Uma Metodologia para o Auto-Ajuste de Aplicações em Grades

Daniel M. Batista[1] **, Nelson L. S. da Fonseca**[1] **,**
Fabrizio Granelli[2] **, Dzmitry Kliazovich**[2]

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

[2]DIT - Univ. of Trento,
Via Sommarive 14, I-38050, Trento, Itália

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

{granelli,klezovic}@dit.unitn.it

Abstract. *This paper introduces a procedure for self-adjusting the schedule of tasks in grids considering the dynamics of resource availability. The procedure involves monitoring, scheduling and task migration. The core of the procedure is based on task migration. The procedure differs from others in the literature by considering the state of network resources. Simulation results derived using the NS-2 simulator demonstrate the efficacy of the procedure.*

Resumo. *Este artigo apresenta uma metodologia para auto-ajustar o escalonamento de tarefas dependentes em grades frente à dinâmica do estado dos recursos. A metodologia baseia-se na execução cíclica dos passos de monitoramento dos recursos da grade, escalonamento e migração das tarefas. Este último passo constitui o cerne da proposta. A metodologia diferencia-se das demais encontradas na literatura pelo fato de considerar o estado da rede durante todos os passos. Resultados de simulações realizadas em um simulador de grades construído sobre o NS-2 comprovam a eficácia da metodologia.*

1. Introdução

Um dos pontos-chave para alcançar automaticidade nas grades é a alocação eficiente dos recursos para a execução das aplicações. Como as grades caracterizam-se pela heterogeneidade dos recursos e por mudanças imprevisíveis nos estados dos mesmos, há a necessidade de se tomar decisões sobre o escalonamento de tarefas das aplicações com o mínimo de interferência humana, a fim de que as aplicações executem no menor tempo possível.

Decisões de alocação que não levam em conta a variação da utilização dos recursos da rede podem implicar em um aumento considerável no tempo de execução de aplicações na grade, principalmente para as aplicações compostas por tarefas que transferem grandes quantidades de dados entre si. Algumas destas aplicações foram apresentadas no evento iGRID2005. Demonstrou-se que elas podem ser responsáveis por aumentar de cinco a oito vezes a utilização dos enlaces da CalREN (*California Research and Education Network*), uma rede de pesquisa nos Estados Unidos [Silvester 2005]. A sensibilidade do desempenho das aplicações nas grades frente ao estado da rede é abordada também em [Kamous-Edwards and Jukan 2006].

Outro aspecto de grande importância que deve ser considerado no auxílio à automatização das grades é o suporte à migração das tarefas. No momento em que uma aplicação é submetida para execução em uma grade, o escalonamento produzido para as tarefas desta aplicação considera o estado atual dos recursos. À medida que o tempo passa, novos processos nos recursos podem ser iniciados ou finalizados, o que modifica o estado dos mesmos, e novos recursos podem ser agregados à grade, o que fornece novas opções de hosts a serem alocados. Essas modificações nas grades devem ser detectadas e, caso seja necessário, tarefas devem ser migradas com o objetivo de minimizar o tempo de execução da aplicação. Esses procedimentos de detecção de mudanças e migração de tarefas, a fim de evitar o aumento no tempo de execução das aplicações nas grades, remetem ao objetivo da Engenharia de Tráfego para Internet, que propõe a utilização de técnicas que otimizem o uso de recursos, a fim de garantir suporte a QoS em uma rede IP.

O presente trabalho apresenta uma metodologia para auto-ajustar o escalonamento de aplicações nas grades, que inclui monitoramento dos recursos computacionais e de comunicação, escalonamento e migração de tarefas. A metodologia, denominada "Engenharia de Recursos para Aplicações em Grades", é inspirada na Engenharia de Tráfego para Internet e objetiva uma alocação eficiente dos recursos a fim de manter o tempo de execução das aplicações baixo e robusto frente às mudanças no estado das grades. A contribuição da metodologia está no fato dela considerar o custo das transferências de dados, que é influenciado pelo estado dos enlaces, tanto no escalonamento quanto na migração das tarefas, sendo que este último depende do progresso de execução das tarefas. O impacto da contribuição está no fato de se permitir que seja realizado o auto-ajuste das aplicações nas grades, independente dos seus requisitos de processamento e comunicação e sem intervenção do usuário. A eficácia da metodologia é comprovada através de experimentos realizados em um simulador de grades desenvolvido sobre o simulador de redes NS-2 [ISI 2004]. O simulador representa uma contribuição adicional deste artigo.

Esquemas para ajustar o escalonamento das aplicações frente a mudanças no estado das grades vêm sendo propostos na literatura desde as primeiras implementações de grades [Allen et al. 2001] [Vadhiyar and Dongarra 2003] [Huedo et al. 2002]. **No entanto, eles negligenciam a importância que os recursos de comunicação têm no tempo de execução das aplicações**, o que não acontece com a metodologia apresentada neste artigo. A metodologia de Engenharia de Recursos considera o estado da rede em todas as etapas. O estado da rede influencia a definição do primeiro escalonamento das tarefas e a definição do host para o qual uma tarefa deve migrar. A rede também é considerada no cálculo do *overhead* das migrações a fim de evitar que o tempo necessário para transferir os dados seja tal que piore o tempo de execução da aplicação.

A metodologia aqui apresentada tem o objetivo de auto-ajustar cada aplicação nas grades individualmente. Não se tem como objetivo a otimização global dos recursos das grades. As aplicações-alvo da metodologia são aquelas formadas por tarefas dependentes, cuja necessidade de transferência de dados seja da ordem de GigaBytes e que executem em uma escala de tempo de horas, tais como aplicações das áreas de física de altas energias e astronomia, aplicações estas que motivaram a criação das grades.

É importante enfatizar que as contribuições da metodologia apresentada neste artigo podem levar as grades a alcançarem a automaticidade, e conseqüente ubiqüidade, prevista pelos cientistas da Computação e esperada pelos setores da sociedade que de-

mandam computação confiável de alto desempenho. Contribuições como essas são tão relevantes que estão destacadas em quatro dos cinco desafios apresentados no relatório "Grandes Desafios da Pesquisa em Computação no Brasil - 2006 – 2016" [SBC 2006].

As próximas seções estão organizadas da seguinte forma: a Seção 2 apresenta conceitos básicos sobre grades e sobre como se escalona tarefas nos middlewares existentes. A Seção 3 apresenta a motivação para se utilizar a metodologia de Engenharia de Recursos, tomando como base o esquema convencional de escalonamento apresentado na Seção 2. A Seção 4 introduz a metodologia. A Seção 5 fornece detalhes do `GridSim-NS`, um simulador utilizado para comprovar a eficácia da metodologia. A Seção 6 descreve os experimentos de simulação realizados e apresenta os resultados alcançados. A Seção 7 discorre sobre trabalhos relacionados e a Seção 8 conclui o artigo.

2. Grades e middlewares

Grades são sistemas que coordenam recursos, não submetidos a um controle centralizado, com o objetivo de fornecer QoS para aplicações [Foster 2002]. A coordenação dos recursos é realizada através de interfaces e protocolos padronizados e de propósito geral.

Aplicações em grades são formadas por um conjunto de programas, ou processos, chamados tarefas. Uma gama de aplicações possui tarefas dependentes entre si e costumam ser descritas por grafos acíclicos orientados (DAGs – *Directed Acyclic Graphs*). Nos DAGs, cada vértice representa uma tarefa da aplicação e cada arco representa uma dependência de dados entre duas tarefas. Os vértices têm pesos referentes à quantidade de instruções que devem ser executadas e os arcos têm pesos referentes à quantidade de bytes que devem ser transmitidos entre as tarefas dependentes. A metodologia apresentada neste artigo é voltada para aplicações formadas por tarefas dependentes entre si. Da mesma forma que [Ma and Zhang 2004] e [Sinnen and Sousa 2005], assume-se neste artigo que os pesos dos DAGs são previamente conhecidos.

Grades são implementadas com o objetivo de realizar uma ligação transparente entre as aplicações e os recursos compartilhados via rede. Middlewares para grades implementam camadas de software a fim de promover tal transparência.

Uma função a ser realizada pelos middlewares para grades é a de escalonar as tarefas das aplicações descritas pelos DAGs. As ações relacionadas com o escalonamento podem ser resumidas nas três fases da Figura 1 [Schopf 2003]. A Fase 1 determina quais recursos estão disponíveis para um usuário em um dado domínio. Há uma filtragem dos recursos disponíveis e um conjunto menor é gerado como saída. A Fase 2 analisa os recursos fornecidos pela Fase 1 e define quais recursos executarão cada uma das tarefas. Nesta fase, o ideal é que a escolha dos recursos seja feita objetivando-se minimizar o tempo de execução da aplicação. Finalmente, na Fase 3, as tarefas são executadas nos recursos definidos pela fase anterior e o resultado da aplicação é devolvido para o usuário.

Nas Fases 2 e 3, fica evidente a dependência que a seleção dos recursos nas grades tem da rede. Enlaces com mais banda disponível são as melhores opções para tarefas que precisam transferir muitos dados entre si. Além disto, o monitoramento do progresso da aplicação deve incluir o monitoramento dos enlaces da rede. Enlaces que não são dedicados exclusivamente para a grade têm a banda disponível dependente de outras aplicações e variações na banda podem justificar a migração de uma tarefa para outro host, a fim de diminuir o tempo total de execução.

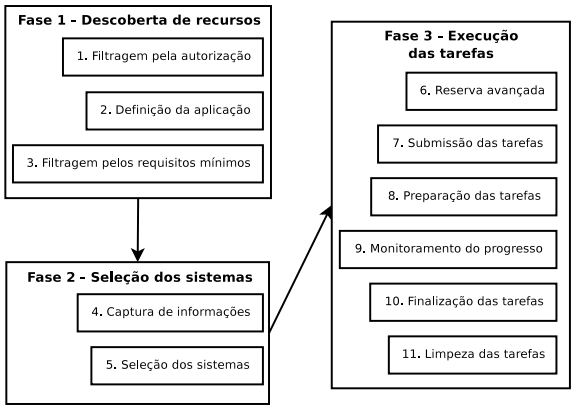


Figura 1. Resumo do escalonamento em grades

No presente artigo assume-se que as tarefas das aplicações requerem ciclos de CPU para execução nos hosts e banda passante para transferência de dados via rede (as grades são descritas por conjuntos de hosts interligados via enlaces de rede), demandas típicas de aplicações de *e-science*. Não se considera outras demandas, tais como requisitos de memória. Pretende-se, no futuro, estender a metodologia para incluir todas as possíveis demandas. A exclusão destas neste momento não limita a contribuição do trabalho.

3. Motivação para a Engenharia de Recursos para Aplicações em Grades

Esta seção ilustra, através de um exemplo, a motivação para se adotar um mecanismo de auto-ajuste para as aplicações nas grades.

É necessário que haja uma análise cuidadosa ao se definir onde cada tarefa deve ser executada na grade. Tanto a capacidade de processamento disponível nos hosts quanto a banda disponível nos enlaces devem ser avaliados. Uma vez as tarefas alocadas, é natural que ocorram mudanças no estado dos hosts da grade e dos enlaces que os interligam. Dessa forma, é necessário observar e medir o estado dos recursos a fim de avaliar o ganho que pode ser alcançado com a migração de tarefas.

Para exemplificar a necessidade de uma metodologia para auto-ajustar o escalonamento de aplicações nas grades, dar-se-á um exemplo. A Figura 2 e a Figura 3 ilustram o cenário de execução de uma aplicação de renderização remota, aplicação típica para execução em grades [Renambot et al. 2003]. Aplicações como esta são muito utilizadas para demonstrar a necessidade de se levar em consideração as características dos recursos de comunicação da grade, visto que elas produzem uma grande quantidade de bytes que devem ser transmitidos via rede.

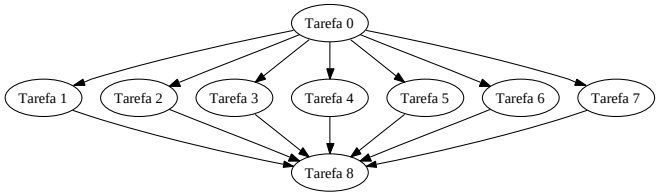
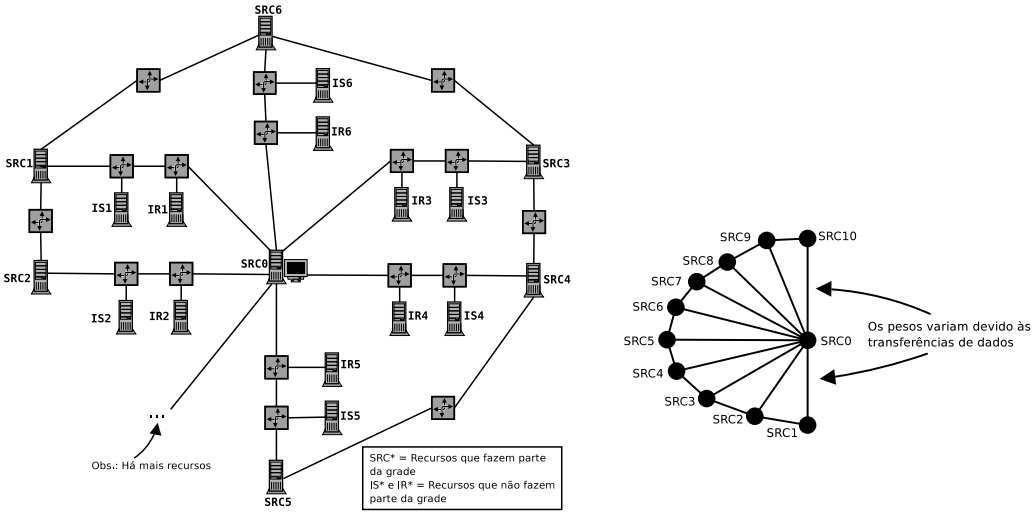


Figura 2. DAG

Pode-se ver na Figura 2 o DAG da aplicação a ser executada na organização virtual

formada pelos hosts SRC_i, que são interligados de acordo com a topologia detalhada na Figura 3(a). Essa topologia remete àquela formada pelos centros de pesquisa principais que compõem a grade LCG do CERN [CERN 2005]. A grade é formada por 11 hosts. Nem todos estão exibidos na figura para facilitar a visualização. Como somente os hosts $SRC_{\{0...10\}}$ fazem parte da grade, é apresentado na Figura 3(b) o grafo da organização virtual formada por eles. É importante observar que o host SRC_0 não descaracteriza a descentralização da grade, já que os demais hosts podem comunicar-se entre si sem a presença do host SRC_0. Além disso, nos experimentos apresentados na Seção 6, esta mesma grade é considerada e o host SRC_0 possui uma capacidade de processamento inferior a dos demais hosts da grade, o que é suficiente para garantir que as tarefas sejam executadas em paralelo ao invés de serem executadas de forma centralizada.



(a) Topologia da rede (b) Grafo da organização virtual

Figura 3. Rede

Se todos os hosts fossem interligados por enlaces dedicados, com a mesma largura de banda e capacidade de processamento disponíveis iguais e exclusivas para a grade, o mapeamento das tarefas seria bem simples e só precisaria ser feito uma única vez; a Tarefa 0 seria a primeira a executar no host SRC_0. Uma das tarefas intermediárias (1 a 7) seria executada no mesmo host SRC_0 e cada uma das demais executariam em um dos outros hosts SRC_i, após os dados de dependência serem transmitidos do host SRC_0. Os resultados das execuções das tarefas intermediárias seriam então enviados para o host SRC_0, no qual a Tarefa 8 seria executada. Entretanto, os tráfegos concorrentes entre os hosts IR_i e IS_i (Figura 3(a)), contribuem com a carga dos enlaces que interligam os hosts da organização virtual, evitando que toda a capacidade seja exclusiva para uma única aplicação na grade. Além desse tráfego concorrente, também há aplicações que variam a capacidade de processamento disponível nos hosts (aplicações locais dos usuários dos hosts SRC_i ou outras aplicações em execução na grade).

Variações no estado dos recursos de processamento e de comunicação, como as apresentadas nesse exemplo, afetam o tempo de execução previsto para as aplicações nas grades. Tarefas sendo executadas em hosts que sofrem queda na capacidade de processamento disponível podem ter seu tempo de execução aumentado, o que afeta o tempo de

execução final da aplicação. O mesmo pode ser dito de tarefas dependentes que transferem seus dados por enlaces que sofrem variações na banda disponível. Recursos adicionados à grade, ou que tenham aumentado as suas taxas de processamento e de comunicação disponíveis, também devem ser monitorados já que eles podem ser alocados para diminuir o tempo de execução das aplicações. Assim, várias questões sobre o desempenho da aplicação podem ser formuladas: vale a pena manter as tarefas nos hosts inicialmente mapeados? Haverá ganhos se uma tarefa for interrompida e transferida para um host com enlaces menos congestionados? A próxima seção apresenta uma metodologia que aborda essas questões através de migrações sugeridas sem a intervenção dos usuários.

4. Metodologia de Engenharia de Recursos para Aplicações em Grades

A Engenharia de Tráfego para Internet concentra-se na otimização do desempenho de redes IP [Awduche et al. 2002]. Ela engloba a aplicação de tecnologias e de princípios metodológicos para medir, modelar, caracterizar e controlar o tráfego da Internet. Diversas técnicas podem ser aplicadas a fim de alcançar os objetivos da Engenharia de Tráfego. Estas técnicas buscam projetar e otimizar uma rede a fim de que ela atenda os requisitos de QoS das aplicações que utilizam a Internet.

A metodologia apresentada no presente trabalho, denominada Engenharia de Recursos para Aplicações em Grades, é inspirada na Engenharia de Tráfego pra Internet. Ela objetiva gerenciar a alocação dos recursos nas grades a fim de diminuir o tempo de execução das aplicações. Assim como na Engenharia de Tráfego para a Internet, a Engenharia de Recursos para Aplicações em Grades é baseada em medições e rearranjo de alocação de recursos a fim de prover a Qualidade de Serviço requisitada, que no caso de aplicações nas grades é traduzida em tempo de execução da aplicação.

A Engenharia de Recursos para Aplicações em Grades consiste em um conjunto de passos que definem um escalonamento inicial das tarefas, monitoram o estado dos hosts da grade e dos enlaces que os interligam e avaliam se aplicações em execução podem ter seu tempo de execução reduzido através de migração de tarefas. **Esta metodologia diferencia-se das demais existentes pelo fato de considerar o custo das transferências de dados tanto no escalonamento inicial quanto na migração das tarefas, sendo que o custo para este último não é fixo e depende do progresso de execução das tarefas.**

A metodologia é utilizada para se escalonar as tarefas de uma única aplicação. Não se tem como objetivo a otimização global dos recursos de uma grade bem como a otimização de um conjunto de aplicações, mas sim a manutenção do tempo de execução da aplicação em um valor menor ou igual àquele inicialmente previsto pelo primeiro escalonamento realizado.

É importante destacar que a metodologia não tem como objetivo definir procedimentos específicos para todos os passos mas sim um procedimento completo que leve em consideração o estado da rede enquanto aplicações estiverem em execução nas grades. Dessa forma, evita-se que a metodologia fique presa a um conjunto específico de procedimentos. Como o cerne da proposta é a migração das tarefas, os passos relacionados com esta ação são os mais detalhados e um algoritmo para a sua realização é apresentado. Nos demais passos propõe-se utilizar técnicas já existentes, tais como estimadores de banda passante em enlaces e escalonadores de tarefas dependentes.

Os passos devem ser executados periodicamente a fim de se capturar variações no

ambiente que possam modificar o tempo de execução das aplicações. A cada execução dos passos, o escalonamento proposto pode diferir do anterior. Caso isso aconteça, as tarefas que forem escalonadas para outros hosts devem ter o impacto da sua migração avaliado. Caso a migração venha a diminuir o tempo de execução da aplicação, a mesma deve ser realizada. A metodologia resume-se nos passos apresentados no fluxograma da Figura 4, que são detalhados logo a seguir:

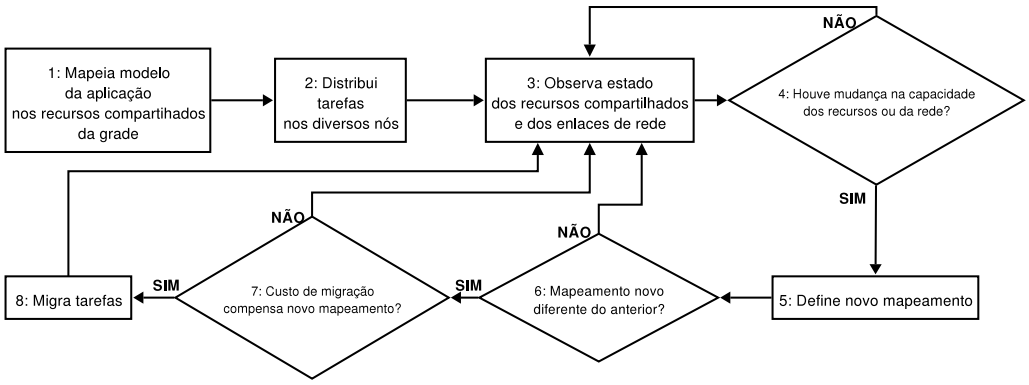


Figura 4. Metodologia de Engenharia de Recursos para Aplicações em Grades

1. A partir de um DAG e um grafo representando, respectivamente, as dependências entre as tarefas da aplicação e a topologia da rede que interliga os vários hosts da grade, é definido um escalonamento que visa responder às seguintes perguntas: i) Em qual host uma tarefa será executada? ii) Em qual instante de tempo as execuções e as transferências de dados devem ocorrer?;
2. O código e os dados necessários para cada uma das tarefas são transferidos para os hosts definidos no passo anterior e as execuções são iniciadas;
3. Durante a execução das tarefas, monitora-se os enlaces e os hosts a fim de se detectar mudanças que possam ocasionar mudanças no tempo de execução da aplicação;
4. Após coletar o estado dos enlaces e dos hosts, compara-se o mesmo com o anterior. Caso haja alguma mudança, define-se um novo mapeamento para as tarefas (Passo 5). Se o estado não mudar, volta-se ao processo de observação (Passo 3);
5. A partir do novo estado da grade, é definido um novo escalonamento para as tarefas. Somente as tarefas que ainda não terminaram sua execução são escalonadas;
6. O novo escalonamento encontrado é comparado com o anterior. Se eles forem iguais, volta-se ao processo de observação (Passo 3). Caso contrário, segue-se para a verificação do ganho alcançado com a migração (Passo 7);
7. Com o novo mapeamento, analisa-se o custo para migrar as tarefas do local atual para o novo local. Caso o custo compense, realiza-se a migração das tarefas (Passo 8). Caso contrário, volta-se ao processo de observação (Passo 3);
8. Dada a decisão de se migrar algumas tarefas, efetua-se a migração e retorna-se ao processo de observação (Passo 3).

O Passo 1 depende de um algoritmo que escalone as tarefas nos hosts, o que não é trivial dado que este é um problema NP-difícil. O algoritmo deve considerar a heterogeneidade da rede e dos hosts ao mesmo tempo em que encontra um escalonamento dentro de um tempo de execução aceitável, que não é fixo e depende tanto das características da aplicação quanto da taxa de variação do estado dos recursos da grade. Uma opção

para este passo é utilizar, ao invés de um único escalonador, um conjunto de escalonadores que seja adaptável aos requisitos temporais do ambiente. Os escalonadores podem ser executados em paralelo e o melhor escalonamento encontrado dentro do intervalo de tempo aceitável ser utilizado. O conjunto proposto pelos autores em [Batista et al. 2007] [Batista et al. 2006] pode ser usado. De um modo geral, eles caracterizam-se por diferentes tempos de execução, sendo que quanto mais rápidos, piores são os escalonamentos encontrados (não é objetivo do presente trabalho detalhar estes escalonadores).

Nos Passos 2 e 8, a metodologia não especifica nenhum protocolo para transferência dos dados. Um protocolo para transferência eficiente de grande quantidade de dados em grades e que pode ser utilizado é o GridFTP [Cannataro et al. 2003]. No Passo 8, considera-se que após interromper a execução de uma tarefa em um dado ponto é possível recomeçar a sua execução do ponto onde ela foi interrompida. Os detalhes de como isso deve ser feito na prática não é tratado pela nossa metodologia mas é possível ser realizado, como descrito em [Roy and Livny 2003], através da utilização de *checkpoints* durante a execução das tarefas.

Nos Passos 3 e 4, devem ser utilizadas técnicas para monitorar e/ou prever o estado dos hosts e dos enlaces. Dois requisitos devem ser atendidos pelas técnicas utilizadas neste passo. O valor referente ao estado atual do recurso monitorado deve ser próximo do valor real e o intervalo de monitoramento deve permitir a detecção de grandes variações no estado dos recursos, mas sem produzir um *overhead* que afete a execução das aplicações. Sem esses dois requisitos, as decisões de escalonamento e migração podem resultar em aumentos no tempo de execução da aplicação, o contrário do que se espera. Na literatura sobre monitoramento em grades, o NWS (*Network Weather Service*) [Wolski et al. 1999] é uma proposta bastante citada e que pode ser utilizada neste passo.

Os Passos 5, 6 e 7 são o núcleo da metodologia e realizam o reescalonamento e a migração das tarefas. Estes passos são responsáveis por avaliar se há ganhos caso migrações sejam realizadas. Apesar de serem passos separados, os três devem ser executados em conjunto e podem utilizar os mesmos algoritmos de escalonamento do Passo 1. O objetivo a ser alcançado com o reescalonamento continua sendo o de diminuir o tempo de execução da aplicação. As únicas diferenças entre o reescalonamento e o escalonamento inicial são a mudança no estado de alguns recursos e o fato de algumas tarefas já terem finalizado suas execuções, o que dispensa a necessidade delas terem um escalonamento encontrado. Dessa forma, é possível reescalonar as tarefas com o mesmo escalonador do Passo 1, bastando que o escalonador receba como entrada o escalonamento anterior, para saber onde as tarefas foram inicialmente escalonadas, o estado atual dos recursos, para avaliar se as mudanças justificam alguma migração, e o instante de tempo atual, a fim de saber quais tarefas já terminaram sua execução e quais são passíveis de migração. O Algoritmo 1 descreve como reutilizar o escalonador do Passo 1.

O funcionamento do algoritmo baseia-se na modificação do DAG da aplicação. Para cada tarefa i do DAG que esteja em execução, é criada uma nova tarefa (i') que representa a possível migração de i. O peso do arco da tarefa i para a tarefa i' deve representar a quantidade de bytes que devem ser transferidos para que a tarefa i possa resumir sua execução em outro host. Já o peso da tarefa i' deve representar a quantidade de instruções que faltam ser executadas pela tarefa i. Uma vez realizadas as mudanças no DAG, executa-se o escalonador. No momento dessa execução, as tarefas que já tiverem

Algoritmo 1 Reescalonamento e migração de tarefas

Entrada: Escalonamento anterior; Estado atual dos hosts e dos enlaces; Instante de tempo

1: **para** cada tarefa i em execução **faça**
2: Mude o peso de i para a quantidade de instruções já executadas.
3: Crie uma nova tarefa i' com peso igual à quantidade de instruções de i que faltam ser executadas.
4: Mova todos os arcos de saída de i para i'.
5: Crie um novo arco ii' com peso igual à quantidade de bytes que precisa ser transferido para que a tarefa i continue sua execução caso seja migrada.
6: Atribua à variável *host* o identificador do host onde i foi escalonada no escalonamento anterior.
7: Crie uma nova restrição para o escalonador que obrigue a tarefa i a ser escalonada no host *host*.
8: **fim para**
9: **para** cada tarefa k finalizada ou que tenha começado a receber dados de dependência **faça**
10: Atribua à variável *host* o identificador do host onde k foi escalonada no escalonamento anterior.
11: Crie uma nova restrição para o escalonador que obrigue a tarefa k a ser escalonada no host *host*.
12: **fim para**
13: Execute o escalonador com as novas restrições.
14: **para** cada tarefa i em execução **faça**
15: **se** host onde i' for escalonada $\neq$ host onde i foi escalonada no escalonamento anterior **então**
16: Migre a tarefa i para o novo host.
17: **fim se**
18: **fim para**

finalizado ou começado a receber dados de suas dependências devem ser mantidas fixas nos hosts alocados anteriormente, a fim de evitar que elas sejam reescalonadas sem necessidade. A migração da tarefa i é realizada caso a tarefa i' seja escalonada para um host diferente do host onde i havia sido escalonada.

O Algoritmo 1 não faz nenhuma suposição a respeito dos pesos de instruções (linha 3 do algoritmo) e de bytes das tarefas migradas (linha 5 do algoritmo). Esses dados devem ser conhecidos previamente, como explicado na Seção 2. Comparando-se os passos de reescalonamento e migração com outras propostas encontradas na literatura, nota-se que a vantagem do Algoritmo 1 deve-se a ele não ser específico para uma única aplicação e tratar os eventos de entrada e saída de recursos da mesma forma que eventos que modifiquem o estado dos recursos. Todos os eventos são considerados como relevantes caso venham a trazer modificações no tempo de execução das aplicações.

Experimentos preliminares foram realizados e comprovaram a eficácia da utilização do algoritmo [Batista 2006]. O seu funcionamento detalhado é apresentado em um dos experimentos na Seção 6.

A metodologia de Engenharia de Recursos para Aplicações em Grades pode ser incluída na arquitetura mostrada na Figura 1, da forma como está detalhada na Figura 5. Os blocos em pontilhado são substituídos pelos passos da metodologia: substitui-se blocos da Fase 2 e da Fase 3 por uma nova fase, que é responsável pela Engenharia de Recursos para Aplicações em Grades. Os blocos não modificados da Fase 3 passam a fazer parte da Fase 4, responsável pelas ações de finalização das aplicações.

A metodologia deve ser seguida pelas aplicações individualmente. No entanto, as informações oriundas do monitoramento de recursos podem ser compartilhadas entre diferentes aplicações. Apesar da metodologia ser executada individualmente por aplicação, o passo 3, responsável pelo monitoramento, garante que o impacto causado por dois escalonamentos simultâneos entre si **não seja ignorado**. A execução de cada aplicação na grade é visualizada pelas demais devido às mudanças nos estados dos recursos alocados.

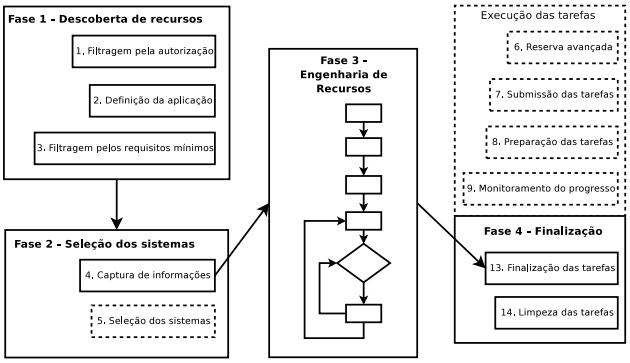


Figura 5. Inclusão da Engenharia de Recursos

5. O simulador GridSim-NS

Uma contribuição adicional deste artigo é o simulador `GridSim-NS`, um simulador de grades desenvolvido pelos autores que foi utilizado em experimentos para comprovar a eficácia da metodologia. O `GridSim-NS` mapeia as tarefas das aplicações nos nós de uma rede de comunicação. O objetivo do simulador é analisar o impacto causado pelas transferências de dados das grades nos recursos de comunicação disponíveis.

A fim de minimizar o trabalho de desenvolvimento, optou-se por implementar o simulador de grades sobre algum simulador de redes já existente. Decidiu-se desenvolver o `GridSim-NS` como uma extensão para o `NS-2`, um simulador de redes moderno, largamente utilizado e de código aberto, o que diferencia o `GridSim-NS` de outros simuladores de grades como os listados em [Sulistio et al. 2005]. A extensão consiste em um conjunto de objetos que é adicionado a um nó simples do `NS-2` a fim de permitir que o mesmo atue como membro de uma grade.

A Figura 6 apresenta a estrutura geral do processo de simulação realizado pelo `GridSim-NS`. Os blocos em cinza representam blocos que precisaram ser criados ou modificados no `NS-2`. Os blocos delimitados pela linha tracejada representam o núcleo do `GridSim-NS`, responsáveis por mapear as tarefas da aplicação na topologia de rede e por gerar o arquivo do cenário que será efetivamente simulado.

A entrada para o simulador é composta de duas topologias: a topologia lógica, que descreve o DAG da aplicação a ser executada, e a topologia da rede de comunicação, que descreve os recursos sobre os quais a grade está implementada. O `GridSim-NS` utiliza geradores das duas topologias, a fim de facilitar a realização de seqüências de experimentos em que uma topologia é mantida fixa e a outra é modificada. Ambos os geradores podem ser ignorados caso o usuário deseje simular um DAG e uma topologia de rede previamente conhecidos. O passo de mapeamento da topologia corresponde ao escalonamento das tarefas. Nesse passo, o usuário pode optar por utilizar um escalonamento personalizado, realizado por algum programa externo ao `NS-2`, ou algum dos algoritmos de escalonamento básicos implementados no simulador. O `GridSim-NS` possui três algoritmos básicos de escalonamento implementados: i) mapeamento aleatório, no qual cada tarefa é escalonada em um host da topologia de forma aleatória; ii) mapeamento ordenado, no qual as tarefas são escalonadas nos hosts levando em consideração o grau de conectividade do host e a distância para os seus vizinhos; e iii) mapeamento pela distância mínima, que funciona da mesma forma que o mapeamento ordenado com a diferença de

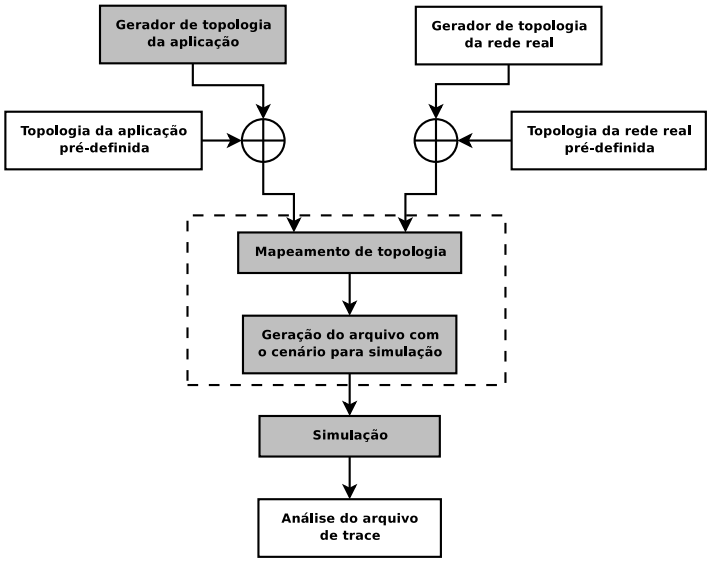


Figura 6. Processo de simulação de grades no `GridSim-NS`

que ele não permite que múltiplas tarefas sejam mapeadas para um mesmo host da rede. Uma vez realizado o escalonamento, é gerado o cenário para a simulação. Em seguida, é realizada a simulação propriamente dita. Neste passo o `GridSim-NS` não influencia no funcionamento do `NS-2` e nenhuma mudança no formato de saída do `NS-2` é realizada, o que permite que a análise do arquivo de *trace* seja realizada de forma manual ou por alguma das diversas ferramentas para análise estatística existentes.

6. Experimentos realizados

A fim de ilustrar a eficácia da metodologia, e em especial os Passos 5 a 7, são relatados, a seguir, experimentos realizados com o objetivo de medir o ganho alcançado com a metodologia apresentada. Todos os experimentos foram realizados no simulador `NS-2` compilado com a inclusão do `GridSim-NS`. As topologias de entrada foram definidas manualmente e o escalonamento foi realizado externo ao simulador. Dos escalonadores desenvolvidos pelos autores [Batista et al. 2007] [Batista et al. 2006], na maioria dos experimentos foi utilizado o TD-R1 para escalonar as tarefas e o PLITD para reescalonar e verificar a necessidade de migração. Ambos os escalonadores implementam o escalonamento de tarefas como um problema de programação linear inteira e considera-se a linha do tempo como discreta. Apesar de haver perda de precisão por esta consideração, os escalonadores executam mais rápido do que se a linha do tempo fosse considerada contínua. A diferença entre o PLITD e o TD-R1 está no fato do segundo relaxar as variáveis inteiras do primeiro, o que o faz executar mais rápido apesar de produzir escalonamentos piores do que o primeiro. Como o escalonamento inicial (Passo 1) deve encontrar um escalonamento para **todas** as tarefas do DAG, optou-se por utilizar o escalonador TD-R1 neste passo, a fim de obter um escalonamento em um intervalo de tempo aceitável. Para a etapa de migração (Passos 5 a 7), como há menos tarefas para serem efetivamente escalonadas, foi utilizado o PLITD. A escolha desses escalonadores fez os passos de escalonamento e migração levarem $\cong 5$ segundos cada um, em um Pentium 4, 3,2GHz com 2GB de RAM. Os experimentos que avaliam o impacto do intervalo de monitoramento também utilizam o escalonador PLMTC. O PLMTC difere do PLITD pelo fato de considerar a

linha do tempo contínua, o que aumenta o seu tempo de execução apesar da qualidade do escalonamento encontrado ser melhor.

Os recursos da grade simulada nos experimentos formam a mesma topologia que foi apresentada na Seção 3 (Figura 3(a)). As bandas disponíveis entre o host SRC_0 e os hosts $SRC_{\{1...10\}}$ inicialmente valem $100Mbps$ para cada par de hosts. As bandas disponíveis nos enlaces que interligam os hosts $SRC_{\{1...10\}}$ entre si, inicialmente valem $33,33Mbps$ para cada par de hosts. Com relação à taxa de processamento disponível, o host SRC_0 possui $1600MIPS$ e os hosts $SRC_{\{1...10\}}$ possuem $8000MIPS$ cada. A aplicação simulada para executar nessa grade, em todos os experimentos, é a mesma apresentada na Seção 3 (Figura 2). Os pesos das Tarefas 0 e 8 são de $7,68 \times 10^6$ milhões de instruções (MI) e das Tarefas 1 a 7 são de $38,4 \times 10^6$ MI. Cada arco de saída da Tarefa 0 possui peso de 2GB e cada arco de entrada da Tarefa 8 possui peso de 10GB. Os intervalos de tempo relacionados com atividades de E/S internas aos hosts foram abstraídos pelo fato de serem relativamente ínfimos se comparados com o tempo de execução da aplicação.

Nos experimentos em que foram gerados tráfegos de interferência, a presença dos mesmos imprime um caráter de aleatoriedade aos experimentos. Por conta disso, utilizou-se o método de *batch means* para derivação do intervalo de confiança. O critério de parada foi o de obtenção de intervalo de confiança com largura máxima de 5% do valor médio estimado e com nível de 95% de confiança. Os intervalos de confiança foram omitidos, tanto na exibição dos resultados em forma de gráfico quando em forma tabular, para uma melhor interpretação visual.

As subseções a seguir resumem os experimentos realizados. A metodologia foi avaliada sob variações de diversos tipos: mudanças na banda disponível dos enlaces, adição de novos recursos de processamento e mudanças nos intervalos de monitoramento.

6.1. Mudanças na banda disponível dos enlaces

Três experimentos são relatados nesta subseção. O primeiro teve o objetivo de verificar o tempo de execução da aplicação na situação ideal, em que todos os recursos são dedicados à grade. O tempo encontrado neste experimento é usado para comparação com o obtido nos experimentos seguintes. O segundo experimento obtém o tempo de execução da aplicação em um cenário onde enlaces sofrem queda na banda disponível mas os passos de reescalonamento e migração não são realizados. O terceiro experimento é semelhante ao segundo com a diferença de que todos os passos do fluxograma da Figura 4 são executados, a fim de que os ganhos com a utilização da metodologia sejam avaliados.

Para o primeiro experimento, a execução do fluxograma correspondeu à seguinte seqüência: **Passo 1**: O escalonamento encontrado para a aplicação mapeia as tarefas da seguinte forma: $0 \rightarrow SRC_0$, $1 \rightarrow SRC_2$, $2 \rightarrow SRC_5$, $3 \rightarrow SRC_8$, $4 \rightarrow SRC_4$, $5 \rightarrow SRC_1$, $6 \rightarrow SRC_9$, $7 \rightarrow SRC_{10}$, $8 \rightarrow SRC_0$. O início de execução da Tarefa 0 se dá no instante $0min$. As Tarefas 1 a 7 iniciam suas execuções no instante $82,66min$. A Tarefa 8 inicia sua execução no instante $175,96min$. O instante de finalização da execução devolvido pelo escalonador é $255,96min$. É importante notar que nenhuma das tarefas intermediárias (1 a 7) executa no host SRC_0 dado que a capacidade de processamento disponível do mesmo é menor do que a dos demais hosts. É mais rápido transferir os dados de dependência para outro host e executar a tarefa nele do que executá-la no host SRC_0.; **Passo 2**: As tarefas são atribuídas aos hosts e a simulação é iniciada no NS-2. As

transferências das dependências entre as tarefas são realizadas via FTP; **Passos 3 e 4**: Um procedimento no NS-2 verifica o estado dos enlaces e dos hosts de 40 em 40 minutos de forma passiva. Como não há nenhuma variação no estado dos mesmos, já que eles estão dedicados à grade, o escalonamento inicial mantém-se fixo durante toda a simulação. O intervalo grande de monitoramento foi utilizado para mostrar que há ganhos mesmo quando não se monitora o estado da grade com grande freqüência. Em outras palavras, a metodologia é vantajosa apesar de não se utilizar um intervalo ótimo para medições.

Para esse primeiro experimento, o tempo de execução da aplicação no NS-2 foi de $257min$, bem próximo do valor encontrado pelo escalonador no Passo 1. A diferença entre o tempo de execução encontrado pelo escalonador e o devolvido pela simulação, mesmo quando não há mudanças na grade, está relacionado com a imprecisão do modelo implementado pelo escalonador.

O segundo experimento realizado teve como objetivo verificar o tempo de execução da aplicação na situação em que os enlaces não são dedicados exclusivamente à grade, e portanto podem sofrer mudanças nas suas bandas disponíveis. A mudança simulada foi a adição de dois tráfegos de interferência. Um entre os hosts IR_2 e IS_2, e outro entre os hosts IR_5 e IS_5. Esses tráfegos afetam a taxa de transmissão dos hosts SRC_2 e SRC_5 ao host SRC_0. Ambos os tráfegos surgem no instante de tempo $90min$, são compostos por fluxos UDP e transferem dados a uma taxa constante de $90Mbps$. Neste experimento, os passos de reescalonamento e migração (Passos 5 a 7) não foram realizados. Sem verificar a necessidade da migração, as Tarefas 1 e 2 permanecem nos hosts SRC_2 e SRC_5 durante toda a execução. Dessa forma, as transferências de dados das Tarefas 1 e 2 para a Tarefa 8 competem com os tráfegos de interferência, como pode ser notado na Figura 7 (a vazão da Tarefa 2 para a Tarefa 8 comporta-se de forma semelhante), que exibe a vazão devido a transferência de dados de dependência da Tarefa 1. A vazão varia bastante e mantém-se próxima à taxa de $10Mbps$, o que é justificado pelo tráfego UDP de $90Mbps$ que consome a maior parte da banda disponível nos enlaces.

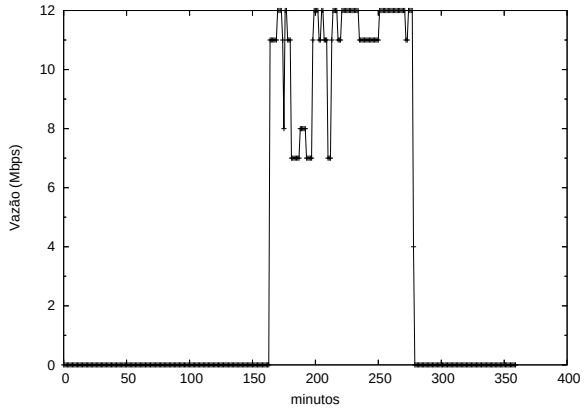


Figura 7. Vazão entre as Tarefas 1 e 8 com tráfego de interferência

Neste segundo experimento, sem a utilização da metodologia completa de Engenharia de Recursos, o tempo de execução no NS-2 foi de $358min$. Comparando esse valor com o alcançado no primeiro experimento, nota-se que as mudanças no estado dos enlaces levaram a um tempo de execução, em média, $\cong$39,3% maior.

O terceiro experimento realizado teve o objetivo de verificar o tempo de execução da aplicação com as mesmas modificações no estado dos enlaces do experimento anterior, com a diferença de que a Engenharia de Recursos foi utilizada por completo. A execução de cada passo da Engenharia de Recursos correspondeu à seguinte seqüência: **Passos 1 e 2**: Igual ao primeiro experimento; **Passos 3 e 4**: O mesmo procedimento descrito no experimento 1 é utilizado para monitorar o estado dos enlaces e dos hosts. Na terceira vez que ele é realizado (no instante igual a $120min$) há a detecção de mudança na banda disponível dos hosts SRC_2 e SRC_5 ao host SRC_0; ela cai de $100Mbps$ para $10,62Mbps$; **Passo 5**: Um novo DAG é construído para a aplicação. Nessa construção, é preciso ter conhecimento da relação entre a quantidade de bytes produzidos pelas tarefas e o tempo de execução das mesmas. Para a aplicação usada no experimento, considera-se que a quantidade de bytes necessários para a migração das tarefas em execução é proporcional ao progresso da execução das tarefas, já que elas produzem os dados de saída de forma proporcional à duração de suas execuções: quanto mais perto de finalizar a execução, uma maior quantidade de bytes precisarão ser enviados na migração de uma tarefa (É importante observar que esse não é um comportamento obrigatório para tarefas de qualquer aplicação nas grades. A consideração realizada é particular a este exemplo). A quantidade de bytes para migração é calculada como $p_i(\sum_{j \in suc(i)} peso(ij) + \sum_{j \in pred(i)} peso(ji))$, onde $p(i)$ é a porcentagem de execução da Tarefa i, $pred(i)$ é o conjunto com as tarefas das quais a Tarefa i depende e $suc(i)$ é o conjunto com as tarefas que dependem de i. As novas tarefas divididas (que representam as tarefas migradas) têm quantidade de instruções igual à quantidade de instruções da tarefa original sendo migrada subtraída da quantidade de instruções que já foram executadas até o instante da migração. Não é considerado *overhead* na quantidade de instruções para as tarefas divididas pois, caso fosse contabilizado, uma tarefa que se mantivesse fixa no mesmo host seria penalizada desnecessariamente. Dessa forma, o DAG modificado no instante de tempo igual a 120 minutos é aquele apresentado na Figura 8.

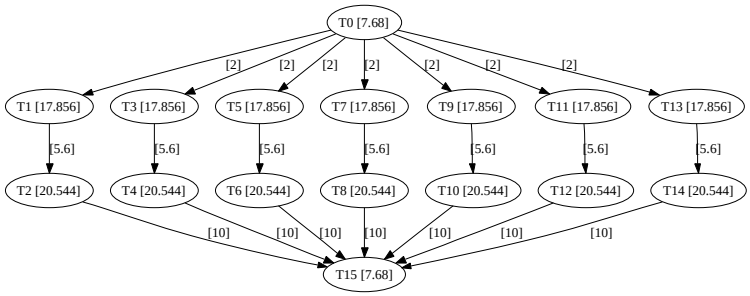
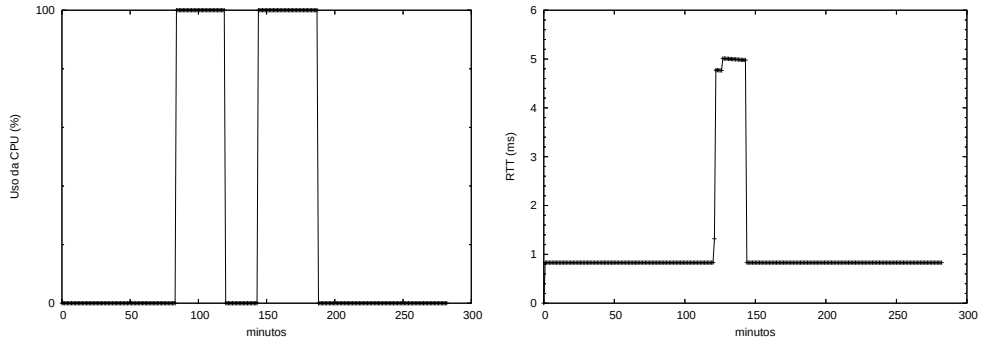


Figura 8. DAG para migração no instante $120min$

Como pode ser notado, as Tarefas 1 a 7 do DAG anterior são divididas, já que no instante $120min$ elas ainda estão em execução e são passíveis de migração. Esse novo DAG e o mapeamento anterior são passados como entrada para o escalonador. Pelo novo escalonamento devolvido, as Tarefas 1 e 2 devem migrar respectivamente para os hosts SRC_3 e SRC_6, a fim de evitar a concorrência do tráfego da aplicação com os tráfegos de interferência. **Passos 6, 7 e 8:** No `NS-2`, as transferências de dados devido às migrações são realizadas no instante $120min$. A Figura 9(a) exibe a interrupção do uso da CPU pela Tarefa 1 entre o instante $120min$ e o instante $143min$ (a interrupção da Tarefa 2 comporta-se de maneira semelhante). O intervalo de tempo em que o uso da CPU é interrompido

coincide com o intervalo em que a migração da Tarefa 1 é realizada do host SRC_2 para o host SRC_3, como pode ser notado no gráfico da Figura 9(b) que exibe o RTT medido entre os hosts SRC_2 e SRC_3. O aumento do RTT entre $120min$ e $143min$ é causado pela transferência dos dados da migração.



(a) Uso da CPU pela Tarefa 1 (b) RTT entre os hosts SRC_2 e SRC_3

Figura 9. Migração da Tarefa 1

Com a realização da migração neste experimento, o tempo de finalização da aplicação no NS-2 foi de $281min$. Comparando este tempo com o que foi encontrado no experimento anterior ($358min$), nota-se que, com a metodologia, houve um ganho médio de ≅21,51% no tempo de execução da aplicação. Comparando o tempo de execução com o primeiro experimento, nota-se que, com a utilização da Engenharia de Recursos, a aplicação executa em um tempo somente ≅9,34% pior, em média, do que no caso em que não houve mudanças no estado dos enlaces.

6.2. Adição de novos recursos de processamento

Nesta subseção, são relatados os experimentos realizados com o objetivo de avaliar o desempenho da Engenharia de Recursos quando são incluídos novos hosts na grade. Nos experimentos, a única modificação na grade ocorre no instante 90min, quando são incluídos 10 novos hosts. Cada um deles tem um enlace ligado a um dos hosts *SRC1* a *SRC10* e outro enlace ligado ao host *SRC0*. A Figura 10 exibe a inclusão do host *SRC16* ligado a *SRC6* (a topologia final completa não é exibida a fim de facilitar a visualização). A taxa de processamento disponível nos novos hosts é de 8000MIPS e a banda disponível nos seus enlaces é de 1Gbps.

Nesse experimento, a metodologia foi utilizada (o monitoramento foi realizado de 40 em 40 minutos) e, no instante 120min, as tarefas 1 a 7 migraram para os hosts que foram incluídos na grade. A aplicação finalizou a sua execução no instante 247min, um tempo inferior ao alcançado sem a inclusão dos novos hosts, que foi 257min.

Na verificação dos ganhos alcançados com a migração, os escalonadores avaliam tanto o estado dos enlaces quanto dos hosts. Para ilustrar esse ponto, o seguinte experimento foi realizado: com as mesmas inclusões descritas na Figura 10, mas com as taxas de processamento disponível dos novos hosts iguais a 4000MIPS, ou seja, uma taxa que vale a metade da considerada anteriormente, os escalonadores não propuseram migrações. A fim de avaliar se a não realização das migrações foi a melhor ação tomada, as migrações foram forçadas de forma manual e o tempo de execução da aplicação foi

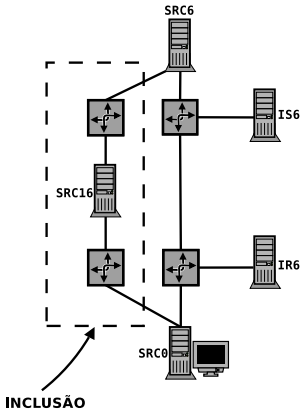
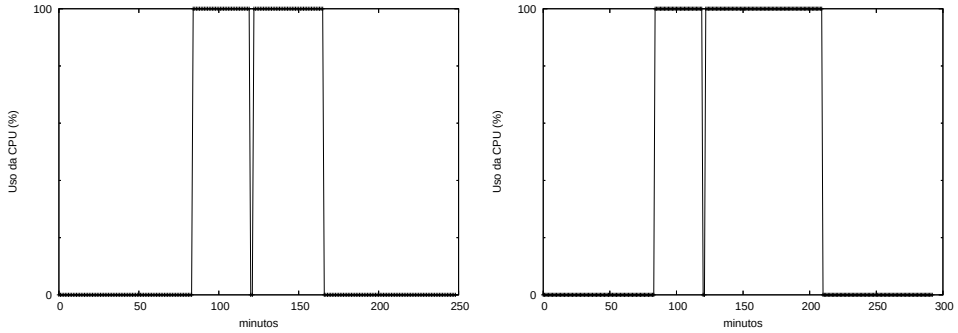


Figura 10. Inclusão do novo host ligado a *SRC6*

de 291min, um tempo maior do que o alcançado caso a migração não tivesse sido realizada (257min), o que comprova que os escalonadores forneceram a melhor opção ao não propor as migrações das tarefas. A Figura 11 apresenta dois gráficos que servem para comparar o tempo de execução da tarefa 1 quando ela migrou para o novo host com taxa de 8000MIPS e quando ela não seguiu a recomendação do escalonador e migrou para o novo host com taxa de 4000MIPS.



(a) Host com 8000MIPS (b) Host com 4000MIPS

Figura 11. Uso da CPU pela tarefa 1 ao migrar para hosts com diferentes taxas de processamento

Quando a migração é feita para o host com taxa de 8000MIPS, a execução da Tarefa 1 finaliza no instante 165min (gráfico da Figura 11(a)). Quando a mesma tarefa é migrada para o host com taxa de 4000MIPS, a sua execução finaliza no instante 209min (gráfico da Figura11(b)), o que faz com que a tarefa de saída do DAG atrase o início de sua execução e aumente o tempo de execução da aplicação.

6.3. Mudanças no instante de monitoramento

Os últimos experimentos realizados ilustram a importância que o intervalos de monitoramento têm no ganho obtido com a utilização da Engenharia de Recursos. Isso se deve ao fato da consideração de que a quantidade de bytes a serem transmitidos para uma tarefa migrar está relacionada com o tempo de execução da aplicação. Além da importância dos instantes de monitoramento, os experimentos desta subseção também avaliam os escalonamentos propostos por dois escalonadores diferentes. A mesma relação entre quantidade

de bytes para migração e tempo de execução das tarefas feita no terceiro experimento da Subseção 6.1 é considerada para as tarefas neste experimento.

Foram realizados 12 experimentos com a utilização da metodologia. Para metade dos experimentos, o Passo 5 da engenharia de tráfego foi executado com o escalonador PLMTC. Na outra metade, foi utilizado o PLITD. Para cada escalonador utilizado, o monitoramento do estado da grade foi realizado uma única vez nos instantes 100min, 110min, 120min, 130min, 140min e 150min. Para todos os experimentos gerou-se, no instante 90min, tráfegos de interferência entre os mesmos hosts dos experimentos relatados na Subseção 6.1 (entre os hosts *IR2* e *IS2*, e entre os hosts *IR5* e *IS5*), o que tornou as tarefas 1 e 2 passíveis de migração. A taxa dos tráfegos de interferência foi de 60Mbps.

A fim de ter um parâmetro para comparação, a aplicação foi executada inicialmente sem a utilização da engenharia de tráfego. Nesse caso, nenhuma tarefa migrou e o tempo de execução da aplicação foi de 276min.

A Tabela 1 apresenta os tempos de execução da aplicação para cada um dos experimentos realizados. Para cada instante do monitoramento, é apresentado o tempo de execução da aplicação de acordo com as sugestões de reescalonamento dadas pelos escalonadores PLMTC e PLITD. Para cada tempo de execução, é apresentada entre parênteses a informação sobre a realização ou não de migração das tarefas 1 e 2 de acordo com o escalonamento devolvido pelos escalonadores.

Tabela 1. Tempos de execução para diferentes instantes de monitoramento (minutos)

Instante	PLMTC	PLITD
100	269 (com migração)	269 (com migração)
110	275 (com migração)	275 (com migração)
120	276 (sem migração)	281 (com migração)
130	276 (sem migração)	287 (com migração)
140	276 (sem migração)	276 (sem migração)
150	276 (sem migração)	276 (sem migração)

Pelos resultados, nota-se que a migração (das tarefas 1 e 2) somente é sugerida pelo PLMTC quando o instante de medição é menor que 120min. A partir desse valor, o escalonador não sugeriu migração e a aplicação terminou sua execução nos hosts inicialmente mapeados. Já o PLITD só não sugeriu migração quando o instante de monitoramento foi maior que 130min, o que foi uma decisão ruim, já que a aplicação levou mais tempo para executar ao seguir as sugestões de migração nos instantes 120min e 130min. Essa decisão ruim deve-se às aproximações do algoritmo implementado pelo PLITD.

A fim de comparar a vantagem alcançada quando não se realiza as migrações de tarefas nos instantes de monitoramento 140 e 150, a aplicação foi executada e a migração das tarefas foi realizada, apesar dos escalonadores não as terem sugerido. Nesses casos, o tempo de execução da aplicação foi, respectivamente, 293min e 299min, o que mostra que as decisões dos escalonadores de não migrar as tarefas foi correta, já que o tempo de execução da aplicação manteve-se em 276min nos dois casos.

6.4. Resumo conclusivo dos experimentos realizados

Pelos experimentos realizados para avaliar o desempenho da Engenharia de Recursos para Grades, nota-se que metodologia é vantajosa. As decisões tomadas, tanto em situações em que há variações nas bandas disponíveis dos enlaces quanto adição de novos recursos, diminuem o tempo de execução da aplicação, atendendo assim o objetivo esperado.

Pôde-se notar ainda a importância de se utilizar um escalonador que dê resultados os mais precisos possíveis, sendo que o PLMTC é uma boa opção de escalonador a ser usado no reescalonamento e migração caso hajam poucas tarefas passíveis de migração. Nota-se, também, que o intervalo de monitoramento influencia nas chances de que migrações dêem bons resultados, já que a partir de um certo valor, o tempo gasto com a migração da tarefa deixa a aplicação mais lenta do que se não fosse feita migração.

7. Trabalhos relacionados

Assim como a metodologia apresentada neste artigo, as propostas apresentadas em [Allen et al. 2001], [Vadhiyar and Dongarra 2003] e [Huedo et al. 2002] baseiam-se na execução periódica das etapas de monitoramento, reescalonamento e migração, com o objetivo de contornar quedas no desempenho da execução de aplicações nas grades.

Em [Allen et al. 2001], o monitoramento e a decisão para a migração de tarefas são implementados levando em consideração dois casos: a inclusão de um novo recurso que melhore o desempenho da execução das aplicações ou a violação de contrato. Assim como apresentado na nossa metodologia, os mecanismos implementados consideram a queda no desempenho da rede como motivo para migrar tarefas, apesar de não serem apresentados experimentos que comprovem essa consideração. Uma diferença desta proposta para a apresentada no presente artigo é que, caso a migração seja necessária, os dados para a tarefa continuar sua execução podem ser enviados para um local de armazenamento antes de seguirem para o novo recurso selecionado, o que pode vir a gerar um gargalo no funcionamento do sistema. Em compensação, com essa abordagem é possível migrar tarefas entre recursos sem ligação direta entre si.

No esquema em [Huedo et al. 2002], a migração das tarefas pode ocorrer em duas situações: mudança no estado dos recursos ou mudança nos requisitos das aplicações. Assim como na nossa metodologia, mudanças no estado da rede são consideradas ao avaliar a necessidade de migração das tarefas, porém o único tipo de mudança considerado é a desconexão dos recursos. Com relação ao escalonamento, é implementado um algoritmo guloso, que fornece o primeiro escalonamento de forma rápida mas que pode implicar na realização de muitas migrações posteriormente, a fim de melhorar o escalonamento.

Em [Vadhiyar and Dongarra 2003], a etapa de migração é a mais focada e ela pode ocorrer nos mesmos casos de [Allen et al. 2001]. Assim como na nossa abordagem, o monitoramento do progresso e a migração de tarefas são tratadas como duas ações a serem realizadas de forma acoplada, a fim de evitar que migrações sejam realizadas quando uma tarefa estiver perto de finalizar sua execução. As migrações só são realizadas caso o ganho a ser alcançado seja superior a 30%. Os próprios autores reconhecem que manter esse valor fixo não é a abordagem ideal. Para o cálculo do custo de migração para o novo recurso, é utilizado um *overhead* fixo que foi coletado com experimentos em ambientes reais, o que não significa que sejam válidos para todas as aplicações. Este cálculo diferen-

cia esta proposta da nossa, já que consideramos que o *overhead* na migração não é fixo e depende do progresso de execução das tarefas.

8. Conclusões e Trabalhos Futuros

Este artigo apresentou uma metodologia para reajustar a alocação dos recursos utilizados por uma aplicação de modo a manter o tempo de execução baixo e robusto frente à dinâmica das grades. A principal contribuição desta metodologia, e o que a diferencia das demais encontradas na literatura, é o fato dela considerar o custo de comunicação causado pelas transferências de dados tanto no escalonamento quanto na migração das tarefas, sendo que o custo para este último não é fixo e depende do progresso de execução das tarefas. Uma contribuição adicional é o simulador `GridSim-NS`, um simulador de grades desenvolvido como extensão para o simulador `NS-2`. A relevância da metodologia está no fato dela auto-ajustar o escalonamento das aplicações como resposta às variações no estado dos recursos de processamento e de comunicação, sem intervenção humana, o que afeta diretamente a automatização e a difusão das grades.

A aplicação da metodologia em vários cenários de simulação mostrou que a mesma consegue ganhos ao migrar as tarefas em situações em que há variações nas bandas disponíveis dos enlaces ou quando há inclusão de recursos com melhor capacidade do que os atuais na grade. Os resultados das simulações também mostram a importância do monitoramento freqüente dos recursos e da utilização de bons escalonadores de tarefas.

Como trabalho futuro, pretende-se realizar a análise de características relacionadas com o tráfego gerado por aplicações nas grades, a fim de se propor modelos que melhor representem as transferências de dados nestes ambientes. Pretende-se, também, realizar um estudo com o objetivo de definir dinamicamente o melhor intervalo de medições a ser utilizado no monitoramento do estado das grades. Um trabalho que encontra-se em andamento é a inclusão, na metodologia, de passos responsáveis por tratar as incertezas presentes nos pesos dos DAGs.

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(4):345–358.

Awduche, D., Chiu, A., Elwalid, A., Widjaja, I., and Xiao, X. (2002). RFC 3272: Overview and Principles of Internet Traffic Engineering.

Batista, D. M. (2006). Engenharia de Tráfego Multi-Camada para Grades. Dissertação de Mestrado, Instituto de Computação – Unicamp. Páginas 74 a 84. Disponível em `http://www.ic.unicamp.br/~batista/dissertacao.pdf`. Último acesso em 30/11/2006.

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.

Batista, D. M., da Fonseca, N. L. S., and Miyazawa, F. K. (2007). A Set of Schedulers for Grid Networks. In *SAC'07: Proceedings of the 2007 ACM Symposium on Applied Computing (Aceito para publicação)*.

Cannataro, M., Mastroianni, C., Talia, D., and Trunfio, P. (2003). Evaluating and Enhancing the Use of the GridFTP Protocol for Efficient Data Transfer on the Grid. *Lecture Notes in Computer Science*, 2840:619 – 628.

CERN (2005). LCG - LHC Computing Grid Project. Disponível em `http://lcg.web.cern.ch/LCG/`. Último acesso em 30/11/2006.

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

Huedo, E., Montero, R. S., and Llrorent, I. M. (2002). An Experimental Framework for Executing Applications in Dynamic Grid Environments. Technical Report 2002-43, NASA Langley Research Center.

ISI (2004). The Network Simulator – ns-2. Disponível em `http://www.isi.edu/nsnam/ns/`. Último acesso em 30/11/2006.

Kamous-Edwards, G. and Jukan, A. (2006). An Optical Control Plane for the Grid Community: Opportunities, Challenges, and Vision. *IEEE Communications Magazine*, 44(3):62–63.

Ma, D. and Zhang, W. (2004). A Static Task Scheduling Algorithm in Grid Computing. *Lecture Notes in Computer Science*, 3033:153 – 156.

Renambot, L., van der Schaaf, T., Bal, H. E., Germans, D., and Spoelder, H. J. W. (2003). Griz: Experience with Remote Visualization over an Optical Grid. *Future Generation Computer Systems*, 19(6):871–882.

Roy, A. and Livny, M. (2003). Condor and Preemptive Resume Scheduling. In *Grid Resource Management: State of the Art and Future Trends (1st edition)*, pages 135–144.

SBC (2006). Grandes Desafios da Pesquisa em Computação no Brasil - 2006 – 2016. `http://www.sbc.org.br/index.php?language=1&subject=8&content=downloads&id=231`. Último acesso em 15/03/2007.

Schopf, J. M. (2003). Ten Actions when Grid Scheduling. In *Grid Resource Management: State of the Art and Future Trends (1st edition)*, pages 15–23.

Silvester, J. A. (2005). CalREN: Advanced Network(s) for Education in California. Páginas 39 a 41. Disponível em `http://isd.usc.edu/~jsilvest/talks-dir/20051021-cudi-merida.pdf`. Último acesso em 30/11/2006.

Sinnen, O. and Sousa, L. A. (2005). Communication Contention in Task Scheduling. *IEEE Transactions on Parallel and Distributed Systems*, 16(6):503–515.

Sulistio, A., Poduval, G., Buyya, R., and Tham, C.-K. (2005). Constructing a Grid Simulation with Differentiated Network Service Using GridSim. In *Proceedings of the 6th International Conference on Internet Computing (ICOMP'05)*.

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.

Wolski, R., Spring, N. T., and Hayes, J. (1999). The Network Weather Service: a Distributed Resource Performance Forecasting Service for Metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768.