MINISTÉRIO DA DEFESA
EXÉRCITO BRASILEIRO
DEPARTAMENTO DE CIÊNCIA E TECNOLOGIA
INSTITUTO MILITAR DE ENGENHARIA
CURSO DE MESTRADO EM SISTEMAS E COMPUTAÇÃO

HENRIQUE DE MEDEIROS KLÔH

MODELO DE ESCALONAMENTO BI-CRITÉRIOS DE WORKFLOW EM GRADES

Rio de Janeiro
2010

INSTITUTO MILITAR DE ENGENHARIA

HENRIQUE DE MEDEIROS KLÔH

MODELO DE ESCALONAMENTO BI-CRITÉRIOS DE WORKFLOW EM GRADES

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientadores:
Prof. Bruno Richard Schulze - D.Sc
Prof. Raquel Coelho Gomes Pinto - D.Sc

Rio de Janeiro
2010

INSTITUTO MILITAR DE ENGENHARIA
Praça General Tibúrcio, 80-Praia Vermelha
Rio de Janeiro-RJ CEP 22290-270

005.75 Klôh, Henrique de Medeiros
K66a Modelo de Escalonamento Bi-critérios de Workflow em Grades/ Henrique de Medeiros Klôh. - Rio de Janeiro: Instituto Militar de Engenharia, 2010.

68 p.: il.

Dissertação (mestrado) – Instituto Militar de Engenharia – Rio de Janeiro, 2010.

1. Workflows. 2. Algoritmo de escalonamento.
I. Título. II. Instituto Militar de Engenharia.

CDD 005.75

INSTITUTO MILITAR DE ENGENHARIA

HENRIQUE DE MEDEIROS KLÔH

MODELO DE ESCALONAMENTO BI-CRITÉRIOS DE WORKFLOW EM GRADES

Dissertação de Mestrado apresentada ao Curso de Mestrado em Sistemas e Computação do Instituto Militar de Engenharia, como requisito parcial para obtenção do título de Mestre em Sistemas e Computação.

Orientadores: Prof. Bruno Richard Schulze - D.Sc
Prof. Raquel Coelho Gomes Pinto - D.Sc

Aprovada em 6 de agosto de 2010 pela seguinte Banca Examinadora:

Prof. Raquel Coelho Gomes Pinto - D.Sc do IME - Presidente

Prof. Bruno Richard Schulze - D.Sc do LNCC

Prof. Antônio Tadeu Azevedo Gomes - D.Sc do LNCC

Prof. Maria Claudia Reis Cavalcanti - D.Sc do IME

Rio de Janeiro
2010

"*Pensar é o trabalho mais pesado que há, e, talvez, seja essa a razão para tão poucas pessoas se dediquem a tal tarefa.*"

HENRY FORD

SUMÁRIO

LISTA DE ILUSTRAÇÕES

RESUMO

O trabalho a seguir apresenta uma proposta de um modelo de escalonamento bi-critérios de Workflows em Grades que tem por base um algoritmo de escalonamento de *Workflows* híbrido bi-critérios com suporte a CoS. O modelo proposto tem por objetivo otimizar os critérios escolhidos pelos usuários e com base na ordem de prioridade por eles especificados. A fim de validar este modelo foi realizado um conjunto de testes que comparam o desempenho do modelo proposto com uma política de escalonamento do tipo *Join the Shortest Queue* (JSQ). Em função dos testes realizados e dos resultados obtidos foi possível observar um ganho de desempenho em termos de tempo, custo e confiabilidade, e uma melhoria na qualidade de serviços ao atender a priorização de critérios definida pelos usuários.

ABSTRACT

The work presented shows a workflows scheduling model on Grids which is based on a hybrid bi-criteria scheduling algorithm with support for Class of service (CoS). The proposed model aims to optimize the criteria chosen by the users, based on their specified priority order. In order to validate this model, it was performed a set of tests that compare the performance of the proposed model with the Join the Shortest Queue (JSQ) scheduling algorithm. The results of the test showed an improvement in the quality of services, as they met the priority criteria set by users, and a performance gain with the use of the proposed algorithm.

1 INTRODUÇÃO

A complexidade com que se revestem os problemas da atualidade, sobretudo aqueles ligados ao estudo de genoma, simulações meteorológicas, médicas e astronômicas, exigem cada vez mais análises interdisciplinares com o conhecimento de áreas tais como física, geofísica, biologia, astronomia, entre outros. Os dados de entrada necessários à solução de problemas são tratados por aplicações e os resultados obtidos podem servir de entrada para outras aplicações, obtendo-se assim um fluxo onde o resultado final será a resolução do problema em questão.

Um *workflow* pode ser visto como um conjunto de aplicações e dados relacionados entre si, onde tarefas são processadas em uma ordem bem definida a fim de cumprir uma meta específica (YU e BUYYA, 2005). *Workflows* científicos têm se tornado cada vez mais necessários na computação científica devido a complexidade, dependência e necessidade de paralelização e sequenciamento do processamento.

Um grande número de cientistas e pesquisadores se utilizam de sistemas de *workflows* para a realização dos seus experimentos. Porém esse tipo de sistema, em geral, necessita de uma maior demanda computacional e o uso de sistemas distribuídos surge como uma solução eficaz.

Os sistemas distribuídos propiciam aos seus usuários um grande poder computacional através da agregação de diversos computadores. As Grades Computacionais são uma classe de sistemas distribuídos que se diferenciam dos demais por permitir um alto grau de heterogeneidade em relação a *hardware*, sistemas operacionais, redes e domínios administrativos. Grades têm por foco o compartilhamento em larga escala de recursos heterogêneos distribuídos geograficamente e o alto desempenho (FOSTER, 2001).

Em se tratando de *workflows*, as Grades computacionais têm sido utilizadas como principal forma de prover a capacidade computacional necessária, pois oferecem uma maior capacidade de processamento com baixo custo e permitem que diversos centros de pesquisa possam trabalhar de forma colaborativa, compartilhando seus recursos e dados.

Apesar da utilização anteriormente descrita, um dos principais problemas encontrados neste ambiente é quanto ao escalonamento de aplicações, principalmente, aplicações *workflow* (WIECZOREK et al., 2008)(WIECZOREK et al., 2009)(MEYER et al., 2006).

A escolha dos recursos a serem alocados a esse tipo de aplicação otimizando o seu uso e permitindo a execução de *workflows* em um ambiente de Grade é uma tarefa complexa, assim como a própria submissão.

O escalonamento de aplicações é tratado como um problema NP-Completo(EL-REWINI et al., 1994), quando consideramos *workflows* esta complexidade é ainda maior, uma vez que sua complexidade cresce mediante o aumento do número de variáveis, nesse caso, os recursos e a quantidade de tarefas e dados que formam o *workflow*. Essa situação torna-se ainda mais desafiadora em ambientes de Nuvem e Grade, por suas características únicas anteriormente citadas. Os algoritmos de escalonamento em sistemas distribuídos que utilizam usualmente recursos homogêneos e dedicados, podem não apresentar resultados satisfatórios nesses ambientes (DONG, 2009).

1.1 JUSTIFICATIVA

O escalonamento de tarefas em Grades é um problema complexo que pode exigir que sejam levados em conta diferentes critérios. Os critérios mais usados atualmente são: o tempo de execução, o custo de execução da tarefa, a confiabilidade e qualidade das informações (WIECZOREK et al., 2008). Uma abordagem multi-critérios, apesar de permitir a agregação de informações relevantes a seleção do recurso, pode aumentar significativamente a complexidade do escalonamento e, dependendo da quantidade de critérios utilizados, impossibilitar o seu uso dado o tempo necessário à sua solução. Por esse motivo, neste trabalho foi escolhida uma abordagem que atenda a mais de um critério e seja capaz de trabalhar com os mesmos de forma **independente e cooperativa**, otimizando inicialmente um critério e, baseado no resultado inicialmente obtido, otimiza o próximo critério em função de uma condição de relaxação pré-estabelecida para o primeiro critério.

Ainda merecem destaque neste trabalho: a redução do tempo de resposta ao usuário, a melhor utilização dos recursos do ambiente de Grade sem alterar o *makespan* das aplicações, permitir que o usuário possa executar o seu *workflow* em um ambiente de Grade sem se preocupar com a escolha dos melhores recursos.

1.2 OBJETIVOS

O planejamento e o desenvolvimento deste trabalho têm como objetivo atingir a seguinte meta:

- Apresentar um modelo de escalonamento que seja capaz de otimizar a submissão e execução de aplicações em um ambiente de Grade, assim como a utilização dos recursos deste ambiente por meio de uma abordagem bi-critérios com CoS (Classes de serviços) onde inicialmente trata da relação usuários/recursos definindo os principais critérios para o usuário através de um acordo em nível de serviço SLA.

Como forma de verificar o objetivo acima citado, serão realizadas comparações para comprovar se o modelo proposto demonstrou bom desempenho ao ser comparado com algumas das principais abordagens de escalonamento já existentes.

Os testes para validação do modelo de escalonamento proposto foram realizados através do ambiente de simulações GridSim (BUYYA e MURSHED, 2002). O simulador foi alterado a fim de proporcionar o ambiente necessário para a realização dos testes.

Este trabalho propõe uma abordagem para o problema de escalonamento de *workflows* em ambientes de Grades que leva em consideração as características específicas destes ambientes e das aplicações *workflows*, tais como: a necessidade de transferência dos dados intermediários das aplicações que compõem o *workflow*, redução de custo e identificação do nível de confiabilidade dos recursos.

O texto deste trabalho se encontra organizado da seguinte forma: o Capítulo 2 apresenta as principais características dos Sistemas Distribuídos, com um maior foco em Grades. Neste capítulo também são apresentados alguns conceitos de aplicações e serviços de sistemas distribuídos que foram considerados neste trabalho; o Capítulo 3 apresenta trabalhos encontrados na literatura que propõem modelos de escalonamento para *workflows* científicos em ambientes de Grades; o Capítulo 4 descreve a implementação do modelo proposto, bem como a definição dos critérios utilizados pelo mesmo; o Capítulo 5 apresenta ainda os experimentos realizados e os resultados obtidos, bem como os comentários sobre os mesmos; por fim, o Capítulo 6 apresenta as considerações finais acerca deste trabalho.

2 SISTEMAS E APLICAÇÕES DE COMPUTAÇÃO DISTRIBUÍDA

Este capítulo apresenta as principais características das Grades e Nuvens que estão relacionadas com a realização deste trabalho. São apresentadas também as principais arquiteturas de sistemas distribuídos, enfatizando as características das Grades.

2.1 *GRADES*

Os *clusters* apesar de atenderem as necessidades dos usuários, principalmente quanto a relação custo/capacidade de processamento, em muitas das soluções propostas possuem dependência de um nó centralizador, além da necessitarem estar próximos geograficamente, preferencialmente em uma mesma rede, o que poderia representar uma limitação em termos de espaço físico.

As Grades Computacionais (*Grid Computing*) surgiram como uma proposta de estenderem a arquitetura dos *clusters* para um ambiente totalmente disperso geograficamente e com recursos heterogêneos. Uma Grade possui uma arquitetura totalmente distribuída onde a falha de um nó não compromete o funcionamento de todo o ambiente. Além de ser formado por recursos de pequeno porte e heterogêneos, os recursos da Grade podem estar dispersos geograficamente e interligados através de uma rede de alta velocidade.

Os grandes desafios a serem trabalhados em Grades são quanto a coordenação e a utilização dos seus recursos, uma vez que esse tipo de ambiente pode ser formado pela união de diversos sítios que encontram-se geograficamente distribuídos. As tecnologias existentes para computação distribuída possuem poucas ferramentas que auxiliem neste problema, por exemplo, a *internet* provê a comunicação entre os domínios, mas não fornecem abordagens para a utilização coordenada de recursos em múltiplos domínios.

Não existe, então, em uma Grade, uma administração única. Cada organização que faz parte da Grade é responsável pela gerência e manutenção dos recursos que compõem o seu sítio.

Neste tipo de ambiente colaborativo, podemos identificar três camadas bem definidas:

- Infra-estrutura

  Essa camada é formada pelos recursos de *software* e de *hardware* que compõe a Grade, interligados através de uma rede física.

- *Middleware*

  O *Middleware* é um conjunto de protocolos, bibliotecas e regras responsável por padronizar e gerenciar a comunicação e informações da Grade. Ele provê uma abstração de mais alto nível dos recursos tornando transparente a utilização do ambiente apesar de ser formado por recursos heterogêneos e estarem dispersos geograficamente.

- Aplicações

  São os aplicativos desenvolvidos e otimizados para melhor utilizar os recursos da Grade.

O ambiente de Grade apesar de se apresentar como uma solução para sistemas distribuídos de forma eficiente quanto ao compartilhamento de dados e recursos, com menor custo, possui limitações do ponto de vista dos usuários quanto a dificuldade de sua utilização e a segurança. A segurança em Grades tem sido aprimorada a cada dia, porém os usuários do ambiente, geralmente, não consideram seguro alocar seus dados e aplicações em recursos localizados em domínios administrativos diferentes, sobre diferentes regras.

2.2 COMPUTAÇÃO EM NUVEM

A Computação em Nuvens (*Cloud Computing*) é um dos mais novos conceitos existentes na computação. O conceito de *Cloud* ou computação nas "nuvens" é, basicamente, a utilização de serviços, aplicações ou recursos disponíveis através da *internet*.

O surgimento da Nuvem foi possível com a evolução das atuais conexões de *internet*, que possibilitam *links* de alta velocidade, e com base em conceitos mais antigos como o de *Utility Computing* e *Software as a Service* (SaaS). O principal conceito apresentado em *Utility Computing* é que a computação se torne disponível como um serviço a ser pago em função do seu uso, assim como luz, água, etc, onde algumas empresas são responsáveis por prover o serviço e os usuários pagam pelo que consumirem deste serviço.

SaaS ou *Software* como Serviço, trata-se de uma forma de trabalho onde o usuário não precisa adquirir o *software* ou a licença do mesmo, instalá-lo ou até mesmo adquirir servidores para o mesmo. Neste modelo paga-se somente uma taxa pelo *Software* utilizado ou pelo tempo de uso.

Esses dois conceitos formam o alicerce do ambiente em Nuvem. Na computação em "nuvens", tem-se então, recursos e aplicações disponíveis através da *internet* onde os usuários pagam pelos recursos consumidos ou tempo de utilização. Existem também aplicações em Nuvem em que não há custos pela sua utilização, como é o caso da ferramenta da Google "Google Docs". Logo, neste ambiente o usuário apenas desfruta dos serviços oferecidos sem se preocupar com o desenvolvimento, armazenamento, manutenção, *backup* e escalonamento, entre outros.

Dentro desta estrutura, onde muitos dos seus recursos podem encontrar-se dispersos em termos de domínios ou geográficos, há uma grande massa de dados que trafega entre os mesmos. Os impactos deste tráfego e a sua distribuição por diversos domínios é hoje considerado um dos elementos prioritários a serem tratados seja sob o ponto de vista legal, comercial ou em termos de desempenho. Quanto a este último aspecto, se considerarmos o custo da comunicação em aplicações que necessitam de alto desempenho, o uso de Nuvens, em muitos casos, poderá se apresentar como um fator restritivo. Este trabalho apresenta no Capítulo 4 uma abordagem que busca minimizar o efeito do custo da comunicação por meio do agrupamento de tarefas.

A contribuição que este novo paradigma tem a oferecer na área comercial passa ainda por um certo receio no que concerne a segurança e controle dos dados, distribuídos por esta Nuvem. Outra preocupação, é quanto a disponibilidade dos recursos e destes dados. Apesar de estarem cobertos por acordos de serviços, há ainda um certo temor quanto a real capacidade desta estrutura garantir total acesso não só aos recursos mas também aos dados lá armazenados. Quando olhamos para o emprego em apoio a aplicações científicas fica clara a contribuição que há na possibilidade de escalar os recursos, sejam próprios ou contratados temporariamente das chamadas Nuvens públicas, para a solução de problemas específicos, reduzindo assim a necessidade de investimento e a subutilização de recursos.

Na área científica, o conceito de Nuvem tem sido amplamente estudado em função desta capacidade de prover recursos sob medida para os usuários. Os usuários podem alocar suas aplicações definindo os recursos necessários para as mesmas pagando somente o que for utilizado, sem se preocupar com a manutenção dos recursos e do ambiente,

utilizando-os no tamanho exato das suas necessidades, seja na forma de *Clusters* ou de plataforma com aplicações dedicadas.

Os sistemas de Nuvem têm por base o uso de recursos virtualizados e com isto é possível responder à demanda, passando a idéia de recursos ilimitados. Isto é possível através do uso de máquinas virtuais, criadas mediante as especificações definidas pelos usuários e alocadas nos recursos reais.

O ambiente de Computação em Nuvens tem sido o foco das pesquisas atuais por ser uma nova frente de trabalho ainda em aberto. Apesar disto, com o alto custo dos recursos e a dificuldade de administração destes, o ambiente de Nuvens já tem conquistado os usuários, pois tem provido serviços com baixo custo sem que o usuário precise se preocupar com a administração dos mesmos como mostrado em (ARMBRUST et al., 2009).

2.3 *WORKFLOWS*

Os problemas com que a sociedade moderna se depara nos dias atuais necessitam de modelos complexos e interdependentes cuja solução depende cada vez mais de alta capacidade computacional. Este problema tem sido encontrado atualmente não só nas áreas de exatas como em todas as demais áreas, principalmente nas pesquisas que envolvem os mais recentes problemas conhecidos pela humanidade. Muitos destes problemas não são solucionados ou simulados por apenas uma aplicação isolada, mas sim por um conjunto encadeado de aplicações. Quando estas aplicações trabalham de forma cooperativa estabelecendo um fluxo de trabalho, damos o nome de *Workflow* (DONG, 2009).

Aplicações *workflows*, ou fluxo de trabalho, têm sido amplamente utilizadas na computação científica para solução dos diversos problemas computacionais existentes. *Workflows* são formados por um conjunto de tarefas que possuem relações entre si. Estabelecendo assim, um fluxo cujos dados de saída se tornam os dados de entrada para o estágio seguinte, até gerar a solução final.

2.3.1 MODELOS DE WORKFLOWS

Os *workflows* podem possuir diferentes níveis de dependências e relacionamentos entre as tarefas que o compõem. Através destas variações, é possível classificar os *workflows* por suas estruturas, componentes e também quanto ao seu dinamismo.

2.3.1.1 ESTRUTURA

Quanto a sua estrutura, os *workflows* podem ser classificados em três principais conjuntos:

- DAG (*Directed Acyclic Graph*)

  Os *workflows* DAG formam com suas tarefas e dependências grafos direcionais e acíclicos, onde cada tarefa corresponde a um nó desse grafo e cada aresta corresponde a uma dependência. As arestas são sempre direcionais e não formam ciclos, estabelecendo assim, que não há dependências recíprocas e formando um sequenciamento bem definido entre as tarefas que formam o *workflow*, um exemplo deste modelo é apresentado na Figura 2.1.

- *Extended Digraph*

  Os *workflows Extended Digraph* são semelhantes aos DAGs, porém possibilitam a criação de ciclos e de condições no grafo. As condições, neste caso, definem o caminho que os dados de saída de uma dada tarefa irão tomar ou até mesmo se eles devem ser enviados para uma tarefa executada anteriormente, criando um ciclo. Por exemplo, após o término da tarefa é feita uma verificação: se for a primeira execução da mesma, o dado é enviado para a tarefa que foi executada anteriormente em relação a tarefa em questão e o fluxo composto por ambas é reexecutado. Ao término da tarefa é refeita a verificação, neste instante, é a segunda execução da mesma. Os dados gerados seguem então outro caminho a fim de prosseguir com a execução do *workflow*, um exemplo deste modelo é apresentado na Figura 2.1.

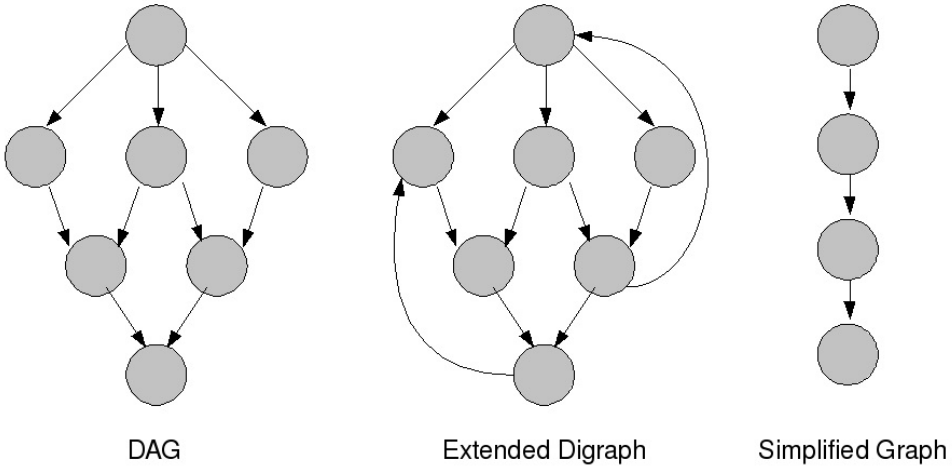


FIG. 2.1: Tipos de estruturas de *workflows*

- *Simplified Graph*

  Os *Workflows Simplified Graph* podem ser definidos como um subconjunto de DAGs com padrões mais simples e mais bem definidos. Exemplos desse tipo de *workflows* são descritos abaixo e mostrados na Figura 2.2:

  – Paralelo: São *workflows* compostos por aplicações que dividem o processamento entre si, trabalhando de forma paralela. Geralmente, esse modelo possui uma aplicação que divide o processamento pelas demais e ao término, recebe e unifica os resultados.

  – Árvore (*Tree-like*): As aplicações formam uma árvore como estrutura.

  – Sequencial (*Pipeline*): As aplicações estabelecem um fluxo único de trabalho, como uma fila.

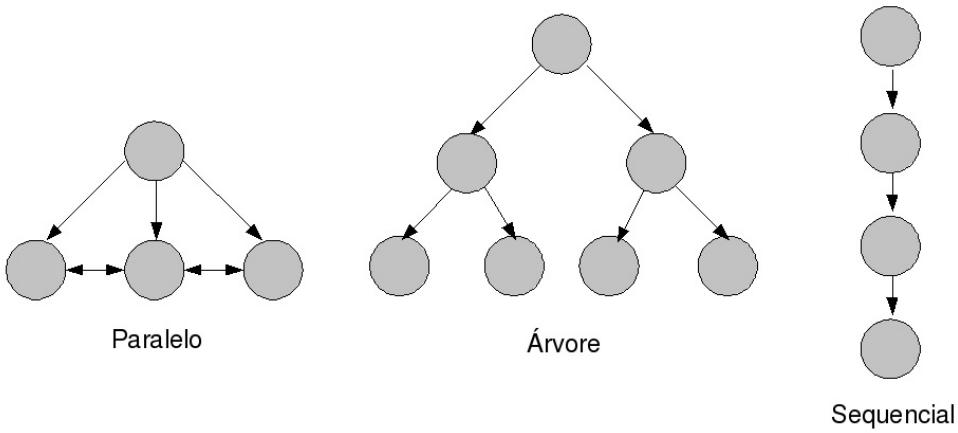


FIG. 2.2: Tipos de *workflows simplified graph*

Dentre as estruturas apresentadas, a DAG tem sido mais amplamente utilizada e trabalhada, principalmente na área científica. Os principais trabalhos existentes de escalonamento de *workflows* (YU e SHI, 2008), (ZHANG et al., 2009), (RAMAKRISHNAN e REED, 2008) têm restringido seus escopos de trabalho em *workflows* DAGs.

2.3.1.2 MODELO DE COMPONENTES

Os *workflows* diferem entre si, também, devido a seus componentes. Um *Worklfow* pode ter como componente principal as tarefas e os dados que o compõem ou focar somente

nas tarefas. Os modelos existentes são apresentados a seguir e exemplificados na Figura 2.3.

- Orientado a tarefas: As tarefas são apresentadas como nós dos grafos e os dados e condições de controle são apresentados como arestas.
- Orientado a tarefas e transferência de dados: Ambos são apresentados como nós do grafo (DEMENEZES, 2008).
- Orientado a dados: Cada nó do grafo representa um dado a ser transformado, esses nós serão alocados em serviços disponíveis nos recursos a fim de realizar as transformações necessárias para obtenção do resultado desejado (IKEDA et al., 2010).

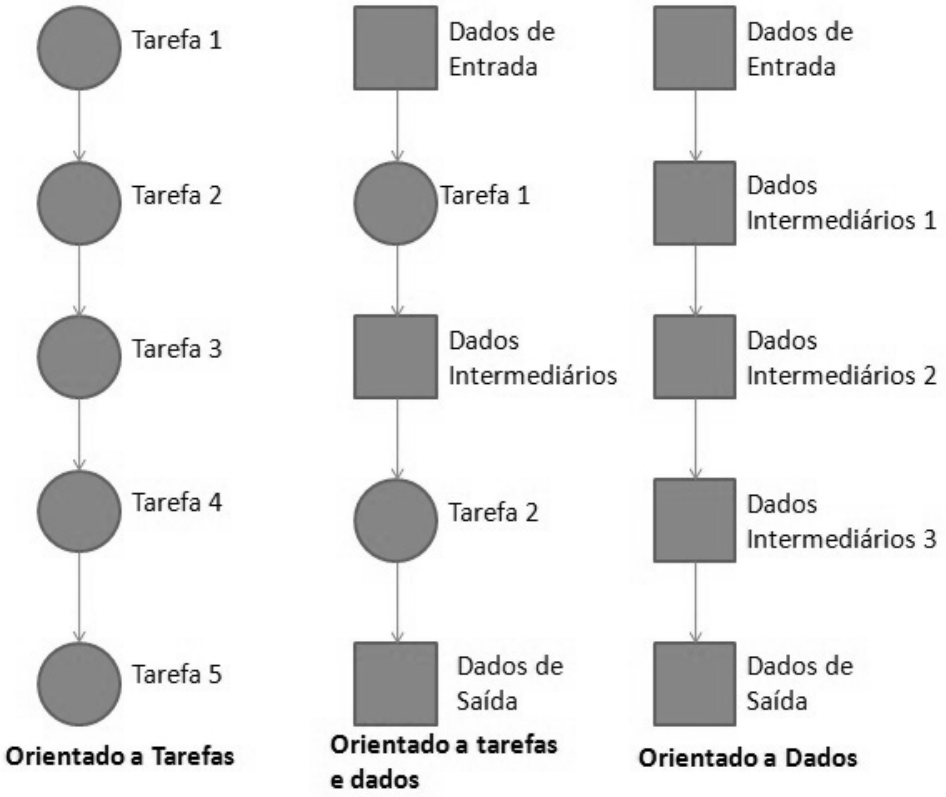


FIG. 2.3: Modelos de Componentes

Dentre os modelos apresentados, o que tem sido mais utilizado é o modelo orientado a tarefas, pois apresenta uma forma bem definida de tarefas e transferência dos dados. Essa abordagem facilita a representação dos DAGs, principalmente, por só necessitar da representação destes dois objetos.

2.3.1.3 DINAMISMO

Os *workflows* podem ser dinâmicos (*Tunable*) ou fixos, como mostrado na Figura 2.4. Quando fixos os mesmos possuem a mesma quantidade de tarefas e estrutura até o fim de sua execução. *Workflows* dinâmicos podem variar a sua estrutura e novas tarefas podem ser adicionadas durante a sua execução.

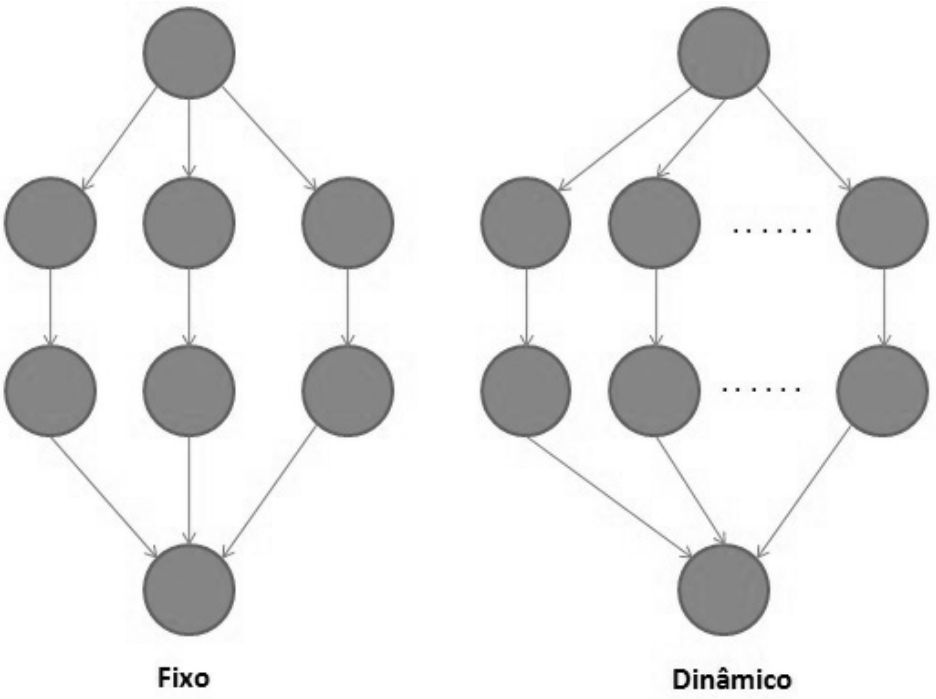


FIG. 2.4: Modelos de dinamismo

2.3.1.4 PROCESSAMENTO DOS DADOS

Existem dois modelos distintos de *workflows* quanto ao processamento dos dados, o modelo que possui uma única entrada e um modelo *pipelined*, como mostrado na Figura 2.5. O primeiro é executado uma vez com uma única entrada. O segundo é executado várias vezes, ao mesmo tempo, porém com valores de entrada diferentes para cada execução.

2.3.2 DEFINIÇÃO DO MODELO

Cada uma das variações possíveis de *workflows* influência diretamente no escalonamento de suas tarefas pelos recursos ou serviços disponíveis. Por este motivo, é importante definir um perfil de *workflow* a ser trabalhado para que o escalonamento possa ser otimizado para

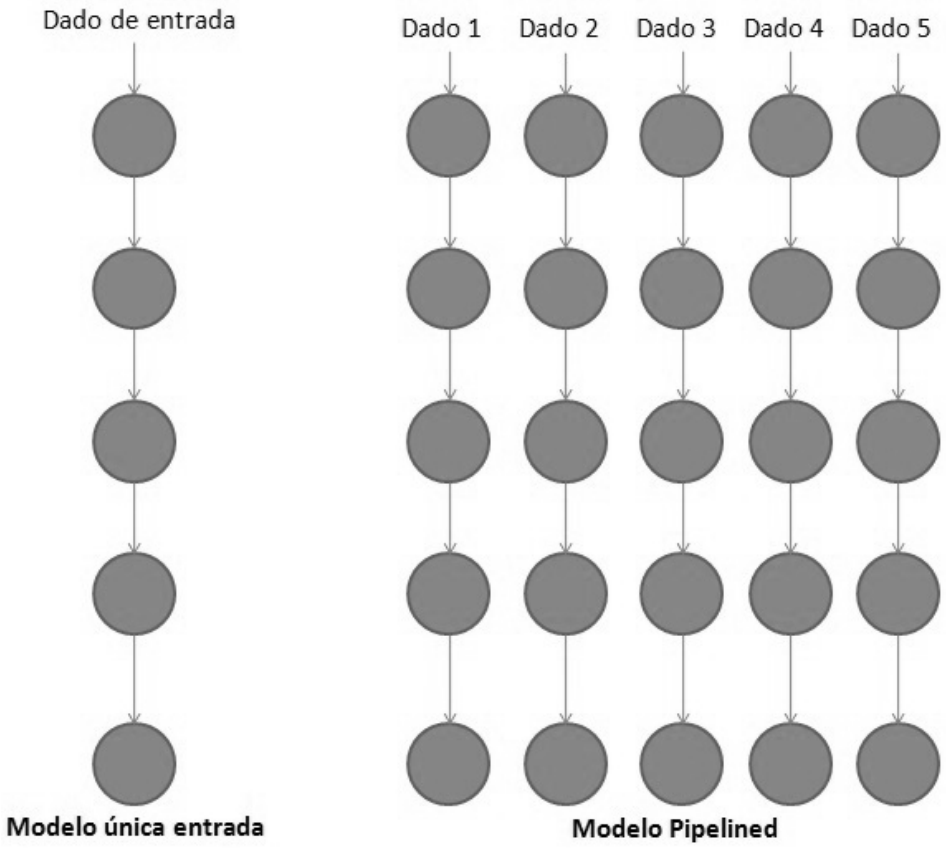


FIG. 2.5: Modelos de processamento de dados

um dado tipo de *workflow*. É apresentado em (WIECZOREK et al., 2009) um estudo detalhado sobre o escalonamento de *workflows*, assim como suas principais características.

Com base no estudo desenvolvido e nos trabalhos relacionados a *workflows* científicos, foi possível definir o perfil de aplicações *workflows* que tem sido mais utilizado e discutido pela comunidade científica. Foi definido que o modelo mais utilizado possui uma estrutura DAG, orientada a tarefas, fixo e com uma única entrada (YU e BUYYA, 2005). Este foi o modelo de *workflow* utilizado neste trabalho.

3 TRABALHOS RELACIONADOS

Um dos problemas mais comuns e complexos em ambientes distribuídos como a computação em Nuvens e as Grades é a seleção dos recursos que melhor atendem uma determinada tarefa. Este problema, mais conhecido como o problema de escalonamento, possui uma grande quantidade de variáveis que influenciam diretamente em suas decisões, como por exemplo, a distribuição dos recursos, heterogeneidade, disponibilidade e desempenho, considerando apenas variáveis do ambiente. Ainda influenciam as variáveis relacionadas ao usuário, tais como o perfil da aplicação e os critérios considerados importantes pelo usuário (tempo de resposta, tempo de submissão, confiabilidade, custo, consumo de banda passante, entre outros).

Há diversos trabalhos que apresentam diferentes abordagens para solução deste problema, como as apresentadas em (MURY et al., 2010),(WIECZOREK et al., 2008),(MEYER, 2007),(DONG, 2009),(IOSUP et al., 2008), (RAMAKRISHNAN e REED, 2008). Por ser um problema NP-Completo, é muito difícil alcançar uma solução ótima que atenda a todas as variáveis envolvidas (EL-REWINI et al., 1994), (YU e BUYYA, 2005).

Quando consideramos aplicações *workflows*, surgem ainda mais variáveis a serem consideradas, como por exemplo as questões da rede e proximidade espacial entre os recursos utilizados para execução de cada tarefa do *workflow*. Tais variáveis tornam-se ainda mais importantes devido a dependência existente entre as tarefas que formam o *workflow*, como mostrado em (WIECZOREK et al., 2009). Por este motivo, os trabalhos existentes visam apresentar diferentes abordagens e restringem o ambiente reduzindo a quantidade de variáveis envolvidas a fim de encontrar a melhor solução para um dado caso.

Foram analisados trabalhos que abordam diferentes estratégias de escalonamento de *workflows* em Grades e Nuvens a fim de identificar as principais características e oportunidades existentes nos ambientes mencionados.

3.1 ESTRATÉGIAS PARA O ESCALONAMENTO DINÂMICO DE *WORKFLOWS* EM *GRIDS*

Em (MEYER, 2007) foi apresentado como tese de doutorado junto a COPPE duas diferentes estratégias de escalonamento de *workflows* dinâmico. O autor trata a relação

usuário/recurso de uma forma diferente da usualmente utilizada. Nessas abordagens os recursos não são vistos de forma individual, mas sim, agrupados através de organizações virtuais. O escalonador seleciona então os sítios e não cada recurso individualmente. Essa seleção é feita através de uma estratégia oportunista ou por proximidade espacial, dependendo da abordagem utilizada.

A estratégia oportunista tem por objetivo escalonar as tarefas do *workflow* pelos sítios da Grade baseado no desempenho dos recursos em submissões anteriores. Inicialmente, a estratégia aloca as tarefas nos sítios que oferecem o maior poder computacional até que o escalonador possua dados suficientes de submissões anteriores para determinar quais recursos apresentam-se mais confiáveis e com melhor desempenho para cada tarefa do *workflow* a ser executada.

Ao fim da realização de testes da estratégia em um ambiente real de Grades, a mesma foi inserida no Sistema de Gerência de *Workflows* Científicos Euryale+ (MEYER, 2007) e aplicada no ambiente *Open Science Grid* (PORDES et al., 2007). Os resultados obtidos apresentam um ganho de até 120% ao serem comparados com as abordagens mais conhecidas de escalonamento.

A outra abordagem proposta é baseada no escalonamento de tarefas por proximidade espacial. Este tipo de estratégia visa estabelecer grupos de tarefas que possuem um maior grau de dependências entre si e alocá-las em sítios mais próximos (com um menor custo de transferência de dados) ou, se possível, a alocação das tarefas no mesmos sítio. Esta estratégia apresentou bons resultados em casos específicos, superiores às mais conhecidas abordagens de escalonamento da área, porém mostrou-se inferior à estratégia oportunista. Para realização de experimentos desta estratégia foi utilizado o simulador GridSim.

Como trabalhos futuros o autor propôs a utilização de ambas as estratégias trabalhando de forma cooperativa, onde as associações dos grupos de tarefas seriam feitas em sítios virtuais e durante a execução do *workflow* é procedido o mapeamento do sítio virtual para o real de acordo com a estratégia oportunista.

3.2 *BI-CRITERIA SCHEDULING OF SCIENTIFIC WORKFOWS*

O artigo (WIECZOREK et al., 2008) apresenta um estudo detalhado do problema de escalonamento de aplicações *workflows* em Grades. No mesmo, são descritos os pontos mais importantes quanto ao escalonamento de *workflows*, tais como: modelo do *workflow*, critérios de escalonamento, processo de escalonamento, modelo de recursos e modelo

da tarefa. Para cada ponto descrito são apresentadas as classificações existentes e os principais trabalhos que utilizam-se dessas variações para desenvolvimento do seu modelo de escalonamento. Com base no estudo realizado, o autor então propôs um modelo de escalonamento bi-critérios para *workflows* científicos.

No modelo bi-critérios foi proposto um novo algoritmo chamado *Dynamic Constraint Algorithm* (DCA) como uma solução para o problema de otimização com dois critérios independentes, comuns em um sistema real (por exemplo, tempo de execução e custo). Primeiro é escolhido o principal critério a ser otimizado (critério primário) e então o usuário estabelece uma porcentagem de variação determinando o quanto a solução final pode diferir da melhor solução para o critério primário, esse espaço de variação é chamado de *sliding constraint*. Após obter o resultado do primeiro critério, é aplicado então o segundo critério, objetivando conseguir o melhor resultado para este, respeitando o limite de variação definido pelo usuário. Entretanto, não leva em consideração a diferenciação das classes de serviços e a tarifação proposta é fixa não diferenciando a oferta de recursos. Também não se utiliza do agrupamento de tarefas para redução do consumo de banda, como proposto no trabalho a seguir.

A abordagem de escalonamento também foi ajustada para diferentes classes de critérios baseados nos estudos apresentados em (WIECZOREK et al., 2007). O modelo proposto foi implementado e testado como parte do ambiente de Grade ASKALON (FAHRINGER et al., 2007).

3.3 *COST-BASED SCHEDULING OF SCIENTIFIC WORKFLOW APPLICATIONS ON UTILITY GRIDS*

O trabalho (YU et al., 2005) apresenta um modelo de escalonamento baseado em custo para *workflows* científicos em Grades utilitárias. As Grades utilitárias têm surgido como um novo modelo para provisão de serviços. Os usuários consomem esses serviços quando necessário e pagam somente pelo que usarem. A provisão de serviços de Grades utilitárias tem sido reforçada pela Grade orientada a serviços. Este trabalho cria uma infraestrutura possibilitando aos usuários consumir os recursos de forma transparente através de um ambiente de rede global seguro, compartilhado e escalável.

Muitos dos usuários de Grades, tais como, pesquisadores em bioinformática e astronomia, necessitam do processamento de aplicações *workflows* em seus experimentos. Para a imposição do paradigma de *workflows* em Grades utilitárias, o custo de execução deve

ser considerado ao escalonar tarefas em recursos.

Dada esta motivação, o trabalho apresenta um algoritmo de escalonamento de *workflows* baseado em custo para aplicações *workflows* com restrições de tempo impostas pelo usuário (tempo crítico). O objetivo do algoritmo de escalonamento proposto é desenvolver um escalonador de *workflow* que miniminize o custo de execução e ainda atenda às restrições de tempo. Para resolver o problema de escalonamento de *workflows* de larga escala de forma eficiente, o *workflow* é dividido em subconjuntos de tarefas que estabelecem um fluxo sequencial e é gerado o escalonamento do *workflow* baseado no escalonamento ótimo de cada um desses subconjuntos. Tarefas que não formam um fluxo único, ou seja possuem dependência de mais de uma tarefa ou existe mais de uma tarefa que depende dela, são separadas e cada uma é executada como um subconjunto.

O usuário definirá também o tempo limite de execução para o *workflow* e este será dividido de uma forma justa entre os subconjuntos de tarefas do *workflow*, através deste tempo o escalonador é capaz de identificar quando uma determinada tarefa está com atraso e reescalonar a mesma.

3.4 *A MULTIPLE QOS CONSTRAINED SCHEDULING STRATEGY OF MULTIPLE WORKFLOWS FOR CLOUD COMPUTING*

A estratégia de escalonamento proposta no trabalho (XU et al., 2009) tem por objetivo possibilitar que sejam usadas múltiplas restrições de QoS em conjunto para o escalonamento de múltiplos *workflows* em Nuvem.

Segundo o autor, a estratégia de escalonamento de *workflows* em Nuvens deve ser considerada como um novo recurso importante a este tipo de ambiente computacional. Primeiramente, por prover serviços para vários usuários e cada usuário possuir necessidades específicas de QoS, e devido também ao problema de alocação dos diversos *workflows* de estruturas diferentes que o ambiente poderá receber para execução ao mesmo tempo.

Para atender a esta necessidade, o autor introduz uma estratégia de escalonamento *multi-workflows* com múltiplas restrições de QoS para Nuvem. Foi gerado então, um escalonador para satisfazer os requisitos de QoS do usuário, minimizar o *makespan* e o custo dos recursos, e aumentar o nível de sucesso quanto ao escalonamento do *workflow*.

No algoritmo, o usuário primeiramente submete o *workflow* com seus requisitos de QoS. O sistema então aloca os serviços apropriados para processamento das tarefas do *workflow* e escalona as tarefas nos serviços de acordo com os requisitos de QoS e o ambiente

de Nuvens.

Para escalonar o *workflow* dinamicamente e otimizar o processo de alocação de recursos, o sistema proposto é formado por três componentes principais formando o fluxo demonstrado na figura 3.1.

- Pré-processador

  Computa quatro atributos de cada tarefa recebida: número de serviços disponíveis, a covariância para tempo e custo, e a estimativa de tempo e custo a serem utilizados. Em um próximo passo o pré-processador submete as tarefas recebidas para a fila do escalonador.

- Escalonador

  O escalonador recomputa os atributos e reorganiza as tarefas na fila de acordo com a estratégia utilizada.

- Executor

  O executor seleciona sequencialmente o melhor serviço para executar as tarefas da fila. Quando a tarefa termina, o executor notifica ao Pré-processador a conclusão da tarefa.

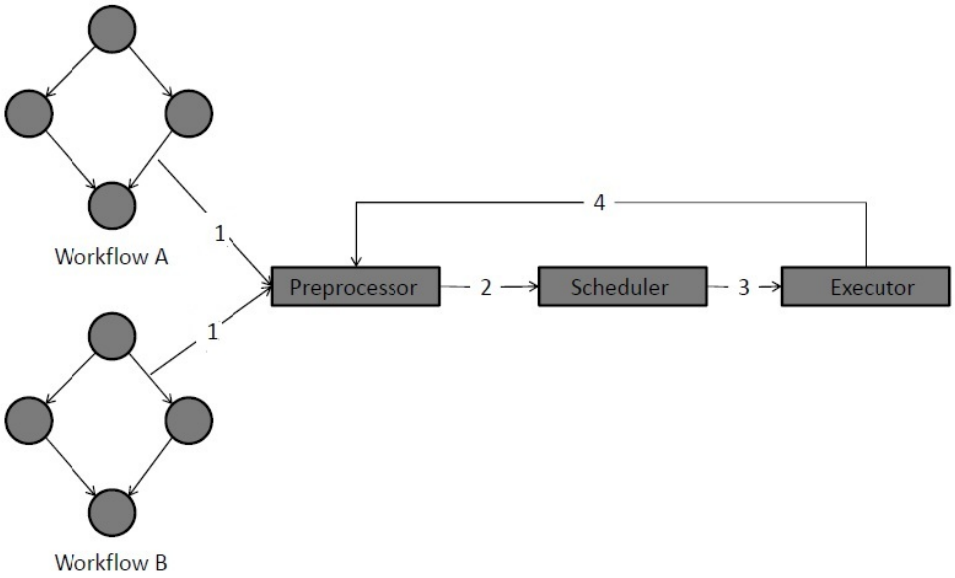


FIG. 3.1: Processo de escalonamento (XU et al., 2009)

Um novo *workflow* inicialmente é submetido ao pré-processador, que computa os atributos de todas as tarefas do *workflow*. Em seguida, o pré-processador insere as tarefas na fila. Em um primeiro momento, somente as tarefas de entrada serão submetidas. Posteriormente, mediante a notificação do término das mesmas, o pré-processador irá verificar se a tarefa sucessora pode ser executada e a submete então.

Quando houver serviços disponíveis e tarefas esperando em fila para serem executadas, o escalonador irá recomputar todos os atributos, reorganizar as tarefas na fila e repetir os seguintes passos:

a) Remover a primeira tarefa da fila;

b) Alocar a tarefa para o melhor serviço disponível;

c) Inserir a tarefa no final da fila se não houver recursos disponíveis para executar a tarefa.

Quando a tarefa é executada com sucesso, o executor irá notificar o pré-processador sobre o status atualizado da tarefa. Este trabalho tem como diferencial para as demais abordagens, a utilização de múltiplas restrições de QoS e uma estratégia de escalonamento que visa otimizar a utilização do ambiente para múltiplos *workflows* submetidos ao mesmo tempo. A abordagem proposta apresentou bons resultados ao ser comparada ao algoritmo de escalonamento dinâmico RANK_HYBD (YU e SHI, 2008).

3.5 *PERFORMABILITY MODELING FOR SCHEDULING AND FAULT TOLERANCE STRATEGIES FOR SCIENTIFIC WORKFLOWS*

Workflows científicos possuem necessidades específicas devido às relações existentes entre as suas tarefas, tais como a necessidade de um maior nível de confiabilidade e questões relacionadas a rede que interliga os recursos utilizados, assim como, a proximidade espacial existente entre estes, tornando a tarefa de seleção dos recursos ainda mais complexa. Em um ambiente com recursos dispersos e distribuídos como os de Grades e Nuvens, essa tarefa é ainda mais delicada (RAMAKRISHNAN e REED, 2008).

Ao utilizar ambientes distribuídos para a execução de aplicações *workflows* um dos pontos fundamentais a ser considerado para que possa atender devidamente às necessidades do usuário, é a questão da confiabilidade dos recursos e a tolerância a falhas. Em um ambiente dinâmico e não dedicado como a maioria das Grades, a falha de parte dos

recursos da Grade torna-se algo esperado. Porém, a forte dependência existente entre as tarefas de uma aplicação *workflow*, faz com que a falha de uma tarefa do *workflow* comprometa toda a sua execução.

Mediante os problemas acima apresentados o autor apresenta um modelo para definição de QoS, possibilitando identificar a degradação de desempenho dos recursos e tratar o problema através de uma estratégia de tolerância a falhas. Nesse artigo é apresentada então, uma extensão para a *virtual grid description* (KEE et al., 2005) que permite aos usuários definirem as configurações desejáveis quanto aos recursos a serem utilizados baseado no modelo de custo.

Através desta linguagem o usuário é capaz de especificar ao sistema a configuração desejável dos recursos e a confiabilidade esperada de rede e de cada recurso. Com base nesses dados o sistema identifica a degradação de desempenho e, se for inferior ao mínimo estipulado pelo usuário, é acionada então a estratégia de tolerância a falhas em vigor para que a tarefa possa ser realocada.

O modelo foi testado através de simulações com dados disponibilizados pelo *Los Alamos National Laboratory* (LANL), utilizando duas diferentes heurísticas de escalonamento, Min-min e Max-min (BRAUN et al., 2001).

3.6 *A GRID-QOS DECISION SUPPORT SYSTEM USING SERVICE LEVEL AGREEMENTS*

(BANDINI et al., 2009) propõe um modelo de Qualidade de Serviço (QoS) em Grades utilizando acordos em nível de serviço (SLA). Neste trabalho os usuários tem prioridade na utilização dos recursos de acordo com o plano de serviço escolhido pelo usuário.

São considerados três níveis de serviço (Premium, Master e Básico) em função dos quais são calculados os custos, diferenciando-os por meio de pesos atribuídos tanto por usuários como pelos provedores dos serviços.

Para a contabilização do custo de execução de cada tarefa, o trabalho apresenta a Fórmula 3.1 que utiliza como fatores de variação deste valor o desempenho do recurso utilizado e o plano de serviço do usuário.

$$C = S + \alpha(S.t) + \beta(S.f) \tag{3.1}$$

Onde,

- C : Custo da submissão;
- S : Tempo de submissão;
- t : Taxa cobrada de acordo com o plano de serviço;
- f : Taxa do recurso, acréscimo a ser cobrado pelo recurso utilizado;
- $\alpha = 1; \beta = 1/8$, sendo α e β os fatores de equilíbrio da fórmula.

Além do modelo de QoS, o trabalho apresenta também um modelo de reserva antecipada de recursos através de uma interface *Web* intuitiva que facilita a interação entre os usuários e o sistema.

4 MODELO DE ESCALONAMENTO PEB-COS

A análise dos trabalhos anteriormente citados no Capítulo 3 possibilitou identificar um conjunto de características consideradas importantes a um algoritmo de escalonamento, para este tipo de ambiente, e que serviram de base para o modelo proposto nesta dissertação. As seguintes características foram implementadas:

- Agrupamento de tarefas que compõem um *workflow* através da identificação de fluxos sequenciais a fim de diminuir o consumo de banda passante;
- Parametrização dos requisitos mínimos para submissão de tarefas e prevenção de falhas;
- Uso de dados de submissões anteriores de cada usuário para auxiliar na seleção do recurso que melhor atenda a um dado usuário;
- Abordagem bi-critérios baseada em um limite de variação do critério primário, definido pelo usuário, para otimização do critério secundário;
- Criação de classes de serviços[1] e a sua tarifação tendo em vista a redução da degradação de recursos e parâmetros de otimização selecionados.

O modelo de escalonamento PEB-CoS (Processo de Escalonamento Bi-critérios com suporte a CoS) tem por propósito permitir um escalonamento bi-critérios, para *workflows* aplicado em ambientes de Grades, com suporte a CoS, possibilitando uma melhor utilização dos recursos, o reescalonamento dinâmico das tarefas em tempo de execução e atendendo aos requisitos estabelecidos nos planos de serviço.

4.1 CLASSES DE SERVIÇO (COS)

Antes de iniciar o processo de escalonamento, cada usuário deverá formalizar um acordo de serviço (*Service Level Agreements* - SLA), no qual serão descritos os requisitos que ele considera importantes para a execução do seu *workflow*, priorizando dois dentre os quatro

[1]Optou-se pelo uso de CoS em detrimento de QoS por não ser o foco deste trabalho algumas características necessárias em ambientes com suporte a QoS, como por exemplo, o controle de admissão.

tipos de critérios considerados neste trabalho. Os critérios abaixo foram estipulados com base naqueles utilizados nos trabalhos apresentados no Capítulo 3, (WIECZOREK et al., 2008), (WIECZOREK et al., 2007).

a) **Tempo de submissão**

   Em um primeiro momento o tempo de submissão é o único critério considerado, alocando as tarefas nos melhores recursos disponíveis priorizando os que possuem menor concorrência. A partir do momento que há um histórico de submissões é possível estimar o tempo médio de execução de cada perfil de tarefa nos respectivos recursos. Neste processo, são consideradas tarefas do mesmo perfil, ou seja, as que possuem os mesmos parâmetros (tamanho do arquivo de entrada, tamanho do executável da tarefa e tamanho do arquivo de saída).

b) **Custo de execução nos serviços**

   O custo de execução se baseia no custo médio de execução de tarefas do mesmo perfil em submissões anteriores. Através dessa informação, é possível estabelecer o custo médio em cada recurso e definir quais oferecem os menores custos.

c) **Nível de confiabilidade dos recursos**

   Cada recurso está sujeito a degradações de desempenho no decorrer da sua utilização, neste trabalho é determinado o nível de confiabilidade dos recursos baseado nos dados coletados de cada um dos recursos, de tempos em tempos, nos experimentos anteriores. Através destes dados, é encontrada a média do desempenho de cada recurso (o desempenho é encontrado através da Fórmula 4.1 apresentada na Seção 4.3), sendo assim, quanto maior essa média menos sujeito a falhas o recurso está.

A abordagem bi-critérios permite que o usuário selecione, dentre os critérios acima apresentados, os dois critérios que considera de maior importância e então defina qual será utilizado pelo escalonador como critério primário e qual será utilizado como critério secundário.

A SLA do usuário também será responsável por definir quais serviços o usuário poderá utilizar. Na mesma constará o plano ao qual o usuário está associado e neles os recursos, o critério primário, o secundário e o limite de variação permitido pelo usuário com relação ao critério primário.

4.2 PLANOS DE SERVIÇO

A fim de possibilitar a diferenciação entre os níveis de classes de serviços selecionados por cada usuário, foram definidos três planos de serviço: básico, master e premium. Os planos de serviço variam entre si em relação à quantidade de recursos disponíveis, à quantidade de MIPS de cada processador dos recursos e ao custo, de forma que o poder computacional agregado dos recursos e o custo tenham um crescimento proporcional dos seus valores para os pacotes acima estipulados.

4.3 DEFINIÇÃO DO ÍNDICE DE DESEMPENHO E DO CUSTO DE SUBMISSÃO

Para a etapa de alocação do recurso, verificou-se a necessidade da utilização de uma fórmula para determinação do desempenho de cada recurso, a fórmula elaborada e utilizada neste trabalho está na Equação 4.1.

$$I = Nl(MIPS/Nt); \tag{4.1}$$

Onde,

- I : Indicador de Desempenho;
- MIPS: Milhões de instruções por segundo que o processador é capaz de desempenhar;
- Nl : Número de núcleos de processamento livres;
- Nt : Número total de núcleos de processamento.

Para contabilização do custo de execução das tarefas em função dos requisitos de CoS acordados no plano de serviço foi utilizada a fórmula apresentada em (BANDINI et al., 2009). A fórmula foi adaptada para este trabalho conforme abaixo especificado:

$$C = S + \alpha(S.f) + \beta(S.I) \tag{4.2}$$

Onde,

- C : Custo da submissão;
- S : Tempo de execução;

- I : Indicador de desempenho do recurso no momento da execução. Originalmente, representava a taxa cobrada de acordo com o plano de serviço;
- f : representa a constante de custo para cada plano (0, 0.4 e 0.8 para os planos básico, master e premium respectivamente). Originalmente, representava a taxa do recurso, que consistia em um acréscimo a ser cobrado pelo recurso utilizado.
- $\alpha = 2$; $\beta = 1/2$, sendo α e β os fatores de equilíbrio da fórmula.

Após a realização dos experimentos iniciais, verificou-se a necessidade da alteração de alguns dos valores utilizados originalmente na fórmula de custo a fim de adequar a mesma ao modelo simulado. Os valores da constante de custo (f) e dos fatores de equilíbrio da fórmula (α e β), foram reajustados em relação à fórmula original a fim de permitir a diferenciação entre os planos de serviço, possibilitando o seu uso tanto para a cobrança em valores monetários quanto na forma de troca de créditos.

4.4 ALGORITMO DE ESCALONAMENTO

O processo de escalonamento é dividido nas seguintes etapas:

a) Inicialmente, o escalonador recebe o *workflow* e identifica os conjuntos de tarefas que formam um fluxo sequencial (fila),

b) O escalonador separa essas tarefas em subconjuntos dentro do *workflow*. Tarefas com mais de uma dependência ou que sejam dependências para mais de uma tarefa, são consideradas como um ponto de convergência e cada uma delas é colocada em um subconjunto unitário. Para essa verificação, o *workflow* é transcrito para um grafo no qual é feito uma busca em profundidade que percorre todos os nós. Quando um nó possui mais de uma aresta incidente ou aponta para mais de um outro nó, esse nó é considerado um ponto de convergência. Quando é encontrada uma sequência de nós que possuem no máximo uma aresta incidente e aponta para apenas um outro nó, essa sequência é agrupada em um subconjunto e escalonada para um mesmo recurso.

c) Ao ser identificado cada fluxo e cada tarefa de convergência como um subconjunto, o escalonador aloca o subconjunto a um recurso a fim de otimizar a utilização dos serviços e diminuir os custos de utilização e consumo de rede.

No processo de associação dos subconjuntos de tarefas criados aos serviços existentes, o escalonador utiliza um algoritmo de escalonamento bi-critérios.

Esse algoritmo será capaz de identificar, através da SLA definida pelo usuário, quais os dois critérios que foram considerados de maior importância. Para a aplicação dos critérios será definida inicialmente a prioridade entre os mesmos (critério primário e critério secundário).

O algoritmo aplicará então o critério primário a fim de obter o melhor resultado. Com base nos dados fornecidos pelo usuário através da SLA, utiliza o limite de variação para este valor inicialmente obtido, a fim de otimizar o segundo critério, encontrando qual serviço oferece o melhor resultado para ambos os critérios.

A fim de prover informações sobre os recursos disponíveis, o algoritmo utiliza um mecanismo de coleta de informações dos recursos armazenando em um banco de dados para que as mesmas possam ser consultadas no momento da seleção dos recursos. No Apêndice A é apresentado o modelo do banco de dados utilizado.

d) Após a seleção do melhor recurso, as tarefas são enviadas para a fila de espera do mesmo. A tarefa é enviada para a execução assim que o recurso estiver com pelo menos uma unidade de processamento livre. Antes da tarefa entrar em execução, o escalonador verifica se o serviço continua atendendo os requisitos mínimos do usuário. Caso tenha ocorrido uma perda de desempenho, a tarefa a ser executada e as tarefas posteriores à mesma e que fazem parte do mesmo subconjunto, são então realocadas para outro serviço.

Este escalonamento segue o modelo de escalonamento híbrido, em que há uma alocação prévia e a verificação da necessidade de uma realocação em tempo de execução, possibilitando uma melhor utilização dos serviços e consequentemente um menor retardo por parte do escalonador em tempo de execução. Diversos trabalhos recentes (BERMAN et al., 2005), (YU e SHI, 2007), (ZHANG et al., 2007), (ZHANG et al., 2009) têm utilizado este modelo apresentando resultados melhores do que as demais abordagens existentes (DONG e AKL, 2006). Como exemplos podemos citar o modelo estático e o dinâmico.

Este trabalho limitou-se a uma abordagem bi-critérios por possibilitar obter uma resposta em tempo útil para o escalonamento de *workflows*. Em uma abordagem multi-critérios a cada novo critério adicionado, é refeita a otimização levando em consideração

o limite de variação dos critérios anteriores. O número de critérios no entanto deve ser limitado pois quanto maior a quantidade de critérios, maior a complexidade do algoritmo.

Assim sendo, com uma abordagem bi-critérios, é possível alcançar uma solução que atenda o valor e o limite de variação estipulados para o primeiro critério, assim como um valor para o segundo critério garantindo os níveis de qualidade de serviço estabelecidos pelo usuário. Com esses passos é possível, no pior caso, encontrar um ótimo local para a escolha do serviço que melhor atenda a uma dada tarefa, como mostrado em (WIECZOREK et al., 2008).

O trabalho aqui apresentado busca reunir as principais características dos trabalhos de escalonamento de *Workflows* estudados, que permita o escalonamento considerando mais de um critério, com suporte a CoS, possibilitando uma melhor utilização dos recursos, possibilitando o reescalonamento dinâmico das tarefas em tempo de execução e atendendo aos requisitos estabelecidos nos planos de serviço.

O processo utilizado neste trabalho foi o de procurar a melhor solução que contemplasse os dois critérios considerados, ou seja a sua otimização. Com a determinação do valor do intervalo de variação por parte do usuário o valor encontrado, na maioria das vezes, não será o valor correspondente a melhor solução de todas as soluções possíveis (ótimo global), mas sim o valor que atenda à restrição imposta pelo limite de variação (ótimo local). No presente texto as expressões ótimo e otimização passarão a ser empregadas com o significado de se obter o melhor valor em função da restrição imposta pelo usuário.

5 RESULTADOS OBTIDOS

Nesta seção serão apresentados os resultados obtidos dos experimentos realizados com o modelo de escalonamento proposto.

5.1 METODOLOGIA

O desenvolvimento deste trabalho tem como foco um ambiente de Grades a fim de possibilitar a realização de experimentos com o modelo de escalonamento de *workflows* científicos proposto e uma das principais abordagens utilizada para servirem de comparação. Devido a toda a complexidade que envolve a utilização desse ambiente, definição e instalação das ferramentas necessárias para gerência de aplicações de *workflow*, optou-se pela utilização de um ambiente de simulação de Grades para que todo o esforço deste trabalho se concentrasse no seu diferencial, a apresentação e análise de desempenho do modelo.

5.1.1 SIMULADOR

A fim de definir qual seria o ambiente de simulação utilizado foram estudados os simuladores apresentados a seguir.

O *GridSim* é uma plataforma desenvolvida pelo Projeto *Gridbus* dedicada a simulação de ambientes de Grades e *Clusters*. Abaixo serão descritas algumas das principais características do *GridSim* (BUYYA e MURSHED, 2002).

- Desenvolvido em Java.
- Possui código aberto, bem documentado e de simples utilização.
- Possui ferramentas de rede que possibilitam a criação de diferentes topologias, a definição da capacidade de cada enlace e a simulação de tráfego em cada enlace.
- Possibilita o uso de duas políticas de alocação: *Space-shared* e *Time-shared*.
- Possibilita a criação de *DataGrids*.
- Permite simular a reserva antecipada de recursos.

- Foi criado sobre a biblioteca de simulação *SimJava* (HOWELL e MCNAB, 1998).
- Não oferece suporte a aplicações *workflow*.

O *GSSim* (*Grid Scheduling Simulator*) (KUROWSKI et al., 2007) é um *framework opensource* dedicado a simulação de escalonamento de aplicações em Grades que foi desenvolvido sobre o *GridSim* possuindo as características anteriormente citadas para o *GridSim*. Porém, neste simulador os escalonadores a serem testados são adicionados como *plugins* e as cargas, usuários e recursos são descritos através de arquivos padrões. O *GSSim* possui uma implementação complexa porém o suporte a tarefas *workflow* não está totalmente implementado.

O ALEA (KLUSÁCEK et al., 2007) é um ambiente de simulação de escalonamento em Grades *opensource* que, assim como o *GSSim*, foi criado sobre o *GridSim* e utiliza arquivos padrões de entrada para recursos, usuários e cargas. Difere do *GSSim* quanto a exemplos de modelos de escalonamento e arquivos de carga, usuários e recursos que o acompanham, além de ser mais simples para a utilização e alteração. Uma outra facilidade oferecida é a geração de gráficos de consumo e desempenho dos recursos em tempo de execução, porém apresenta uma grande complexidade para sua utilização devido a falta de documentação.

O *SimGrid* (CASANOVA, 2001) é um *framework opensource* baseado em simulações para avaliação de algoritmos e heurísticas em *Clusters*, Grades e P2P. É um projeto da *University of California at San Diego* e foi desenvolvido na linguagem C. O *SimGrid* provê um ambiente para simulação de escalonamento de aplicações *Bag-of-Tasks* e aplicações do tipo *workflow* através do módulo *SimDag*. Apesar de possibilitar a criação de topologias de redes e simulação de tráfego de rede que permite uma grande proximidade com o mundo real, o *SimGrid* mostrou-se mais complexo e deficiente em termos de documentação e ferramentas se comparado ao *GridSim*, como mostrado no artigo (BUYYA e SULISTIO, 2008).

Uma análise comparativa das características de cada simulador avaliado para este trabalho é apresentada na Figura 5.1.

O *GridSim* foi escolhido por ser simples, desenvolvido em Java e por possuir uma boa documentação se comparado com os demais simuladores estudados.

Tendo em vista que o *GridSim* oferece a infraestrutura para experimentos apenas com um modelo de escalonamento descentralizado e para aplicações sem nenhum nível de dependência, foi necessário adaptá-lo para o escalonamento centralizado, hierárquico

	GridSim	SimGrid	GSSim	ALEA
Distribuição	*Opensource*	*Opensource*	*Opensource*	*Opensource*
Linguagem	Java	C	Java	Java
Nível de Documentação	Boa	Pouca	Pouca	Pouca
Suporte a aplicações workflows	Não	Sim	Sim	Não
Modelo de escalonamento	Descentralizado	Descentralizado	Centralizado	Centralizado

FIG. 5.1: Características dos Simuladores

e com suporte a aplicações de *Workflows*, tratando das dependências existentes entre as mesmas. No apêndice C são apresentadas as alterações feitas para a execução da simulação deste trabalho.

5.1.2 CONFIGURAÇÃO DO AMBIENTE DE SIMULAÇÃO

Para a realização das simulações é necessário especificar os recursos e usuários que compõem a simulação. A cada usuário teremos associado um conjunto de tarefas e a SLA acordada com este. Para especificação dos usuários, assim como as tarefas e a SLA relacionadas aos mesmos, foi criado um padrão de descrição de usuários através de XML (*schema* apresentado no Apêndice B).

Desta forma, cada arquivo XML representa um usuário no modelo. Neste XML constará o plano de serviço ao qual este está associado, os critérios primário e secundário, assim como o limite de variação considerado por este usuário. Seguindo as especificações de CoS será descrito um modelo do *Workflow* que considere as tarefas, o tamanho do arquivo de entrada, o seu tamanho e o tamanho do arquivo de saída de cada uma delas e suas dependências. Um exemplo é mostrado nas Figuras 5.2 (XML) e 5.3 que representa o modelo de *workflow* utilizado neste trabalho.

Os usuários se diferenciaram pelo tamanho de suas tarefas como forma de simular a sua complexidade, aderindo a planos de serviço diferentes e prioridade de critérios distintos, criando um ambiente de simulação complexo e mais próximo da realidade.

Nas simulações realizadas foram utilizados 12 usuários para cada simulação, sendo 4 em cada plano de serviço. Desses 4 usuários em cada plano, 2 consideram como critério primário e secundário o tempo e o custo, respectivamente, e 2 consideram o custo e o tempo. Para cada plano de serviço foi permitido um limite de variação de 20% para o

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<User name="user_4" >
  <planoServico>BASICO</planoServico>
  <criterioPrimario>CONFIABILIDADE</criterioPrimario>
  <criterioSecundario>TEMPO</criterioSecundario>
  <variacao>10</variacao>
  <Workflow>
    <Task id="0">
      <lenght>995550</lenght>
      <fileSize>995550</fileSize>
      <outputSize>995550</outputSize>
      <nextTasks>
        <id>1</id>
        <id>2</id>
        <id>3</id>
        <id>6</id>
      </nextTasks>
    </Task>
    <Task id="1">
      <lenght>995550</lenght>
      <fileSize>995550</fileSize>
      <outputSize>995550</outputSize>
      <dependencies>
        <id>0</id>
      </dependencies>
      <nextTasks>
        <id>4</id>
      </nextTasks>
    </Task>
    ...

  </Workflow>
</User>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<User name="user_5" >
  <planoServico>MASTER</planoServico>
  <criterioPrimario>TEMPO</criterioPrimario>
  <criterioSecundario>CONFIABILIDADE</criterioSecundario>
  <variacao>10</variacao>
  <Workflow>
    <Task id="0">
      <lenght>995550</lenght>
      <fileSize>995550</fileSize>
      <outputSize>995550</outputSize>
      <nextTasks>
        <id>1</id>
        <id>2</id>
        <id>3</id>
        <id>6</id>
      </nextTasks>
    </Task>
    <Task id="1">
      <lenght>995550</lenght>
      <fileSize>995550</fileSize>
      <outputSize>995550</outputSize>
      <dependencies>
        <id>0</id>
      </dependencies>
      <nextTasks>
        <id>4</id>
      </nextTasks>
    </Task>
    ...

  </Workflow>
</User>
```

FIG. 5.2: XMLs de definição dos parâmetros de cada usuário

critério primário a fim de permitir a otimização do critério secundário, variando-se ainda, a quantidade de recursos e de MIPS (Milhões de Instruções Por Segundo) por processador em cada recurso. Durante os experimentos tomou-se o valor de MIPS dos recursos como forma de simular o limite de carga de cada recurso em função do plano de serviço proposto.

Com o objetivo de realizar a comparação do consumo de banda passante sem o agrupamento de tarefas e com o agrupamento, cada um dos usuários simulados submete um *workflow* com 10 tarefas que possui a estrutura mostrada na Figura 5.3. Nos experimentos realizados para validação do modelo, foi utilizado apenas o modelo de *workflow* apresentado, concentrando a diversidade de experimentos na variação do ambiente, dos critérios considerados e do limite de variação considerado por cada usuário.

Para os experimentos variou-se a quantidade de recursos, assim como os MIPS dos mesmos para cada plano de serviço conforme especificado abaixo:

- Básico : 6 recursos, 100 MIPS por processador;
- Master : 8 recursos, 130 MIPS por processador;
- Premium : 7 recursos, 200 MIPS por processador;

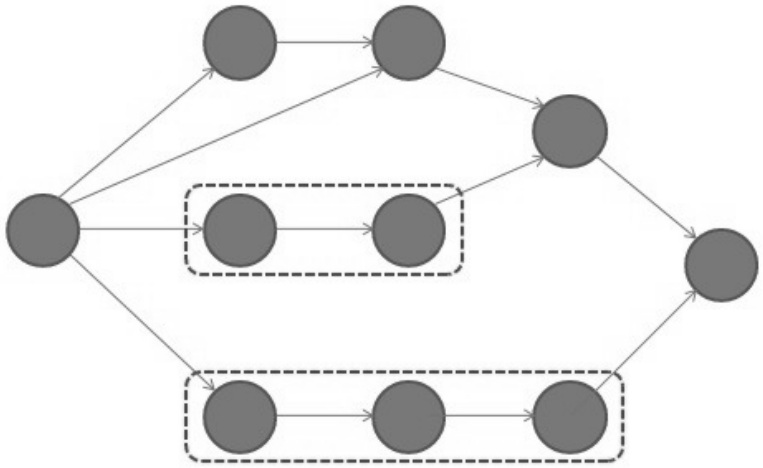


FIG. 5.3: Modelo de *workflow* utilizado

Os valores acima foram arbitrados a fim de possibilitar a diferenciação entre os planos de serviço.

Nos experimentos realizados foi utilizada a topologia de rede apresentada na Figura 5.4. Nesta topologia os recursos são distribuídos igualmente pelos três roteadores onde cada um representa um sítio. Todos os *links* possuem capacidade de 1 Gigabit por segundo e *delay* de propagação de 10 milisegundos. Entre os usuários e o escalonador há um roteador e entre este e os recursos há pelo menos menos mais um roteador.

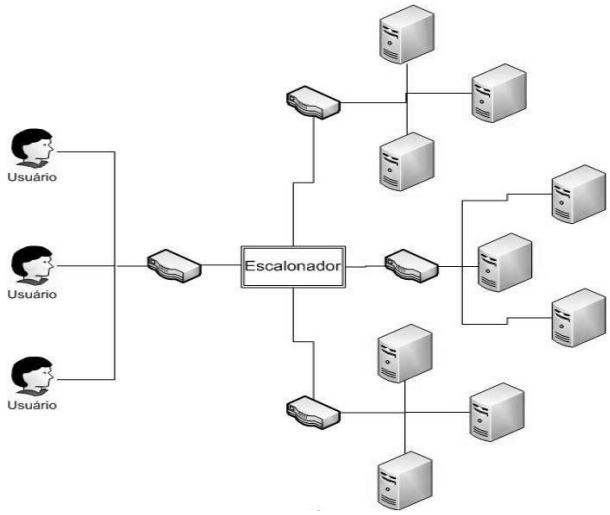


FIG. 5.4: Topologia da rede

Para efeito de comparação os experimentos foram realizados utilizando-se o algoritmo *Join the Shortest Queue* (JSQ) (LIN e RAGHAVENDRA, 1996) como base, o qual associa a primeira tarefa da fila do escalonador ao recurso que possuir a menor fila.

5.2 ANÁLISE DOS RESULTADOS

Para os experimentos foram coletadas 40 amostras dos valores de tempo e custo, e feito o cálculo de sua média. O intervalo de confiança utilizado para a determinação da precisão dos resultados foi de 95%, com o modelo de distribuição *T Student*, com um grau de liberdade.

5.2.1 EXPERIMENTO 1

Nos experimentos iniciais, nos limitamos a utilização de uma abordagem de escalonamento estático que leva em consideração os níveis de QoS, a ordenação de preferência dos critérios (neste caso, tempo de execução e custo) e o agrupamento de tarefas (diminuição do tráfego de rede). Neste modelo, as tarefas dos usuários recebidas pelo escalonador, são alocadas na fila local do recurso selecionado. No momento em que a tarefa for executada, é verificado se suas dependências já foram concluídas, caso não tenham sido, a tarefa retornará ao final da fila do mesmo recurso.

Nos resultados a seguir serão apresentados dois usuários, onde estes apresentam as mesmas características quanto aos critérios selecionados e a prioridade entre eles, variando apenas os valores referentes aos dados de cada tarefa (tamanho do arquivo de entrada, tamanho do executável da tarefa e tamanho do arquivo de saída) que compõem os seus *workflows*.

Para otimização dos critérios de tempo e custo no modelo proposto, foram utilizados os dados de submissões anteriores de cada tarefa. Em um primeiro momento não há esses dados e o algoritmo utiliza uma abordagem semelhante ao modelo JSQ, nas submissões posteriores são utilizados os dados armazenados de cada seleção e tempo de execução anteriores para encontrar quais os melhores recursos para o perfil de cada tarefa.

Para os resultados apresentados a seguir usaremos as seguintes notações:

- UB (1 ou 2) - Usuário base utilizando algoritmo JSQ (-C para custo, -T para tempo);
- U (1 ou 2) - Usuário utilizando o algoritmo proposto (-C para custo, -T para tempo);

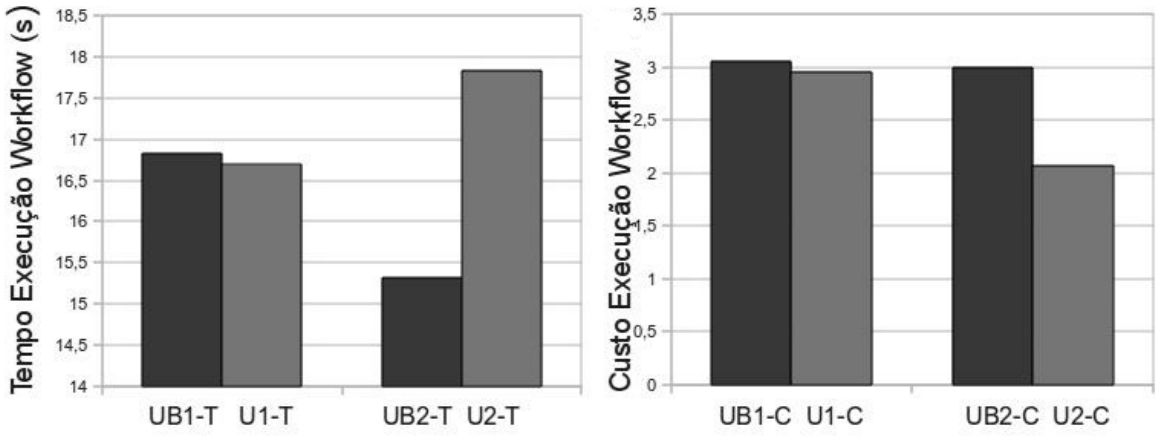


FIG. 5.5: Valores de tempo e custo dos usuários 1 e 2, em relação ao usuário base, no plano Premium com 20% de limite de variação

As figuras 5.5, 5.6 e 5.7 apresentam os resultados obtidos para os planos premium, master e básico em que os usuários tinham optado pelo tempo como critério primário e o custo como critério secundário.

A figura 5.5 apresenta os resultados obtidos em relação ao tempo de execução para dois usuários com o plano Premium e 20% de limite de variação. Para o usuário 1 (U1) obteve-se ganho de tempo e custo em relação ao JQS (UB1) de 0,8% e 3,23% respectivamente. Porém, para o usuário 2 (U2-T) houve perda de desempenho de 17,9% em relação ao usuário base 2 (UB2-T) em relação ao tempo, mantendo porém esta perda dentro dos 20% do limite permitido. Em compensação obteve-se um ganho de 43,8% quanto ao custo se comparado ao usuário base (UB2-C).

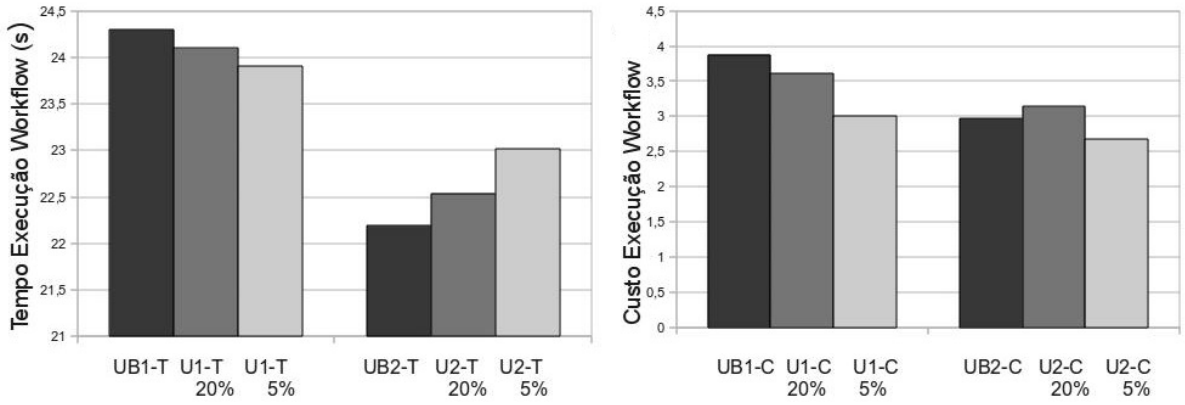


FIG. 5.6: Valores de tempo e custo dos usuários 1 e 2, em relação ao usuário base, no plano Master com 20% e 5% de limite de variação

Na figura 5.6 são apresentados os resultados obtidos para o plano master com 20% e

5% de limite de variação, e o algoritmo JSQ. Neste experimento optou-se por adicionar mais um valor de variação para análise do efeito e comportamento do limite de variação no modelo proposto. O usuário 1 (U1-T e U1-C) obteve uma otimização maior de tempo e custo de 0,8% e 7,0% respectivamente para o caso de 20% de limite de variação e de 1,6% e 22,5% respectivamente para o caso de 5% de limite de variação em relação ao usuário base (UB1-T e UB1-C). Já o usuário 2, para o critério de tempo (U2-T), teve uma perda de 1,6% e 3,7% para 20% e 5%, respectivamente, em relação ao tempo do usuário base (UB2-T) na tentativa de otimização do critério secundário. Este obteve um ganho quanto ao custo (U2-C) de 5,8% quando o limite de variação foi de 20%, mas para o limite de variação de 5%, este foi 9,6% menor. Apesar do limite de variação determinar o quanto de perda do critério primário o usuário considera aceitável na tentativa de otimização do critério secundário, haverá casos em que não será possível otimizar o critério secundário.

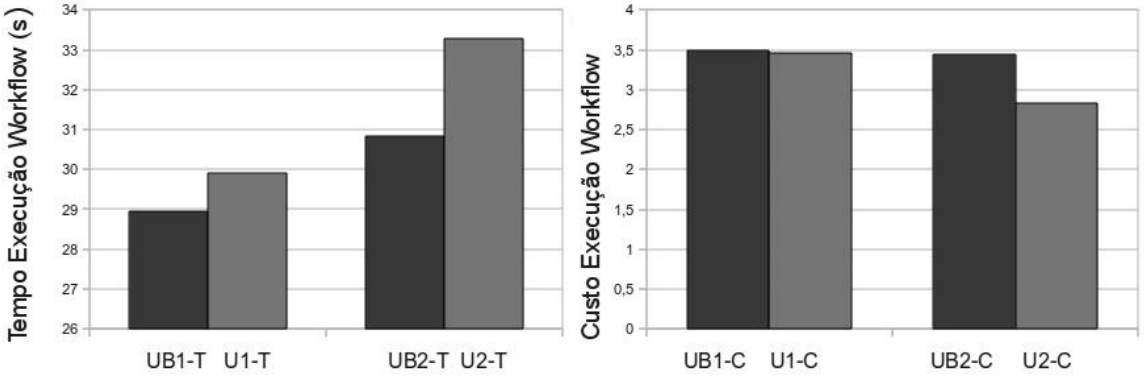


FIG. 5.7: Valores de tempo e custo dos usuários 1 e 2, em relação ao usuário base, no plano Básico com 20% de limite de variação

Para o plano básico, na figura 5.7, percebemos que na abordagem proposta houve uma perda de desempenho tanto para o usuário 1 quanto para o usuário 2 em relação ao modelo JSQ, 3,2% e 7,9% (U1-T e U2-T) respectivamente. Apesar disso, a perda manteve-se dentro do limite estabelecido e permitiu a otimização do critério secundário obtendo resultados melhores do que o JSQ para ambos os usuários em relação ao custo, sendo 0,9% para o usuário 1 e 17,7% para o usuário 2.

Em todos os experimentos foi possível notar que o modelo cumpriu a sua finalidade ao otimizar o tempo e o custo. Mesmo quando se mostrou pior em relação ao tempo quando comparado ao JSQ, este valor se manteve dentro do limite de variação especificado otimizando o custo. Há ainda que alertar a importância do valor escolhido para o limite de variação, nem sempre o maior valor significa a possibilidade de otimização.

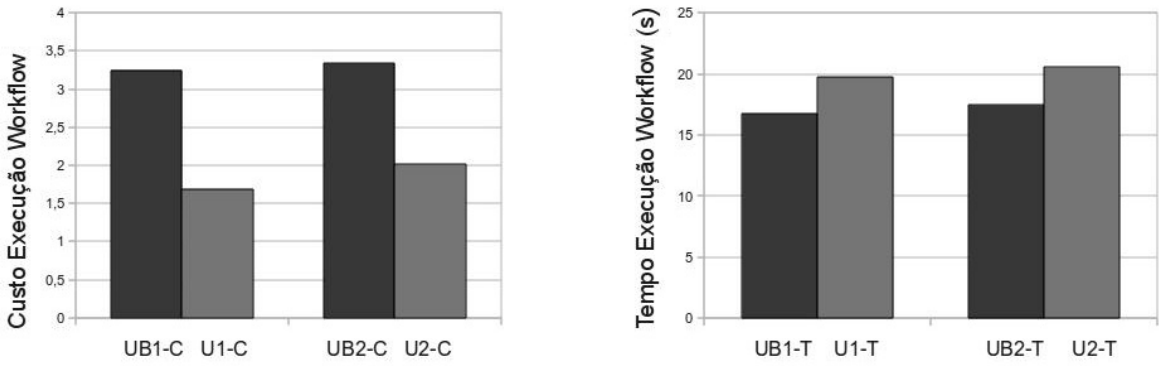


FIG. 5.8: Valores de custo e tempo dos usuários 1 e 2, em relação ao usuário base, no plano Premium com 20% de limite de variação

As figuras 5.8, 5.9 e 5.10 apresentam os resultados obtidos para os planos premium, master e básico em que os usuários tinham optado pelo custo como critério primário e o tempo como critério secundário, invertendo as prioridades apresentadas nos primeiros experimentos.

Quanto ao custo como critério primário, para o plano premium a figura 5.8 mostra que para ambos os usuários (U1-C e U2-C) o algoritmo proposto obteve uma significativa otimização para o custo, 43,8% e 39,9% respectivamente. Porém, já a otimização do critério secundário (tempo) gerou um ganho inferior ao obtido com o algoritmo JSQ. Este resultado era esperado pois, apesar do algoritmo procurar otimizar o critério secundário, é limitado pelo limite de variação do critério primário, assim, dificilmente ela será do mesmo nível da obtida com um algoritmo dedicado a apenas um critério, neste caso, a otimização do tempo.

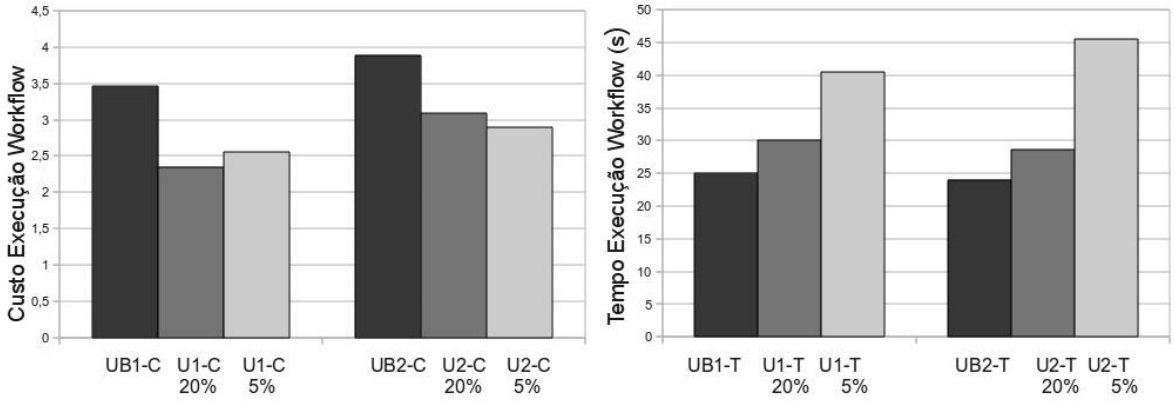


FIG. 5.9: Valores de custo e tempo dos usuários 1 e 2, em relação ao usuário base, no plano Master com 20% e 5% de limite de variação

Na figura 5.9 percebemos que para o plano master o modelo proposto alcançou um melhor desempenho tanto para 20% quanto para 5% de limite de variação. Nesta figura destaca-se que na maioria dos casos quanto menor for o valor de variação menor será a possibilidade de otimização para o critério secundário.

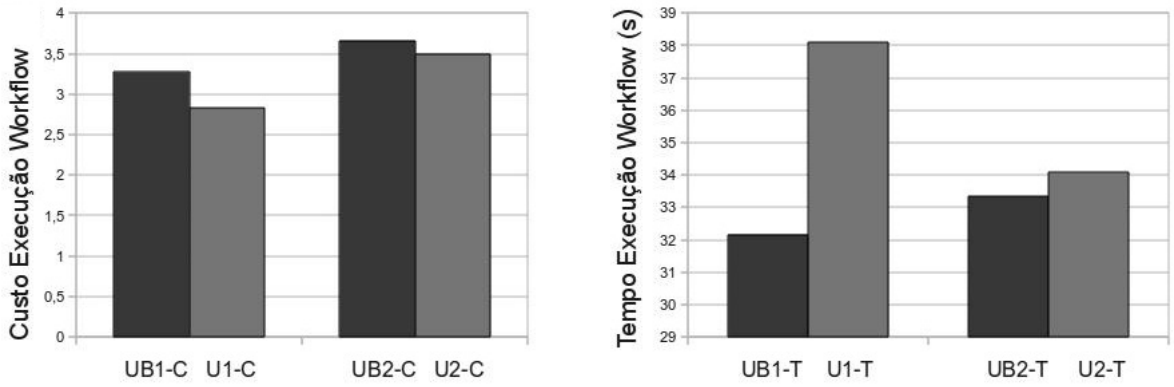


FIG. 5.10: Valores de custo e tempo dos usuários 1 e 2, em relação ao usuário base, no plano Básico com 20% de limite de variação

Na figura 5.10, em relação ao plano básico, foi possível otimizar o custo para os usuários 1 e 2 (U1-C e U2-C), obtendo um ganho de 13,6% e 4,3% respectivamente. Porém, a otimização do critério secundário (tempo) foi inferior 18,4% e 2,1% respectivamente (U1-T e U2-T), ao comparar-se com o tempo de execução do algoritmo JSQ.

Nos primeiros experimentos foi possível notar que quando é utilizado o tempo como critério primário, é alcançada uma maior otimização para o custo como critério secundário. Isso se dá pelo fato do tempo estar diretamente relacionado ao custo, sendo a base para o seu cálculo. Nos últimos experimentos apresentados foi possível notar que esta recíproca não é verdadeira, pois não é garantida a otimização do tempo, como critério secundário, quando o custo é considerado como critério primário.

As figuras 5.11 e 5.13 apresentam a variação nos valores de tempo de execução para o caso do limite de variação de 5% e 20%. É importante observar que ao utilizar a variação de apenas 5%, considerando o tempo como critério primário (Figura 5.11), o algoritmo tem um pico inicial que é explicado pelo seu período de aprendizagem, porém após este pico, ele estabiliza em um mesmo valor igual ao da variação de 20%. Entretanto, apresentou uma variância maior (6,9) se comparado ao de 20%(2,86). Ainda foi possível notar que, desconsiderando o período de aprendizagem, ambos obtiveram valores de variância melhores do que o algoritmo JSQ, como mostrado no quadro apresentado na Figura 5.12.

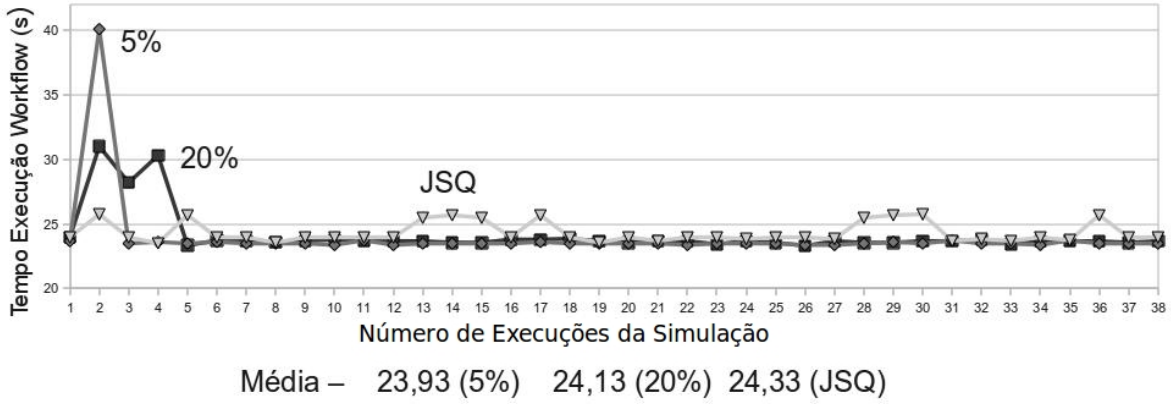


FIG. 5.11: Valores tempo no plano Master com 20% e 5% de limite de variação em relação ao JSQ

Abordagens	Considerando o período de aprendizado	Desconsiderando o período de aprendizado
5%	6,9	0,008
20%	2,86	0,01
JSQ	0,6	0,6

FIG. 5.12: Valores da variância das abordagens apresentadas

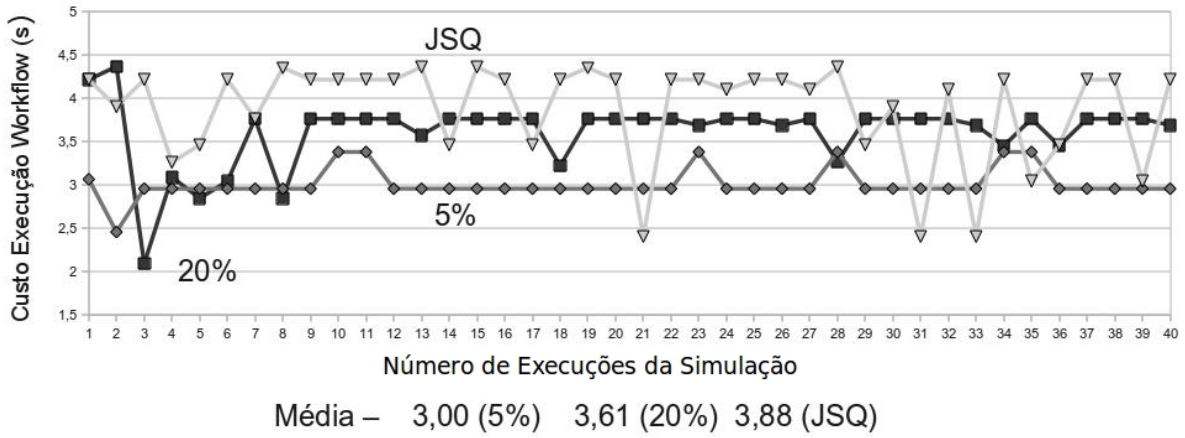


FIG. 5.13: Valores custo no plano Master com 20% e 5% de relaxação em relação ao JSQ

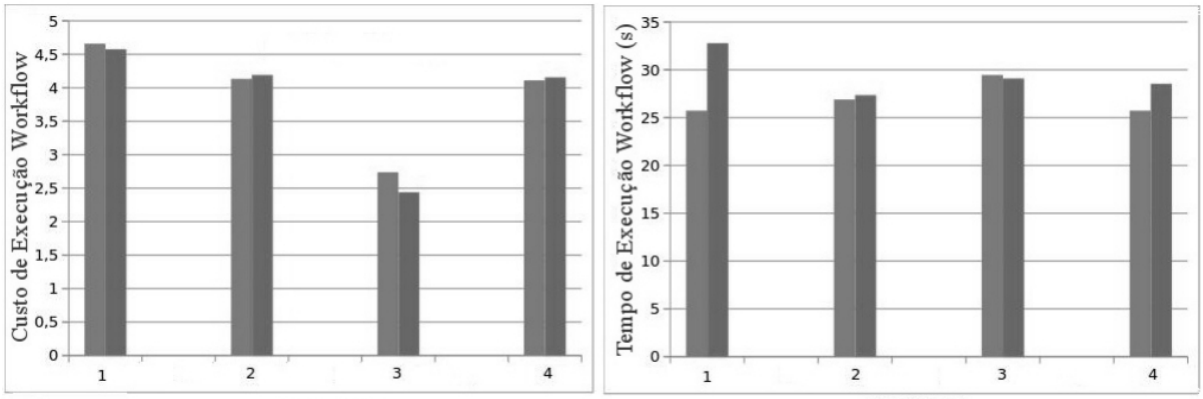
Quanto ao custo, como mostrado na figura 5.13, pode-se observar que ambos os limites de variação obtiveram melhores resultados do que o JSQ, inclusive destaca-se uma menor variabilidade para o limite de 5%, mostrando-se assim superior aos demais. Este resultado comprova que o algoritmo proposto consegue otimizar o custo para o usuário. Também

levanta o questionamento de quanto seria o valor ótimo para o limite de variação.

5.2.2 EXPERIMENTO 2

A seguir serão apresentados os valores obtidos com a inclusão do tempo de comunicação entre as tarefas do *workflow* que levam em consideração os efeitos do agrupamento de tarefas. Os gráficos nas figuras 5.14 e 5.15 apresentam os valores de tempos e custos para os usuários 1 e 2 respectivamente para as seguintes abordagens:

- (1) Abordagem sem o agrupamento de tarefas utilizando o algoritmo JSQ.
- (2) Abordagem com o agrupamento de tarefas utilizando o algoritmo JSQ.
- (3) Abordagem com o agrupamento de tarefas utilizando o algoritmo bi-critérios com limite de variação de 5%.
- (4) Abordagem com o agrupamento de tarefas utilizando o algoritmo bi-critérios com limite de variação de 20%.



CUSTO

	1	2	3	4
Usuário 1	4,66	4,14	2,74	4,11
Usuário 2	4,57	4,19	2,44	4,15

TEMPO

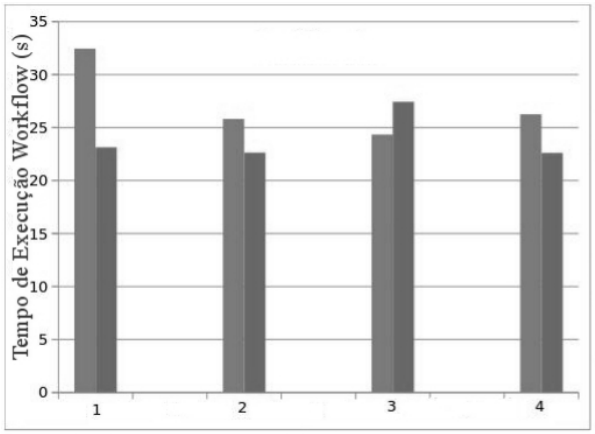
	1	2	3	4
Usuário 1	25,73	26,89	29,46	25,73
Usuário 2	32,79	27,35	29,09	28,53

FIG. 5.14: Resultados obtidos pela inclusão do agrupamento de tarefas com usuários priorizando custo e tempo

Na análise do gráfico sobre tempo de execução apresentado na figura 5.14 foi possível notar que para o usuário 1 nas abordagens 1 e 4 obtiveram valores iguais. Ao comparar a abordagem 3 (5% de variação) com a abordagem 1, esta obteve valores de tempo 14,49%

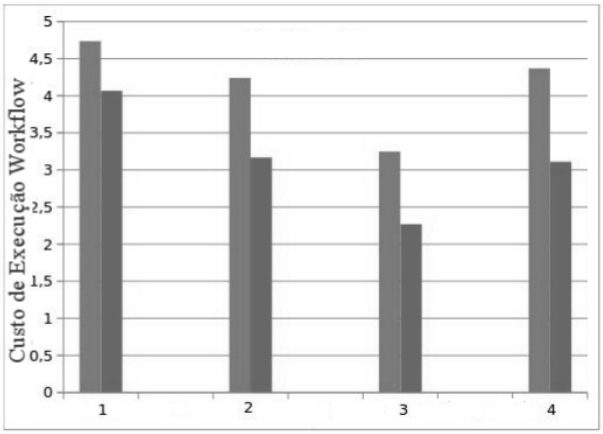
maior para o usuário 1. Estes valores são esperados considerando que a abordagem 3 tem por objetivo principal a otimização do custo como critério primário e o tempo como critério secundário. Com relação a abordagem 2, apesar de neste caso ter apresentado valores 4,5% maior, para o usuário 2 a mesma foi 16,57% menor do que a abordagem 1. Para o usuário 2 destaca-se a diferenciação do tempo nas abordagens que utilizam o agrupamento de tarefas em relação a JSQ sem o agrupamento, esta inclusive chegou a apresentar valores até 16% maior do que as demais abordagens. Ao utilizarmos o agrupamento de tarefas, é possível notar que a diferença de tempo entre os usuários 1 e 2 diminui, isto se deve à otimização que é feita com o usuário 2, que possui arquivos maiores em relação ao usuário 1. Essa diferença de tamanho entre o usuário 1 e 2 fica clara quando comparamos o tempo destes usuários na abordagem JSQ sem o agrupamento e utilizando o agrupamento, que torna os tempos dos usuários 1 e 2 mais próximos.

Quanto ao custo foi possível notar que o agrupamento de tarefas também influenciou positivamente o resultado, chegando a obter um valor de 41,25% melhor para o usuário 1 e 46,6% para o usuário 2. Foi possível notar também que para o limite de variação de apenas 5% a otimização do critério primário (custo) ficou cerca de 40% menor do que as demais porém o seu critério secundário (tempo) foi superior.



TEMPO

	1	2	3	4
Usuário 1	32,45	25,82	24,34	26,26
Usuário 2	23,13	22,63	27,43	22,60



CUSTO

	1	2	3	4
Usuário 1	4,74	4,24	3,25	4,37
Usuário 2	4,07	3,17	2,27	3,11

FIG. 5.15: Resultados obtidos pela inclusão do agrupamentode tarefas com usuários priorizando tempo e custo

A figura 5.15 apresenta os valores de tempo e custo para os usuários 1 e 2 considerando as abordagens acima citadas, porém alterando a prioridade dos critérios (tempo como

critério primário e custo como secundário). Para o usuário 1 foi possível notar o ganho significativo em relação ao tempo quando utilizamos o agrupamento de tarefas (este ganho chegou a ser de 24,9% em relação às demais abordagens). Em relação ao usuário 2 o tempo foi semelhante na maioria das abordagens com exceção da abordagem 3 que apresentou tempos até 18,6% maior para o usuário 2.

Comparando-se a abordagem 1 e 2 foi possível, novamente, notar a importância do agrupamento de tarefas pois ambas utilizam o mesmo algoritmo de seleção de recursos (JSQ), diferenciando-se somente pelo agrupamento. A abordagem com o agrupamento obteve valores de até 22,11% menores quanto ao custo em relação à abordagem 1. A abordagem que apresentou os melhores resultados de custo foi a abordagem 3 (5% de variação) que obteve ganhos de até 44,23%. A princípio, a otimização do critério secundário deveria ser melhor com 20% de limite de variação, porém neste experimento foi possível ver que em alguns casos a abordagem com 5% é melhor. Percebeu-se que este fato se dá nos casos em que a rede é o principal fator limitador. A abordagem de 20% distribuiu melhor as tarefas do *workflow* pelos diversos recursos, pois com um limite de variação maior a tendência é que haja mais recursos que atendam aos requisitos estabelecidos, podendo assim distribuir a carga entre estes. Já a abordagem com 5% tende a centralizar mais as tarefas de um mesmo *workflow* em um só recurso, pois a redução na variação limita a quantidade de recursos que atendem às necessidades especificadas pelo usuário, limitando assim a um conjunto menor de recursos, em alguns casos, a somente um recurso.

Em função dos resultados anteriormente apresentados a utilização do agrupamento de tarefas em conjunto com a abordagem bi-critérios, mostrou-se mais vantajosa melhorando a utilização dos recursos assim como a otimização dos critérios tempo e custo.

5.2.3 EXPERIMENTO 3

Os gráficos apresentados nas figuras 5.16, 5.17 e 5.18 apresentam o nível de utilização dos recursos nos experimentos realizados para os planos básico, master e premium. Com a adição do critério confiabilidade foram realizados experimentos considerando 8 usuários e 12 recursos, com 2 núcleos cada e capacidade de 100, 150 e 200 MIPS para os planos básico, master e premium respectivamente, totalizando 24 usuários e 36 recursos. Desses foram escolhidos arbitrariamente 6 recursos e 4 usuários para análise dos resultados. Quanto aos usuários, foram utilizadas as mesmas especificações para cada plano de serviço facilitando a análise comparativa entre os mesmos. Nos gráficos de ocupação dos recursos foi utilizada

a escala logaritma a fim de facilitar o entendimento e a sua análise. Em cada gráfico os recursos A, B e C representam recursos que estão mais sujeitos a falhas e os recursos D, E e F correspondem aos que não falharam em nenhum momento. Recursos considerados como falhos, são aqueles que por pelo menos um período de tempo do experimento sofreram uma perda de desempenho, simulado por meio da redução da sua capacidade de MIPS. O recurso passa então por um período com sua capacidade reduzida mas volta à sua capacidade real depois de algum tempo. Assim, foi estipulado que do conjunto de recursos que estão mais sujeitos a falhas, aleatoriamente são selecionados, a cada vez, dois destes e é reduzido assim a sua capacidade em MIPS. Isto faz com que haja recursos com um maior pencentual de falhas do que outros. Em cada gráfico é apresentada uma amostragem de três recursos mais sujeitos a falhas e três que não sofreram este problema.

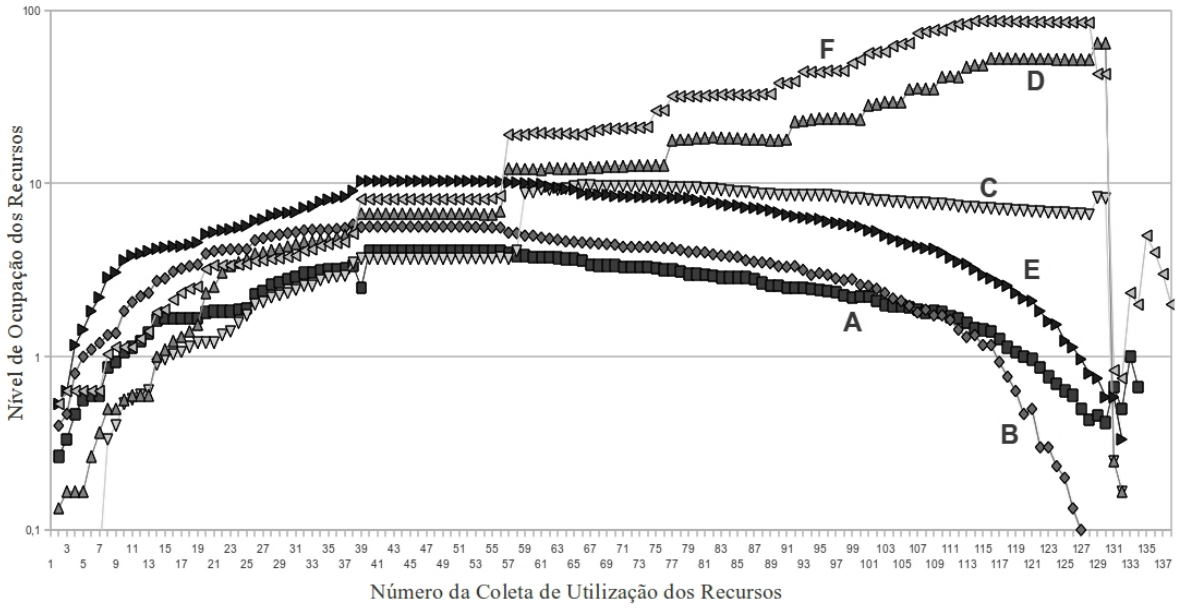


FIG. 5.16: Nível de utilização dos recursos do Plano Básico

O gráfico 5.16 apresenta a utilização dos recursos do plano básico. Nele podemos ver que os recursos A, B e C apresentam falhas e possuem utilização inferior aos recursos D, E e F que não falharam. Porém o recurso C, a partir do instante 60, começou a receber mais tarefas até mesmo do que o recurso E que é considerado confiável, o motivo para isto é que este recurso foi menos sujeito a falha do que os outros dois e possuía uma boa média de custo ou tempo. Outro fator que contribuiu é que neste experimento 50% dos usuários consideraram como critério primário a confiabilidade e os outros 50%

consideraram o tempo ou custo. Assim sendo, os usuários que priorizaram o tempo ou o custo tendem a utilizar estes recursos falhos por oferecerem um alto desempenho devido a sua baixa concorrência, reduzindo neste caso a importância da confiabilidade.

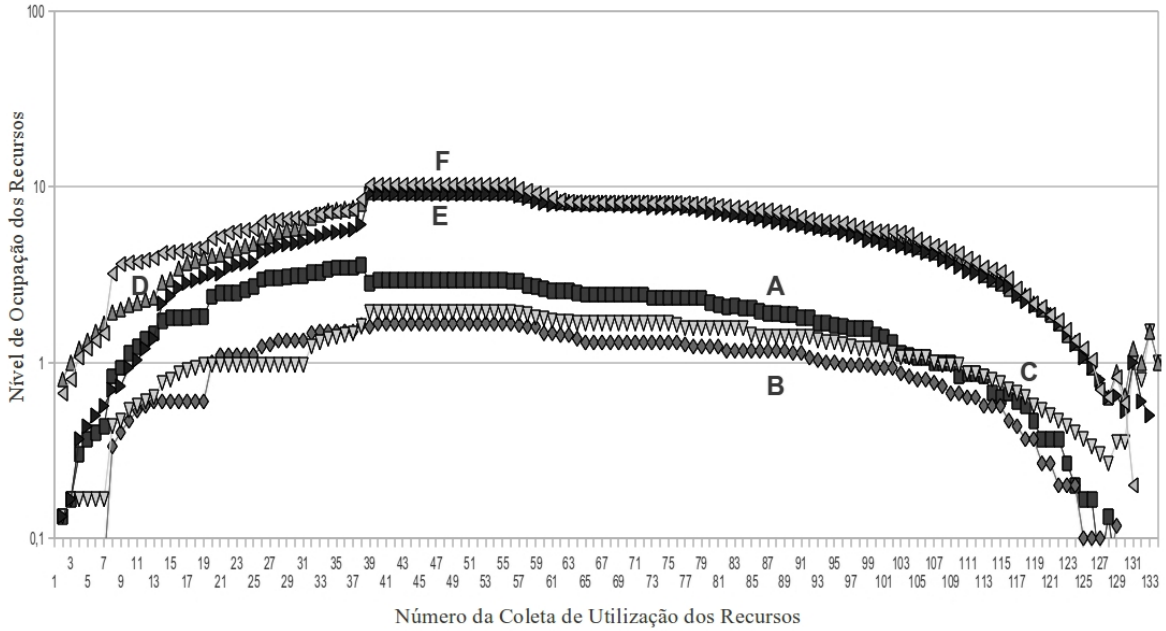


FIG. 5.17: Nível de utilização dos recursos do Plano Master

O gráfico apresentado na figura 5.17, apresenta a utilização dos recursos para o plano master. Neste gráfico é possível notar a diferença quanto a utilização dos recursos considerados mais e menos confiáveis, onde os recursos considerados mais confiáveis são mais utilizados do que os demais e há um balanceamento de carga entre estes. Outro detalhe importante é quanto ao recurso A que, apesar de estar sujeito a falhas, não lhe foi atribuído tantas falhas quanto aos recursos B e C e assim recebeu uma quantidade maior de tarefas. É possível notar também algumas variaçãos maiores em alguns recursos. Quando esta variação é positiva, deve-se a este estar recebendo mais tarefas por ser considerado mais confiável (recursos E e F, coleta 39). Quando esta variação é negativa, deve-se a degradação de desempenho do mesmo (recurso A, coleta 39).

O gráfico 5.18 apresenta a utilização para o plano Premium. Neste gráfico, além da diferenciação quanto a utilização dos planos, destacam-se dois detalhes importantes. Primeiro, o recurso A obteve um índice de utilização superior aos outros dois recursos com falhas pelos mesmos motivos apresentados anteriormente. Neste gráfico foi possível perceber que alguns recursos falhos são utilizados mesmo considerando a confiabilidade na

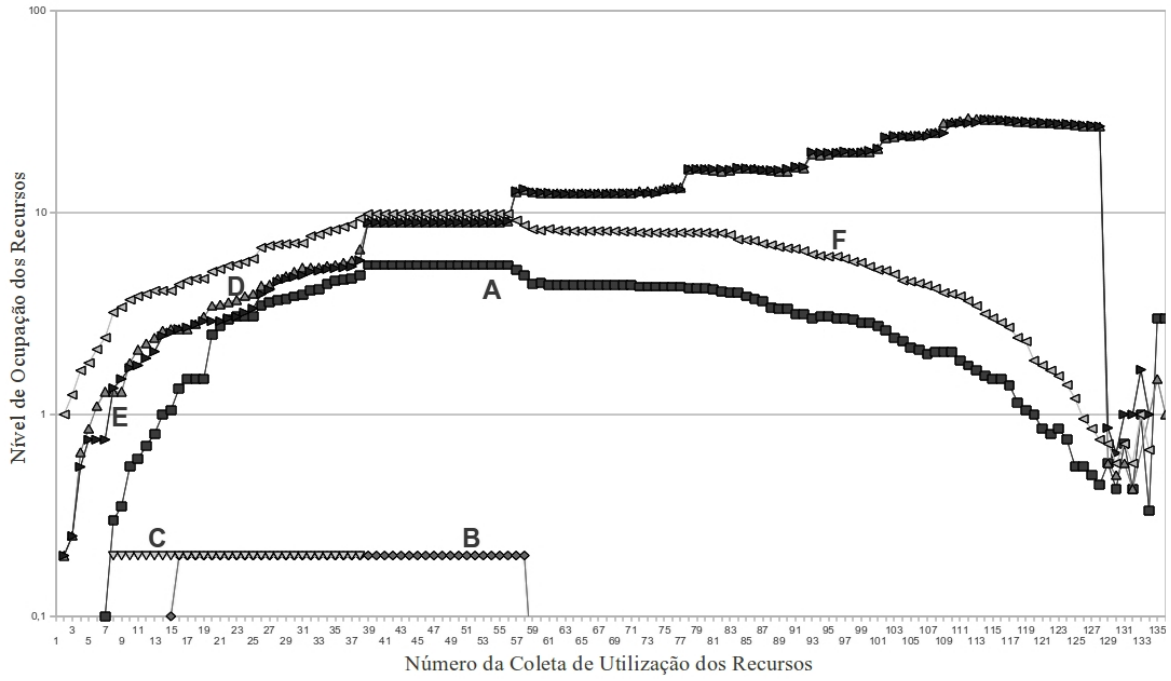


FIG. 5.18: Nível de utilização dos recursos do Plano Premium

seleção dos recursos, isto se dá devido aos usuários que têm como prioridade a otimização do tempo ou custo, o que causa a seleção dos recursos menos utilizados devido ao seu baixo nível de confiabilidade. O segundo detalhe importante a ser analisado é quanto à migração de tarefas. No instante 39, é possível notar que cessa a utilização do recurso C com o respectivo aumento da utilização dos recursos D e E, fruto da migração das tarefas alocadas nos recursos que tiveram degradação naquele instante. O mesmo fato torna a se repetir para o recurso B no instante 58 deslocando-se para os recursos E e F.

Nos parágrafos anteriores foram apresentados o efeito do critério de confiabilidade na utilização dos recursos. A seguir será apresentado o efeito que a otimização deste critério acarretou em termos de tempo e custo para cada um dos planos de serviço.

Os gráficos da figura 5.19 apresentam os valores de tempo e custo para o usuário 1 que tem como critério primário o tempo e secundário a confiabilidade para os planos básico, master e premium. Neste gráfico foi possível notar a otimização de até 30,6% para o tempo ao ser comparado com o algoritmo JSQ. Com relação a confiabilidade, apesar desta se contra por ao critério tempo, foi possível mesmo assim, otimizar o critério tempo (primário) em função da existência do limite de variação, permitindo selecionar recursos que apesar de menos confiáveis são menos concorridos, alcançando assim um

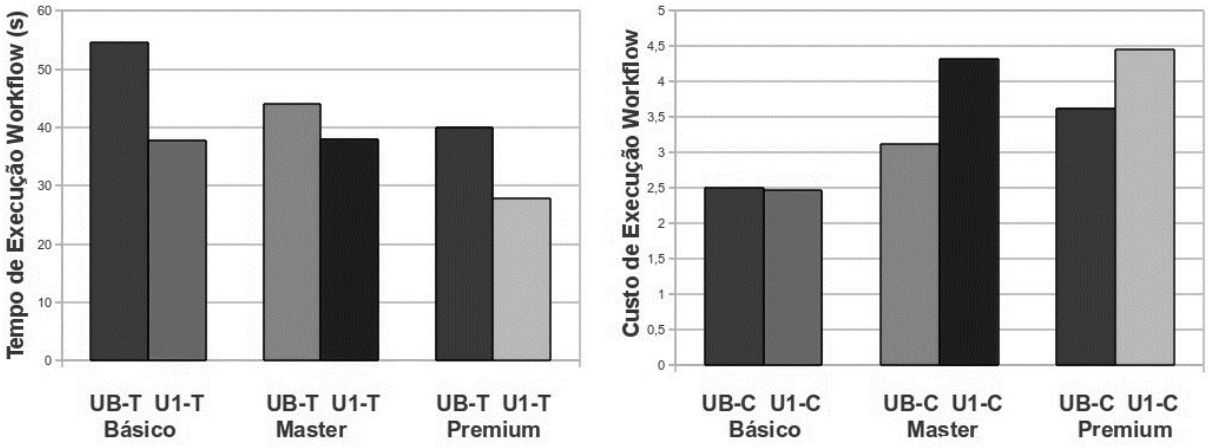


FIG. 5.19: Valores de tempo e custo para os planos básico, master e premium para o usuário 1 JSQ e modelo proposto com 10% de variação

ponto intermediário que permitiu a otimização de ambos. Desta forma não alcançamos o melhor resultado para apenas um dos critérios, mas sim, o melhor resultado que atenda a ambos os critérios selecionados. Quanto ao custo, que não foi otimizado para este usuário, obteve-se um pequeno ganho de 1,34% para o plano básico e perdas de 38,5% e 22,6% para os planos master e premium respectivamente como era esperado pois o objetivo era, em primeiro lugar, a otimização do tempo.

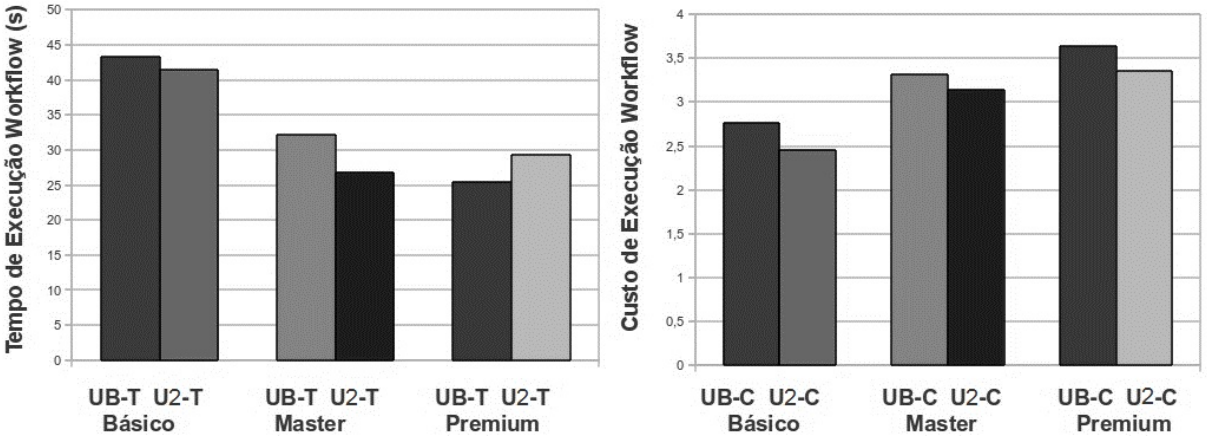


FIG. 5.20: Valores de tempo e custo para os planos básico, master e premium para o usuário 2 JSQ e modelo proposto com 10% de variação

A figura 5.20 apresenta os valores de tempo e custo para o usuário 2 nos três planos utilizados. Este usuário visa a otimização do nível de confiabilidade e do tempo como critérios primário e secundário, respectivamente. Neste gráfico foi possível notar a otimização nos planos básico e master de até 16,58% e 11,2% para tempo e custo respectivamente. Para

o plano premium, o tempo não obteve uma otimização se comparado ao algoritmo JSQ, obtendo inclusive uma perda de 15,52%. Em compensação, quanto ao custo, o usuário obteve um ganho de cerca de 7,6% em relação ao usuário base. Nestes gráficos destaca-se o fato de mesmo priorizando a confiabilidade como critério primário foi possível otimizar na maioria dos casos o tempo e até mesmo o custo.

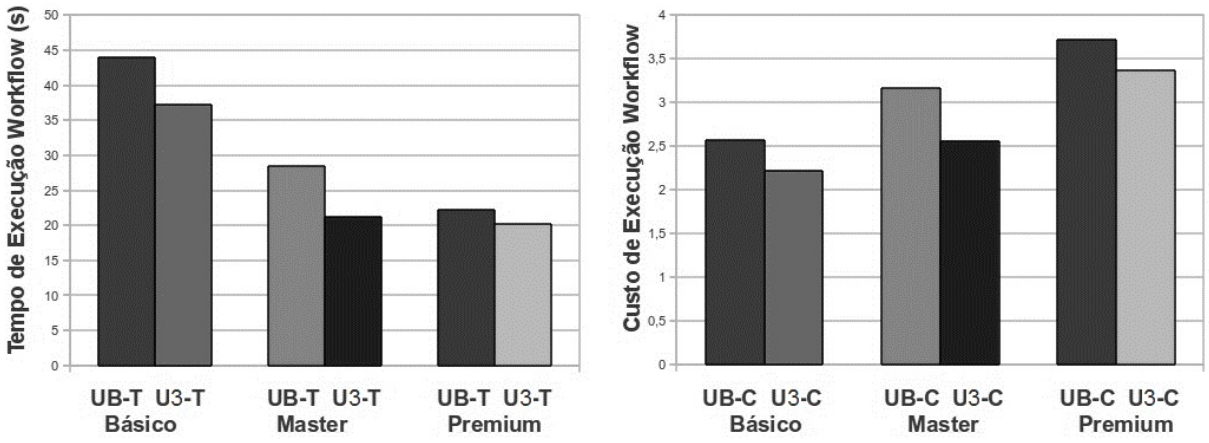


FIG. 5.21: Valores de tempo e custo para os planos básico, master e premium para o usuário 3 JSQ e modelo proposto com 10% de variação

Na figura 5.21 são apresentados os gráficos de tempo e custo para o usuário 3 nos três planos apresentados. O usuário 3 visa a otimização do nível de confiabilidade e do custo como critérios primário e secundário respectivamente. Para este usuário foi possível otimizar tanto o tempo quanto o custo para os três planos de serviço, alcançando os melhores resultados para o plano master onde o ganho foi de 25,3% para o tempo e 18,95% para o custo em relação ao usuário base.

A figura 5.22 apresenta os gráficos de tempo e custo para o usuário 4 para os planos básico, master e premium otimizando como critério primário o custo e secundário a confiabilidade. Para o plano básico não foi possível obter a otimização se comparado ao modelo JSQ obtendo perdas de 3,42% e 18,9% para tempo e custo respectivamente. Já nos planos master e premium foi possível obter ganhos de até 28,03% para o tempo e 6,83% para o custo. Foi possível ver neste usuário então, a questão da priorização dos recursos com falhas devido a sua baixa utilização objetivando o menor custo e, por consequência, ao utilizar estes recursos menos concorridos, obtem-se um menor tempo.

Nestes últimos experimentos foram levados em consideração a otimização dos critérios de tempo, custo e confiabilidade. Por meio dos resultados obtidos foi possível verificar que critérios como confiabilidade e tempo são, de certa forma, contraditórios e necessitam

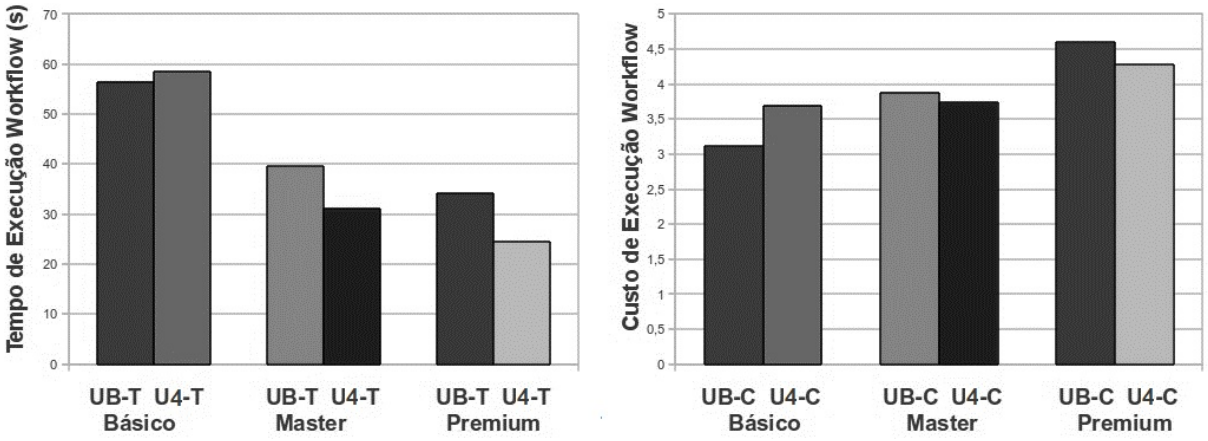


FIG. 5.22: Valores de tempo e custo para os planos básico, master e premium para o usuário 4 JSQ e modelo proposto com 10% de variação

ser tratados de forma independente, não impedindo porém serem trabalhados de forma cooperativa.

5.2.4 CONCLUSÃO DA ANÁLISE

Critérios tais como tempo e custo podem ser trabalhados em conjunto por estarem co-relacionados e serem selecionados sobre o perfil de tarefa/usuário. Já critérios como o nível de confiabilidade, estão relacionados aos recursos e a seleção deve ser tratada de uma forma diferente, não relacionada ao perfil da tarefa, mas sim ao perfil do recurso. Desta forma, dificilmente será possível alcançar a otimização de todos critérios, mas é possível alcançar uma solução satisfatória para cada um destes por meio do balanceamento dos valores possíveis de serem obtidos.

6 CONSIDERAÇÕES FINAIS

Este capítulo tem o objetivo de explicitar as conclusões obtidas com a realização deste trabalho, assim como apresentar as contribuições alcançadas e os possíveis trabalhos futuros identificados.

6.1 CONCLUSÕES

A proposta deste trabalho foi a elaboração de um algoritmo de escalonamento de tarefas que otimize mais de um critério, procurando combinar características das propostas anteriores, destacando-se condicionar a distribuição das tarefas em função de uma ordem de prioridade para os critérios definidos pelos usuários. Pelos resultados obtidos neste trabalho foi possível a comprovação da possibilidade de utilização de um algoritmo de escalonamento multi-paradigmas e multi-critérios em ambientes de Nuvens e Grades. Podemos considerá-lo multi-paradigma por utilizar diversas abordagens trabalhando de forma cooperativa, tais como: utilização de modelos de CoS, agrupamento de tarefas, e escalonamento hierárquico e hibrído. Podemos considerá-lo multi-critérios por utilizar uma abordagem de seleção de recursos que leva em consideração mais de um critério.

Neste trabalho apresentou-se um modelo que possibilita a seleção de recursos através de uma abordagem em que dois critérios são utilizados como parâmetro de seleção. Nesta abordagem o usuário é capaz de selecionar, dentre um conjunto de critérios, os dois critérios que considera mais importantes e estabelecer uma prioridade entre estes. Este método possibilitou alcançar resultados promissores com o modelo de *workflow* utilizado para os critérios selecionados, ao ser permitido ao usuário estabelecer um limite de variação para o critério primário de forma a também otimizar o critério secundário, semelhante aos alcançados através das abordagens dedicadas a cada um destes critérios, separadamente.

Nas simulações realizadas foi submetido um modelo de *workflow* DAG em que se procurou avaliar a otimização dos critérios tempo, custo e confiabilidade dos recursos. Um dos principais critérios a ser considerado na execução de *workflows*, o critério de banda passante, foi otimizado pelo algoritmo, em um primeiro passo, através do uso do agrupamento de tarefas. Durante os experimentos percebeu-se ainda a importância do limite de variação estipulado pelo usuário para a otimização e solução final do problema.

Nos experimentos comparou-se o algoritmo proposto com o algoritmo JSQ, utilizado como base de referência. A análise destes experimentos demonstrou que o algoritmo foi capaz de otimizar os critérios de acordo com os parâmetros estabelecidos pelo usuário.

Através dos experimentos foi possível também notar a influência positiva do agrupamento de tarefas para o critério de banda passante, reduzindo a transferência de arquivos intermediários pela rede e contribuindo para a melhoria da utilização dos recursos. Quando considerado este critério, foi possível notar ganhos de até 24,9% para o custo e 44,23% para o tempo. Com relação aos critérios otimizados através da abordagem bi-critérios, foi possível notar a melhora dos resultados obtidos em cada um destes, se comparados aos valores obtidos com a abordagem JSQ. Dos valores encontrados, destacam-se os valores de custo que apresentaram ganhos de até 47,60%.

Quanto ao tempo, os resultados foram considerados satisfatórios, pois obteve-se valores próximos ao encontrado com o algoritmo JSQ que visa a otimização apenas do critério de tempo. Para este critério, a abordagem proposta obteve ganhos de até 24,9% quando o tempo foi considerado como critério primário e o custo como critério secundário.

No que se refere ao critério de confiabilidade foi possível notar a diferenciação quanto a utilização dos recursos que foram considerados mais confiáveis e menos confiáveis com o uso do algoritmo proposto.

De maneira geral, os objetivos principais do trabalho foram alcançados, uma vez que o modelo de escalonamento bi-critérios com suporte a CoS para ambientes de Grades proposto atendeu a demanda do escalonamento de aplicações de *workflows*.

6.2 CONTRIBUIÇÕES ALCANÇADAS

Podemos em função dos resultados obtidos destacar as seguintes contribuições:

- Criação de um modelo capaz de oferecer uma abordagem de escalonamento que trabalha com o agrupamento de tarefas em um primeiro momento a fim de otimizar a utilização dos recursos e reduzir o consumo de banda passante da rede, cuja seleção desses recursos atenda a um modelo de qualidade de serviço com base em um algoritmo de seleção bi-critérios.

- Validação da proposta de otimização no escalonamento diferenciando os critérios selecionados por cada usuário e suas prioriades, baseado em planos de serviço e que

levou em consideração a confiabilidade dos recursos utilizados contribuindo para o aperfeiçoamento de propostas e modelos desenvolvidos sobre o assunto.

6.3 TRABALHOS FUTUROS

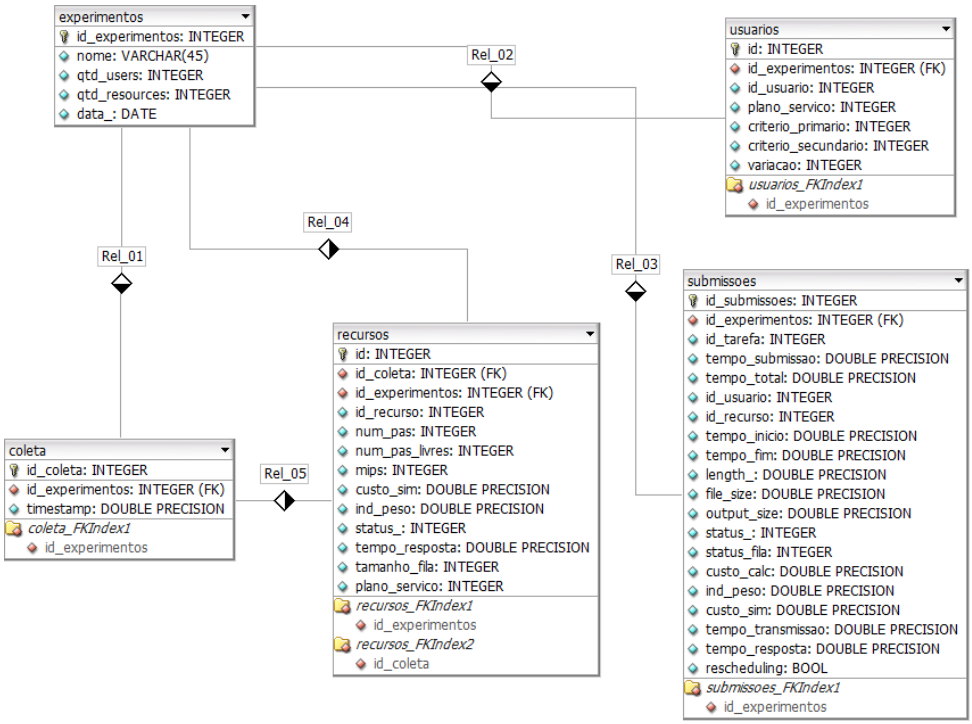
O fato de que os objetivos iniciais estabelecidos para este trabalho tenham sido alcançados e apresentados na forma de contribuições para a comunidade científica não significa que o modelo aqui apresentado não possa ser aperfeiçoado ou que tenha esgotado o assunto. Novas propostas neste contexto podem ser apresentadas cobrindo algumas lacunas. Os itens a seguir foram identificados como possíveis trabalhos futuros:

- Realização de experimentos com o modelo proposto considerando diferentes modelos de *workflow* DAG.
- Estudo da abordagem proposta em um ambiente de Nuvens a fim de verificar o seu comportamento.
- Implantação do modelo proposto em ambiente real de Grades ou Nuvens.
- Estudo e definição de novos critérios a serem incorporados ao modelo.
- Estudos e aplicação de novas estratégias de escalonamento para cada critério utilizado.
- Aprimoramento do modelo de agrupamento de tarefas considerando dependências indiretas.
- Desenvolvimento de um algoritmo que seja capaz de encontrar o melhor limite de variação do critério primário para cada usuário.

7 APÊNDICE A

7.1 BANCO DE DADOS

A figura abaixo representa a modelagem do banco de dados utilizado neste trabalho.



8 APÊNDICE B

8.1 *SCHEMA* DO XML

Neste anexo é apresentado o *schema* usuario.xsd que define os XMLs de representação dos parâmetros de cada usuário.

```
- <xsd:schema>
  - <xsd:element name="User">
    - <xsd:complexType>
      - <xsd:sequence>
          <xsd:element ref="planoServico" minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="criterioPrimario" minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="criterioSecundario" minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="variacao" minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="Workflow" minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
        <xsd:attribute name="name" type="xsd:string" use="required"/>
      </xsd:complexType>
    </xsd:element>
  - <xsd:element name="Workflow">
    - <xsd:complexType>
      - <xsd:sequence>
          <xsd:element ref="Task" minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  - <xsd:element name="Task">
    - <xsd:complexType>
      - <xsd:sequence>
          <xsd:element ref="lenght" minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="fileSize" minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="outputSize" minOccurs="1" maxOccurs="1"/>
          <xsd:element ref="dependencies" minOccurs="0" maxOccurs="unbounded"/>
          <xsd:element ref="nextTasks" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
        <xsd:attribute name="id" type="xsd:integer" use="required"/>
      </xsd:complexType>
    </xsd:element>
  - <xsd:element name="dependencies">
    - <xsd:complexType>
      - <xsd:sequence>
          <xsd:element ref="id" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  - <xsd:element name="nextTasks">
    - <xsd:complexType>
      - <xsd:sequence>
          <xsd:element ref="id" minOccurs="0" maxOccurs="unbounded"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="planoServico" type="xsd:string"/>
```

```
    <xsd:element name="criterioPrimario" type="xsd:string"/>
    <xsd:element name="criterioSecundario" type="xsd:string"/>
    <xsd:element name="variacao" type="xsd:integer"/>
    <xsd:element name="lenght" type="xsd:integer"/>
    <xsd:element name="fileSize" type="xsd:integer"/>
    <xsd:element name="outputSize" type="xsd:integer"/>
    <xsd:element name="id" type="xsd:integer"/>
</xsd:schema>
```

9 APÊNDICE C

9.1 SIMULADOR GRIDSIM

O simulador *GridSim* é uma API que foi utilizada a fim de possibilitar a simulação do ambiente de Grade. Neste trabalho foi desenvolvida uma aplicação que utiliza uma versão alterada desta API com o intuito de solucionar algumas das restrições do simulador original e adicionar as funcionalidades necessárias a este trabalho. Para utilização do *GridSim* algumas rotinas e classes do simulador tiveram que ser inseridas ou alteradas.

O simulador tem como principais características:

- Permite a simulação de ambientes de Grades e P2P, onde cada "personagem" (usuários, recursos) é uma entidade do simulador (uma *Thread*) e trocam mensagens entre si.
- Permite a definição do ambiente de rede utilizado e a simulação do tráfego desta rede.
- Permite a criação dos usuários e tarefas relacionadas ao mesmo.
- Cada usuário possui acesso direto para alocação de suas tarefas nos respectivos recursos, caracterizando um escalonamento descentralizado.
- Não oferece suporte a aplicações *workflow*.

Para propiciar um ambiente de escalonamento centralizado e hierárquico, foram criadas duas novas classes que foram inseridas na aplicação desenvolvida utilizando a API apresentada, o Escalonador Global e o Escalonador Local. Cada um destes possui uma fila do tipo FIFO (*First In, First Out*) na qual as tarefas são alocadas. No escalonador local de cada recurso é realizado o controle de execução do *workflow*. Antes da tarefa ser executada é verificado se todas as suas dependências foram concluídas, caso alguma de suas dependências não tenha sido concluída, a tarefa retorna ao final da fila do mesmo recurso enquanto aguarda o término da execução de suas dependências.

Para poder dar suporte a aplicações de *workflows* e possibilitar a simulação da degradação de desempenho dos recursos, na API do *GridSim* foram acrescentadas as seguintes funcionalidades:

- Na classe que corresponde a cada tarefa, foram acrescentadas as informações descritas no XML do usuário sobre as relações existentes entre as tarefas que compõem o *workflow*. Sendo assim, cada tarefa possui as informações de quais tarefas são pré-requisitos para sua submissão e quais tarefas dependem de sua execução.

- Novos modelos de mensagens entre entidades do simulador foram criados para acréscimos de funcionalidades, tais como, a simulação de falhas através do envio de um sinal ao recurso para geração das mesmas e envio de um conjunto de tarefas para os escalonadores.

Através das alterações descritas acima, foi possível simular as seguintes características do modelo:

- Construir um ambiente de Grade com um modelo des escalonamento centralizado e híbrido.

- Submissão de aplicações *workflows*.

- Simulação de falhas dos recursos.

- Reescalonamento das tarefas.

- Criação de classes de recursos.

10 REFERÊNCIAS BIBLIOGRÁFICAS

ARMBRUST, M., FOX, A., GRIFFITH, R., JOSEPH, A. D., KATZ, R. H., KONWINSKI, A., LEE, G., PATTERSON, D. A., RABKIN, A., STOICA, I., e ZAHARIA, M. **Above the clouds: A berkeley view of cloud computing**. Technical Report UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009. URL `http://www.eecs.berkeley.edu/Pubs/TechRpts/2009/EECS-2009-28.html`.

BANDINI, M., MURY, A. R., SCHULZE, B., e SALLES, R. **A grid-qos decision support system using service level agreements**. Em *Congresso da Sociedade Brasileira de Computacao de 2009, CSBC 09*, 2009.

BERMAN, F., CASANOVA, H., CHIEN, A., COOPER, K., DAIL, H., DASGUPTA, A., DENG, W., DONGARRA, J., JOHNSSON, L., KENNEDY, K., KOELBEL, C., LIU, B., LIU, X., MANDAL, A., MARIN, G., MAZINA, M., MELLOR-CRUMMEY, J., MENDES, C., OLUGBILE, A., PATEL, J. M., REED, D., SHI, Z., SIEVERT, O., XIA, H., e YARKHAN, A. **New grid scheduling and rescheduling methods in the grads project**. *Int. J. Parallel Program.*, 33(2):209–229, 2005. ISSN 0885-7458.

BRAUN, T. D., SIEGEL, H. J., BECK, N., BOLONI, L. L., MAHESWARAN, M., REUTHER, A. I., ROBERTSON, J. P., THEYS, M. D., YAO, B., HENSGEN, D., e FREUND, R. F. **A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems**, 2001.

BUYYA, R. e MURSHED, M. M. **Gridsim: A toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing**. *CoRR*, cs.DC/0203019, 2002.

BUYYA, R. e SULISTIO, A. **Service and utility oriented distributed computing systems: Challenges and opportunities for modeling and simulation communities**. Em *ANSS-41 '08: Proceedings of the 41st Annual Simulation Symposium (anss-41 2008)*, págs. 68–81, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3143-4.

CASANOVA, H. **Simgrid: a toolkit for the simulation of application scheduling**. Em *Proceedings of the First IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid 2001*, págs. 430–437, 2001.

DE MENEZES, J. G. M. **Gerência distribuída de dados em workflows de bioinformática**. Dissertação de Mestrado, Instituto Militar de Engenharia, 2008.

DONG, F. ***Workflow scheduling algorithms in the Grid***. Tese de Doutorado, Queen's University, Abril 2009. URL

http://qspace.library.queensu.ca/bitstream/1974/1795/1/Dong_Fangpeng_200904_PhD.pdf.

DONG, F. e AKL, S. G. **Technical report no. 2006-504 scheduling algorithms for grid computing: State of the art and open problems**, 2006.

EL-REWINI, H., LEWIS, T. G., e ALI, H. H. ***Task scheduling in parallel and distributed systems***. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994. ISBN 0-13-099235-6.

FAHRINGER, T., PRODAN, R., DUAN, R., HOFER, J., NADEEM, F., NERIERI, F., QIN, J., SIDDIQUI, M., TRUONG, H.-L., VILLAZON, A., WIECZOREK, MAREK, e PODLIPNIG, S. **Askalon: A development and grid computing environment for scientific workflows**. Em TAYLOR, I. J., DEELMAN, E., GANNON, D., e SHIELDS, M. S., editores, *Scientific Workflows for Grids*, Workflows for e-Science. Springer Verlag, 2007. URL http://www.springer.com/dal/home/generic/search/results?SGWID=1-40109-22-173663723-0. ISBN-10: 1-84628-519-4, ISBN-13: 978-1-84628-519-6.

FOSTER, I. **The anatomy of the grid: Enabling scalable virtual organizations**. *INTERNATIONAL JOURNAL OF SUPERCOMPUTER APPLICATIONS*, 15(3):2001, 2001.

HOWELL, F. e MCNAB, R. **SimJava: a discrete event simulation package for Java with applications in computer systems modelling**. Em *First International Conference on Web-based modelling and simulation*. Society for Computer Simulation, 1998.

IKEDA, R., SALIHOGLU, S., e WIDOM, J. **Provenance-based refresh in data-oriented workflows**. Technical report, Stanford University, 2010. URL http://ilpubs.stanford.edu:8090/962/.

IOSUP, A., SONMEZ, O., ANOEP, S., e EPEMA, D. **The performance of bags-of-tasks in large-scale distributed systems**. Em *HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing*, págs. 97–108, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-997-5.

KEE, Y.-S., LOGOTHETIS, D., HUANG, R., CASANOVA, H., e CHIEN, A. A. **Efficient resource description and high quality selection for virtual grids**. Em *CCGRID '05: Proceedings of the Fifth IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05) - Volume 1*, págs. 598–606, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7803-9074-1.

KLUSÁCEK, D., MATYSKA, L., e RUDOVÁ, H. **Alea - grid scheduling simulation environment**. Em *PPAM*, págs. 1029–1038, 2007.

KUROWSKI, K., NABRZYSKI, J., OLEKSIAK, A., e WEGLARZ, J. **Grid scheduling simulations with gssim**. Em *ICPADS '07: Proceedings of the 13th International*

Conference on Parallel and Distributed Systems, págs. 1–8, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 978-1-4244-1889-3.

LIN, H.-C. e RAGHAVENDRA, C. **An approximate analysis of the join the shortest queue (jsq) policy**. *Parallel and Distributed Systems, IEEE Transactions on*, 7(3): 301 –307, mar 1996. ISSN 1045-9219.

MEYER, L. A. V. C., SCHEFTNER, D., VÍCKLER, J.-S., MATTOSO, M., WILDE, M., e FOSTER, I. T. **An opportunistic algorithm for scheduling workflows on grids**. Em *VECPAR*, volume 4395 of *Lecture Notes in Computer Science*, págs. 1–12. Springer, 2006. ISBN 978-3-540-71350-0. URL `http://dblp.uni-trier.de/db/conf/vecpar/vecpar2006.html`.

MEYER, L. A. V. C. ***Estratégias para o escalonamento dinâmico de workflows em Grid***. Tese de Doutorado, COPPE/UFRJ, Julho 2007. URL `http://teses.ufrj.br/COPPE_D/LuizAntonioVivacquaCorreaMeyer.pdf`.

MURY, A. R., SCHULZE, B., e GOMES, A. T. A. **Task distribution models in grids: towards a profile-based approach**. *Concurr. Comput. : Pract. Exper.*, 22 (3):358–374, 2010. ISSN 1532-0626.

PORDES, R., PETRAVICK, D., KRAMER, B., OLSON, D., LIVNY, M., ROY, A., AVERY, P., BLACKBURN, K., WENAUS, T., WARTHWEIN, F., FOSTER, I., GARDNER, R., WILDE, M., BLATECKY, A., MCGEE, J., e QUICK, R. **The open science grid**. *Journal of Physics: Conference Series*, 78(1):012057, 2007. URL `http://stacks.iop.org/1742-6596/78/i=1/a=012057`.

RAMAKRISHNAN, L. e REED, D. A. **Performability modeling for scheduling and fault tolerance strategies for scientific workflows**. Em *HPDC '08: Proceedings of the 17th international symposium on High performance distributed computing*, págs. 23–34, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-997-5.

WIECZOREK, M., PRODAN, R., HOHEISEL, A., WIECZOREK, M., PRODAN, R., e HOHEISEL, A. **Taxonomies of the multi-criteria grid workflow scheduling problem**, 2007.

WIECZOREK, M., HOHEISEL, A., e PRODAN, R. **Towards a general model of the multi-criteria workflow scheduling on the grid**. *Future Gener. Comput. Syst.*, 25 (3):237–256, 2009. ISSN 0167-739X.

WIECZOREK, M., PODLIPNIG, S., PRODAN, R., e FAHRINGER, T. **Bi-criteria scheduling of scientific workflows for the grid**. Em *CCGRID '08: Proceedings of the 2008 Eighth IEEE International Symposium on Cluster Computing and the Grid*, págs. 9–16, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3156-4.

XU, M., CUI, L., WANG, H., e BI, Y. **A multiple qos constrained scheduling strategy of multiple workflows for cloud computing**. *Parallel and Distributed Processing with Applications, International Symposium on*, 0:629–634, 2009.

YU, J. e BUYYA, R. **A taxonomy of scientific workflow systems for grid computing**. *SIGMOD Rec.*, 34(3):44–49, 2005. ISSN 0163-5808.

YU, J., BUYYA, R., e THAM, C. K. **Cost-based scheduling of scientific workflow application on utility grids**. Em *E-SCIENCE '05: Proceedings of the First International Conference on e-Science and Grid Computing*, págs. 140–147, Washington, DC, USA, 2005. IEEE Computer Society. ISBN 0-7695-2448-6.

YU, Z. e SHI, W. **An adaptive rescheduling strategy for grid workflow applications**. *Parallel and Distributed Processing Symposium, International*, 0:115, 2007.

YU, Z. e SHI, W. **A planner-guided scheduling strategy for multiple workflow applications**. Em *ICPPW '08: Proceedings of the 2008 International Conference on Parallel Processing - Workshops*, págs. 1–8, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3375-9.

ZHANG, Y., KOELBEL, C., e COOPER, K. **Hybrid re-scheduling mechanisms for workflow applications on multi-cluster grid**. Em *CCGRID '09: Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, págs. 116–123, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3622-4.

ZHANG, Y., KOELBEL, C., e KENNEDY, K. **Relative performance of scheduling algorithms in grid environments**. Em *CCGRID '07: Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid*, págs. 521–528, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2833-3.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo