Escalonamento de Workflows em Condições de Incerteza

Luiz F. Bittencourt e Edmundo R. M. Madeira

[1]Instituto de Computação - Universidade Estadual de Campinas (UNICAMP)
Av. Albert Einstein, 1251 - Cidade Universitária - Campinas - São Paulo

{bit,edmundo}@ic.unicamp.br

Abstract. *Grids and clouds are utilized for the execution of applications composed of dependent tasks, usually modeled as workflows. To efficiently run the application, a scheduler must distribute the components of the workflow in the available resources using information about processing capacities and bandwidth from the target system, as well as information about duration of tasks and communication between tasks in the workflow. However, such information may be subject to imprecisions, thus not reflecting what is observed during the execution. In this paper we evaluate a scheduling heuristic in the presence of uncertainties in the input data. Results shown that the schedule is negatively affected by the non-foreseen variation in resources capacities during the execution of the application.*

Resumo. *Grades e nuvens têm sido utilizadas na execução de aplicações compostas por tarefas dependentes, geralmente modeladas como workflows. Para que a execução seja eficiente, um escalonador deve distribuir as partes do workflow nos recursos disponíveis, utilizando informações prévias sobre as capacidades de processamento e largura de banda do sistema-alvo, assim como informações sobre a duração e comunicação das tarefas componentes do workflow. No entanto, tais informações podem ser imprecisas, não refletindo o que de fato é encontrado durante a execução. Neste artigo avaliamos uma heurística de escalonamento na presença de incertezas nos dados de entrada. Resultados mostram que o escalonamento é afetado negativamente pela mudança não prevista nas capacidades dos recursos durante a execução da aplicação.*

1. Introdução

Em grades, o escalonamento deve considerar características como heterogeneidade e variação de desempenho. Enquanto a heterogeneidade traz diferentes capacidades de processamento e de transmissão de dados para uma mesma tarefa em diferentes recursos, o compartilhamento dos recursos computacionais traz variações de desempenho durante a execução das tarefas. Essa característica de incerteza durante a execução das aplicações reflete-se em imprecisão nos dados de entrada do escalonador. A variabilidade da informação de entrada do escalonador está atrelada ao conceito de *qualidade da informação* [Casanova et al. 2000].

Já nas nuvens computacionais, a virtualização de recursos de processamento teoricamente contorna o problema do compartilhamento, provendo isolamento entre usuários. Entretanto, enlaces de comunicação internos e externos continuam sendo compartilhados

sem o isolamento necessário, trazendo incertezas na comunicação entre tarefas em recursos computacionais distintos. Como tais canais podem interligar tanto recursos de processamento dentro de uma mesma nuvem quanto fora dela, estes podem atravessar enlaces da internet cujo desempenho é difícil de estimar. Além disso, concorrência de barramento e de acesso a disco em recursos computacionais virtualizados também podem interferir no tempo de processamento das aplicações dos usuários [Senna et al. 2010]. Portanto, incertezas em estimativas de tempo também estão presentes em nuvens computacionais, interferindo na qualidade da informação.

Além de variações de desempenho dos recursos de processamento e rede, estimativas incorretas em relação às aplicações são outro ponto de incerteza que contribui para variabilidade de estimativas em relação ao tempo real de execução, afetando a qualidade da informação. Tais estimativas podem ter origem no modelo de programação, onde o programador da aplicação as fornece, ou em execuções passadas da mesma aplicação. Em ambos os casos o escalonador realizará o escalonamento utilizando informações de duração de tarefas e transmissões de dados que não condizem com a realidade. Como conseqüência, as estimativas do escalonador não se concretizam, distorcendo o tempo de duração estimado da aplicação.

Estimativas incorretas de tempo de transmissão de dados são especialmente importantes quando da execução de tarefas dependentes. Atualmente, uma variada gama de aplicações científicas modeladas como *workflows* tem sido executada em grades computacionais e nuvens. Assim, tempos estimados de execução de tarefas e de transmissão de dados tornam-se particularmente importantes nesse tipo de aplicação.

Neste artigo realizamos um estudo preliminar de uma heurística de escalonamento de *workflows* em sistemas heterogêneos, chamada *Heterogeneous Earliest Finisht Time* (HEFT) [Topcuoglu et al. 2002], sob condições de incerteza. Heurísticas dessa natureza podem ser utilizadas em grades e nuvens, onde a qualidade da informação é potencialmente afetada por estimativas incorretas de tempos de execução e transmissão de dados. Os resultados obtidos mostram que a qualidade da informação é importante no escalonamento, o que traz à tona a relevância do desenvolvimento de algoritmos que sejam cientes de incerteza. Dessa forma, a contribuição deste artigo está em: (i) trazer a discussão da incerteza à tona, mostrando que em ambientes atuais como grades e nuvens ela existe; e (ii) mostrar que a incerteza da informação é relevante no resultado obtido para o escalonamento, e que, portanto, estudos com esse foco ainda podem contribuir para a melhoria dos algoritmos.

Este artigo está organizado da seguinte maneira. Na Seção 2 apresentamos trabalhos que estudam o escalonamento de tarefas dependentes em sistemas distribuídos heterogêneos. Em seguida, na Seção 3 apresentamos características de nuvens e grades e definimos o problema de escalonamento de grafos acíclicos direcionados, indicando potenciais problemas ocasionados por incerteza nos dados de entrada. O algoritmo HEFT, utilizado nas simulações, é brevemente descrito na Seção 4, elucidando potenciais problemas advindos de estimativas incorretas feitas pelo escalonador. Na Seção 5 apresentamos os resultados das simulações e uma discussão sobre como o problema pode ser abordado. Conclusão e trabalhos futuros são apresentados na Seção 6.

2. Trabalhos Relacionados

O estudo e aprimoramento constante de algoritmos e heurísticas é fundamental para melhorar o desempenho na execução de *workflows* em nuvens e grades [Taylor et al. 2007]. Sendo o escalonamento de tarefas um problema NP-Completo, exceto em alguns casos simples [Pinedo 2008], a maior parte dos esforços em algoritmos de escalonamento estão direcionados às heurísticas e meta-heurísticas [Grajcar 1999]. Há alguns trabalhos que utilizam algoritmos de aproximação [Phillips et al. 1997] e outros que utilizam abordagens de programação linear inteira [Genez et al. 2012].

Mais especificamente, a área de escalonamento de tarefas em sistemas distribuídos vem sendo bastante explorada. Tanto sistemas distribuídos homogêneos [Hakem and Butelle 2005] quanto heterogêneos [Yu and Buyya 2006, Sakellariou and Zhao 2004a] possuem vasta bibliografia no escalonamento de tarefas dependentes. No escalonamento de tarefas em sistemas distribuídos, o objetivo a ser otimizado mais encontrado na literatura é a minimização do *makespan*, ou seja, a busca por um escalonamento que resulte em um tempo de execução mínimo para as tarefas. Descrições extensas sobre escalonamento em grades computacionais são feitas em [Dong and Akl 2006] e [J.Yu et al. 2008].

Batista e Fonseca [Batista and da Fonseca 2011] apresentam um algoritmo baseado em lógica fuzzy para o escalonamento de *workflows* com incertezas em grades computacionais. Os autores comparam o algoritmo proposto com algoritmos estáticos para sistemas heterogêneos, incluindo o HEFT. Entretanto, apenas uma avaliação superficial da incerteza na largura de banda com um DAG de 4 nós é apresentada para os algoritmos estáticos, impossibilitando conclusões sobre a diferença de comportamento quando há e quando não há incerteza. Neste trabalho apresentamos uma avaliação com DAGs de aplicações reais com incertezas tanto em canais de comunicação quanto em capacidade de processamento.

Outras abordagens para tratamento de incertezas, como escalonamento dinâmico [Allen et al. 2001], escalonamento adaptativo [Bittencourt and Madeira 2008], re-escalonamento [Sakellariou and Zhao 2004b], e auto-ajuste [Batista et al. 2008], são encontradas na literatura. Enquanto tais abordagens são reativas, o que discutimos no presente artigo é o desenvolvimento de uma abordagem que visa prevenir que tais flutuações afetem a execução de modo que ações reativas não sejam necessárias. Adicionalmente, um monitoramento mais intenso é necessário para ações reativas, o que pode aumentar o grau de incerteza por causa de efeitos de intrusão, assim como a migração desnecessária de tarefas pode introduzir *overhead* e prejudicar a execução da aplicação [Batista and da Fonseca 2011].

3. Conceitos Básicos e Definições

Nesta seção apresentamos uma visão geral do problema de escalonamento de *workflows* e dos sistemas de grade e nuvem, associando suas características com a relevância da qualidade da informação. É importante notar que utilizamos a terminologia *tarefa* para designar um nó do *workflow* a ser executado. Essa terminologia deriva da palavra em inglês *job*, que é usada para designar um nó de um *workflow*, seja este uma chamada a um serviço ou uma tarefa auto-contida. Com isso visamos diferenciar o termo *serviço* no

sentido de *serviço computacional oferecido pela nuvem* do termo *serviço* relacionado a um serviço que compõe um *workflow*.

3.1. Grades e Nuvens

Enquanto grades computacionais são sistemas colaborativos compartilhados, nuvens tem como objetivo oferecer computação como serviço isolando os recursos computacionais através da virtualização [Armbrust et al. 2010]. A grade, como um grande sistema compartilhado, permite que processadores e enlaces de rede sejam sobrecarregados por aplicações de usuários diferentes que executam concorrentemente no sistema. Por outro lado, uma nuvem computacional procura isolar cada usuário, fornecendo um ambiente de execução teoricamente independente de outras aplicações executando no mesmo sistema.

Nuvens computacionais abstraem detalhes dos usuários, os quais não mais necessitam de conhecimento sobre a tecnologia da infraestrutura que suporta a nuvem. Tipicamente, nuvens envolvem o fornecimento de recursos dinamicamente escaláveis e freqüentemente virtualizados sobre a Internet em um modelo de pagamento pelo uso. Nuvens computacionais são geralmente classificadas em três diferentes tipos: *software as a service* (SaaS), *platform as a service* (PaaS), e *infrastructure as a service* (IaaS). O modelo de infraestrutura como serviço é mais genérico, onde a nuvem disponibiliza ao consumidor recursos computacionais e privilégios administrativos sobre eles. De maneira simplificada, podemos entender o modelo IaaS como um conjunto de servidores virtuais acessíveis através da Internet, o que vem sendo um grande foco de pesquisas em gerência e otimização [Zhang et al. 2010]. Uma nuvem IaaS pode ser classificada em três tipos: (i) pública, que oferece recursos virtualizados no modelo de pagamento pelo uso; (ii) privada, ou nuvem particular, que pode ser vista como uma grade computacional virtualizada; e (iii) híbrida, que combina nuvens privadas e públicas.

O escalonamento em nuvens híbridas pode trazer ao usuário um bom aproveitamento dos recursos locais da nuvem privada e a utilização da elasticidade proporcionada pela nuvem pública quando a demanda exceder a capacidade dos recursos privados. Entretanto, cuidado adicional deve ser tomado para que a execução da aplicação não extrapole limites de orçamento estipulados previamente. Nesse contexto, o estudo das incertezas contribui também para evitar gastos monetários desnecessários quando nuvens públicas são utilizadas.

Podemos identificar diversos focos de incerteza em sistemas de grade e nuvem. Nas grades, o compartilhamento de recursos pode atrasar a execução das aplicações. Em nuvens, apesar da virtualização permitir que um recurso físico seja compartilhado e apresentado aos usuários como diversos recursos independentes, a utilização de processadores, discos e rede de um mesmo recurso computacional físico potencialmente cria atrasos em barramentos, unidades de disco e enlaces, o que contribui para a existência de incertezas nas estimativas de execução [Senna et al. 2010]. Além disso, enlaces da Internet podem ser utilizados em uma nuvem híbrida para transmissão de dados entre a nuvem privada e a nuvem pública, o que dificulta a estimativa do tempo de comunicação entre tarefas.

Neste trabalho modelamos um sistema heterogêneo distribuído $\mathcal{R}$ como um conjunto de recursos computacionais $\mathcal{R} = \{r_1, r_2, ..., r_k\}$. Cada recurso $r_i \in \mathcal{R}$ possui uma capacidade de processamento $p_{r_i} \in \mathbb{R}^+$ associada, além de um conjunto de enlaces

$\mathcal{L}_i = \{l_{i,1}, l_{i,2}, ..., l_{i,m}\}$, $1 \leq m \leq k$, onde $l_{i,j} \in \mathbb{R}^+$ é a largura de banda do enlace entre os recursos r_i e r_j, com $l_{i,i} = \infty$. Como discutido anteriormente, a incerteza presente no tempo de execução das tarefas pode ter origem na variabilidade apresentada pelas capacidades de processamento $p_{r_i} \in \mathbb{R}^+$ e largura de banda $l_{i,j} \in \mathbb{R}^+$, que são utilizadas como entrada do escalonador.

3.2. Escalonamento

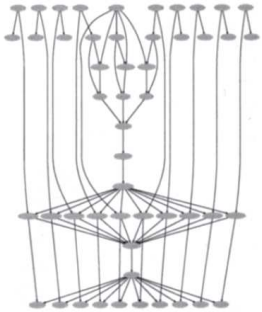
Para explorar sistemas computacionais distribuídos, como grades e nuvens, em toda sua amplitude os recursos disponíveis devem ser eficientemente utilizados. O escalonamento é essencialmente um processo de tomada de decisão que permite o compartilhamento de recursos entre uma dada quantidade de atividades a serem realizadas. Para isso, o escalonador atua na organização da seqüência de atividades em máquinas existentes, dessa forma determinando o escalonamento das atividades nos recursos disponíveis para executá-las. O tempo de execução de cada tarefa, e conseqüentemente o tempo de conclusão da execução de todas as tarefas, é intrínseco ao escalonamento gerado. Portanto, o escalonador é responsável por decidir qual tarefa computacional (ou serviço) deve ser executado em qual recurso computacional, tendo grande influência na eficiência do sistema. O escalonador deve levar em consideração diversos aspectos, como o tempo e custo de computação das tarefas, tempos e custos de transmissão de tarefas e serviços, custos de transmissão de dados, localização das fontes de dados, *overheads* de virtualização, etc. Com isso, a presença de imprecisão em dados estimados para esses parâmetros afetam a qualidade do escalonamento e a eficiência do sistema.

Em sistemas distribuídos a função objetivo mais freqüentemente encontrada na literatura é a minimização do tempo de execução (*makespan*), alcançado por uma alocação propícia das tarefas nos recursos disponíveis. Em nuvens computacionais é comum que a função objetivo também englobe um tempo máximo de execução para todas as tarefas (*deadline*) e os custos monetários envolvidos [Bittencourt and Madeira 2011].

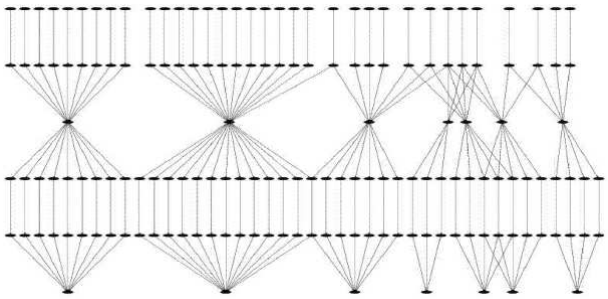
Diversas definições para o problema geral de escalonamento podem ser encontradas na literatura [Pinedo 2008, Haupt 1989, Gonzalez 1977]. Neste artigo utilizamos uma definição do problema para escalonamento em sistemas distribuídos, como segue: seja $\mathcal{A} = \{a_1, a_2, ..., a_n\}$ o conjunto de tarefas submetidas pelos usuários e que serão executadas no sistema distribuído heterogêneo $\mathcal{R}$. O escalonador é responsável pela construção de um sequenciamento de tarefas de $\mathcal{A}$ nos recursos de $\mathcal{R}$ seguindo um ou mais objetivos a serem otimizados.

3.3. Workflows

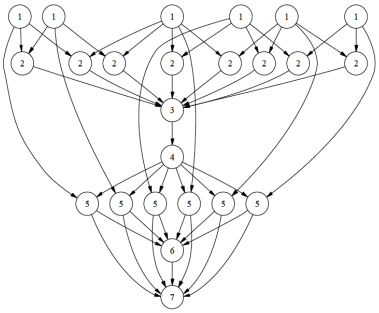
Tarefas submetidas para execução em uma nuvem podem ser, de forma geral, divididas em duas classes: tarefas independentes e tarefas dependentes. Tarefas dependentes têm um fluxo de dados em que cada tarefa depende de dados computados por outras tarefas. Um conjunto de tarefas dependentes pode ser chamado de *workflow*. Esse paradigma tem sido usado amplamente na representação de processos científicos [Taylor et al. 2007]. Com o surgimento das grades computacionais, o escalonamento de *workflows*, particularmente aqueles de aplicações de e-Ciência, passou a receber atenção substancial. Algumas aplicações que usam esse paradigma são Montage (Figura 1(c), de [Deelman et al. 2005]); AIRSN (Figura 1(a), de [Zhao et al. 2005]); LIGO (Figura



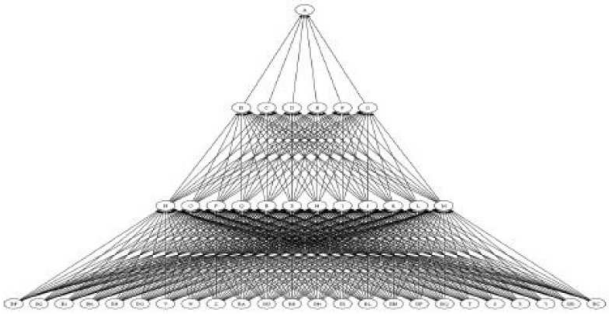
(a) AIRSN



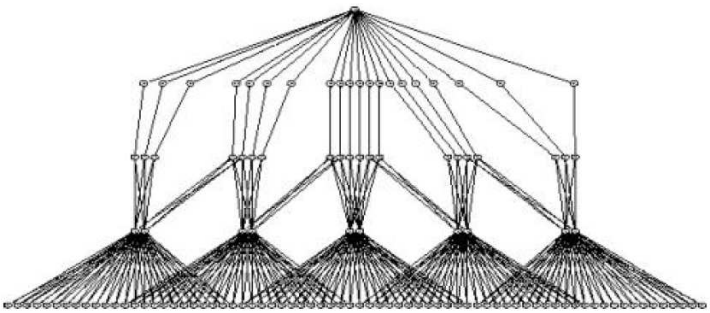
(b) LIGO



(c) Montage



(d) Chimera-1



(e) Chimera-2

Figura 1. Exemplos de aplicações de *workflow*.

1(b), de [Deelman et al. 2002]); e, Chimera-1 e Chimera-2 (Figuras 1(d) e 1(e), de [Annis et al. 2002]).

Em um *workflow* a execução de cada tarefa deve respeitar as precedências impostas por tais dependências. Então, o escalonador deve tratar das dependências de dados, custos de comunicação entre tarefas e custos de computação das tarefas componentes do processo.

DAGs tornaram-se o padrão na representação de *workflows* acíclicos. Com a evolução da e-Ciência, esses *workflows* tornaram-se maiores e passaram a demandar mais poder computacional. Essas demandas dos *workflows* científicos podem ser supridas utilizando grades e nuvens computacionais em conjunto com escalonadores eficientes. Entender a qualidade da informação e sua influência nos algoritmos de escalonamento pode contribuir para o aprimoramento da eficiência dos escalonadores, consequentemente colaborando para melhoria da execução dos workflows.

Um *workflow* composto por tarefas dependentes é tipicamente modelado como um grafo acíclico direcionado (directed acyclic graph - DAG). Para cada nó n de um DAG não existe qualquer caminho direcionado que inicia e termina em n. Um processo é representado por um DAG $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ com n nós (ou tarefas), onde $t_a \in \mathcal{V}$ é uma tarefa do *workflow* com um custo de computação (peso) $w_{t_a} \in \mathbb{R}^+$ associado, e $e_{a,b} \in \mathcal{E}$, é uma dependência de dados entre t_a e t_b com custos de comunicação $c_{a,b} \in \mathbb{R}^+$ associados. Assim:

- $\mathcal{V}$ é o conjunto de $n = |\mathcal{V}|$ nós, e $t_i \in \mathcal{V}$ representa uma tarefa atômica que deve ser executada em um recurso;
- $\mathcal{E}$ é o conjunto de $e = |\mathcal{E}|$ arestas direcionadas (arcos), onde $e_{i,j} \in \mathcal{E}$ representa dependência de dados entre t_i e t_j. Isso implica que t_j não pode iniciar sua execução antes que t_i termine e envie os dados necessários à t_j;

Adicionalmente, o modelo de workflow pode conter desvios condicionais, onde tarefas podem ou não executar dependendo de resultados de seus predecessores. DAGs condicionais podem implicar em re-escalonamento de tarefas mesmo quando não há incerteza nos dados de entrada.

Os custos de computação $w_{t_a} \in \mathbb{R}^+$ e de comunicação $e_{a,b} \in \mathcal{E}$ estão sujeitos a incertezas advindas de estimativas incorretas, sejam tais estimativas realizadas pelo usuário ou por um sistema de estimativa baseado em execuções passadas da mesma aplicação.

3.4. Qualidade da Informação

A qualidade da informação no escalonamento refere-se à precisão dos parâmetros de entrada do escalonador, que englobam: (i) informações sobre o conjunto de recursos, como as capacidades de processamento e largura de banda disponíveis; e (ii) informações sobre a aplicação, como tempo de execução das aplicações e tamanho das dependências de dados. Dessa forma, a qualidade da informação é um conceito derivado da incerteza, que por sua vez é resultado da soma de um conjunto de estimativas imprecisas que são utilizadas pelo escalonador no processo de tomada de decisão.

O modelo de qualidade da informação utilizado neste artigo baseia-se em [Casanova et al. 2000]. Assumimos que, inicialmente, todas as estimativas são 100% precisas, isto é, os custos de computação e comunicação estimados e informados ao escalonador, assim como as capacidades de processamento e largura de banda, correspondem àquelas que observadas na execução do *workflow*. Assim, para simular a variabilidade da qualidade da informação na presença de incertezas, introduzimos nos dados de entrada do escalonador um erro percentual que é uniformemente distribuído no intervalo $[-p, +p]$, onde p é um valor entre 0% e 100%. Uma qualidade da informação perfeitamente precisa possui $p = 0$, enquanto, por exemplo, $p = 10$ implica que cada estimativa informada ao escalonador será aleatoriamente incrementada ou decrementada em no máximo 10% durante a execução.

4. O Algoritmo HEFT

O algoritmo *Heterogeneous Earliest Finish Time* (HEFT) [Topcuoglu et al. 2002] foi desenvolvido para o escalonamento de DAGs em sistemas distribuídos heterogêneos. O HEFT é um algoritmo notadamente reconhecido por sua simplicidade e baixa complexidade. Esses dois fatores aliados aos bons escalonamentos gerados nos mais variados

cenários tornaram o HEFT um dos algoritmos mais populares no escalonamento estático de *workflows* em sistemas heterogêneos.

Utilizando a técnica de *list scheduling*, onde uma lista de prioridades para os nós a serem escalonados é criada, o HEFT primeiramente atribui um peso para cada nó e arco (ou aresta direcionada) do DAG. O peso de cada nó é calculado como sendo o custo médio de computação da tarefa em cada máquina, enquanto o peso de cada arco é a média dos custos de comunicação para o arco sobre todos os pares de recursos. Em seguida, um valor de classificação, $rank_u$, é computado atravessando o DAG de trás para frente. O atributo de prioridade $rank_u$ para cada tarefa t_i é definido da seguinte forma:

$$rank_u(t_i) = \overline{w_i} + \max_{t_j \in suc(t_i)} (\overline{c_{i,j}} + rank_u(t_j)), \quad (1)$$

onde $\overline{w_i}$ é a média do tempo de execução de t_i sobre todos os recursos, $suc(t_i)$ é o conjunto de sucessores imediatos de t_i, e $\overline{c_{i,j}}$ é a média dos custos de comunicação entre t_i e t_j. Dessa forma, o $rank_u$ de uma tarefa t é o peso de t mais o valor máximo resultando do peso de cada filho de t adicionado ao peso do arco que conecta esse filho a t. Em seguida, as tarefas são escalonadas uma a uma em ordem não crescente de $rank_u$ no recurso que resultar o menor tempo estimado de término (*estimated finish time* - EFT), que é computado utilizando as capacidades de processamento e dos enlaces em conjunto com os custos de computação e comunicação associados ao DAG. Durante a busca pelo recurso que proporciona o menor tempo estimado de término, o HEFT considera a inserção de tarefas em espaços oriundos das transmissões das dependências de dados.

A breve descrição do HEFT apresentada nesta seção permite fazer uma análise prévia da importância da precisão dos dados de entrada em algoritmos de escalonamento de tarefas dependentes. A prioridade $rank_u$, assim como o tempo estimado de término (EFT) e outros atributos utilizados durante o escalonamento, é computada para todas as tarefas do DAG se baseia totalmente em informações sobre os recursos e sobre as tarefas componentes do DAG. Se as capacidades dos recursos e/ou duração das tarefas, fornecidas previamente ao escalonador, sofrem variações durante a execução, todo o processo de escalonamento é afetado e a lógica na qual a heurística baseia-se perde eficiência.

Considere o DAG da Figura 2, com os custos de tarefas e dependências associados. Adicionalmente, considere 3 recursos em 2 grupos diferentes. Em um grupo, recursos $R0$ e $R1$ com capacidade de processamento de 133 e 130, respectivamente. No outro grupo, o recurso $R2$ com capacidade de processamento de 118. A largura de banda entre dois recursos do mesmo grupo é 10, enquanto para pares de recursos em grupos distintos é 5. Um escalonamento hipotético é mostrado na Figura 2, onde as partes hachuradas representam desempenho melhor que o estimado e as áreas em cinza representam desempenho pior que o estimado. Mesmo com um desempenho melhor na execução das tarefas 6, 7 e 10, um desempenho aquém do esperado no recurso $R2$, mesmo que proporcionalmente menor à melhoria nos recursos $R0$ e $R1$, afeta negativamente o escalonamento gerado, atrasando a execução da tarefa 11.

5. Experimentos

Nesta seção apresentamos resultados obtidos através de simulações do algoritmo *Heterogeneous Earliest Finish Time* (HEFT) [Topcuoglu et al. 2002]. A métrica utilizada é o

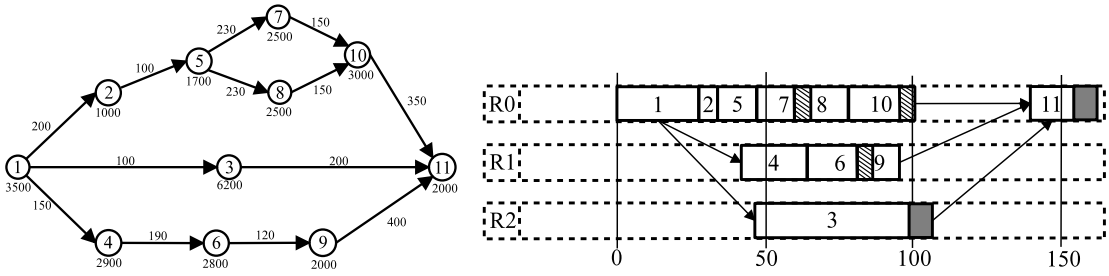


Figura 2. DAG e escalonamento hipotético com influência de incertezas.

makespan médio, ou seja, o tempo de término da última tarefa do *workflow*.

5.1. Configurações da simulação

As simulações foram conduzidas utilizando 2, 5 e 10 recursos heterogêneos no sistema alvo, com cada recurso r_i possuindo capacidade de processamento $p_{r_i} \in \mathbb{R}^+$ obtida aleatoriamente de uma distribuição uniforme no intervalo $(10, 100)$. Os recursos estavam totalmente conectados por uma rede heterogênea, e cada enlace $l_{i,j} \in \mathbb{R}^+$ entre um par de recurso r_i e r_j teve sua largura de banda obtida aleatoriamente de uma distribuição uniforme no intervalo $(10, 100)$. As médias apresentadas foram calculadas sobre 500 execuções para cada cenário.

Cinco DAGs diferentes foram utilizados nas simulações, todos correspondendo a aplicações de *workflow* encontradas no mundo real apresentadas anteriormente na Figura 1. São elas:

- Montage: 24 tarefas [Deelman et al. 2005];
- AIRSN: 51 tarefas [Zhao et al. 2005];
- LIGO: 166 tarefas [Deelman et al. 2002];
- Chimera-1: 43 tarefas [Annis et al. 2002];
- Chimera-2: 123 tarefas [Annis et al. 2002].

Os custos de computação (peso) $w_{t_a} \in \mathbb{R}^+$ para cada tarefa t_a dos DAGs foram obtidos aleatoriamente de uma distribuição uniforme no intervalo $(500, 4000)$. Para cada tipo de DAG, executamos simulações utilizando diferentes relações entre a quantidade de computação e a quantidade de comunicação total do *workflow*, isto é, utilizando *communication to computation ratio* (CCR) de 0.5, 1.0 e 2.0. Um $CCR = 2.0$ significa que o *workflow* gasta duas vezes mais tempo realizando comunicação (isto é, transmitindo dados das dependências) do que realizando computação (isto é, executando cada tarefa do DAG).

Para avaliar o comportamento do HEFT na presença de incertezas, utilizamos os valores de $p = 0$, $p = 20$ e $p = 50$ aplicados às capacidades de computação e às larguras de banda dos enlaces. Portanto, $p = 0$ é o escalonamento onde as informações de entrada do escalonador correspondem exatamente ao que é encontrado durante a execução. Para, por exemplo, $p = 20$, após o algoritmo HEFT ter feito o escalonamento do DAG, as capacidades de processamento e as larguras de banda foram incrementadas ou decrescidas em no máximo 20% do valor original informado ao algoritmo, como no modelo sugerido em [Casanova et al. 2000]. Esse caso representa uma situação onde os recursos de processamento e de transmissão de dados poderiam ter um desempenho até 20% melhor ou

pior que o informado ao algoritmo de escalonamento. Note que a estimativa dada ao escalonador não é pessimista nem otimista, pois a variação de desempenho é aleatoriamente atribuída como positiva ou negativa.

5.2. Resultados

A Figura 3 mostra os resultados obtidos para o DAG Montage. Observamos que para qualquer CCR e qualquer quantidade de recursos a qualidade da informação é importante, resultando em um *makespan* estimado incorretamente, sempre para menos, em todos os casos.

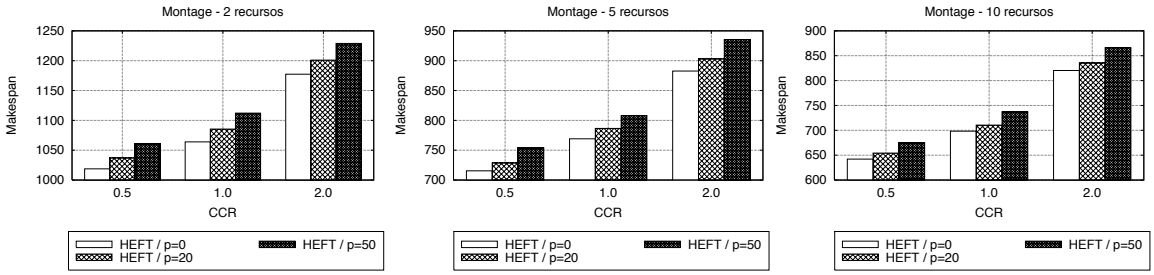


Figura 3. Makespan médio para o DAG Montage.

Resultados para o DAG AIRSN são mostrados na Figura 4. A importância da qualidade da informação continua evidente, com as incertezas resultando sempre em um aumento do *makespan*, ainda que essas incertezas tenham sido geradas de forma aleatória para diminuir e também para aumentar a capacidade estimada inicialmente. Para este DAG, observamos ainda uma leve tendência de aumento do erro da estimativa com o aumento do CCR.

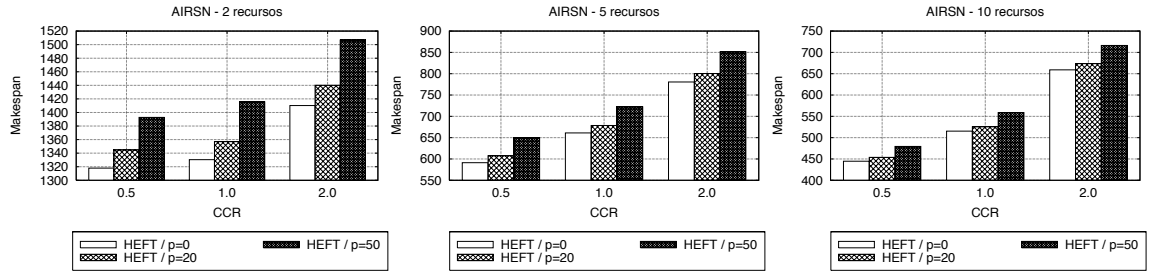


Figura 4. Makespan médio para o DAG AIRSN.

A qualidade da informação mostrou-se ainda mais importante nos resultados do *workflow* LIGO, apresentados na Figura 5. A incerteza introduziu erro de estimativa do *makespan* de até 14% para essa aplicação quando existiam 10 recursos e $p = 50\%$. A mesma tendência de leve aumento do erro com o aumento do CCR foi observada. Além disso, observamos uma tendência de aumento do erro na estimativa do *makespan* com o aumento do número de recursos, sendo de 4% para $p = 50$ com 2 recursos, de 10% para 5 recursos e de 14% para 10 recursos. Atribuímos essa tendência ao maior espalhamento das tarefas desse DAG, que possui 168 nós, quando existe uma maior quantidade de recursos disponíveis. Se por um lado com poucos recursos a comunicação entre tarefas é suprimida, já que tarefas dependentes no mesmo recurso têm custo de comunicação

igual a zero, por outro lado uma distribuição maior de tarefas entre mais recursos tende a aumentar a quantidade de comunicação realizada. Com isso, a variabilidade introduzida na largura de banda torna-se tão importante quanto a variabilidade da capacidade de processamento.

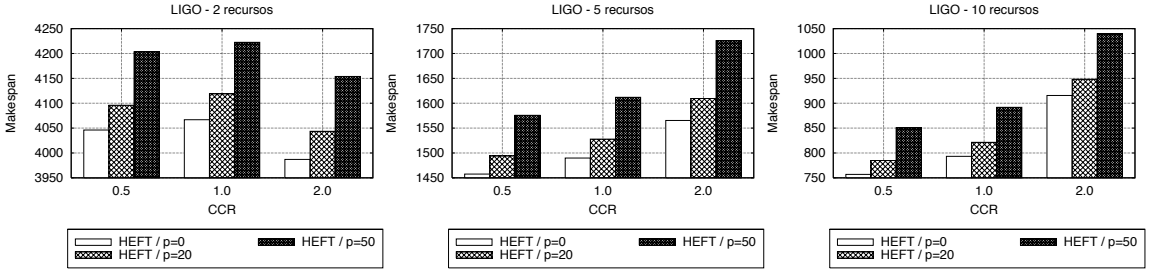


Figura 5. Makespan médio para o DAG LIGO.

Os resultados para o DAG Chimera-1 (Figura 6) também apresentaram uma tendência de aumento no erro com o aumento do CCR e da quantidade de recursos. Nesse caso, a variação do erro foi de 7% (2 recursos, CCR=2) a 17% (10 recursos, CCR=2). Apesar desse *workflow* ser menor que o LIGO, ele apresenta uma alta densidade de arestas. Assim, a largura de banda torna-se mais importante quando a quantidade de recursos é maior, pois com as tarefas mais espalhadas, um número maior de transmissões de dados são realizadas.

Resultados do DAG Chimera-2 corroboram aqueles obtidos para o DAG Chimera-1. O aumento da quantidade de recursos culmina em aumento do erro na estimativa do *makespan* quando incertezas existem, tendo o erro variado entre 5% (2 recursos, CCR=2) e 15% (10 recursos, CCR=2).

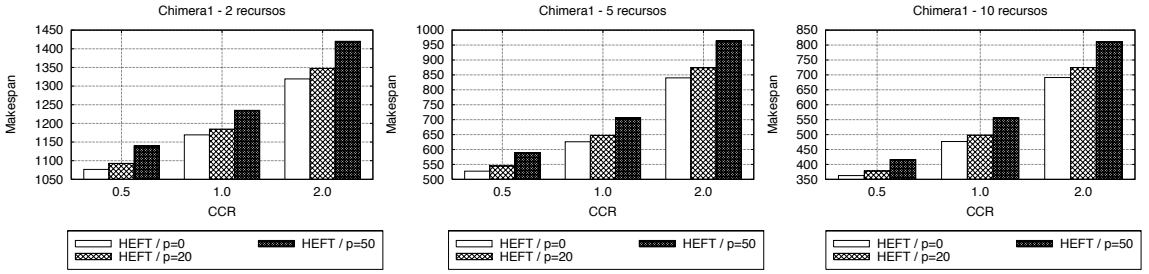


Figura 6. Makespan médio para o DAG Chimera-1.

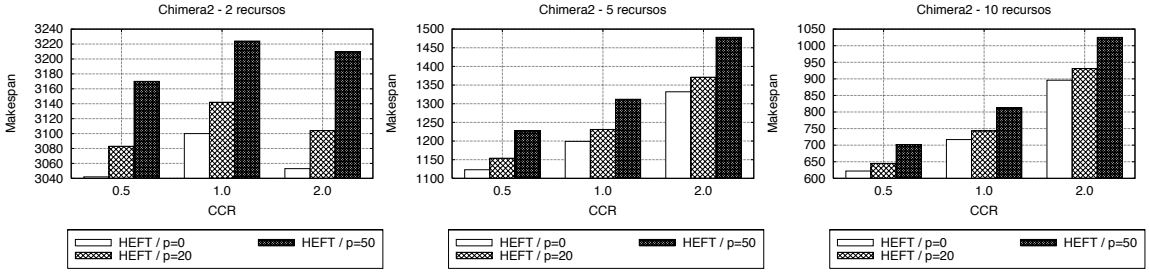


Figura 7. Makespan médio para o DAG Chimera-2.

5.3. Discussão e análise

Os resultados obtidos sugerem que os dados de entrada fornecidos ao escalonador, quando sujeitos à incertezas, podem afetar significativamente o *makespan* real da aplicação quando comparado ao estimado pelo escalonador. Em ambientes de grade computacional, tal erro de estimativa pode acarretar no atraso da execução das aplicações. Já em ambientes de nuvem, além do atraso na execução da aplicação, o usuário pode ter seu orçamento afetado pela maior utilização de recursos computacionais pagos, enfatizando a importância do desenvolvimento de escalonadores cientes de incertezas também para esses ambientes.

O algoritmo HEFT, assim como diversos outros algoritmos de escalonamento de *workflows*, baseiam-se fortemente na premissa de que os dados de entrada com informações sobre os DAGs e sobre os recursos são precisos. Com isso, os atributos computados para as tarefas do DAG, que são extremamente importantes na decisão de escalonamento, utilizam apenas valores absolutos dos dados de entrada. Na prática, tais dados são difíceis de obter com precisão, o que introduz perda na qualidade da informação e gera incertezas no processo de escalonamento e execução das tarefas. Para minimizar o impacto negativo da diferença entre a estimativa e a execução real, algoritmos de escalonamento a serem desenvolvidos podem considerar como entrada distribuições estatísticas de desempenho dos recursos e de tempos e requisitos das aplicações ao invés de apenas valores absolutos na computação dos atributos. Com isso, tais algoritmos evitariam abordagens reativas, como escalonamento dinâmico, anulando o impacto e diminuindo a complexidade do monitoramento no sistema distribuído.

6. Conclusão

Dados de entrada de escalonadores são determinantes na qualidade do escalonamento gerado. Entretanto, tais dados são difíceis de obter de forma precisa por uma gama de motivos, dentre os quais destacamos em grades e nuvens: (i) heterogeneidade dos recursos torna difícil prever tempo de execução da mesma aplicação em máquinas diferentes; (ii) aplicações diferentes demandam tempos de computação diferentes que nem sempre são conhecidos previamente; (iii) concorrência na utilização de recursos computacionais podem aumentar o tempo de execução estimado; e (iv) indisponibilidade de dados sobre a infra-estrutura utilizada, como no caso de largura de banda em nuvens e suas interconexões, impede que dados precisos sejam utilizados.

Neste artigo realizamos um estudo preliminar do comportamento da heurística *Heterogeneous Earliest Finish Time* (HEFT) [Topcuoglu et al. 2002] quando os dados de entrada utilizados estão sujeitos a incertezas. A discussão apresentada em conjunto com os resultados obtidos mostra que a imprecisão dos dados de entrada afeta negativamente o desempenho do escalonamento gerado, tendo como conseqüência o aumento do tempo de execução do *workflow* e, no caso da execução em nuvens, potencial aumento de custos monetários. Portanto, este estudo traz à tona a importância do desenvolvimento de algoritmos que considerem a qualidade da informação de entrada em sistemas como grades e também nuvens.

Como trabalho em andamento, estamos desenvolvendo um algoritmo que utiliza custos relativos e intervalos de valores em arestas e nós do DAG ao invés de um único valor absoluto. Os resultados obtidos por esse algoritmo serão utilizados para direcionar o

desenvolvimento de um algoritmo que utilize distribuições estatísticas como informações de entrada sobre o DAG e o sistema-alvo.

Agradecimentos

Os autores agradecem à FAPESP (2009/15008-1), CNPq e RNP pelo apoio financeiro.

Referências

Allen, G., Angulo, D., Foster, I., Lanfermann, G., Liu, C., Radke, T., Seidel, E., and Shalf, J. (2001). The cactus worm: Experiments with dynamic resource discovery and allocation in a grid environment. *International Journal of High Performance Computing Applications*, 15:2001.

Annis, J., Zhao, Y., Voeckler, J., Wilde, M., Kent, S., and Foster, I. (2002). Applying Chimera virtual data concepts to cluster finding in the Sloan Sky Survey. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE Conference on Supercomputing*, pp 1-14, Baltimore, EUA. IEEE Computer Society Press.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Communications of the ACM*, 53:50–58.

Batista, D. M. and da Fonseca, N. L. S. (2011). Robust scheduler for grid networks under uncertainties of both application demands and resource availability. *Computer Networks*, 55(1):3–19.

Batista, D. M., da Fonseca, N. L. S., Miyazawa, F. K., and Granelli, F. (2008). Self-adjustment of resource allocation for grid applications. *Computer Networks*, 52(9):1762–1781.

Bittencourt, L. F. and Madeira, E. R. M. (2008). A performance-oriented adaptive scheduler for dependent tasks on grids. *Concurrency and Computation: Practice and Experience*, 20(9):1029–1049.

Bittencourt, L. F. and Madeira, E. R. M. (2011). HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds. *Journal of Internet Services and Applications*, 2(3):207–227.

Casanova, H., Legrand, A., Zagorodnov, D., and Berman, F. (2000). Heuristics for scheduling parameter sweep applications in grid environments. In *Heterogeneous Computing Workshop, 2000. (HCW 2000) Proceedings. 9th*, pages 349 –363.

Deelman, E., Kesselman, C., Mehta, G., Meshkat, L., Pearlman, L., Blackburn, K., Ehrens, P., Lazzarini, A., Williams, R., and Koranda, S. (2002). GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists. In *HPDC '02: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing*, Edinburgh, UK. IEEE Computer Society.

Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, G. B., Good, J., Laity, A., Jacob, J. C., and Katz, D. S. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237.

Dong, F. and Akl, S. G. (2006). Scheduling algorithms for grid computing: state of the art and open problems. Technical report, Queen's University School of Computing, Kingston, Canada.

Genez, T. A. L., Bittencourt, L. F., and Madeira, E. R. M. (2012). Workflow Scheduling for SaaS / PaaS Cloud Providers Considering Two SLA Levels. In *IEEE/IFIP Network Operations and Management Symposium (NOMS 2012)*. IEEE.

Gonzalez, Jr., M. J. (1977). Deterministic processor scheduling. *ACM Comput. Surv.*, 9(3):173–204.

Grajcar, M. (1999). Genetic list scheduling algorithm for scheduling and allocation on a loosely coupled heterogeneous multiprocessor system. In *DAC '99: Proceedings of the 36th ACM/IEEE conference on Design automation*, Nova Iorque, EUA. ACM Press.

Hakem, M. and Butelle, F. (2005). Dynamic critical path scheduling parallel programs onto multiprocessors. In *19th International Parallel and Distributed Processing Symposium*. IEEE Computer Society.

Haupt, R. (1989). A survey of priority rule-based scheduling. *OR Spectrum*, 11(1):3–16.

J.Yu, Buyya, R., and Ramamohanarao, K. (2008). Workflow scheduling algorithms for grid computing. *Studies in Computational Intelligence*, 146:173–214.

Phillips, C., Stein, C., and Wein, J. (1997). Task scheduling in networks. *SIAM Journal on Discrete Mathematics*, 10(4):573–598.

Pinedo, M. L. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, New York.

Sakellariou, R. and Zhao, H. (2004a). A hybrid heuristic for dag scheduling on heterogeneous systems. In *13th Heterogeneous Computing Workshop, (IPDPS'04 Workshops)*. IEEE Computer Society.

Sakellariou, R. and Zhao, H. (2004b). A low-cost rescheduling policy for efficient mapping of workflows on grid systems. *Scientific Programming*, 12(4):253–262.

Senna, C. R., Bittencourt, L. F., and Madeira, E. R. M. (2010). Performance evaluation of virtual machines in a service-oriented grid testbed. In *Intl. Conference on High Performance Computing & Simulation (HPCS 2010)*, Caen, France. IEEE CS Press.

Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M. (2007). *Workflows for e-Science. Scientific Workflows for Grids*. Springer.

Topcuoglu, H., Hariri, S., and Wu, M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274.

Yu, J. and Buyya, R. (2006). Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms. *Sci. Program.*, 14(3,4):217–230.

Zhang, Q., Cheng, L., and Boutaba, R. (2010). Cloud computing: state-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1(1):7–18.

Zhao, Y., Dobson, J., Foster, I., Moreau, L., and Wilde, M. (2005). A notation and system for expressing and executing cleanly typed workflows on messy scientific data. *SIGMOD Records*, 34(3):37–43.