a sua execução em um único instante de tempo. As restrições (D2) garantem que uma tarefa só poderá finalizar sua execução em um instante de tempo suficiente para que todas suas instruções sejam executadas. As restrições (D3) definem que a j-ésima tarefa só pode iniciar a sua execução após as transferências dos dados de dependência. As restrições (D4) garantem que cada host só pode executar no máximo uma (1) tarefa em qualquer instante de tempo e as restrições (D5) definem o domínio das variáveis $x_{j,t,k}$.

A subseção seguinte descreve o escalonador PLITD-FUZZY, implementado com o objetivo de tratar as incertezas nas aplicações.

3.2. Escalonador PLITD-FUZZY

O escalonador PLITD-FUZZY é implementado através de modificações no escalonador PLITD-MODIF necessárias para permitir a criação de um escalonador com capacidade de manipular números fuzzy. Os números fuzzy representam as incertezas presentes na descrição das aplicações, ou seja, nos pesos dos arcos e das tarefas dos DAGs submetidos para execução nas grades. Dessa forma o escalonador passa a receber como entrada DAGs com pesos representados por intervalos e fornece escalonamentos que são gerados levando em consideração uma faixa de valores e não um único valor como ocorre com o escalonador PLITD-MODIF.

Além das várias informações de descrição do DAG e da grade que devem ser passadas como entrada para o escalonador, e que estão descritas nos dois primeiros parágrafos da Subseção 3.1, a formulação do escalonador PLITD-FUZZY precisa receber como entrada o menor tempo de execução possível da aplicação, T_{min}, que vale $\min(\{TI_{k|k \in V_H}\}) \times \sum_{i \in MC} I_i$. Esse intervalo de tempo corresponde ao tempo que a aplicação levaria para executar caso todas as tarefas do menor caminho do DAG executassem no *host* mais rápido (MC é o conjunto de todas as tarefas do menor caminho entre as tarefas 1 e n em termos de quantidade de instruções).

As incertezas para as quais o escalonador PLITD-FUZZY é projetado, referem-se aos pesos do DAG da aplicação a ser escalonada. Assim como modelado em [Carole et al. 2007], considera-se que os valores de I_i e $B_{i,j}$ são números fuzzy triangulares, para reproduzir situações em que o erro na estimativa do peso pode ser tanto para mais quanto para menos. Assim, uma tarefa i é descrita como tendo I_i instruções com

uma incerteza de $\sigma\%$ sobre esse valor. A quantidade de instruções da tarefa i portanto é representada, utilizando a notação de números fuzzy, por $[\underline{I_i}, I_i, \overline{I_i}]$ onde $\underline{I_i} = I_i(1-\frac{\sigma}{100})$ e $\overline{I_i} = I_i(1+\frac{\sigma}{100})$. De modo similar, as demandas de comunicação do DAG são representadas por $[\underline{B_{i,j}}, B_{i,j}, \overline{B_{i,j}}]$ com uma incerteza $\rho\%$. $\underline{B_{i,j}} = B_{i,j}(1-\frac{\rho}{100})$ e $\overline{B_{i,j}} = B_{i,j}(1+\frac{\rho}{100})$.

A formulação do programa inteiro resolvido pelo escalonador PLITD-FUZZY é descrita a seguir:

Maximize λ

sujeito a

$$1 - \frac{f_n - T'_{min}}{T'_{max} - T'_{min}} \geq \lambda \qquad \text{(F1)}$$

$$\sum_{t \in \mathcal{T}'} \sum_{k \in V_H} x_{j,t,k} = 1 \qquad \text{para } j \in V_D; \qquad \text{(F2)}$$

$$x_{j,t,k} = 0 \qquad \text{para } j \in V_D, \quad k \in V_H, \quad t \in \{1, \ldots, \lceil \underline{I_j} TI_k \rceil\}; \qquad \text{(F3)}$$

$$\sum_{k \in \delta(l)} \sum_{s=1}^{\lceil t - \overline{I_j} TI_l - \overline{B_{i,j}} TB_{k,l} \rceil} x_{i,s,k} \geq \sum_{s=1}^{t} x_{j,s,l} \qquad \text{para } j \in V_D, \quad ij \in A_D, \quad \text{para } l \in V_H, \quad t \in \mathcal{T}'; \qquad \text{(F4)}$$

$$\sum_{j \in V_D} \sum_{s=t}^{\lceil t + \underline{I_j} TI_k - 1 \rceil} x_{j,s,k} \leq 1 \qquad \text{para } k \in V_H, \quad t \in \mathcal{T}', \quad t \leq \lceil T'_{max} - \underline{I_j} TI_k \rceil; \qquad \text{(F5)}$$

$$x_{j,t,k} \in \{0, 1\} \qquad \text{para } j \in V_D, \quad l \in V_H, \quad t \in \mathcal{T}'. \qquad \text{(F6)}$$

Diferentemente do escalonador PLITD-MODIF, o objetivo do escalonador PLITD-FUZZY é a maximização do grau de satisfação λ ($\in [0, 1]$), que é inversamente proporcional ao tempo de execução da aplicação dado pelo escalonamento. O tempo de execução da aplicação é limitado pelos valores de $T'_{min} = T_{min}(1 - \frac{\sigma}{100})$ e $T'_{max} = T_{max}(1 + \frac{\sigma}{100})$. Os valores de T_{max} e T_{min} não podem ser utilizados diretamente por conta da incerteza presente nos pesos do DAG. A Figura 3 exibe o gráfico com a relação entre λ e o tempo de execução da aplicação f_n.

A fim de incluir a nova faixa de valores de f_n ($\in [T'_{min}, T'_{max}]$) e a relação com o grau de satisfação λ, a função do gráfico da Figura 3 está representada no escalonador PLITD-FUZZY pela restrição (F1). O intervalo de tempo no qual as tarefas podem executar passam a ser representadas pela variável $\mathcal{T}' = \{1, \ldots, T'_{max}\}$.

Como I_i e $B_{i,j}$ são números fuzzy, as restrições no escalonador PLITD-MODIF

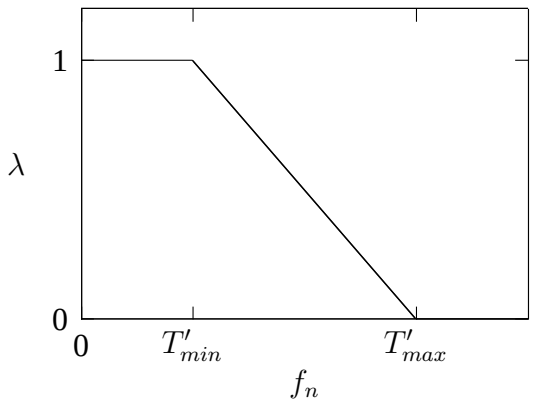


Figura 3. Grau de satisfação.

que utilizam ambos os valores ((D2), (D3) e (D4)) são modificadas de acordo com essa característica. A restrição (D2) é modificada e torna-se a restrição (F3). Essa restrição determina que uma tarefa (j) não pode terminar sua execução até que todas as suas instruções tenham sido completadas. Como não é possível saber o número exato de instruções, o valor I_j é substituído por $\underline{I_j}$, já que a quantidade mínima de instruções é o único valor corretamente conhecido.

A restrição (D3) é modificada e torna-se a restrição (F4). A restrição (F4) define que a tarefa j só pode começar sua execução depois que todas as tarefas predecessoras terminarem suas execuções e transferirem os dados requeridos pela tarefa j. Para evitar que a tarefa j comece sua execução antes dos dados de dependência terem sido transferidos, I_j e $B_{i,j}$ são substituídos pelos valores máximos dados, respectivamente, $\overline{I_j}$ e $\overline{B_{i,j}}$.

A última restrição modificada é a restrição (D4), que se torna a restrição (F5). A restrição (F5) estabelece que cada *host* só pode executar no máximo uma tarefa por vez. Como o tempo mínimo de execução de uma tarefa é conhecido, só é possível garantir que um *host* vai executar a tarefa durante esse tempo, motivo pelo qual o valor I_j é substituído por $\underline{I_j}$. T'_{max} é utilizado em substituição de T_{max} a fim de representar todo o possível intervalo de execução das tarefas.

4. Avaliação de desempenho

Para verificar a eficácia do escalonador PLITD-FUZZY, compara-se, através de simulações, o seu desempenho com o do escalonador PLITD-MODIF em ambientes com incertezas nos pesos das aplicações. Dessa forma, é possível avaliar o impacto das modificações diretamente relacionadas com o tratamento das incertezas, já que a diferença entre os escalonadores PLITD-FUZZY e o PLITD-MODIF diz respeito à presença dessas modificações.

Cada escalonador recebe como entrada dois grafos. Um representa o DAG da aplicação e outro representa os recursos da grade. Em todos os experimentos relatados neste artigo o mesmo DAG é utilizado como entrada para os escalonadores. O DAG é construído a partir do modelo da aplicação de astronomia Montage [Blythe et al. 2005], uma aplicação real que é executada em grades, com pesos definidos de forma aleatória seguindo uma distribuição uniforme com média de 72×10^6 bits para os pesos dos arcos

e com média de $31,5 \times 10^{12}$ instruções para os pesos das tarefas. A Figura 4 ilustra o DAG da aplicação utilizada nos experimentos. Apesar desse DAG ser passado como entrada para os escalonadores, os escalonamentos propostos são aplicados em versões modificadas desse DAG, que incluem graus específicos de incerteza. Em [NASA] há uma lista de várias aplicações e projetos baseados no DAG que foi utilizado nos experimentos.

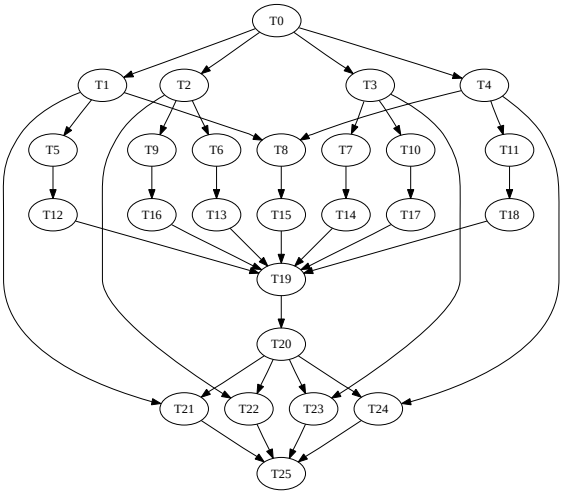


Figura 4. DAG utilizado nos experimentos.

É importante observar que apesar do escalonador ser destinado a aplicações formadas por tarefas dependentes, ele pode ser utilizado para escalonar aplicações formadas por tarefas independentes, mais conhecidas como *Bag of Tasks* (BoTs). No caso de BoTs, deve-se construir um DAG com tarefas virtuais de entrada e de saída e com arcos virtuais. Todos esses arcos e tarefas virtuais devem ter peso zero.

Além dos experimentos apresentados neste artigo com o DAG da Figura 4, vários outros DAGs foram utilizados em outros experimentos. Os resultados alcançados com o DAG da Figura 4 são os únicos apresentados neste artigo por serem representativos dentre todos os experimentos realizados e dado a limitação de espaço.

Vinte grades foram utilizadas nos experimentos. Cada uma delas foi criada de forma pseudo-aleatória pelo método Doar-Leslie [Doar and Leslie 1993], um método muito utilizado para gerar topologias de rede com características semelhantes àquelas da Internet. Com a utilização dessas topologias, é possível obter casos próximos à realidade, tornando os resultados encontrados mais representativos. Todas as grades utilizadas nos experimentos possuem 50 *hosts* e pesos médios de $\frac{9}{5{,}25 \times 10^{12}}$ minutos/instrução (9726MIPS) para os *hosts* e $\frac{2}{12 \times 10^{9}}$ minutos/bit (100Mbps) para os enlaces.

Os valores de incerteza considerados nos experimentos variaram no conjunto $\{40\%, 100\%, 200\%, 400\%\}$, o que equivale a valores de QoI de, respectivamente, $\{72, 43\%, 50\%, 33, 33\%, 20\%\}$ e segue as faixas de valores utilizadas nos experimentos apresentados em [Blythe et al. 2005] e [Sakellariou and Zhao 2004]. É importante observar que os valores utilizados nos experimentos foram escolhidos levando-se em consideração as informações encontradas atualmente na literatura.

Apesar do PLITD-FUZZY suportar o tratamento de incertezas tanto nas demandas de processamento quanto nas demandas de comunicação, devido a limitação de espaço,

os resultados apresentados neste artigo focam nas incertezas nos pesos dos arcos do DAG, isso equivale a valores de $\sigma = 0$ e $\rho \in \{40\%, 100\%, 200\%, 400\%\}$.

A fim de avaliar de forma correta a robustez dos escalonadores, o escalonamento proposto por eles foi aplicado em 20 versões modificadas dos DAGs originais. Cada uma das 20 versões modificadas dos DAGs foram geradas através do aumento nos pesos dos arcos. Os aumentos foram definidos de forma aleatória segundo uma distribuição uniforme dentro da faixa de 0 a $x\%$, com $x \in \{40, 100, 200, 400\}$. O valor de x utilizado nos experimentos é referenciado nas subseções a seguir como "incerteza ocorrida".

Todos os escalonadores foram implementados em `C` e a biblioteca de otimização `Xpress` versão `2006A.1` [Dash Optimization] foi utilizada. Todas as execuções dos escalonadores foram realizadas em um computador equipado com um processador Pentium IV, 3,2GHz e 2,5GB de memória RAM.

As subseções a seguir apresentam os resultados obtidos com os experimentos que avaliaram a qualidade do escalonador em termos do *speedup* gerado (*speedup*$= \frac{T_{max}}{makespan}$ e quanto maior o seu valor, melhor o escalonamento), utilização da rede na grade e tempo de execução em várias situações onde os pesos reais dos arcos do DAG foram na prática diferentes daqueles passados como entrada para os escalonadores. Os intervalos de confiança exibidos nos gráficos apresentados a seguir foram calculados considerando-se um nível de confiança de 95%.

4.1. Speedup

A Figura 5 exibe o gráfico que plota o *speedup* médio do escalonador PLITD-MODIF e do escalonador PLITD-FUZZY, projetado com diferentes valores de ρ, em função de diferentes valores de incerteza ocorrida (x) nos pesos dos arcos do DAG da Figura 4.

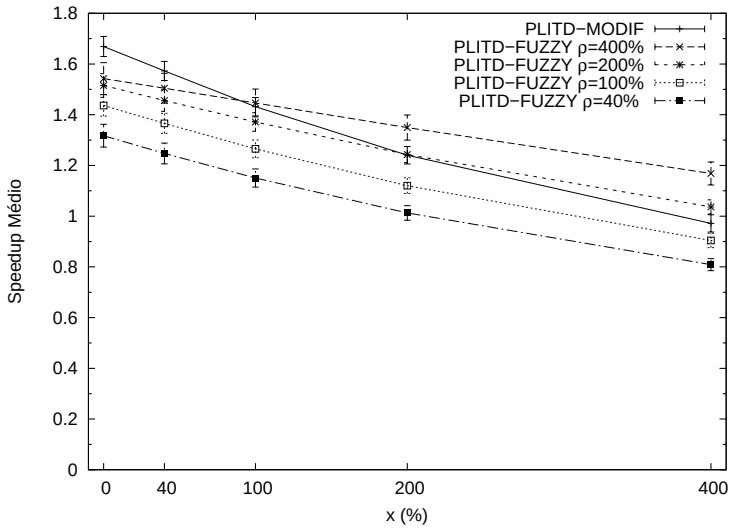


Figura 5. *Speedup* médio do PLITD-MODIF e do PLITD-FUZZY.

Pelo gráfico da Figura 5, nota-se que para qualquer valor de x, os escalonamentos propostos pelo escalonador PLITD-MODIF são melhores do que os escalonamentos do escalonador PLITD-FUZZY projetado com $\rho \leq 100\%$. O escalonador PLITD-MODIF também produz *speedups* melhores do que o escalonador PLITD-FUZZY projetado com

$\rho = 200\%$ mas somente quando $x \leq 40\%$. Entretanto, quando o escalonador PLITD-FUZZY é projetado esperando um alto valor de incerteza ($\rho = 400\%$), ele só produz *speedups* piores do que o escalonador PLITD-MODIF para o caso em que o DAG é corretamente descrito ($x = 0\%$).

Pode-se notar, também, na Figura 5, que o *speedup* do escalonador PLITD-MODIF decresce duas vezes mais rápido do que o *speedup* do escalonador PLITD-FUZZY com $\rho = 400\%$, em função do aumento de x. O *speedup* médio produzido pelo escalonador PLITD-FUZZY com $\rho = 400\%$ é $\cong 20,27\%$ maior do que aquele produzido pelo escalonador PLITD-MODIF quando $x = 400\%$. Além de ser pior do que o escalonador PLITD-FUZZY neste caso, o escalonamento proposto pelo escalonador PLITD-MODIF é pior, até mesmo, do que o caso em que todas as tarefas do DAG são executadas de forma serial no *host* mais rápido, dado que o *speedup* médio obtido foi < 1.

A partir desses resultados, conclui-se que o escalonador PLITD-FUZZY, em termos de *speedup*, comporta-se melhor quando projetado com a expectativa de altos valores de incerteza, mesmo quando as incertezas reais ocorridas na prática não ocorram dentro da faixa de valores esperados. Isso pode ser explicado pela falta de precisão introduzida pelas técnicas de otimização fuzzy quando sua flexibilidade está limitada, o que ocorre quando ρ é baixo. Entretanto, quando a flexibilidade decorrente da incerteza aumenta, por conta de valores maiores para ρ, a habilidade que as técnicas fuzzy têm de lidar com as incertezas destaca-se, levando a resultados melhores do que quando as técnicas não são utilizadas.

Pela tendência dos gráficos exibidos na Figura 5, em situações onde os valores de ρ fossem maiores do que os utilizados nos experimentos, haveria uma diminuição do paralelismo e aumento da concentração das tarefas em um número limitado de hosts. Em tais cenários, o *speedup* alcançado tenderia a 1. A Subseção 4.2 fornece comentários a cerca da utilização da rede em função da incerteza na quantidade de dados transferidos entre as tarefas.

4.2. Utilização da rede

Como os experimentos realizados aplicam a incerteza somente nos arcos do DAG, é de se esperar que os ganhos obtidos pelo escalonador PLITD-FUZZY com relação ao escalonador PLITD-MODIF sejam decorrentes da melhor utilização da rede. O escalonador PLITD-MODIF, por não implementar nenhum tratamento de incertezas, propõe escalonamentos que utilizam os enlaces de rede na expectativa de que os pesos passados como entrada sejam os pesos gerados pela aplicação na prática. Caso os pesos mostrem-se menores do que os valores reais, o intervalo de tempo necessário para transferir os bits de dependência torna-se maior do que aquele previsto, o que aumenta as chances de aumento no tempo de execução da aplicação.

Por outro lado, um escalonador projetado para tratar diferentes valores de incerteza, como o escalonador PLITD-FUZZY, propõe escalonamentos mais robustos através da diminuição do paralelismo das tarefas. Tarefas dependentes mas interligadas por arcos com uma alta incerteza possuem uma probabilidade alta de serem executadas em um único *host*, ao invés de serem executadas em *hosts* diferentes. Dessa forma, a transferência dos dados de dependência pelos enlaces da grade deixa de existir, diminuindo as chances de aumento no tempo de execução da aplicação.

A Figura 6 exibe um gráfico que permite a análise da utilização dos recursos de comunicação da grade em função dos escalonadores executados. Comprovam-se as expectativas em torno da maior utilização da rede por parte do escalonador PLITD-MODIF.

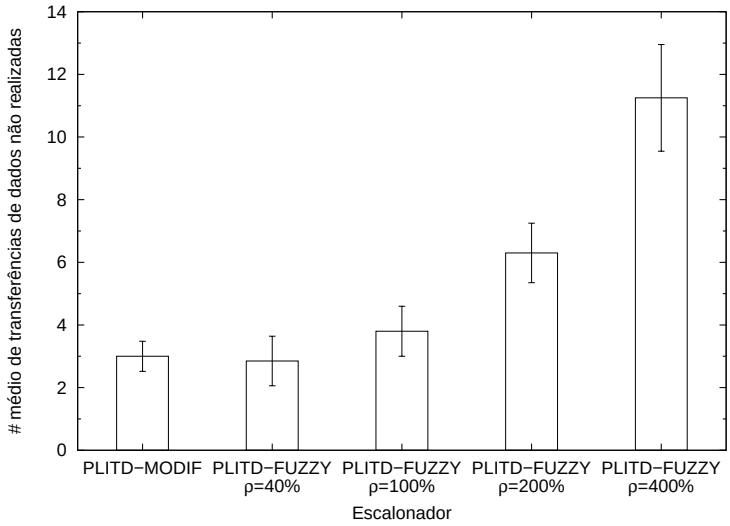


Figura 6. Número médio de dependências de dados que não utilizaram a rede.

O DAG utilizado nos experimentos possui 39 arcos, o que significa 39 potenciais dependências de dados que utilizarão os enlaces entre os recursos de processamento das grades simuladas. O gráfico da Figura 6 plota a quantidade média dessas dependências de dados que **não** foram transferidas via rede para cada um dos escalonadores utilizados nos experimentos. É possível notar que dentre todos os escalonadores, o escalonador PLITD-MODIF é o escalonador que mais utiliza os recursos de comunicação, visto que em média somente 3 dependências de dados do DAG não são transferidas via rede. Quando o escalonador PLITD-FUZZY foi utilizado, a quantidade de dependências de dados não transferidas via rede mostrou-se diretamente proporcional à incerteza esperada, ρ. O escalonador que apresentou os melhores valores de *speedup* sob altas incertezas, o PLITD-FUZZY com $\rho = 400\%$, foi o escalonador que fez menos uso da rede. Em média, cerca de 11 dependências de dados não foram transferidas via rede, um aumento de cerca de 266.67% em relação aos números apresentados pelo escalonador PLITD-MODIF.

Os resultados comprovam a expectativa de que escalonadores que tratam a incerteza propõe uma menor utilização da rede do que aqueles escalonadores que não tratam. Esse comportamento é apresentado a fim de contornar possíveis aumentos no tempo de execução, decorrentes de aumentos nos tempos de transferência dos dados de dependência.

4.3. Tempo de execução

Independente do objetivo a ser alcançado com o escalonamento de aplicações em grades, o tempo de execução do escalonador sempre desempenha um papel importante. A dinâmica da grade faz com que escalonamentos propostos em um instante de tempo t tornem-se sub-ótimos, caso ocorram mudanças significativas no estado dos recursos dentro de um intervalo de tempo Δt.

A Figura 7 exibe o gráfico que plota o tempo de execução médio de todos os escalonadores utilizados nos experimentos.

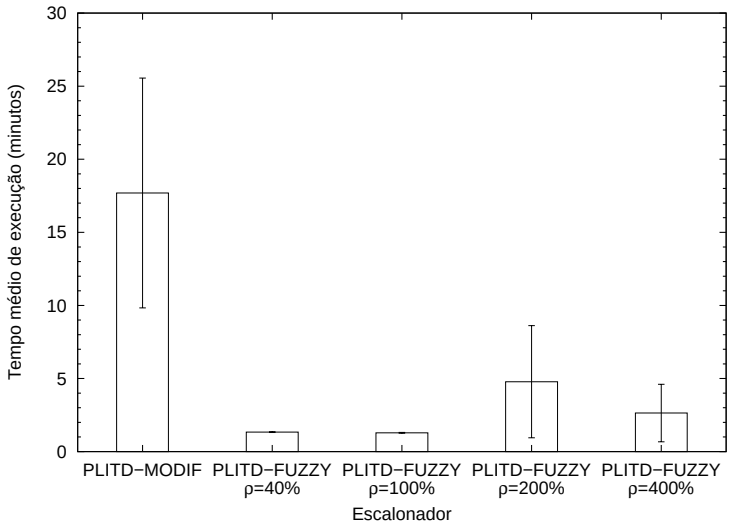


Figura 7. Tempo de execução médio dos escalonadores.

Pelo gráfico da Figura 7, observa-se que o escalonador PLITD-FUZZY apresenta tempos de execução médios menores do que aqueles apresentados pelo escalonador PLITD-MODIF. Por exemplo, o escalonador PLITD-FUZZY com $\rho = 400\%$ executa, em média, durante um intervalo de tempo $\cong 85,08\%$ menor do que o escalonador PLITD-MODIF.

Por esses resultados, nota-se que o escalonador PLITD-FUZZY, além de fornecer bons resultados quando a expectativa de incerteza é alta e além de fazer menos uso dos recursos de comunicação do que o escalonador PLITD-MODIF, requer menos tempo de processamento do que o escalonador PLITD-MODIF. Essa característica do PLITD-FUZZY torna sua implementação mais adequada para grades que apresentem altas taxas de variação de disponibilidade de recursos.

5. Trabalhos relacionados

O impacto negativo da ausência de tratamento de incerteza por parte dos escalonadores pode ser observado em [Blythe et al. 2005]. O objetivo do trabalho é propor e comparar duas abordagens diferentes de escalonamento de aplicações formadas por tarefas dependentes. A primeira abordagem escalona as tarefas sem ter noção do DAG como um todo, ou seja, considera apenas os pesos computacionais. A segunda abordagem leva em consideração também os pesos de comunicação entre tarefas dependentes. Assim como neste artigo, os experimentos realizados utilizam um DAG derivado da aplicação de astronomia Montage. Diferente de outros trabalhos que propõem escalonadores de tarefas para grades, em [Blythe et al. 2005] há uma análise do impacto causado pela incerteza nos pesos das tarefas, e nos pesos das dependências de dados, sobre o *makespan* das aplicações. Mostra-se que o *makespan* das aplicações cresce com o aumento da incerteza (em certos casos chega a aumentar 400%). A faixa de valores utilizada na análise é tomada como referência na definição dos valores utilizados nos experimentos que avaliam o desempenho do escalonador PLITD-FUZZY neste artigo. Apesar de reconhecer o impacto negativo

da incerteza, em [Blythe et al. 2005] não são propostas modificações nos escalonadores a fim de diminuí-lo.

Em [da Silva and Scherson 2000] propõem-se um escalonador que tem por objetivo utilizar valores reais de tempo de execução das tarefas para aumentar o desempenho de aplicações *I/O bound* e de aplicações interativas. Assim como no escalonador PLITD-FUZZY, há a consideração das demandas de comunicação entre as tarefas de uma aplicação no escalonamento, apesar da proposta ser voltada para *clusters*. A proposta é baseada no monitoramento das tarefas durante as suas execuções a fim de agrupá-las em diferentes classes: *I/O bound*, *communication intensive* ou *computation intensive*. Cada classe é representada como um conjunto fuzzy. O grau de pertinência das tarefas é computado utilizando medições realizadas nos *hosts* que depois são passadas para um estimador bayesiano. Caso seja detectada mudança de classe de uma tarefa, o escalonador busca preencher os intervalos de tempo vazios que surgem (por conta de espera por algum dado de E/S ou por conta de comunicação) com novas tarefas que possam ser executadas durante o tempo de espera. A análise de desempenho da proposta é realizada em um simulador que modela uma topologia de rede local com 8 e 16 máquinas. Os resultados apresentados mostram ganhos a nível de melhor utilização das máquinas bem como aumento na vazão do sistema. As principais diferenças para a proposta apresentada neste artigo são o fato de serem tomadas ações reativas caso sejam detectadas mudanças nos pesos esperados das tarefas e o fato das simulações considerarem ambientes homogêneos, com poucos *hosts* e com uma topologia de rede que é irreal pra grades. O escalonador PLITD-FUZZY toma ações preventivas e os experimentos de simulação empregados constituíram um ambiente heterogêneo, uma das principais características das grades.

O estudo apresentado em [Cirne 2001] tem o objetivo de criar um modelo que descreva a carga de trabalho imposta por aplicações paralelas a supercomputadores. Dentre as várias métricas coletadas em relatórios de 4 supercomputadores, o tempo de execução real e o tempo de execução requisitado pelo usuário para cada tarefa são avaliados em termos de correlação. Várias distribuições de probabilidade são derivadas a fim de permitir a criação de cargas de trabalho sintéticas para simulação em termos de instante de chegada de tarefas, cancelamento de tarefas, tempo de execução real das tarefas e tempo de execução previsto pelo usuário. A principal diferença para o trabalho apresentado neste artigo vem do fato de que em [Cirne 2001] as tarefas nunca excedem o tempo de execução previsto pelos usuários. Caso as tarefas não finalizem até o tempo previsto, o ambiente de execução força as suas finalizações. Adotar esse comportamento em grades traria diversos problemas visto que, pelo fato dos recursos não serem dedicados e nem homogêneos, torna-se impraticável prever o tempo de execução corretamente para todas as possibilidades de escalonamento. Assim como em [da Silva and Scherson 2000], a proposta diferencia-se da apresentada neste artigo por analisar o impacto das incertezas no tempo de execução para ambientes paralelos formados por recursos homogêneos. Outra diferença está no fato de serem consideradas tarefas independentes. Por outro lado, o trabalho é complementar ao nosso no sentido de avaliar os requisitos de aplicações executadas em ambientes paralelos reais.

Em [Bölöni and Marinescu 2002] é apresentada uma proposta para tratar a incerteza baseada na classificação de tarefas críticas em termos de robustez. Tarefas mais críticas são aquelas cujo atraso na execução influenciam diretamente no tempo de

execução total da aplicação. Uma vez detectadas as tarefas mais críticas, elas são alocadas para *hosts* que são separados para executá-las exclusivamente. Uma diferença da abordagem adotada em [Bölöni and Marinescu 2002] e o escalonador PLITD-FUZZY é o fato daquele não considerar o impacto causado pelas incertezas nos requisitos de comunicação das aplicações. Assim como avaliado nos experimentos deste artigo, em [Bölöni and Marinescu 2002] os ganhos alcançados com incertezas baixas ($\leq 40\%$) não foram expressivos. Diferente do realizado neste artigo, os autores não conduziram experimentos com incertezas maiores.

Em [Real et al. 2003], apresenta-se o módulo responsável pela descrição da capacidade dos recursos disponíveis em uma grade gerenciada pelo *middleware* ISAM (Infraestrutura de Suporte às Aplicações Móveis). O módulo tem como objetivo devolver uma descrição da grade que seja o mais próxima possível do real. Valores capturados por sensores na grade são passados como entrada para um algoritmo baseado em uma rede bayesiana que (i) auto-ajusta os seus valores com relação a mudanças dinâmicas no ambiente; (ii) usa algoritmos de aprendizagem para predizer o estado dos recursos; e (iii) considera os estados no passado para prever o estado atual. Com a descrição mais próxima do real, o módulo resolve o problema de incerteza no estado dos recursos porém não trata a incerteza na descrição das aplicações, ponto que é tratado neste artigo. Assim como na avaliação de desempenho realizada neste artigo, em [Real et al. 2003] há comentários a respeito do tempo de execução dos escalonadores. Diferente do proposto neste artigo, não há menção sobre tratamento de incertezas relacionadas com o tempo de transmissão de dados entre tarefas dependentes das aplicações. Os experimentos realizados mostram ganho de 21,3% com relação a um escalonamento que utiliza a descrição do NWS (*Network Weather Service*), um serviço de monitoramento e previsão muito utilizado em grades, porém só é avaliado o impacto das incertezas na capacidade de processamento disponível de uma grade homogênea formada por 20 *hosts*.

Em [Sakellariou and Zhao 2004], apresenta-se uma proposta para tratar incertezas dinamicamente. Se uma tarefa extrapola o seu tempo de execução esperado, ela pode ser passível de migração. O que define se a tarefa será ou não passível são dois valores (*slack* e *spare time*) que definem até quando uma tarefa pode aumentar seu tempo de execução sem que o *makespan* da aplicação seja modificado. Os experimentos realizados mostram que os algoritmos utilizados no reescalonamento exercem papel fundamental no tratamento das incertezas quando a QoI varia de 10 a 50% em uma grade de apenas 3 a 8 *hosts*. Entretanto, apesar da formulação considerar o impacto na transmissão dos dados, ela não considera o efeito da QoI no caso de imprecisões no tempo de transferência dos dados da aplicação, diferente do realizado neste artigo. Uma diferença importante para o trabalho apresentado aqui é o fato dos experimentos realizados em [Sakellariou and Zhao 2004] não considerarem incertezas diferentes daquelas esperadas.

Uma análise de carga de trabalho semelhante ao [Cirne 2001], mas voltado para grades, é apresentado em [Oikonomakos et al. 2007]. Um *cluster* que faz parte da grade EGEE (*Enabling Grids for E-sciencE*) teve seus relatórios analisados a fim de se buscar modelos estatísticos que descrevessem o tempo entre chegadas de tarefas bem como o tempo de execução de cada tarefa. Chegou-se à conclusão que o primeiro exibe dependência de longa duração enquanto o segundo é melhor modelado por um processo hiperexponencial com 3 estados. A semelhança com o trabalho apresentado neste artigo

está no fato de, no *cluster* analisado, os usuários fornecerem uma previsão do tempo de execução de suas tarefas. Entretanto, na análise realizada, tais valores não são utilizados na criação dos modelos. Nas aplicações analisadas os recursos mais utilizados são os recursos de processamento. Poucos bits são transferidos entre as tarefas (a grande maioria das dependências de dados é da ordem de KB) e não há a proposta de modelos estatísticos que descrevam a utilização da rede por parte das aplicações. Um problema da análise realizada é o fato dela se concentrar em um único *cluster* de uma grade, o que exige mais experimentos a fim de se tirar conclusões genéricas a respeito da carga de trabalho para outras grades, bem como para aplicações *data-intensive*.

Em [Carole et al. 2007], o tratamento da incerteza nas tarefas de uma aplicação é realizado independente da origem da incerteza. Caso uma tarefa leve mais tempo do que o esperado para executar, a proposta não se preocupa se a causa foi um erro do usuário na hora de descrever os requisitos ou se foi alguma variação no estado dos recursos alocados. Uma diferença do trabalho em [Carole et al. 2007] para o apresentado neste artigo é o fato dele considerar falhas nas tarefas, o que pode levar as mesmas a não executarem. Outra diferença é o fato da proposta em [Carole et al. 2007] ser voltada para grades que negociam SLAs (*Service Level Agreements*) com as aplicações. Após avaliar as incertezas de uma aplicação, a proposta devolve como saída a decisão sobre aceitar ou não a aplicação na grade. Assim como assumido no escalonador PLITD-FUZZY, os requisitos das aplicações em [Carole et al. 2007] são modelados como números fuzzy triangulares.

6. Conclusões e Trabalhos Futuros

Aplicações descritas de forma incorreta por usuários de grades podem levar a perdas de desempenho imprevisíveis caso não seja implementado nenhum mecanismo que tolere graus de incerteza. Neste artigo, apresentou-se o escalonador PLITD-FUZZY, um escalonador que utiliza técnicas de otimização fuzzy com o objetivo de fornecer escalonamentos robustos frente às incertezas nos pesos das tarefas de DAGs, tanto em termos de processamento quanto em termos de comunicação.

Experimentos de simulação realizados com o DAG de uma aplicação executada em grades e várias grades geradas com características semelhantes àquelas da Internet, comprovaram a eficácia do escalonador em situações onde há uma expectativa alta de incerteza na descrição dos requisitos de comunicação. O escalonador PLITD-FUZZY mostrou-se robusto em termos de *speedup* produzido, tempo de processamento para fornecer o escalonamento e utilização dos recursos de comunicação das grades. Os resultados alcançados pelo escalonador PLITD-FUZZY foram comparados com os resultados de um escalonador que não implementa as técnicas de otimização fuzzy empregadas no primeiro. Pelas comparações, o escalonador PLITD-FUZZY produziu escalonamentos com *speedups* até 20,27% maiores, enquanto o tempo de execução foi até 85,08% menor, do que os equivalentes produzidos pelo escalonador sem as técnicas de otimização. Dessa forma, pode-se afirmar que os ganhos alcançados foram decorrentes das técnicas empregadas.

Como trabalhos futuros, pretende-se: (i) estudar os ganhos do escalonador PLITD-FUZZY em ambientes onde haja ao mesmo tempo incertezas nos requisitos de comunicação e nos requisitos de computação; (ii) investigar o *trade-off* entre soluções fornecidas pelo escalonador PLITD-FUZZY e soluções fornecidas por esquemas de auto-

adaptação baseados em reescalonamento e migração; (iii) comparar os ganhos do escalonador PLITD-FUZZY com escalonadores que não considerem a rede através da transformação dos arcos dos DAGs em tarefas virtuais; e (iv) avaliar o escalonador PLITD-FUZZY em situações onde os pesos fornecidos pelo usuário sejam maiores do que os pesos reais das tarefas.

Referências

Batista, D. M., da Fonseca, N. L. S., Granelli, F., and Kliazovich, D. (2007). Uma Metodologia para o Auto-Ajuste de Aplicações em Grades. In *Anais do XXVII Congresso da Sociedade Brasileira de Computação – VI WPerformance*, pages 475–494.

Batista, D. M., da Fonseca, N. L. S., and Miyazawa, F. K. (2006). Escalonadores de Tarefas em Grades. In *Anais do XXVI Congresso da Sociedade Brasileira de Computação – V Wperformance*, pages 73–92.

Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., and Kennedy, K. (2005). Task Scheduling Strategies for Workflow-based Applications in Grids. In *IEEE International Symposium on Cluster Computing and Grids (CCGRID'05)*, volume 2, pages 759–767.

Bölöni, L. and Marinescu, D. C. (2002). Robust Scheduling of Metaprograms. *Journal of Scheduling*, 5:395–412.

Carole, F., Garibaldi, J. M., and Ouelhadj, D. (2007). Fuzzy Grid Scheduling Using Tabu Search. In *Proceedings of IEEE International Fuzzy Systems Conference*, pages 1–6.

Casanova, H., Zagorodnov, D., Berman, F., and Legrand, A. (2000). Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. In *HCW '00: Proceedings of the 9th Heterogeneous Computing Workshop*, page 349, Washington, DC, USA. IEEE Computer Society.

Chiang, S.-H., Arpaci-Dusseau, A., and Vernon, M. K. (2002). The Impact of More Accurate Requested Runtimes on Production Job Scheduling Performance. In *8th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2002)*, pages 103–127. Springer-Verlag Berlin Heidelberg.

Chiang, S.-H., Mansharamani, R. K., and Vernon, M. K. (1994). Use of Application Characteristics and Limited Preemption for Run-to-Completion Parallel Processor Scheduling Policies. In *SIGMETRICS '94: Proceedings of the 1994 ACM SIGMETRICS Conference on measurement and Modeling of Computer Systems*, pages 33–44, New York, NY, USA. ACM Press.

Cirne, W.; Berman, F. (2001). A Comprehensive Model of the Supercomputer Workload. In *IEEE International Workshop on Workload Characterization (WWC-4)*, pages 140–148.

da Silva, F. A. B. and Scherson, I. D. (2000). Improving Parallel Job Scheduling Using Runtime Measurements. In *IPDPS '00/JSSPP '00: Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*, pages 18–38, London, UK. Springer-Verlag.

Dash Optimization. The Xpress-Optimizer. Disponível em `http://www.dashoptimization.com/home/products/products_optimizer.html`. Último acesso em 1 fev 2008.

Doar, M. and Leslie, I. M. (1993). How Bad is Naive Multicast Routing? In *Proc. IEEE INFOCOM'93*, pages 82–89.

Foster, I. (2002). What is the Grid? A Three Point Checklist. *GRIDToday*, 1(6).

Foster, I. and Kesselman, C. (1999). Computational Grids. In *The Grid: Blueprint for a New Computing Infrastructure*, chapter 2. Morgan-Kaufman.

Lee, C. B., Schwartzman, Y., Hardy, J., and Snavely, A. (2004). Are User Runtime Estimates Inherently Inaccurate? In *10th International Workshop on Job Scheduling Strategies for Parallel Processing (JSSPP 2004)* , pages 253–263. Springer-Verlag Berlin Heidelberg.

NASA. Applications of Montage. Disponível em `http://montage.ipac.caltech.edu/applications.html`. Último acesso em 8 mai 2008.

Oikonomakos, M., Christodoulopoulos, K., and Varvarigos, E. M. (2007). Profiling Computation Jobs in Grid Systems. In *Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2007)*, pages 197–204.

Papadimitriou, C. H. and Steiglitz, K. (1998). *Combinatorial Optimization – Algorithms and Complexity*, pages 363–366. Dover Publications.

Prodan, R. and Fahringer, T. (2005). Dynamic Scheduling of Scientific Workflow Application on the Grid: a Case Study. In *SAC'05: Proceedings of the 2005 ACM symposium on Applied computing*, pages 687–694, New York, NY, USA. ACM Press.

Real, R., Yamin, A., da Silva, L., Frainer, G., Augustin, I., Barbosa, J., and Geyer, C. (2003). Resource Scheduling on Grid: Handling Uncertainty. In *Proceedings of the Fourth International Workshop on Grid Computing*, pages 205–207.

Sakellariou, R. and Zhao, H. (2004). A Low-cost Rescheduling Policy for Efficient Mapping of Workflows on Grid Systems. *Scientific Programming*, 12:253 – 262.

Skillicorn, D. B. (2002). Motivating Computational Grids. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'02)*, pages 401–406.

Sun, X. and Wu, M. (2005). GHS: A Performance System of Grid Computing. In *19th IEEE International Parallel and Distributed Processing Symposium*. `http://doi.ieeecomputersociety.org/10.1109/IPDPS.2005.234`. Accessed at 21/05/2006.

Vadhiyar, S. S. and Dongarra, J. J. (2003). A Performance Oriented Migration Framework for the Grid. In *3rd IEEE/ACM International Symposium on Cluster Computing and the Grid(CCGRID'03)*, pages 130–137.

Zhao, H. and Sakellariou, R. (2006). Scheduling Multiple DAGs onto Heterogeneous Systems. In *Proceedings of the 20h International Parallel and Distributed Processing Symposium (IPDPS 2006)*, pages 130–143.