ESTRATÉGIAS PARA O ESCALONAMENTO DINÂMICO DE WORKFLOWS EM GRID

Luiz Antônio Vivacqua Corrêa Meyer

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Profa. Marta Lima de Queirós Mattoso, D.Sc.

Prof. Jano Moreira de Souza, Ph.D.

Profa. Inês de Castro Dutra, Ph.D.

Prof. Nelson Francisco Favilla Ebecken, D.Sc.

Prof. João Eduardo Ferreira, D.Sc.

Prof. José Palazzo Moreira de Oliveira, D.Sc.

RIO DE JANEIRO, RJ, - BRASIL

JULHO DE 2006

MEYER, LUIZ ANTÔNIO VIVACQUA CORRÊA

Estratégias para o Escalonamento Dinâmico de Workflows em Grid [Rio de Janeiro] 2006

IX, 100 p. 29,7 cm (COPPE/UFRJ, D.Sc., Engenharia de Sistemas e Computação, 2006)

Tese – Universidade Federal do Rio de Janeiro, COPPE

1. Computação em Grade 2. Escalonamento de tarefas

I. COPPE/UFRJ II. Título (série)

AGRADECIMENTOS

Gostaria de agradecer a Profa. Marta Mattoso pela sua competência e empenho na orientação deste trabalho. A profa. Marta se mostrou sempre disponível, paciente e disposta a ajudar. O seu incentivo e os seus questionamentos foram fundamentais para a realização desta tese.

Também quero agradecer aos demais professores da Coppe com quem tive a oportunidade de conviver ao longo do mestrado e doutorado, e que de alguma forma contribuíram para que este trabalho fosse concluído. Agradeço também a equipe da secretaria do programa pela ajuda nas questões administrativas.

Agradeço ao Prof. Ian Foster pela oportunidade oferecida para trabalhar junto ao seu grupo de pesquisa na Universidade de Chicago durante o período do meu doutorado sanduiche e pela orientação nos trabalhos que realizamos juntos. Agradeço também a Jens Voeckler, Doug Shaftner e em especial a Mike Wilde que além da ajuda no trabalho de pesquisa, proporcionou a mim e a minha família todo o apoio na nossa estadia em Chicago.

Agradeço ao IBGE a licença concedida para que pudesse me dedicar ao programa de doutorado, e a CAPES e ao CNPq pelos auxílios financeiros concedidos.

Cristina e Viviane, obrigado pelo constante apoio, incentivo e compreensão ao longo desta jornada.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

ESTRATÉGIAS PARA O ESCALONAMENTO DINÂMICO DE WORKFLOWS EM GRID

Luiz Antônio Vivacqua Corrêa Meyer

Julho/2006

Orientador: Marta Lima de Queirós Mattoso
Programa: Engenharia de Sistemas e Computação

O escalonamento de tarefas é o aspecto preponderante para o desempenho de workflows científicos em Grid. A dinamicidade do Grid torna complexa a decisão de escalonamento de dados e programas. As soluções de escalonamento existentes em ambientes de Grid caracterizam-se por serem aleatórias, ou seja, não levam em consideração as mudanças no desempenho dos sítios que definem o ambiente de execução. Esta tese apresenta alternativas para o escalonamento dinâmico de tarefas para contemplar o comportamento variável do ambiente e permitir a modelagem de especificidades da aplicação. A solução apresentada utiliza um selecionador de sítios, cujo objetivo é escolher, dinamicamente, o local para executar cada tarefa do workflow no Grid. Duas estratégias para seleção de sítios foram desenvolvidas. Uma estratégia é baseada na criação de grupos de tarefas por proximidade espacial e a outra estratégia é oportunista. Estas estratégias foram avaliadas em experimentos realizados em um ambiente de simulação e em um ambiente real de Grid. Elas encontram-se disponíveis no sistema Euryale+ em sintonia com os sistemas Condor e DagMan, sobre o ambiente operacional de Grid Globus. Foram também desenvolvidos um simulador para execução de workflows e um gerador de workflows abstratos. Os resultados dos experimentos mostram que as estratégias desenvolvidas superam o desempenho das soluções de escalonamento dinâmico existentes.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

DYNAMIC STRATEGIES FOR SCHEDULING WORKFLOWS ON GRID

Luiz Antônio Vivacqua Corrêa Meyer

July/2006

Advisor: Marta Lima de Queirós Mattoso
Department: Computing and Systems Engineering

Task scheduling is the main aspect for scientific workflows' performance in Grid. The dynamic nature of the Grid makes the scheduling decision complex. The usual scheduling solutions do not consider the performance changes in the Grid sites that define the execution environment. This thesis presents alternatives for dynamic scheduling tasks on Grid resources. The solution captures the variable nature of the environment, allows the modeling of application particularities and is based on a site selection mechanism that during the workflow execution chooses a site to run each task. Two site selection strategies were developed, the clustering strategy, which is based on the creation of clusters of jobs by their spatial proximity, and the opportunistic strategy. The strategies were evaluated through experiments using simulation and real Grid environments. Their implementations are available in the Euryale^{+} planner that works together with Condor, DagMan and Globus software. A workflow simulation engine and an abstract workflow generator were also developed. The results show that both strategies overcome the current dynamic scheduling solutions.

ÍNDICE

ÍNDICE DE FIGURAS

I - INTRODUÇÃO

I.1 - Motivação

Recursos científicos como programas e dados podem apresentar características de distribuição, heterogeneidade e em muitos casos, um enorme volume de informação. Freqüentemente, os cientistas necessitam combinar estes recursos para executar uma função de mais alto nível. Neste caso, os programas e dados são combinados em uma cadeia de execução de forma que a saída de um processamento serve como entrada para o seguinte. Como observado por Deelman et al. (2003), comunidades científicas como os físicos, astrônomos, biólogos entre outras, não estão mais desenvolvendo suas aplicações como códigos monolíticos. Ao invés disto, diversos componentes estão sendo combinados para o processamento dos dados. Neste cenário, as aplicações científicas podem ser vistas como workflows científicos (MEDEIROS et. al, 1995).

Um workflow pode ser caracterizado como um fluxo de encadeamento de tarefas e suas dependências para execução. Os componentes típicos de um workflow científico são dados e programas. Num workflow científico, esses programas, em geral, processam grandes conjuntos de dados. Portanto, um problema que pode surgir durante a execução de um workflow científico é o tempo necessário para finalizar o seu processamento. A execução de alguns programas pode ser demorada acarretando consequentemente em um tempo elevado para processar o workflow como um todo. Assim, o paralelismo pode ser utilizado para melhorar o desempenho desses programas e de workflows científicos em geral.

Uma possibilidade de explorar o paralelismo é utilizar uma máquina com processamento paralelo. Entretanto, esta solução apresenta algumas desvantagens tais como o alto custo de hardware e software. Outra possibilidade é explorar um ambiente distribuído como um grupo de PC (*PC-cluster*). Este ambiente pode prover desempenho equivalente ao obtido com as máquinas paralelas, porém com custos substancialmente inferiores. Uma terceira alternativa é a utilização de um ambiente de Grid (FOSTER, KESSELMANN, 2004). Os Grids vêm emergindo como plataformas para alto

desempenho e para a integração de recursos em rede. Um Grid pode ser definido como um ambiente de processamento onde recursos heterogêneos e distribuídos, administrados por organizações independentes, podem ser compartilhados e agregados para formar um supercomputador virtual.

Um típico cenário encontrado nas aplicações científicas é ter um conjunto de dados para serem processados por um workflow cujos componentes são programas legados ou de terceiros. Logo, estes programas devem ser tratados como componentes "caixas-pretas" já que não é possível modificar o seu código. Embora o paralelismo de dados seja usualmente empregado, existem diversas alternativas para se executar um workflow científico porque os programas e dados podem ser distribuídos de várias maneiras e diversas estratégias para seleção de recursos para o processamento podem ser utilizadas. Cabe ao sistema de gerência de workflows cuidar da execução correta e eficiente do workflow. Para avaliar as diferentes estratégias de alocação de recursos, foram realizados experimentos com workflows em bioinformatica em PC-*Cluster* (MEYER, MATTOSO, 2004, MEYER et al., 2005a). Os resultados obtidos evidenciaram o impacto do escalonamento dos componentes do workflow. O processamento de um workflow científico em um ambiente de Grid é ainda mais difícil do que em uma máquina paralela ou em um *Cluster* de PC devido à heterogeneidade e a natureza dinâmica dos recursos de um Grid.

O escalonamento de um workflow em um Grid é hoje um tópico de pesquisa em aberto e de grande interesse por parte da comunidade científica, conforme pode ser observado nas publicações dos principais veículos da ciência da computação. Esse escalonamento consiste no mapeamento e gerência da execução de suas tarefas em recursos compartilhados que não estão sob o controle direto do sistema de gerência do workflow (YU, BUYYA, 2005). Assim, achar uma única solução de escalonamento que seja a melhor para os diversos tipos de workflows é um problema difícil porque tanto as aplicações como os ambientes de Grid podem apresentar diferentes características.

Em muitas áreas científicas como, por exemplo, na astronomia e na geografia, os dados modelados possuem coordenadas espaciais e representam características de alguma região do espaço celestial ou da terra. As aplicações que processam os dados

espaciais são geralmente divididas em tarefas que processam partições do espaço representadas por um conjunto de arquivos. Em outras classes de aplicações científicas, como na bioinformática, grandes conjuntos de dados são freqüentemente fragmentados em uma série de arquivos e a execução de um programa que processa este conjunto de dados decomposta em um conjunto de tarefas, cada qual responsável pelo processamento de um fragmento. O processamento destes tipos de workflow em Grid impõe desafios porque em geral eles são:

- Compostos por um número grande de tarefas;
- Necessitam realizar um número elevado de transferências de arquivos;
- Demandam alto espaço de armazenamento e
- Apresentam um compartilhamento de arquivos em alta escala.

Várias iniciativas para melhorar o desempenho de aplicações científicas em Grid podem ser encontradas na literatura. Existem diversos trabalhos endereçando os problemas de transferência e compartilhamento de arquivos através do escalonamento de tarefas baseados na localização dos dados em cenários de Grid de dados (CASANOVA et al. 2000, RANGANATHAN, FOSTER 2001, 2002, 2003b, CAMERON et al. 2003a, 2003b, CHAKRABARTI 2004). Entretanto, as cargas de trabalho analisadas nestes estudos apresentam um compartilhamento de arquivos em pequena escala e consistem de um conjunto de tarefas independentes submetidas por diferentes usuários espalhados por diferentes localidades. Logo as soluções propostas se baseiam no escalonamento de tarefas isoladas.

Outros trabalhos propõem estratégias para o escalonamento de tarefas de um workflow em Grid visando à redução do tempo de execução de um workflow como um todo (BUYYA, YU, 2005). O escalonamento estático (antes de iniciar a execução) de todas as tarefas do workflow utilizando a reserva de recursos ou estimativas de desempenho é proposto por ALHUSAINI et al. (1999), DEELMAN et al. (2004) e MANDAL et al. (2005). Entretanto, a reserva de recursos computacionais é uma caraterística que pode não estar disponível no ambiente de Grid. Neste caso, visto que o ambiente de execução de um Grid pode ser muito dinâmico, associar as tarefas do workflow às localidades do Grid previamente, pode não produzir a melhor alternativa

porque no momento da execução de uma tarefa o recurso escolhido pode estar sobrecarregado, ou mesmo nem estar disponível.

Técnicas de agrupamento de tarefas de um workflow (DEELMAN et al., 2005) e a utilização de múltiplas máquinas no controle da execução do workflow (VARGAS et al., 2005) beneficiam o desempenho devido à redução do número de tarefas a escalonar e da carga de trabalho na máquina de submissão. Entretanto, a escolha dos sítios para execução das tarefas dos grupos é realizada de forma aleatória.

Em outros sistemas como o Triana (TAYLOR et. al., 2003, GOODALE, et al., 2005), BioOpera (BAUSH et al., 2003) e Taverna (OINN et al., 2004) o escalonamento das tarefas é dinâmico porém deve ser realizado pelo próprio usuário através de ferramentas de monitoramento da execução. Técnicas tradicionais para o escalonamento dinâmico como, por exemplo, o mapeamento das tarefas de forma aleatória ou circular, estão disponíveis em sistemas de planejamento de workflows como Pegasus (DEELMAN et al., 2003, 2004) e Euryale (VDS USER GUIDE, 2005). Entretanto, esses sistemas desconsideram o comportamento dinâmico do Grid e acabam por apresentar um desempenho bem aquém do esperado, conforme testes realizados (MEYER et al., 2006a, 2006b).

Em resumo, existe hoje no Grid, um conjunto de ferramentas disponíveis para submissão de tarefas, cópia de arquivos, descoberta e localização de recursos, em especial destaca-se o GLOBUS (FOSTER, 2005). Entretanto, quando se trata do escalonamento de um workflow, as soluções existentes ainda são inadequadas devido, principalmente, à especificidade das aplicações científicas e ao dinamismo de um ambiente como o Grid. Conforme analisado, essas soluções podem ser caracterizadas por serem totalmente estáticas ou com um dinamismo muito limitado. Nos casos em que está previsto algum grau de dinamismo as soluções encontradas deixam o escalonamento a cargo do usuário ou adotam estratégias de característica aleatória. Em diversas situações, o numero de tarefas a escalonar impede qualquer planejamento manual. Além do mais, o tempo de processamento e a complexidade inerente aos componentes dos workflows científicos também impede que o usuário se responsabilize por esse escalonamento. Já o escalonamento aleatório tende a gerar soluções com

desempenho aleatório também, como pode ser observado em MEYER et al. (2006a, 2006b).

I.2 - Objetivo

Considerando-se o impacto do escalonamento de tarefas e dados no desempenho da execução de workflows científicos em ambientes de Grid, o objetivo desta tese é apresentar estratégias eficientes para o escalonamento dinâmico de tarefas. A solução apresentada pretende suplantar os problemas de desempenho encontrados nas soluções existentes através do planejamento dinâmico das tarefas do workflow. A solução, denominada de Euryale+, engloba o comportamento variável do Grid ao mesmo tempo em que oferece recursos para a modelagem de especificidades da aplicação.

Para alcançar os objetivos foi proposto um selecionador de sítios, cujo objetivo é escolher, dinamicamente, o local para executar cada tarefa do workflow no Grid. O selecionador de sítios tira proveito das diversas ferramentas já existentes nos ambientes Globus/Condor/Euryale e acrescenta a possibilidade de o usuário optar pelo escalonamento dirigido à aplicação ou dirigido a explorar dinamicamente o potencial dos recursos do Grid. Para avaliar o selecionador de sítios Euryale+, foram desenvolvidas duas novas estratégias para o escalonamento de workflows científicos em ambientes de Grid. Uma estratégia baseada no agrupamento por proximidade espacial e uma estratégia oportunista.

A estratégia de agrupamento por proximidade espacial explora a idéia de criar grupos de tarefas baseados na região de processamento destas tarefas e é recomendada para o processamento de workflows espaciais com grande compartilhamento de acesso aos arquivos pelas tarefas. Seu foco principal é diminuir o número de transferências de arquivos durante a execução do workflow. Além da definição de grupos, a estratégia utiliza um algoritmo para re-estruturação das tarefas do workflow e um algoritmo para escalonamento das tarefas.

A estratégia oportunista para o escalonamento de tarefas leva em consideração a dinamicidade do ambiente e explora a idéia de escalonar as tarefas do workflow para os locais onde elas provavelmente irão executar mais rápido, baseado na observação do comportamento dos diferentes locais do Grid em relação às tarefas prévias do workflow. Seu foco principal é reduzir o tempo total de processamento do workflow. Em nossa estratégia oportunista, o escalonamento é dinâmico, ou seja, no momento em que a tarefa está apta a ser executada e não leva em consideração estimativa de desempenho ou reserva de recursos.

A avaliação da estratégia de agrupamento por proximidade espacial foi conduzida em um ambiente de simulação e os resultados mostram uma redução significativa no número de transferências de arquivos no Grid e conseqüente melhora no desempenho do workflow, quando comparado às melhores soluções existentes. A estratégia tira proveito da localidade dos dados fazendo uso da replicação dinâmica dos arquivos e do escalonamento das tarefas de forma a reduzir o número de réplicas criadas e o número de transferências realizadas. No caso da estratégia oportunista, a sua avaliação foi conduzida em um ambiente real de Grid. Três workflows diferentes explorando padrões usuais de composição entre os componentes foram utilizados. Os resultados obtidos mostram ganhos de desempenho de até 150% em relação às outras estratégias convencionais avaliadas.

Além das duas abordagens para escalonamento de tarefas, esta tese contribui com duas ferramentas para simulação de workflows: uma para o projeto e a outra para a execução. Elas podem ser integradas para gerar e avaliar o desempenho de workflows expressos como grafos acíclicos diretos (DAG). A *Linguagem para Geração de Workflow* (*WGL*) é uma linguagem de alto nível para a especificação de DAG e permite que o usuário facilmente defina workflows com diferentes formatos e cargas de trabalho. Um interpretador denominado *Gerador de Workflow* (GenWF) processa a *WGL* e produz uma definição do workflow com todos os componentes e dependências. O simulador de Workflow permite que sejam feitas avaliações de desempenho de acordo com vários algoritmos de escalonamento e é capaz de processar definições de workflows criadas pela WGL.

I.3 - Organização da Tese

O restante desta tese está organizado da seguinte forma: o Capítulo II descreve alguns aspectos tecnológicos ligados ao ambiente de Grid e ao gerenciamento de workflows científicos neste ambiente. São discutidas as principais características relacionadas à especificação e ao gerenciamento de workflows científicos e analisados os principais projetos na área. O capítulo II se encerra com uma análise comparativa dos trabalhos relacionados ao proposto nesta tese.

O terceiro capítulo descreve a arquitetura da proposta de implementação do Euryale+. Inicialmente, é fornecida uma visão geral dos sistemas de infra-estrutura básica utilizados na implementação. Uma vez que as abordagens de escalonamento foram implementadas no sistema VDS, uma descrição detalhada da arquitetura do sistema de planejamento de workflows no VDS é fornecida. São descritos os principais componentes da solução Euryale+.

O quarto capítulo descreve a estratégia de agrupamento por proximidade espacial. São detalhados os algoritmos para geração de grupos, alteração de estrutura do Dag e escalonamento de tarefas, bem como o ambiente de simulação desenvolvido. O capítulo termina com a descrição dos experimentos realizados para avaliação da proposta.

O capítulo V descreve a estratégia oportunista para escalonamento de tarefas. São detalhados os algoritmos para escalonamento e monitoramento das tarefas submetidas e fornecida uma visão do ambiente real de Grid utilizado bem como os experimentos realizados para a sua avaliação.

No capítulo VI, são analisados os resultados experimentais obtidos em simulações e em execuções reais dos experimentos discutidos nos capítulos IV e V. O capítulo VII conclui esta tese e aponta para direções futuras.

II - GERENCIAMENTO DE WORKFLOWS CIENTÍFICOS EM GRID

II.1 – Introdução

O modelo de computação em Grid tem sido descrito fazendo-se uma analogia as redes de transmissão de eletricidade. Quando ligamos algum aparelho elétrico na tomada, nós não fazemos idéia de onde a eletricidade é gerada. A companhia elétrica responsável pelo fornecimento deste serviço fornece uma interface para um sistema complexo de geração e transmissão. A visão de Grid é similar (ou almeja ser), ou seja, diversos recursos computacionais geograficamente distribuídos podem ser agregados para formar um supercomputador virtual. Porém, estes recursos não necessitam ser visíveis ao usuário, da mesma forma que um consumidor não tem visibilidade de onde a eletricidade disponível em sua residência é gerada. A infra-estrutura e os recursos tecnológicos que formam um Grid devem apoiar o compartilhamento e o uso coordenado destes recursos.

O objetivo deste capítulo é descrever as principais características do processamento de workflows científicos em ambientes de Grid. Inicialmente são comentadas algumas características particulares dos workfows científicos em relação à definição e ao escalonamento das tarefas. Na terceira seção são relacionados alguns dos principais sistemas voltados para o gerenciamento de workflows científicos em Grid. Na quarta seção nós concluímos o capítulo com uma análise dos recursos providos por estes sistemas.

II.2 - Workflow Científico

De acordo com o WFMC (*Workflow Management Coalition*) (WORKFLOW MANAGEMENT COALITION, 1995), Workflow é a automação total ou parcial de um

processo de negócio, durante o qual documentos, informações ou tarefas são passadas de um participante para outro, de acordo com um conjunto de regras de procedimento. Um Sistema de Gerenciamento de Workflow é definido como um sistema que de forma completa define, gerencia e executa workflows. A execução dos programas é baseada em uma representação da lógica do workflow no computador.

Embora as definições de Workflow façam referência a "processos de negócio", na realidade, os sistemas para gerenciamento de workflows vêm sendo utilizados não só em aplicações comerciais convencionais, mas também em aplicações científicas. Diversas comunidades científicas como físicos, astrônomos, biólogos, geólogos entre outras vêm desenvolvendo suas aplicações através da combinação de diferentes programas e dados formando os chamados workflows científicos.

Existem diferenças entre workflows comerciais e científicos. Segundo MEDEIROS et al. (1995), em um ambiente científico, o cientista em geral irá especificar seu Workflow diretamente, enquanto que em um ambiente comercial, esta especificação será normalmente realizada pelo administrador do sistema. Outra característica apontada em seu trabalho é a necessidade de se ter um registro das execuções do Workflow científico para permitir a re-execução e reprodução de resultados obtidos previamente.

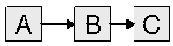
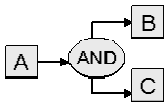
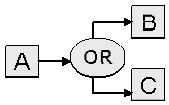
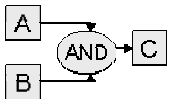
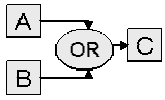
Outras características que distinguem os workflows comerciais dos científicos são apontadas por ALTINTAS et al. (2004). Os workflows comerciais necessitam de complexos fluxos de controle e requerem a coordenação de um número pequeno de troca de mensagens e documentos. Os workflows científicos por sua vez são mais orientados ao fluxo de dados. Normalmente os dados são heterogêneos, distribuídos, com grande volume e são obtidos através da análise de outro conjunto de dados. Além disto, os workflows científicos precisam lidar com programas legados ou escritos por terceiros, que em geral também são heterogêneos e sem disponibilidade do código fonte.

Os componentes de um Workflow podem ser modelados de acordo com vários padrões de controle de fluxo. Estes padrões endereçam funcionalidades distintas. De

acordo com Aalst et al., (2003), a maioria das linguagens para especificação de workflows implementa os padrões básicos listados na tabela 1.

Em geral, workflows científicos podem ser representados como uma estrutura de um grafo direto acíclico (DAG) ou como um grafo com ciclos. No primeiro caso, todos os cinco componentes básicos mostrados na tabela 1 podem ser empregados. No segundo caso, também podem ser utilizadas interações para forçar a repetição de seções do Workflow.

Tabela 1 – Padrões básicos para controle de workflows (Aaalst et al., 2003)

Padrão	***Nome***	***Descrição***
	Seqüencial	A execução de uma tarefa só pode acontecer após a execução da tarefa que a precede.
	Divisão paralela	As tarefas B e C têm que ser executadas após o término da tarefa A.
	Escolha exclusiva	A tarefa B ou a tarefa C tem que ser executada após a tarefa A.
	Sincronização	A tarefa C só pode executar quando as tarefas A e B terminarem
	Fusão simples	A tarefa C pode executar se a tarefa A ou a tarefa B terminarem.

II.2.1 - Especificação

Muitos sistemas de gerenciamento de workflow científico disponibilizam uma interface gráfica para permitir que o usuário especifique a sua aplicação através da seleção de seus componentes. Em outros casos, uma linguagem de workflow é usada para a definição da aplicação. Através destas ferramentas, o projeto de um workflow pode ser especificado de uma forma abstrata ou concreta. Na forma abstrata, os componentes do workflow são especificados sem fazer referência a recursos específicos do Grid. Na forma concreta, as tarefas estão atreladas a recursos específicos.

Pelo uso de diferentes níveis semânticos, workflows abstratos podem especificar seus componentes sem levar em consideração detalhes sintáticos de invocação de programas ou do formato de dados. Por exemplo, em um workflow abstrato de bioinfomática, um usuário poderia especificar um componente que execute uma tarefa de busca de seqüências por homologia em uma base de genomas e quando o sistema gerasse o workflow concreto, um programa executável que realiza esta tarefa seria associado. Em um outro nível semântico, mas também abstrato, o usuário poderia especificar os componentes do workflow através de seus nomes lógicos, tais como Blast (ALTSCHUL ET AL., 1990) e PDB (PROTEIN DATA BANK). Entretanto, em ambas as situações, existe a necessidade de haver catálogos ou repositórios que permitam o mapeamento dos componentes abstratos em concretos. Por exemplo, NCBI-Blast e NCBI PDB version 5.2.3.

No contexto desta tese, nós assumimos as seguintes definições:

- *Workflow abstrato* é um conjunto de programas e arquivos de dados referenciados pelo seu nome lógico e suas dependências entre si.

- *Workflow concreto* é um conjunto de programas e arquivos de dados referenciados pelos seus nomes físicos, e a especificação do seu local de execução no Grid. Um nome lógico não caracteriza um workflow concreto uma vez que um programa ou arquivo de dados pode estar associado a múltiplos nomes físicos. Este é o nível mais baixo de uma definição de workflow.

II.2.2 - Planejamento

O mapeamento de um workflow abstrato em um workflow concreto é chamado de planejamento (também chamado de escalonamento de tarefas). Este mapeamento consiste em selecionar recursos específicos, arquivos de dados e tarefas adicionais (por exemplo, uma tarefa para transferir um arquivo de um local para outro) para formar um workflow concreto que possa ser executado em um ambiente de Grid (DEELMAN et

al., 2003). Esse planejamento envolve três dimensões para a tomada de decisão: 1) gerência; 2) abrangência; 3) instante.

Existem três categorias principais para fazer a **gerência** da tomada de decisão (YU e BUYYA, 2005):

- *Centralizado* - Um único escalonador central toma as decisões de mapeamento para todas as tarefas do workflow.
- *Hierárquico* - Um gerente central controla a execução do workflow e mapeia partes do workflow para outros escalonadores.
- *Decentralizado* - Não existe um controlador central. Os vários escalonadores distribuídos podem se comunicar e mapear um para outro, partes do workflow.

A **abrangência** do mapeamento diz respeito ao número de tarefas que estão sendo planejadas. Em geral, as decisões sobre o mapeamento podem ser do tipo:

- *Local* - Neste caso o escalonamento é feito baseado em informações da tarefa corrente.
- *Global* - O escalonamento é feito tendo-se por base todas as tarefas do workflow.

Uma terceira característica importante do planejamento refere-se ao **instante** em que as decisões são tomadas. Neste caso, as decisões podem ser de dois tipos:

- *Estática* - O workflow concreto é gerado antes da execução das tarefas.
- *Dinâmica* - O workflow concreto é gerado durante a execução das tarefas.

Por fim, uma função alvo pode guiar o mapeamento das tarefas em um workflow. O objetivo final do mapeamento pode ser, por exemplo, a obtenção de uma solução que forneça um menor tempo de execução do workflow, um menor custo financeiro de execução (supondo que provedores de recursos vendam serviços), um menor volume e tempo de comunicação ou a seleção de recursos que forneçam maior confiabilidade.

II.3 - Principais Sistemas de Workflows

O objetivo desta seção é fazer uma breve descrição de alguns dos sistemas mais importantes no gerenciamento e execução de workflows. Foram selecionados doze sistemas porque além de serem bastante referenciados na literatura, são voltados para aplicações científicas e utilizados em ambientes de Grid. A análise prioriza as características de especificação e planejamento, por considerarmos as mais relevantes para o desempenho.

II.3.1 - Triana

Triana (TAYLOR et. al., 2003, GOODALE, TAYLOR, WANG, 2005) é um ambiente gráfico para computação em Grid e fornece um portal que permite que o usuário especifique a composição de seu workflow através da seleção de componentes disponíveis. O sistema foi inicialmente desenvolvido com a finalidade de ajudar os cientistas nas tarefas de análise de dados. Triana foi posteriormente estendido para o uso em ambiente de Grid e permite a composição de serviços remotos como serviços web.

Um grafo de tarefas em XML é gerado a partir da ferramenta de composição e o sistema possui diversos construtores que permitem a especificação de ciclos e condições lógicas. Serviços remotos também podem ser selecionados para compor o workflow através de consultas a um serviço de diretório. O workflow como um todo pode ser distribuído para ser executado em uma única máquina remota ou o usuário pode distribuir partes do workflow para mais de uma máquina. Para distribuir a execução do workflow, primeiro o usuário tem que criar grupos de tarefas que representam subseções do workflow.

Triana também permite a execução de programas legados através do sistema GAT (*Grid Application Toolkit*) (ALLEN et. al., 2005) desenvolvido no projeto GridLab (ALLEN et. al., 2003). O escalonamento de tarefas no GAT é feito pelo

GRMS (*GridLab Resource Management System*) (GRMS, 2003). A seleção de recursos para a execução das tarefas pode ser feita diretamente pelo usuário antes da execução do workflow, ou pelo próprio GRMS de acordo com uma lista de requisitos especificados para a tarefa. Neste caso, GRMS usará um algoritmo baseado nos critérios especificados para escolher a melhor opção para executar o serviço. Entretanto, detalhes do algoritmo não são fornecidos na documentação do projeto.

O escalonamento de tarefas no Triana não é totalmente transparente para os usuários, uma vez que estes têm que especificar diretamente o recurso para executar a tarefa. Quando Triana utiliza o GAT, não é mostrado como o escalonador toma a decisão se mais de um recurso satisfaz os requisitos para a tarefa.

II.3.2 - Askalon

Askalon (WIECZOREK, PRODAN, FAHRINGER, 2005) é um projeto desenvolvido pela Universidade de Insbruck na Austria que fornece um ambiente para o desenvolvimento e a execução de aplicações no Grid. Workflows são especificados através de uma interface gráfica ou usando a linguagem AGWL (*Abstract Grid Workflow Language*) e podem expressar grafos diretos acíclicos ou estruturas mais sofisticadas contendo repetições e desvios condicionais.

O workflow abstrato é automaticamente mapeado para o workflow concreto e executado no Grid. O escalonamento das tarefas é estático, global e realizado por algoritmos genéticos. A performance de cada componente do workflow é estimada, baseada nos resultados coletados em uma fase de treinamento, que mede o tempo de execução de cada tarefa com diferentes cargas em cada máquina do Grid. A estimativa de desempenho do workflow é obtida combinando-se os dados adquiridos na fase de treinamento com modelos analíticos. Nos experimentos reportados, os autores assumem alta disponibilidade e controle sobre os recursos do Grid. Em particular, é assumido que o escalonador possui informações precisas sobre os recursos disponíveis no Grid e que não ocorrem falhas durante a execução do workflow.

II.3.3 - Grads

O projeto Grads (*Grid Application Development Software*) (MANDAL et al., 2005, COOPER et al., 2004) é fruto de uma colaboração de diversas universidades americanas e tem por objetivo simplificar a utilização de recursos distribuídos heterogêneos. Grads fornece ferramentas de programação e um ambiente de execução para facilitar o desenvolvimento de aplicações para o Grid. No Grads, as aplicações são encapsuladas como objetos de programa configuráveis (COP). Um COP inclui o código da aplicação, um mapeador que determina como mapear as tarefas de uma aplicação para um conjunto de recursos e um modelo de performance que faz estimativas do desempenho das aplicações no conjunto de recursos.

GrADS fornece ferramentas para prototipagem que de forma semi-automática constróem o modelo de performance e o mapeador para o COP. O escalonamento é estático e centralizado e tem por objetivo reduzir o tempo total de execução do workflow. O escalonador faz uso de serviços como MDS (*Monitoring and Directory Service)* (CZAJKOWSKIE et al., 2001) NWS (*Network Weather Service*) (WOLSKI et al., 1999) para obter informações sobre os recursos disponíveis no Grid. O escalonador avalia cada recurso para cada componente do workflow. Esta avaliação é calculada usando uma soma ponderada do tempo de execução esperado do componente em um determinado recurso e o tempo esperado para a transferência de dados. Grads possui um componente ("Binder") que executa em todos os nós do Grid e que é responsável por criar o executável da aplicação, fazendo a compilação do código e submetendo para a execução.

Grads utiliza monitores que periodicamente coletam dados sobre a execução. Se durante a execução do workflow o desempenho de um componente não reflete o esperado, este componente pode ser re-escalonado para outro recurso.

II.3.4 - BioOpera

O projeto BioOpera (BAUSCH, PAUTASSO, ALONSO, 2003) foi desenvolvido no Instituto Federal de Tecnologia da Suíça em Zurique e fornece recursos para o desenvolvimento e execução de aplicações biológicas em ambientes distribuídos.

Os workflows biológicos são especificados através de uma ferramenta gráfica que permite que o cientista defina os arquivos de entrada e saída bem como as dependências entre as tarefas. As tarefas são selecionadas de uma biblioteca de componentes. A representação gráfica é transformada em uma representação textual pelo uso de uma linguagem de programação interna. Toda vez que um novo programa é registrado, o usuário especifica os parâmetros de entrada e saída, a forma de invocação do programa e em que locais ele pode ser executado. Estas informações são armazenadas em um banco de dados e podem ser dinamicamente alteradas. O banco de dados também armazena informações sobre registros de execução, carga e disponibilidade dos recursos.

Durante a execução do workflow, tarefas independentes podem ser escalonadas em paralelo caso existam recursos computacionais disponíveis. O usuário pode interagir com o BioOpera para determinar onde executar uma tarefa bem como para conferir resultados intermediários. Também é permitido, suspender ou reiniciar a execução de tarefas. O escalonamento é dinâmico e local. BioOpera não utiliza o conceito de workflow abstrato.

II.3.4 - GridFlow

GridFlow (CAO et al., 2003) é definido como um sistema para gerenciamento de workflows com o objetivo de permitir ao usuário a construção, simulação, execução e monitoramento de workflows em ambientes de Grid. Um workflow é representado como um fluxo de diferentes atividades onde cada atividade é representada por um sub-

workflow. Um sub-workflow por sua vez define um fluxo de tarefas relacionadas que precisam ser executadas em uma determinada seqüência dentro de um Grid local. Uma tarefa é o menor elemento em um workflow e é geralmente uma aplicação paralela em PVM ou MPI.

Um portal disponibiliza uma interface gráfica para que o usuário defina a sua aplicação. Para construir um workflow, o usuário precisa definir as propriedades de cada sub-workflow, tarefas e a respectiva ordem de execução. O workflow abstrato é descrito por uma especificação em XML. O usuário pode definir o mapeamento das tarefas diretamente ou deixar a cargo do próprio sistema. GridFlow fornece três funcionalidades: simulação, execução e monitoração. A simulação ocorre antes da execução do workflow e produz o escalonamento das tarefas. A execução é realizada de acordo com o resultado da simulação. As ferramentas de monitoração permitem o acesso a informações em tempo real sobre o estado de execução das tarefas.

O escalonamento do workflow é realizado em dois níveis. Primeiro, a ferramenta PACE é usada para estimar a performance dos recursos no Grid. Uma vez que um recurso seja escolhido para executar a tarefa, o gerenciador de recursos TITAN localizado em cada nó é usado para o escalonamento no Grid local.

A documentação do GridFlow não deixa claro como os usuários interagem com o portal para especificar o workflow. Aparentemente, o sistema não fornece um repositório que permita que o usuário selecione os componentes do workflow. O mapeamento do workflow abstrato para o workflow concreto também não está bem claro, e o seu uso em aplicações reais não é reportado.

II. 3.5 - Unicore

Unicore (STREIT et al., 2006) é um projeto desenvolvido com apoio do governo alemão que disponibiliza um software básico para Grid cujo objetivo é fornecer uma interface uniforme para recursos de computação distribuídos. O sistema possui uma interface gráfica que dá assistência ao usuário na criação de tarefas simples ou

workflows. Os worflows podem ter estruturas de grafos acíclicos ou estruturas mais complexas com desvios ou grupos de repetição.

A especificação do workflow é feita na forma concreta, e o escalonamento das tarefas é estático e centralizado. O sistema provê ferramentas para monitorar a execução do workflow. A interface gráfica também disponibiliza funções para o gerenciamento de dados. Estas funções permitem que o usuário especifique transferência de dados entre diferentes nós do Grid antes ou depois da execução de uma tarefa. Durante a execução, o sistema realiza as transferências de dados sem a intervenção do usuário.

II.3.6 - Taverna

Taverna (OINN et al., 2004) é um sistema de gerenciamento de workflow desenvolvido para o projeto myGrid (GREENWOOD et al., 2002, GOBLE et al., 2003). O objetivo do sistema é fornecer um conjunto de ferramentas para composição e execução de workflows de bioinformática.

O sistema disponibiliza uma interface gráfica para a especificação e execução de workflows baseados na composição de serviços Web. Estes workflows podem ter a estrutura de grafos acíclicos, são especificados de forma abstrata e também podem ser descritos através da linguagem Scufl (*Simple Conceptual Unified Flow Language*).

O workflow é executado pela ferramenta Freefluo que é integrada ao Taverna. Freefluo é uma ferramenta Java para orquestração de workflows e é responsável pela invocação dos serviços e pela transferência de dados intermediários. Freefluo pode invocar serviços web, serviços WSDL e serviços Soaplab.

Taverna também fornece um ambiente amigável para o usuário manipular o workflow, selecionar os recursos, executar e monitorar os resultados. Outra característica importante do projeto é que durante a execução de um workflow, a informação da proveniência dos dados é gravada mostrando em que condições cada tarefa foi executada.

II.3.7 - Karajan

Karajan (LASZEWSKI, HATEGAN, 2005a) compreende uma linguagem e um sistema para a especificação e execução de workflows. Karajan é parte do Java CoG Kit (LASZEWSKI, HATEGAN, 2005b). O sistema de execução é baseado no Globus Toolkit enquanto que a linguagem de definição é baseada em XML. Karajan suporta a definição de workflows com estrutura de grafos acíclicos bem como o uso de repetições e desvios condicionais.

O sistema disponibiliza uma interface baseada em linha de comando e uma interface gráfica que permite ao usuário monitorar o andamento da execução do workflow. Pontos de parada podem ser inseridos para suspender a execução de tarefas específicas.

Karajan define três tipos de tarefas em Grid: *grid:execute* que executa um programa localmente ou remotamente; *grid:transfer* que transfere arquivos entre locais do Grid e *grid:authentication* que autentica o usuário no Grid. O sistema permite que o usuário defina uma ação para o tratamento de erros durante a execução de uma tarefa.

O escalonamento de tarefas é realizado diretamente pelo usuário na medida em que o workflow é definido na sua forma concreta.

II.3.8 - Kepler

Kepler (ALTINTAS et al., 2005) é um sistema de gerenciamento de workflows desenvolvido pelas Universidades da Califórnia de San Diego e Santa Barbara. O sistema fornece uma interface gráfica que permite que o cientista defina e execute seu workflow. O sistema utiliza o conceito de *ator* para descrever abstrações de componentes reutilizáveis para um workflow em um ambiente de Grid. As tarefas individuais do workflow são implementadas como *atores* abstratos ou concretos. Existe uma biblioteca de atores abstratos para autenticação de usuários, cópia de arquivos, execução de tarefas, monitoramento, armazenamento de resultados, acesso à banco de

dados, entre outros. Os atores abstratos são então instanciados por atores concretos em tempo de execução. Por exemplo, um ator abstrato para cópia de arquivos pode ser instanciado para diversos tipos de ator concreto tais como FTP, GridFTP ou scp.

Kepler permite a definição de workflows com estrutura de grafos acíclicos e os workflows gerados são traduzidos para a linguagem MoML (*Modeling Markup Language*) e podem ser salvos e re-executados. A maioria dos atores são processos Java que rodam localmente, porém serviços web também podem ser usados como atores e incorporados a biblioteca para executarem tarefas remotas no Grid.

Como nos projetos Taverna, Triana e Karajan, parece que fica a cargo do usuário a decisão de onde executar uma tarefa no Grid, ou seja, o usuário deve informar qual ator concreto irá mapear determinada tarefa representada pelo ator abstrato.

II.3.9 - GRAND

GRAND (Vargas et al., 2004, 2005) é uma arquitetura cujo principal objetivo é lidar com problemas de gerenciamento de carga na máquina de submissão. A arquitetura provê a submissão distribuída de tarefas baseado na partição do DAG que define o workflow.

A definição do workflow é feita através da linguagem GRID-ADL. A GRID-ADL é compilada gerando um grafo de tarefas. O grafo é então fragmentado gerando partições que são submetidas por máquinas diferente no Grid. O escalonamento das tarefas é feito de forma hierárquica, dinamicamente e aleatória. Os resultados apresentados mostram uma melhoria no desempenho quando foram utilizados mais de uma máquina de submissão em workflows constituídos por tarefas independentes em testes realizados em uma rede local.

II.3.10 - Condor / DagMan

Condor (THAIN, D., TANNENBAUM, T., LIVNY, M., 2005) é um sistema em lote especializado no gerenciamento de tarefas que executem em estações de trabalho que se comuniquem através de uma rede. Os usuários submetem suas tarefas para o Condor, que as coloca em uma fila, executa-as e informa o resultado ao usuário quando as tarefas terminam. Condor descobre as máquinas para executar os programas, mas não faz o escalonamento das tarefas baseado em dependências.

Condor-G (FREY et al., 2001) é uma extensão do Condor, e permite que as tarefas sejam descritas no arquivo de submissão do Condor. Porém, quando submetidas para execução, o Condor utiliza a interface Globus GRAM para submeter à tarefa para execução em um recurso remoto.

DagMan (*Directed Acyclic Graph Manager*), (CONDOR TEAM, 2003), é um meta-escalonador para tarefas Condor. DagMan submete tarefas para o Condor na ordem representada pelo grafo e processa os resultados. O grafo de tarefas é definido em um arquivo de entrada que especifica a lista dos programas no grafo, pré e pós-processamento para cada programa, a descrição das dependências entre os programas e o número de re-execuções para o caso de falhas.

DagMan não utiliza o conceito de workflow abstrato e os usuários são responsáveis por especificar o local de execução de cada tarefa antes de sua submissão.

II.3.11 - Pegasus

Pegasus (DEELMAN, E, BLYTHE, J., GIL, Y., KESSELMAN, C., et al., 2004, DEELMAN, E., SINGH, G., SU, M., BLYTHE, J., et al., 2005) é um sistema para planejamento de workflows em Grid e é desenvolvido pelo ISI (Information Sciences Institute) na Universidade Southern Califórnia como parte do projeto GriPhyn-VDS, inicialmente chamado Chimera (FOSTER et al., 2002, FOSTER et al., 2003), Pegasus transforma um workflow abstrato em um workflow concreto através de uma série de

refinamentos. O workflow abstrato é composto de tarefas e arquivos de entrada e saída descritos em termos de nomes lógicos. O objetivo do Pegasus é mapear estas tarefas para os recursos disponíveis no Grid para a sua execução.

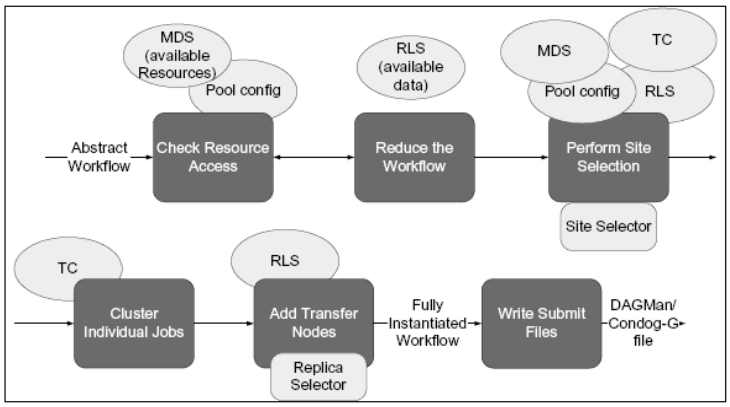


Figura 1 - Etapas do planejamento do Pegasus (DEELMAN et al., 2005)

A figura 1 ilustra os passos executados pelo Pegasus durante o processo de refinamento do workflow. Primeiramente, o Pegasus consulta o serviço *MDS* do Globus e o arquivo com a configuração dos recursos do Grid (*Pool config*) para verificar que locais estão disponíveis para a execução das tarefas. O *MDS* fornece informação sobre características no Grid, tais como número de processadores existentes e quantidade de memória disponível nos diferentes locais. O arquivo de configuração por sua vez, fornece informações sobre o ambiente de execução, tais como a identificação do servidor de *GridFTP* e do serviço de localização de réplicas (*RLS*).

O próximo passo é possivelmente modificar o workflow abstrato baseado na disponibilidade dos arquivos de dados referenciados no workflow. Pegasus consulta o serviço de localização de réplicas para determinar quais arquivos intermediários já estão disponíveis no Grid. Baseado nestas informações, Pegasus pode reduzir o workflow de forma a conter somente as tarefas necessárias para gerar os arquivos finais. Em um caso extremo, se todos os arquivos finais a serem produzidos pelo workflow já existirem, nenhuma tarefa será executada, com exceção talvez da transferência dos dados para um local especificado pelo usuário e respectivo registro no *RLS*. Esta redução do workflow é feita baseada na suposição de que é mais eficiente acessar um dado do que ter que re-gerá-lo.

Uma vez que o workflow mínimo é gerado, a escolha dos recursos para execução das tarefas é executada (*seleção de sítio*). Esta seleção dos recursos é feita de forma centralizada, local, estática e baseada nas informações de disponibilidade e da localização dos dados de entrada. O sistema dispõem de dois algoritmos para seleção de recursos: seleção aleatória e circular. Estes algoritmos consultam as informações disponíveis no MDS, no arquivo de configurações, no catálogo de transformações e no serviço de réplicas para definirem o local de execução de cada programa. Adicionalmente, os próprios usuários podem definir seus mecanismos de seleção e informar o sistema para fazerem a chamada destes algoritmos.

Pegasus também oferece a opção de agrupamento de tarefas (SINGH, G., KESSELMAN, C., DEELMAN, E., 2005). Esta opção pode trazer benefícios no desempenho quando o workflow é composto por um número grande de tarefas rápidas, sem dependências entre si e que podem ser escalonadas para serem executadas em um mesmo recurso.

II.3.12 - Euryale

Euryale (VDS USER GUIDE, 2005) é um sistema para planejamento de workflows em Grid, desenvolvido na Universidade de Chicago, e assim como o Pegasus, também faz parte do projeto GriPhyn-VDS. Euryale realiza o planejamento do workflow de forma centralizada, local e dinâmica e implementa um recurso de tolerância à falhas através do replanejamento de tarefas. Um workflow concreto é gerado a partir de um workflow abstrato fornecido pelo usuário. Euryale gera a especificação de execução de cada tarefa em um arquivo de submissão do Condor que é submetido ao Grid pelo sistema DagMan. Durante a execução do workflow, DagMan é o responsável por decidir qual tarefa deve ser executada e iniciar para cada tarefa apta a ser executada o seu pré e pós-processamento.

A escolha do sítio para a execução de uma tarefa é feita dinamicamente no momento em que a tarefa é eleita pelo DagMan para ser executada. Esta seleção é feita

por um componente externo denominado selecionador de sítio. A comunicação entre o sistema Euryale e o selecionador de sítio externo é feita através de um arquivo temporário que fornece entre outras informações a identificação da tarefa, os arquivos a serem processados, e a relação de sítios disponíveis. Esta relação de sítios disponíveis é construída a partir dos catálogos de sítios, réplicas e transformações mantidas pelo sistema.

Euryale disponibiliza duas estratégias para a seleção de sítios: constante e aleatória. A primeira serve basicamente para testar a execução remota de um componente do workflow. Entretanto, a sua arquitetura permite que os próprios usuários desenvolvam seus algoritmos para escalonamento para serem utilizados no planejamento das tarefas. Antes de submeter o workflow para execução, o usuário informa ao sistema qual algoritmo deve ser utilizado para seleção de sítios.

II.3.13 – Outros Trabalhos Correlatos

Além dos sistemas já citados, encontramos na literatura diversos trabalhos relacionados à execução de aplicações científicas em ambientes distribuídos. No campo da bioinformática, por exemplo, existem trabalhos explorando o processamento paralelo em operações para a comparação e alinhamento de seqüências. Pappas (1994) utilizou uma rede de estações de trabalho e uma implementação utilizando o PVM para obter ganhos no desempenho na execução do programa Blast. Costa e Lifschitz (2003) apresentam diferentes abordagens para a distribuição de consultas para o processamento do Blast frente a diferentes bancos de dados. Os autores implementam uma interface desenvolvida em MPI e comparam os ganhos no desempenho de acordo com várias políticas de replicação e fragmentação da base de dados em um ambiente de cluster de PCs. Yarkhan e Dongarra (2003) implementam a paralelização do programa FastA em um ambiente de Grid usando uma interface MPI e estratégias de replicação de dados. Entretanto, estes trabalhos exploram o paralelismo para melhorar o desempenho de uma única aplicação fortemente acoplada. O foco desta tese é apresentar estratégias para a

melhoria do desempenho na execução de um conjunto de tarefas associadas, que constitui um workflow.

Em um trabalho inicial (MEYER, MATTOSO, 2004), fizemos uma avaliação de diversos sistemas de gerência de workflow científico não limitados ao ambiente de Grid, tais como WASA (MEDEIROS et al., 1995), METEOR (KOCHUT et al., 2003) e gRNA (BHOWMICK et al., 2003). Alem disto, apresentamos uma caracterização do processamento paralelo de workflows e apresentamos heuristicas para execução paralela do workflow. As soluções analisadas foram implementadas em um ambiente de *PC-cluster* utilizando o escalonamento estático e diferentes abordagens para alocação e replicação dos componentes do workflow.

Em cenários de Grid de Dados, existem diversos trabalhos que exploram estratégias de processamento baseadas na localidade dos dados. CASANOVA et al. (2000) propõem um algoritmo para escalonamento de tarefas onde arquivos multi-referenciados são pré-alocados estrategicamente no Grid de forma a aumentar o reuso. Ranganathan e Foster (2003a, 2003b) avaliam um conjunto de algoritmos para o escalonamento de tarefas e replicação de arquivos em Grid, e o impacto da largura de banda da rede, padrões de acesso dos usuários e distribuição de dados no desempenho da execução de tarefas. A avaliação é realizada em um ambiente de simulação e os resultados mostram benefícios ao se escalonar tarefas para locais onde os dados de entrada existem e assincronamente replicar arquivos populares pelos nós do Grid. O impacto de transferir arquivos de entrada e saída no desempenho geral de um Grid é estudado por Shan et al. (2004). Os autores assumem que os arquivos de entrada residem no mesmo local onde a tarefa é submetida e que não são reutilizados por diferentes tarefas.

Nestes trabalhos, o tipo de processamento consiste de tarefas isoladas submetidas por diferentes usuários distribuídos em diferentes locais do Grid. Estes trabalhos assumem que o maior problema é a concorrência por recursos compartilhados entre aplicações diferentes e não apresentam soluções para otimizar o desempenho de um único workflow. Se o workflow é complexo e apresenta um alto grau de

compartilhamento de recursos entre as tarefas, essa concorrência por recursos ocorre dentro do próprio workflow. Neste caso, as soluções de concorrência para usuários distintos não se aplica.

No contexto do projeto VDS alguns trabalhos, também endereçam o problema de desempenho na execução de workflows e merecem destaque. O sistema GNARE (SULAKHE et al., 2005) implementa um ambiente computacional de alta performance confeccionado especialmente para análise de genomas. O escalonador define não só o local de execução de cada tarefa, mas também o número de seqüências que cada tarefa deve processar. A decisão é baseada em informações coletadas dos sítios remotos e da fila mantida pelo Condor na máquina de submissão. Entretanto, as soluções de escalonamento estão atreladas a aplicação específica.

Dumitrescu e Foster (2005) avaliaram no Grid3 (FOSTER et al., 2004) o desempenho do programa Blast de acordo com vários algoritmos de escalonamento. Nestes experimentos, os autores utilizam um ambiente que considera as políticas de alocação dos recursos de acordo com a organização virtual do usuário. O escalonamento é realizado baseado em informações coletadas dos diversos sítios do Grid que informam o percentual de recursos utilizados por cada organização virtual. Baseado nestas informações é construída uma lista com os melhores locais para escalonamento de uma tarefa. Esta lista é então passada para o escalonador que escolhe o local de execução de acordo com o algoritmo implementado. Embora esta solução seja benéfica por reduzir as chances do escalonador escolher um sítio onde a tarefa teria poucas chances de executar rapidamente devido à política de alocação de recursos do sítio, os sítios vinculados a organização virtual do usuário podem apresentar desempenhos distintos, o que não é captado dinamicamente pelo escalonador.

II.4 – Avaliação dos trabalhos relacionados

A tabela 2 resume as principais características dos sistemas apresentados na seção II.3 levando-se em consideração alguns aspectos da estrutura, forma de especificação, planejamento e escalonamento das tarefas do workflow.

Tabela 2 - Características dos Sistemas de Gerenciamento de workflow em Grid

				Planejamento			
Sistema	Estrutura	Modelo	Tarefa	Gerência	Abran-gência	Instante	Seleção de Sítio
Triana	Grafo complexo	Abstrato	Serviço Web	Centralizada	Local	Dinâmico	Usuário Aleatório
Askalon	Grafo complexo	Abstrato	Job	Centralizada	Global	Estático	Predição
Grads	DAG	Abstrato	Job	Centralizada	Global	Estático	Predição
BioOpera	DAG	Concreto	Job	Centralizada	Local	Dinâmico	Usuário
GridFlow	DAG	Abstrato	Job Paralelo	Hierárquico	Global	Estático	Predição
Unicore	Grafo complexo	Concreto	Job	Centralizada	Global	Estático	Usuário
Taverna	DAG	Abstrato Concreto	Serviço Web	Centralizada	Local	Dinâmico	Usuário
Karajan	Grafo complexo	Concreto	Job	Centralizada	Global	Estático	Usuário
Kepler	DAG	Abstrato Concreto	Serviço Web	Centralizada	Global	Estático	Usuário
GRAND	DAG	Abstrato	Job	Hierárquica	Local	Dinâmico	Aleatório
DagMan	DAG	Concreto	Job	Centralizada	Global	Estático	Usuário
Pegasus	DAG	Abstrato	Job	Centralizada	Global	Estático	Aleatório
Euryale	DAG	Abstrato	Job	Centralizada	Local	Dinâmico	Aleatório

A maioria dos sistemas permite que o usuário defina o workflow em termos abstratos. A especificação abstrata é vantajosa se o sistema possuir um mecanismo automático para o planejamento da execução das tarefas. Desta forma, o usuário fica livre da incumbência de ter que decidir por si próprio qual sítio é o mais apropriado a ser selecionado para execução de uma tarefa ou para transferência de um arquivo. Porém, se o workflow é composto por centenas ou milhares de tarefas e transferências

de arquivos, é desejável que a sua especificação seja feita em um nível abstrato mais elevado de forma a simplificar para o usuário esta especificação. Esta idéia é explorada pelo GRAND, porém em um nível de especificação semelhante ao de uma linguagem de programação, o que dependendo do perfil de usuário, pode não ser trivial.

A especificação de um workflow em termos concretos faz com que o usuário seja o responsável pelo desempenho do sistema, uma vez que ele passa a ter que selecionar os recursos para execução dos componentes. Os sistemas que fazem o gerenciamento de workflows cujos componentes são serviços web são em geral planejados pelo próprio usuário. Entretanto, esses sistemas têm sido utilizados por aplicações compostas por poucas tarefas e onde o foco maior é a integração e a interoperabilidade dos componentes.

Não existe uma única estratégia que forneça a melhor solução para o mapeamento de um workflow abstrato em um workflow concreto. As aplicações científicas podem apresentar diferentes características e necessidades de processamento, volume de dados, visualização de resultados, etc. Se por um lado, o planejamento global e estático pode levar em consideração a estrutura geral do grafo que representa a estrutura do workflow e aplicar algoritmos sofisticados para o escalonamento das tarefas, este tipo de planejamento não leva em consideração as mudanças repentinas que o ambiente de Grid está sujeito. Uma vez que o Grid pode ser um ambiente extremamente dinâmico, este tipo de estratégia pode não trazer benefícios para o desempenho. Assim, o planejamento estático parece ser mais adequado a ambientes de Grid onde exista a possibilidade da reserva prévia de recursos.

Entretanto, os sistemas analisados que fazem o planejamento dinâmico do workflow não conseguem captar o comportamento do Grid nem modelar especificidades que podem ser encontradas em classes de aplicações, seja por empregarem técnicas aleatórias de escalonamento, seja por deixarem a cargo do próprio usuário a responsabilidade do escalonamento. As propostas de criação de grupos de tarefas e de utilização de múltiplas máquinas para controlar a execução do workflow

empregada nos sistemas Pegasus e GRAND não resolvem por completo a questão de desempenho mas apresentam uma solução que pode ser complementar ao problema.

A análise desses sistemas aponta o Euryale como aquele com as características que mais se aproximam da solução necessária para o problema do desempenho. Embora as estratégias disponíveis para a seleção de sítio sejam ingênuas, a sua arquitetura permite o acoplamento de novos algoritmos. As soluções de especificação e desempenho para workflows desenvolvidas nesta tese combinam as características de definição abstrata, escalonamento dinâmico e replanejamento de tarefas encontradas no Euryale com as vantagens de uma especificação menos trabalhosa, algoritmos não aleatórios para seleção de sítios e replanejamento de tarefas configurável.

III – EURYALE^{+}: UMA ARQUITETURA COM ESCALONAMENTO DINÂMICO

III.1 - Introdução

Apesar de existirem no Grid, um conjunto de ferramentas para submissão de tarefas, replicação de arquivos e localização de recursos, o seu uso é em geral difícil para um cientista não especialista em computação. Se o workflow for constituído por um conjunto pequeno de componentes, o próprio cientista pode ser o responsável pela especificação e pelo planejamento de sua execução, dependendo das suas habilidades no manuseio destas ferramentas. Entretanto, se a aplicação for formada por centenas ou milhares de tarefas, é imprescindível que existam ferramentas que automatizem o projeto e o planejamento de execução do workflow no Grid.

O objetivo deste capítulo é descrever a arquitetura de Euryale+, com a proposta de solução para o escalonamento dinâmico de workflows científicos em Grid. As propostas de escalonamento de tarefas desenvolvidas nesta tese fazem uso de um conjunto de sistemas largamente utilizados em Grid. Em particular, nossas soluções foram implementadas no contexto do projeto GriPhyN-Virtual Data System (VDS) que disponibiliza um conjunto de ferramentas para definição, planejamento e execução de workflows em Grid .

Inicialmente é feita uma breve descrição do sistema Globus que é o sistema de infra-estrutura básica mais utilizado nos ambientes de Grid. Em seguida, são comentadas algumas características do sistema DagMan utilizado pelo VDS para fazer a gerência de workflows. A quarta seção detalha a arquitetura do sistema de planejamento Euryale enquanto que a última seção descreve as extensões promovidas para implementação das soluções gerando o Euryale+.

III.2 - Globus Toolkit

O "Globus Toolkit (GT4)" (FOSTER, 2005) é um sistema aberto, desenvolvido em conjunto por diversas instituições de pesquisa, universidades e colaboradores individuais, e provê um conjunto de recursos para a construção de sistemas computacionais em Grid. A figura 2 ilustra os principais componentes do GT4. Um conjunto de serviços de infra-estrutura implementa interfaces para segurança de acesso, gerenciamento de dados, gerenciamento e execução de programas, monitoramento e descobrimento de recursos, além de ferramentas para instalação e execução de serviços Web desenvolvidos nas linguagens Java, C e Python. As caixas com bordas pontilhadas indicam serviços disponibilizados no *toolkit,* mas ainda em fase de testes.

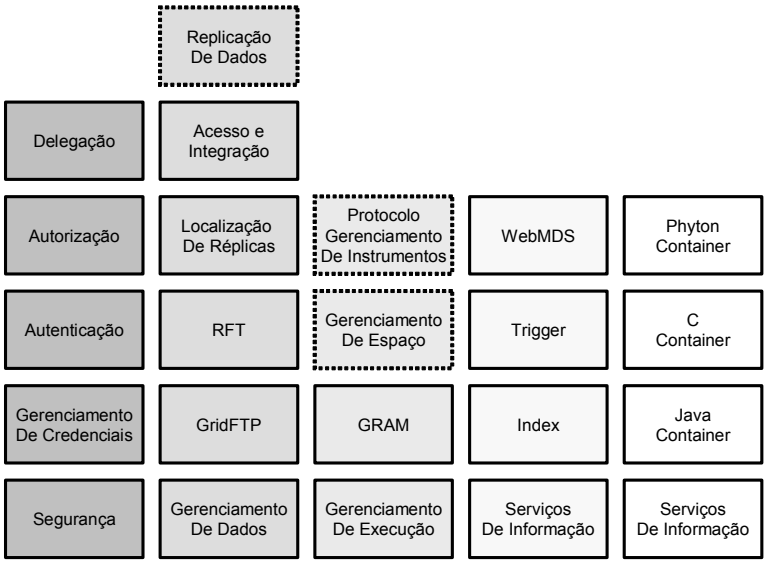


Figura 2 – Componentes principais do GT4 (adaptado de FOSTER, 2005)

- **Segurança**

Os componentes de segurança do GT4 implementam protocolos para proteção de mensagens, autenticação de usuários, e delegação. Por *default*, todo usuário e recurso computacional possui uma credencial de chave pública. Cada entidade pode validar a credencial de outra entidade, usar estas credenciais para estabelecer um canal seguro de comunicação e agir em nome de um usuário remoto por um determinado período de tempo.

- **Gerenciamento de Dados**

Os componentes para o gerenciamento de dados do GT4 são: GridFTP, RFT, RLS, OGSA-DAI e DRS. O GridFTP implementa um serviço de alto desempenho para transferência de arquivos, enquanto que o serviço RFT (*Reliable File Transfer*) gerencia múltiplas transferências de arquivos que utilizem o GridFTP. O registro e a localização de réplicas são implementados pelo serviço RLS (*Replica Location Service*). O serviço OGSA-DAI (*Globus Data Access and Integration*) possibilita o acesso e a integração de dados residentes em banco de dados relacionais ou em formato XML. O DRS (*Data Replication Service*) combina os recursos do GridFTP e do RLS para fornecer um serviço para replicação de dados.

- **Gerenciamento de execução**

O GRAM (*Globus Resource and Allocation Manager*) é o componente básico do GT4 para submeter, monitorar e controlar tarefas em computadores remotos e possui interface para vários escalonadores como Condor e PBS. O gerenciamento de espaço é responsabilidade do componente WMS (*Workspace Management Service*) enquanto que o GTPC (*Grid TeleControl Protocol*) é um serviço voltado para a gerência de instrumentos e tem sido usado em microscópios e equipamentos para detecção de terremotos.

- **Serviços de informação**

Os serviços de informação permitem que o usuário obtenha uma descrição dos recursos do Grid. O GT4 fornece dois serviços agregadores que coletam dados de fontes de informação: *Index Service* e *Trigger Service*. O primeiro serviço provê informações sobre o estado de outros serviços no Grid, enquanto que o segundo pode ser usado para construir notificações por email para avisar administradores de sistemas em caso de falhas. O serviço *WebMDS* pode ser usado para criar visões especializadas dos dados coletados pelo *Index Service*.

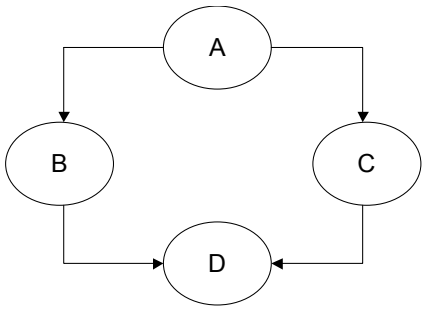
- **Construção de serviços**

O GT4 inclui software para possibilitar o desenvolvimento de componentes que implementem interfaces para serviços Web. O GT4 disponibiliza "containers" para a instalação e execução de serviços Web escritos em Java, C ou Python.

III.3 – Condor/DagMan

Como já mencionado no capítulo II, DagMan é um meta-escalonador para tarefas Condor. DagMan submete tarefas para o Condor na ordem representada pelo grafo de tarefas e processa os resultados. O grafo que representa o workflow é definido em um arquivo e especifica a lista das tarefas, a localização das rotinas para pré e pós-processamento de cada tarefa, caso existam, a descrição das dependências e o número de re-execuções para o caso de falhas. A figura 3 ilustra um arquivo de descrição com o respectivo DAG e um arquivo de submissão da tarefa do Condor.

```
# Filename: diamond.dag
#
Job A A.condor
Job B B.condor
Job C C.condor
Job D D.condor
#
Script PRE A top_pre.csh
Script PRE B mid_pre.perl $JOB
Script POST B mid_post.perl $JOB
$RETURN
Script PRE C mid_pre.perl $JOB
Script POST C mid_post.perl $JOB
$RETURN
Script PRE D bot_pre.csh
#
PARENT A CHILD B C
PARENT B C CHILD D
Retry C 3
```




```
# Filename:diamond_job.condor
executable = /path/diamond.exe
output = diamond.out.$(cluster)
error = diamond.err.$(cluster)
log = diamond_condor.log
universe = vanilla
```

Figura 3 – Arquivos de descrição do DAG e submissão de tarefa (CONDOR TEAM, 2003)

Cada nó do DAG representa uma tarefa que é descrita no arquivo Condor para submissão. Este arquivo define os locais onde se encontra o executável além dos arquivos de saída, erros e log.

DagMan também permite que o número de rotinas de pré e pós-processamento bem como o número de tarefas executando concorrentemente seja definido por meio de parâmetros de execução. Uma vez submetido, o workflow pode ser monitorado através de comandos que listam o estado das tarefas.

III.4 - Virtual Data System

O projeto GriPhyn Virtual Data System (VDS) define uma arquitetura para integrar arquivos, programas e computações para a produção de dados fornecendo um conjunto de ferramentas para expressar, executar e registrar os resultados de workflows.

O sistema combina uma linguagem para a especificação de workflows abstratos, denominada *Virtual Data Language* (VDL) e um catálogo denominado *Virtual Data Catalog* (VDC) que armazena em XML a definição do workflow. O VDS também disponibiliza dois sistemas para o planejamento de workflows, já referenciados no capítulo II:

- *Pegasus* baseado no planejamento estático, global;
- *Euryale* baseado no planejamento dinâmico e local e que vem sendo utilizado na construção de protótipos e pesquisas.

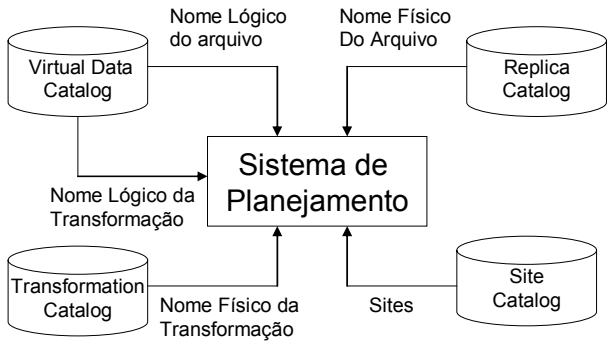


Figura 4 - Componentes do VDS para o planejamento

Além do VDC, o sistema incorpora mais três catálogos responsáveis por informar ao sistema de planejamento a localização dos programas executáveis, arquivos de dados e as configurações do Grid. Eles são chamados respectivamente *Transformation Catalog*, *Replication Catalog* e *Site Catalog*. A figura 4 ilustra os componentes do VDS para o planejamento de workflows.

Nas próximas subseções serão descritos em detalhes como são especificados os workflows para o VDS e a arquitetura do sistema de planejamento Euryale.

III.4.1 - Especificação do Workflow

A linguagem VDL permite que o usuário defina um workflow usando dois tipos de declaração: *Transformação* e *Derivação*. Uma transformação (TR) define logicamente um programa com seus parâmetros de execução, arquivos de entrada e arquivos de saída.

```
TR BM::Blast(input I, output O)
  {
  argument = "-i "${input:I};
  argument = "-o "${output:O};
  argument = "-p blastp -T"; }
DV DV1->BM::Blast(
    F1=@{input:"I1"},
    F2=@{output:"O1"});
DV DV2->BM::Blast(
    F1=@{input:"I2"},
    F2=@{output:"O2"});
DV DV3->BM::Blast(
    F1=@{input:"I3"},
    F2=@{output:"O3"});
DV DV4->BM::Blast(
    F1=@{input:"I4"},
    F2=@{output:"O4"});
DV DV5->BM::Blast(
    F1=@{input:"I5"},
    F2=@{output:"O5"});
```

Figura 5 - Definição em VDL para o workflow

Uma derivação (DV) corresponde a uma execução de uma transformação e informa os nomes lógicos dos arquivos de entrada e saída da transformação.

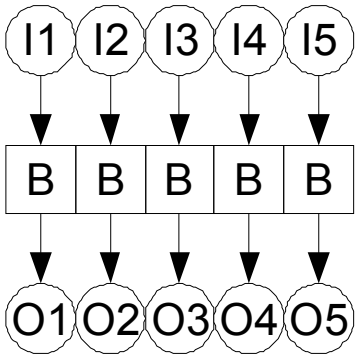


Figura 6 - Representação gráfica para o workflow

A figura 5 exemplifica o uso da VDL. Este exemplo define o workflow ilustrado na figura 6 que representa uma execução do programa Blast. Como pode ser observado, o workflow é composto por uma única transformação (TR) e cinco derivações (DV), cada uma processando um único arquivo de entrada e produzindo um arquivo de saída.

III.4.2 - Arquitetura de Planejamento do Sistema Euryale

O sistema Euryale realiza o planejamento de um workflow em duas etapas: Uma antes da execução e outra durante a execução do workflow. A figura 7 ilustra estas duas etapas do planejamento.

Primeiramente, Euryale recupera do VDC a definição abstrata do workflow e produz os arquivos de descrição do Condor para submissão de cada tarefa e mais um arquivo com a descrição do DAG. Este arquivo lista a relação das tarefas do workflow e eventuais dependências.

Durante a execução do workflow é feita a segunda fase do planejamento. O sistema DagMan para gerenciamento de workflow invoca o Euryale antes da execução de cada tarefa, para que seja concretizado o mapeamento dos programas e arquivos nos

arquivos Condor de submissão. Euryale é novamente invocado pelo DagMan, toda vez que uma tarefa termina a sua execução.

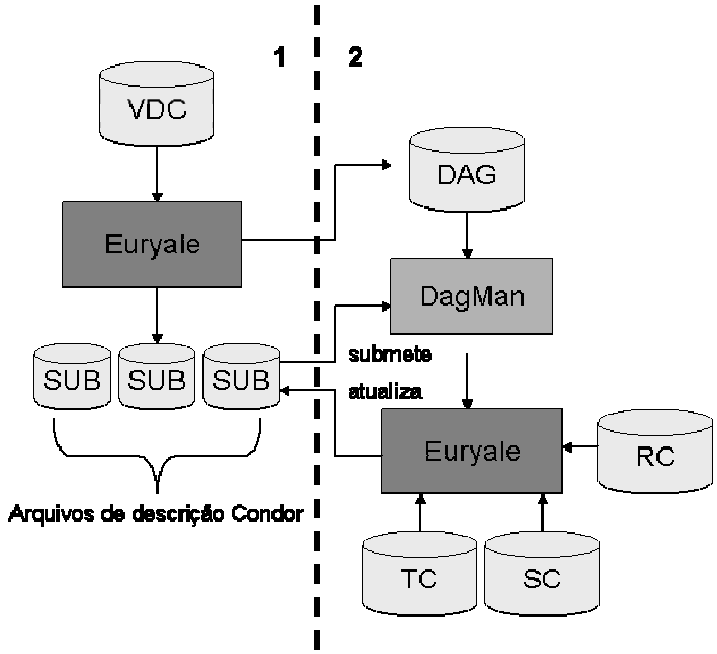


Figura 7 - Planejamento em duas fases do sistema Euryale

Para realizar a concretização de uma tarefa, Euryale utiliza além do VDC, as informações dos catálogos de transformação (TC), sítios (SC) e réplicas (RC). O catálogo VDC fornece os nomes lógicos dos arquivos de entrada e saída, e da transformação a ser executada. O catálogo de transformação especifica como uma transformação deve ser executada, ou seja, informa o nome do executável, a localização do programa e seus argumentos. O catálogo de sítios é responsável por fornecer informações sobre os locais candidatos a execução do workflow. Este catálogo informa para cada sítio do Grid o nome do servidor de GridFTP, o diretório para transferência do arquivos de entrada, o diretório para executar a tarefa, entre outra informações. O catálogo de réplicas fornece a localização completa, ou seja, o nome físico de cada réplica associada a um arquivo lógico. Este serviço pode ser implementado com o RLS do Globus Toolkit, ou por intermédio de um Sistema Gerenciador de Banco de Dados.

Euryale realiza o planejamento de execução do workflow da seguinte maneira:

1. Com as informações coletadas dos catálogos, Euryale cria uma lista de locais candidatos a executar a tarefa;
2. Euryale chama um componente externo denominado Selecionador de Sítio passando a lista de candidatos e espera por uma solução, ou seja, o sítio no Grid que irá executar a tarefa.
3. Uma vez definido o local de execução, Euryale cria os diretórios temporários remotos para armazenamento de dados e execução da tarefa;
4. Euryale consulta o catálogo de réplicas e transfere para o local de execução os arquivos de entrada que não possuírem réplicas no local.
5. Euryale atualiza o arquivo de descrição do Condor e termina sua execução.

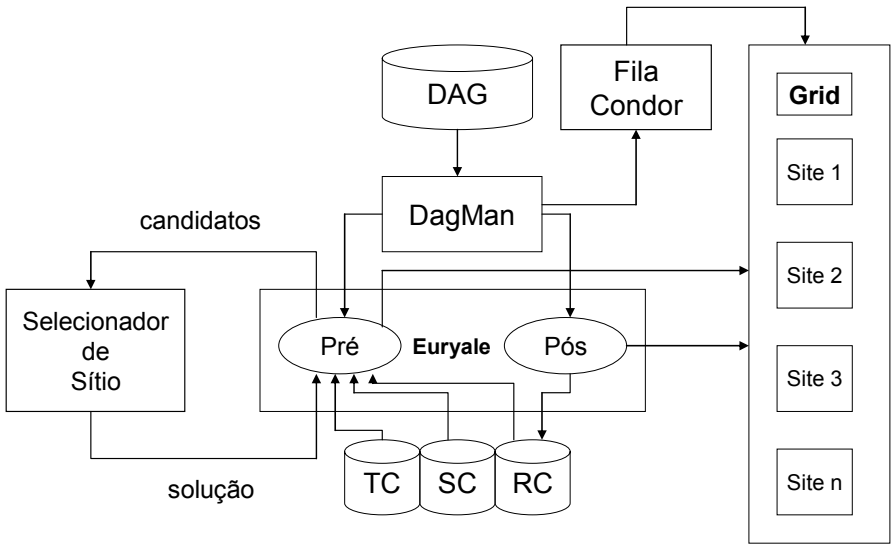


Figura 8 - Detalhamento da arquitetura do Euryale

O Selecionador de Sítio é o modulo responsável por implementar as estratégias de execução do workflow e seu objetivo é escolher o local para executar cada tarefa do workflow. A figura 8 detalha a arquitetura de planejamento do Euryale e sua interação com o Selecionador de Sítio e o DagMan. Toda a vez que uma tarefa termina, DagMan invoca o Euryale que verifica o código de execução do programa. Caso não tenha ocorrido nenhum erro, os diretórios temporários criados remotamente para a execução da tarefa são apagados e os arquivos produzidos registrados no catálogo de réplicas.

III.5 – Proposta da Arquitetura Euryale^{+}

A seleção do sítio para executar uma tarefa é a parte mais importante do planejamento dinâmico de um workflow para a obtenção de desempenho em ambientes de Grid. As soluções propostas nesta tese para o planejamento de workflows foram implementadas como extensões ao sistema Euryale. A escolha recaiu sobre o Euryale por este sistema apresentar uma arquitetura de planejamento dinâmica, disponibilizar o acesso ao código fonte de suas rotinas e trabalhar com selecionadores de sítios externos. Estas características são vantajosas porque permitem o desenvolvimento de novas estratégias de planejamento que podem ser acopladas ao sistema.

As estratégias promovidas para o planejamento dinâmico de workflows contemplam aspectos de gerência dinâmica dos dados, replanejamento de tarefas e soluções não aleatórias para seleção de sítios. Estas estratégias para escalonamento são baseadas na geração de grupos de tarefas e na observação do desempenho corrente dos sítios no Grid. Além das estratégias de escalonamento, identificamos a necessidade de facilitar a especificação do workflow através de um nível semântico mais alto o que acarretou na implementação de uma metalinguagem para especificação abstrata do workflow. Nas próximas subseções são discutidas as limitações existentes no Euryale que motivaram as extensões promovidas em sua arquitetura e o detalhamento da arquitetura Euryale^{+}.

III.5.1 - Extensões para Definição Abstrata do Workflow

Embora a linguagem VDL viabilize a especificação de um workflow através dos dois tipos simples de declaração - transformação e derivação, a sua especificação para workflows mais complexos e com mais tarefas pode ser uma atividade trabalhosa. Por exemplo, se o workflow mostrado na figura 5 ao invés de cinco derivações, tivesse mil derivações, seria praticamente inviável para um usuário especificar diretamente em um editor de texto este workflow. Consequentemente, é muito freqüente que usuários do

VDS sejam obrigados a escrever pequenos programas para a geração automática das especificações em VDL do workflow. Buscando endereçar este problema da geração da definição abstrata do workflow, desenvolvemos uma ferramenta composta de uma meta-linguagem denominada *Linguagem para Geração de Workflow* (WGL) e de um interpretador denominado Gerador de Workflow (GenWF). A WGL permite que o usuário defina em poucas declarações a estrutura de seu workflow. O GenWF interpreta a WGL e gera a VDL completa para o workflow, bem como o arquivo com a especificação do DAG para o processamento pelo Condor-DagMan. A figura 9 ilustra os componentes para a geração automática da definição do workflow.

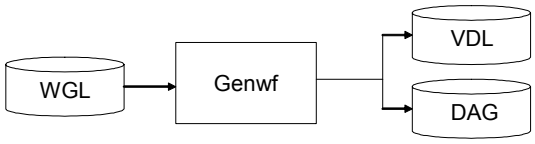


Figura 9 - Geração automática do Workflow

III.5.1.1 - Linguagem para Geração de Workflow (WGL)

A WGL consiste de dois comandos: *Dataset* e *Program*. O comando *Dataset* é usado para modelar os arquivos de entrada e saída enquanto que o comando *Program* modela as características de cada tarefa do workflow. A sintaxe dos dois comandos é mostrada a seguir:

Dataset (nome, número_arquivos, tamanho)
Program (nome, dataset_entrada, dataset_saida, número_instâncias, duração)

Um *Dataset* representa um conjunto de arquivos e é descrito por três atributos: nome, número de arquivos e o tamanho de cada arquivo. Quando da geração da VDL, o nome lógico de cada arquivo de um Dataset é gerado pela concatenação do nome do *Dataset* com um número seqüencial. O atributo tamanho é definido para o caso do usuário querer gerar uma massa de dados para testes para o workflow. Desta forma é possível não só gerar a definição do workflow, como também instanciar todos os componentes deste workflow para a sua execução no Grid.

Cada programa é descrito por cinco atributos: nome, identificação do *Dataset* de entrada, identificação do *Dataset* de saída, número de instâncias do programa e duração.

A identificação do *Dataset* de entrada e saída pode ser especificada como uma lista de elementos separados por virgula e compreendidos entre dois colchetes. A duração do programa pode ser utilizada para gerar workflows de teste para serem executados com programas que simulem a duração real das tarefas.

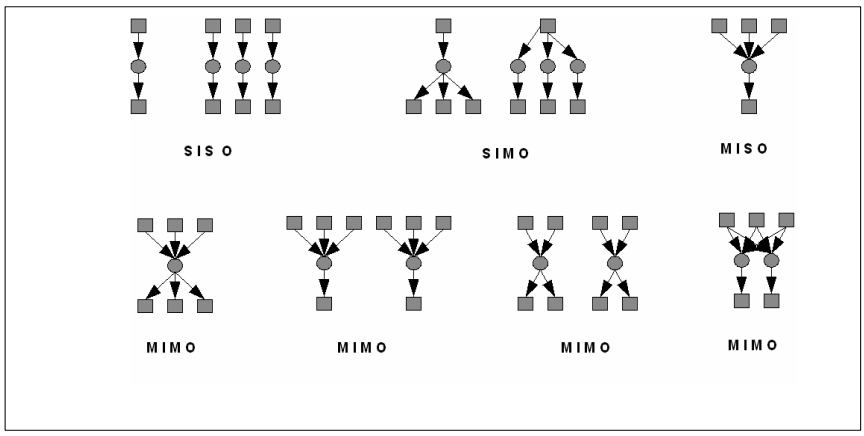


Figura 10 - Padrões de relacionamentos implementáveis pela WGL

Os relacionamentos obtidos através da combinação destes dois comandos permitem a especificação de workflows expressos como DAG com as mais diversas formas. A figura 10 ilustra os relacionamentos possíveis de serem implementados com a ferramenta. Quatro padrões de associação são definidos:

- *SISO* (*Single Input Single Output*) - Um arquivo de entrada, um arquivo de saída;
- SIMO (*Single Input Multiple Output*) - Um arquivo de entrada, muitos de saída;
- MISO (*Multiple Input Single Output*) - Muitos arquivos de entrada, um de saída;
- MIMO (*Multiple Input Multiple Output*) - Muitos arquivos de entrada, muitos de saída.

A figura 11 mostra a especificação em WGL para gerar a VDL do exemplo mostrado na figura 5.

```
Dataset (I:5:10)
Dataset (O:5:10)
Program (Blast:I:O:5)
```

Figura 11 - WGL para geração da VDL na figura 5

Pode-se observar que com apenas três comandos é possível gerar toda a VDL necessária para definir o workflow. Se ao invés de cinco derivações, o workflow executasse mil derivações, as únicas mudanças necessárias seriam alterar de 5 para 1.000 o número de arquivos nos comandos *Dataset* e o número de instâncias do programa Blast.

III.5.1 - Extensões para o planejamento de tarefas

Por *Default*, o sistema Euryale disponibiliza somente estratégias aleatórias de escalonamento de tarefas. Estas estratégias não conseguem captar o comportamento dinâmico do Grid nem tampouco aspectos específicos que podem ser importantes para o planejamento de aplicações como por exemplo à localização dos dados. Além disto, o comportamento do Euryale com relação aos dados de entrada para o processamento de uma tarefa é o de transferir os arquivos para o sítio selecionado e uma vez concluída a execução, excluir do sítio estes arquivos transferidos. Tal comportamento não permite que as estratégias existentes se beneficiem da replicação dinâmica de arquivos durante parte ou totalidade da execução do workflow.

Em diversas classes de aplicações, é bastante comum que um mesmo conjunto de arquivos de entrada tenha que ser processado por diferentes tarefas de um mesmo workflow. Portanto, dispor de um mecanismo que adie a exclusão de arquivos pode trazer benefícios ao desempenho e eliminar transferências desnecessárias.

Um outro aspecto importante também relacionado à gerência dinâmica dos dados no Grid, é o espaço utilizado por estes dados ao longo da execução do workflow. Pode ser que seja necessário controlar quais arquivos devem permanecer replicados e quais dever ser excluídos para permitir um melhor aproveitamento do espaço em disco nos diversos sítios.

Apesar de implementar um recurso de tolerância à falhas através do replanejamento de tarefas, este replanejamento é iniciado pelo Euryale após duas horas de inatividade de uma tarefa, ou seja, em casos em que a tarefa é submetida mas

permanece na fila remota sem ser executada durante este período. Como o tempo de execução das tarefas pode variar consideravelmente de um workflow para outro, é desejável dispor de um mecanismo mais flexível que detecte automaticamente este tipo de situação e promova o replanejamento de tarefas enfileiradas.

A figura 12 ilustra a arquitetura Euryale^{+}. As extensões promovidas (realçadas na figura) visam endereçar as deficiências discutidas anteriormente. O Selecionador de Sítio é o componente da arquitetura que informa ao sistema de planejamento o local de execução de cada tarefa. Esta tese estende a família de algoritmos para seleção de sítios do Euryale com mais duas estratégias não aleatórias.

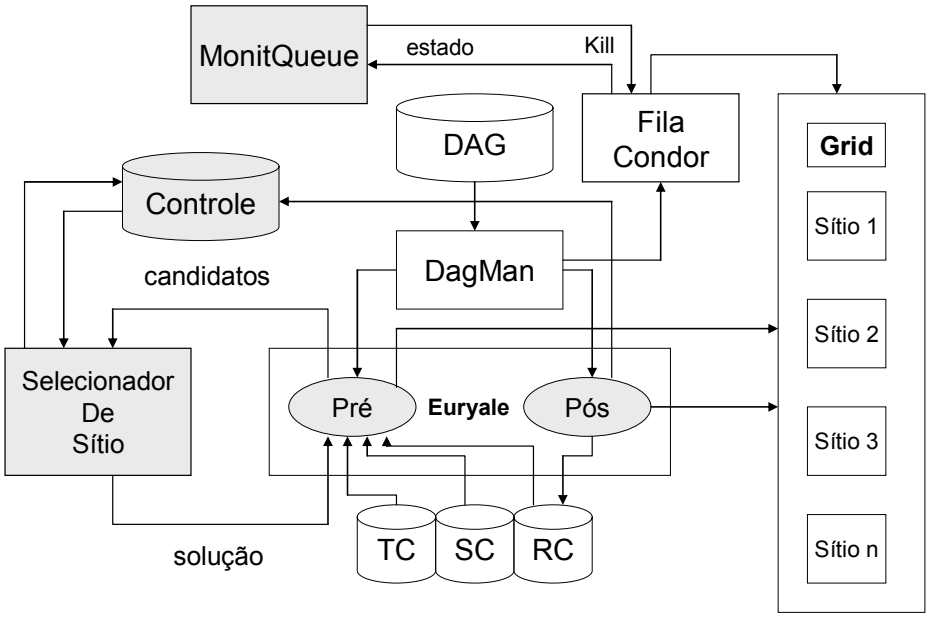


Figura 12 - Arquitetura do Euryale^{+} com as extensões promovidas

A estratégia por agrupamento de tarefas visa reduzir o número de transferências redundantes de arquivos durante a execução do workflow. Para isto, os arquivos são temporariamente replicados no Grid porque serão acessados por tarefas no contexto da mesma aplicação. O algoritmo de escalonamento define o local onde cada tarefa deve ser executada baseado na identificação do grupo ao qual a tarefa está relacionada.

A implementação desta estratégia implicou na modificação do comportamento das rotinas de pré e pós-processamento do Euryale. Assim, durante o pré-processamento

de uma tarefa, uma vez que o selecionador de sítios tenha definido local de execução e os arquivos não existentes no sítio tenham sido transferidos, estes arquivos passam a ser registrados no catálogo de réplicas. Desta forma, eles ficam disponíveis para também serem utilizados por outras tarefas do workflow. Analogamente, a rotina de pós-processamento das tarefas também sofreu modificações de forma a inibir a exclusão dos arquivos de entrada transferidos para o sítio de execução após o processamento de uma tarefa. Como o escalonamento é realizado tendo por base um projeto de agrupamento de tarefas, a definição dos grupos e as informações de mapeamento entre grupos, tarefas e sítios são mantidas na base de controle.

Na estratégia baseada na observação do desempenho, o selecionador de sítio toma a sua decisão baseado no comportamento dos sítios durante a execução das tarefas do workflow previamente submetidas. Para isto, toda vez que uma tarefa é escalonada são guardadas na base de controle as informações do sítio escolhido. Da mesma forma, quando uma tarefa é completada, a rotina de pós-processamento registra o sucesso e o horário de conclusão. A figura 13 ilustra o esquema conceitual da base controle.

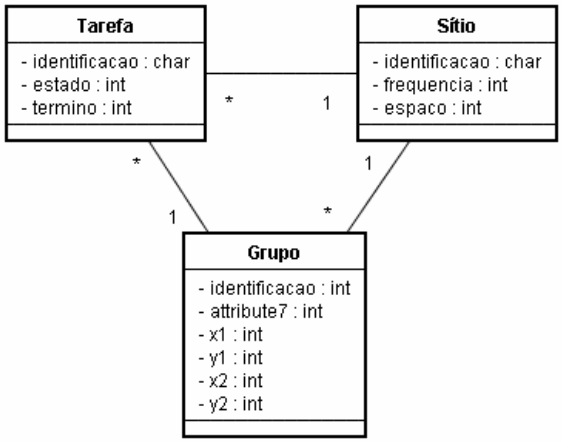


Figura 13 - Esquema conceitual da base de controle

O Monitor de Fila (MonitQueue) é o último componente da arquitetura Euryale^{+}. Sua função é monitorar a fila de tarefas submetidas que é mantida localmente pelo Condor e cancelar aquelas tarefas não inicializadas remotamente após um determinado período de tempo. Este tempo de espera deve ser informado pelo usuário toda vez que

um workflow é submetido para execução pelo DagMan. Assim, caso um determinado sítio não esteja processando as tarefas recebidas em um tempo razoável, este fraco desempenho não comprometerá o desempenho do workflow como um todo na medida em que as tarefas serão canceladas, replanejadas e redirecionadas para outros sítios.

Os capítulos IV e V fazem o detalhamento das extensões introduzidas para o planejamento de workflows descrevendo as estratégias e os algoritmos que apoiam as implementações.

IV – PLANEJAMENTO DE WORKFLOWS ESPACIAIS

IV.1 Introdução

Em muitas áreas científicas como na astronomia e na geografia, os dados modelados possuem coordenadas espaciais e representam características de alguma região do espaço sideral ou da terra. Em geral, estas características espaciais são modeladas usando-se duas ou três dimensões, mas outras dimensões podem ser utilizadas para representar atributos adicionais tais como o componente temporal. Vários sistemas de coordenadas específicas do domínio do problema são empregados e os arquivos freqüentemente são nomeados e catalogados pelas coordenadas da região que representam.

As aplicações que processam os dados espaciais são geralmente divididas em tarefas que processam partições do espaço representadas por um conjunto de arquivos. Como os arquivos que caracterizam uma determinada região do espaço podem ser referenciados por mais de uma tarefa, e as tarefas podem referenciar um conjunto grande de arquivos, é interessante dispor de uma estratégia de minimize o número de transferências destes arquivos durante a execução do workflow.

O objetivo deste capítulo é descrever a estratégia para o processamento de workflows espaciais baseada no agrupamento de tarefas pelo conhecimento da proximidade espacial. Ele está organizado da seguinte forma: na segunda seção é relatada a motivação para o desenvolvimento da estratégia enquanto que na terceira seção é feito o detalhamento da aplicação espacial. Em seguida são apresentados os componentes da estratégia, constituída de um algoritmo para a definição de grupos, de um algoritmo para transformação do DAG e de um algoritmo para escalonamento de tarefas. A última seção descreve os experimentos realizados para avaliação da estratégia.

IV.2 - Motivação

A oportunidade de explorar o conhecimento da proximidade espacial para melhorar o desempenho de workflows espaciais pode ser encontrada em várias aplicações usadas no projeto "*Sloan Digital Sky Survey*" (SDSS PROJECT, 2006). O SDSS é um projeto de pesquisa em astronomia cujo objetivo é criar um banco de dados público com informações de aproximadamente um quarto do espaço sideral, além de um mapa tridimensional de aproximadamente um milhão de galáxias e quasares.

Um dos workflows definidos no projeto SDSS é o "*Coadd*". Este workflow foi executado uma única vez no ambiente Grid3 (FOSTER et al., 2004), com alta participação dos técnicos no planejamento manual de suas tarefas e consumiu aproximadamente setenta dias para ser concluído. Uma análise inicial deste workflow indicou que as dificuldades encontradas no processamento foram provenientes de dois fatos: a) a maioria das tarefas referenciava um grande número de arquivos que também eram referenciados por tarefas processando regiões próximas do espaço, ou seja, existia um alto grau de compartilhamento de arquivos; b) no mecanismo empregado para executar o workflow, não havia formas de controlar a quantidade de espaço para o armazenamento dos arquivos nos sítios.

Os problemas relatados no processamento do *Coadd* não são específicos a este workflow e podem ser encontrados em outras aplicações espaciais nas quais as tarefas processam regiões sobrepostas do espaço. Isto é comum em situações onde o resultado do processamento para um determinado ponto no espaço, depende das condições da vizinhança. Como forma de endereçar os dois problemas levantados foi desenvolvida uma estratégia para o planejamento do workflow levando-se em consideração dois aspectos: primeiro, os arquivos de entrada de uma tarefa são mantidos no sítio ao final do processamento para possibilitar o reuso por tarefas adjacentes; segundo, estes arquivos devem ser removidos dos sítios na medida em que eles não sejam mais necessários, visto que replicar todos os arquivos do workflow é inviável devido ao enorme volume de dados.

A estratégia desenvolvida é baseada na criação de grupos de tarefas por afinidade espacial e do escalonamento destas tarefas no Grid de forma a tirar proveito dos arquivos previamente processados. Os algoritmos desenvolvidos reduzem o número de transferências de arquivos no Grid e aumentam o desempenho do workflow. A estratégia tira vantagem da localidade dos dados empregando a replicação dinâmica dos arquivos e o escalonamento das tarefas do workflow em uma ordem que reduz o número de réplicas criadas.

IV.3 - SDSS/Coadd

O workflow *SDSS/Coadd* é formado por 44.400 tarefas que processam 2.5 Terabytes armazenados em 544.500 arquivos distintos. Estes mais de quinhentos mil arquivos de entrada são referenciados 5.419.370 vezes. Na média, um arquivo é referenciado por aproximadamente dez tarefas diferentes. Estes arquivos contêm imagens tiradas de regiões do espaço sideral que são catalogadas em um sistema de coordenadas formado por três dimensões: *right ascencion* (ra), *declination* (dec) e filtro espectral. Por simplicidade nós tratamos ra, dec e filtro como se fossem coordenadas (X,Y,Z) em um espaço tridimensional.

O padrão pelo qual as tarefas do workflow fazem a travessia do espaço bi-dimensional para processar seus arquivos determina como os grupos devem ser criados.

Estes padrões podem naturalmente variar entre as diferentes aplicações com relação às dimensões, abrangência, etc. Por exemplo, em um espaço bi-dimensional, uma aplicação pode ter as suas tarefas processando arquivos espalhados sobre uma única dimensão enquanto outras aplicações podem ser caracterizadas por possuírem tarefas processando arquivos distribuídos por ambas dimensões. A figura 14 ilustra estes dois tipos de varredura, onde "A" representa um arquivo e "T" uma tarefa. De fato, várias versões do algoritmo de processamento implementado no workflow *SDSS/Coadd* utilizam estes dois padrões de varredura. No experimento realizado, as tarefas do workflow fazem a varredura de arquivos nas duas dimensões (ra e Dec).

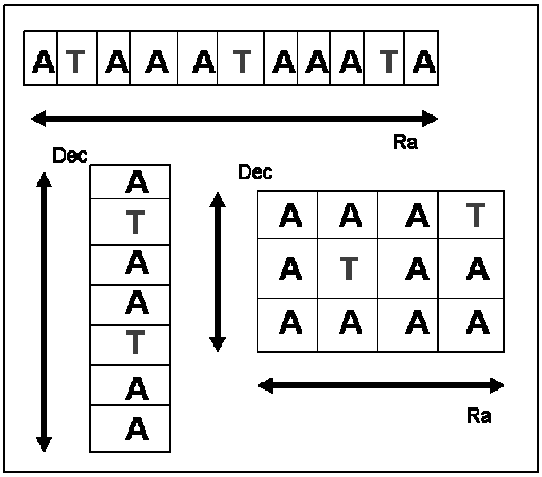


Figura 14 - Padrões de varredura espacial

IV.4 - Estratégia de Agrupamento por Proximidade Espacial

Uma possível abordagem para se processar workflows espaciais é transferir dinamicamente os arquivos de entrada necessários, e uma vez processada a tarefa, excluí-los do sítio. Esta estratégia requer um mínimo espaço em disco para o processamento do workflow. Entretanto, o número de transferências de arquivos entre os sítios do Grid pode ser gigantesco. Uma vez que o tempo necessário para transferir todos os arquivos de entrada de uma tarefa pode ser maior que o tempo de processamento da mesma, esta estratégia, apesar de eficiente no manuseio do espaço em disco, pode acarretar em um mau desempenho.

A replicação prévia de todos os arquivos também pode ser empregada, o que reduz ao mínimo o número de transferências de arquivos entre os sítios, mas pode não ser viável se não houver espaço disponível nos sítios. Uma solução intermediária consiste na replicação dos arquivos enquanto houver espaço disponível. Embora ajude a reduzir o número de transferências de arquivos, esta estratégia não consegue captar o compartilhamento dos arquivos pela tarefas do workflow se for realizada de forma aleatória.

Técnicas de agrupamento são largamente utilizadas em muitas áreas da ciência da computação. Os Sistemas Gerenciadores de Banco de Dados, por exemplo,

disponibilizam uma técnica de armazenamento de dados em grupos de registros para permitir que blocos de dados de tabelas que se referenciam sejam armazenados contiguamente. Desta forma, operações como a junção pode ser executada mais rapidamente através da redução de entrada/saída. A mineração de dados é outra área que explora os algoritmos de agrupamento para os dados como forma de caracterizar grupos de clientes baseados nos padrões de compra, grupos de genes e proteínas com funcionalidade similar, categorização de documentos na web, entre outros.

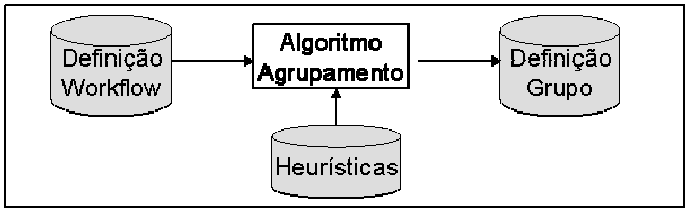


Figura 15 - Projeto de Definição de Grupos

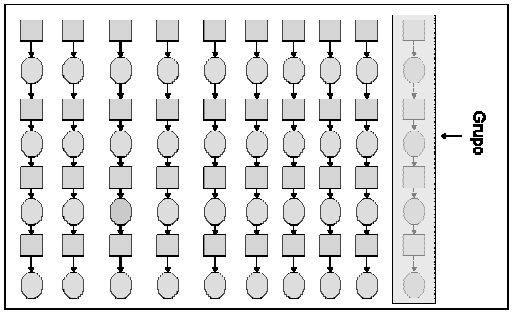


Figura 16 - Workflow com seus componentes em seqüências de tarefas

Grupos de tarefas podem ser computados de diversas formas. Dada uma definição de um workflow, o conjunto de tarefas, arquivos e dependências pode ser extraído e passado para um algoritmo que através de heurísticas pode produzir um projeto de grupos. A figura 15 ilustra esta abordagem. Nos casos onde o workflow é formado por uma série de execuções seqüenciais de tarefas como mostrado na figura 16, o projeto é simples e intuitivo, agrupando todas as tarefas que estão em seqüência no workflow. Desta forma, se uma tarefa que processa um arquivo é escalonada para executar no mesmo local da tarefa que gerou este arquivo, nenhuma transferência de arquivo se faz necessária.

IV.4.1 - Geração de Grupos por Proximidade Espacial

Em aplicações espaço-orientadas, a habilidade para automaticamente descobrir grupos de tarefas pelo compartilhamento de arquivos é mais desafiador, se o algoritmo tiver que descobrir como as tarefas se relacionam entre si baseados nos padrões de acesso aos arquivos. Este processo pode ser computacionalmente intensivo e não tira proveito das informações presentes nos sistemas de coordenadas correlatas das tarefas.

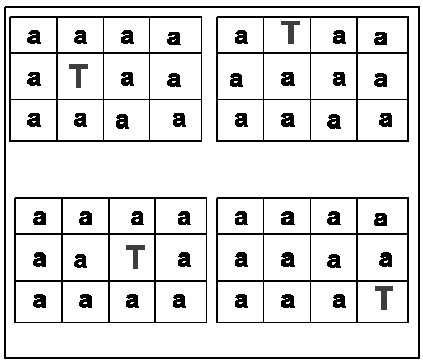


Figura 17 - Tarefas sem compartilhamento espacial

O objetivo da estratégia por agrupamento é reduzir o número de transferências de arquivos durante a execução do workflow. A idéia é criar grupos de tarefas por proximidade espacial e durante a execução do workflow, escalonar todas as tarefas de um mesmo grupo para um mesmo sítio. Cabe ressaltar que se o compartilhamento de arquivos entre as tarefas for pequeno ou inexistir, a estratégia proposta é inútil. A figura 17 ilustra esta situação.

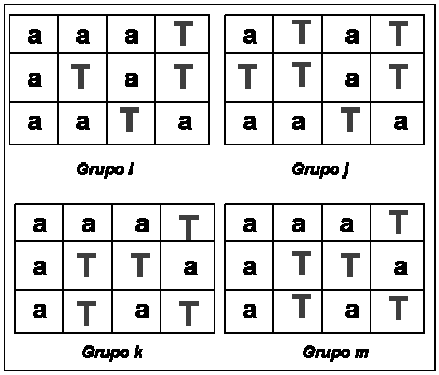


Figura 18 - Tarefas com alto grau de compartilhamento espacial

Entretanto, se existir um alto grau de compartilhamento de arquivos processados por tarefas adjacentes na mesma região do espaço como mostrado na figura 18, a estratégia por agrupamento de tarefas baseado na proximidade espacial pode trazer benefícios muito grandes pela redução no número de transferências de arquivos e pelo aumento de desempenho.

Uma métrica importante a ser analisada no projeto é o tamanho gerado para os grupos de tarefas. Se por um lado a geração de um número pequeno de grupos pode reduzir o número de transferências de arquivos entre os sítios, pode também produzir grupos com inúmeras tarefas e consequentemente requerer um espaço de armazenamento alto. De forma análoga, a geração de um número grande de grupos reduz o espaço de armazenamento para cada grupo, mas pode acarretar no aumento de transferências de arquivos. Portanto, o algoritmo deve produzir grupos tais que o espaço em disco necessário para processar as suas tarefas seja alocável nos sítios do Grid e ao mesmo tempo possibilite o reuso dos arquivos para reduzir o número de transferências.

Como premissa para a estratégia, é assumido o conhecimento de todas as tarefas do workflow bem como suas coordenadas espaciais e respectivos arquivos de entrada e saída. A figura 19 ilustra o algoritmo SPCL para geração de grupos.

Algoritmo: *SPCL*
Entrada: Conjunto de tarefas do Workflow
Conjunto de arquivos de entrada
$f(T) \rightarrow G_i$
Saida: Conjunto G de grupos de tarefas
Passos:
1. Determinar a grade de processamento das tarefas
2. Determinar a abrangência espacial das tarefas
3. Determinar as coordenadas de cada grupo
4. Associar cada tarefa a um grupo baseado nas coordenadas dos grupos e tarefas

Figura 19 - Algoritmo *SPCL* para criação de grupos de tarefas

O primeiro passo do algoritmo é determinar a grade de processamento das tarefas. Como cada tarefa esta associada a uma coordenada, a grade de processamento é

determinada pelos limites inferiores e superiores das coordenadas x e y das tarefas do workflow. O segundo passo é definir a abrangência espacial de cada tarefa, ou seja, no caso do workflow Coadd, descobrir em quais direções (no caso X e Y) as tarefas processam os arquivos e qual a distância (medida em termos de pontos) que a maioria destes arquivos estão de cada tarefa. Uma vez conhecida a grade de processamento das tarefas e suas abrangências espaciais, pode-se definir as coordenadas de cada grupo na grade.

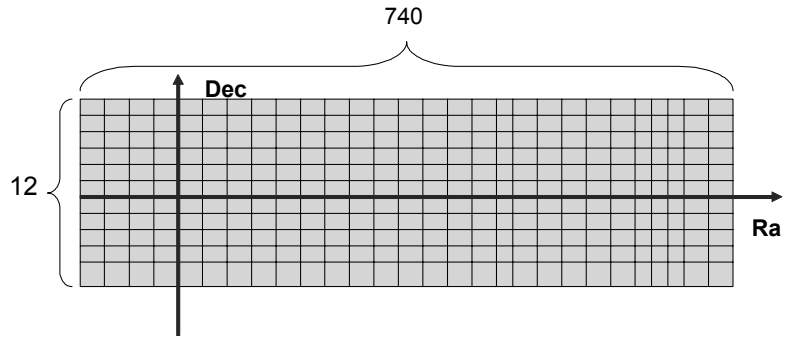


Figura 20 - Grade de processamento das tarefas do workflow Coadd

Uma análise da aplicação Coadd mostrou que a grade de processamento das tarefas é definida por doze pontos diferentes no eixo Y (Dec), setecentos e quarenta pontos diferentes no eixo X (ra) e cinco pontos distintos no eixo Z (filtro). Cada tarefa opera em um ponto específico (X, Y, Z) e processa arquivos relacionados a pontos (X, Y) adjacentes. A figura 20 ilustra a grade de processamento do Workflow. Além disto, a maioria dos arquivos neste workflow são referenciados por tarefas distantes três pontos em declinação e oito pontos em Ra. A aplicação destas dimensões para geração de grupos de tarefas produziu 372 grupos: 368 grupos com 120 tarefas e 4 grupos com 60 tarefas cada. O último passo do algoritmo, associa cada tarefa do workflow a um único grupo. Esta associação é baseada nas coordenadas de cada tarefa e de cada grupo.

IV.4.2 - Transformação do DAG

Uma vez que o número de grupos produzidos pelo algoritmo *SPCL* pode ser muito maior que o número de sítios disponíveis no Grid para processar o workflow, tarefas pertencentes a grupos diferentes terão que executar em um mesmo sítio. Para

garantir que uma grande parte dos arquivos que são compartilhados pelas tarefas de um mesmo grupo estarão disponíveis no sítio é fundamental que tarefas de diferentes grupos não sejam escalonadas para a execução simultânea no mesmo sítio. Ter dois ou mais grupos executando simultaneamente suas tarefas em um mesmo sítio pode fazer com que o espaço reservado para armazenamento de dados do workflow em um sítio fique esgotado.

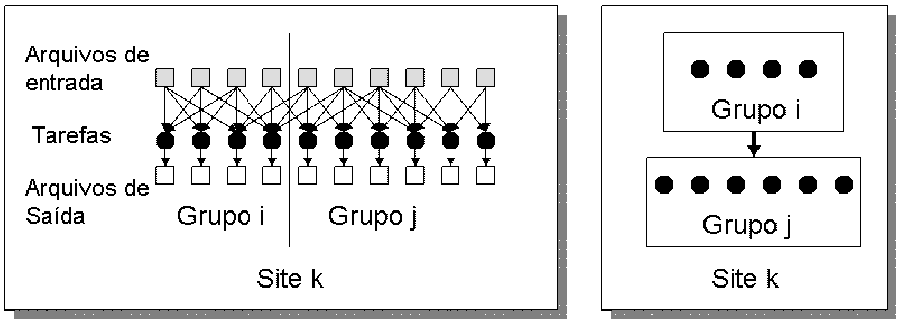


Figura 21 - Transformação do Dag

O não escalonamento simultâneo de tarefas de grupos diferentes em um mesmo sítio é garantido pelo algoritmo de transformação de DAG ilustrado graficamente na figura 21. O algoritmo pressupõe que as tarefas são independentes entre si. Supondo que as tarefas dos grupos *i* e *j* estão escalonadas para executarem no sítio *k*, então para maximizar o reuso dos arquivos de entrada pelas tarefas de um mesmo grupo, todas as tarefas do grupo *i* devem ser escalonadas antes das tarefas do grupo *j*.

O algoritmo de transformação do DAG altera a estrutura do DAG criando padrões de seqüências (dependências) entre as tarefas de grupos distintos. Assim, cada tarefa do grupo *j* tem como predecessoras todas as tarefas do grupo *i*. A figura 22 mostra o algoritmo para alteração do DAG. Para cada sítio definido, o algoritmo recupera o conjunto de grupos associados. A partir deste ponto, são criadas as dependências entre as tarefas associadas aos grupos distintos. O sistema Condor/DagMan permite que estas dependências sejam criadas pelo uso dos comandos "PARENT" e "CHILD". Estes comandos informam uma lista de tarefas predecessoras e sucessoras respectivamente.

Algoritmo: *AlteraDag*
Entrada: Conjunto S { s_i } de sítios
Conjunto G { g_i } de grupos associados ao Sítio s_i
Conjunto T { t_i } de tarefas associadas ao Grupo g_i
Saída: Arquivo F com dependências "PARENT - CHILD" entre as tarefas
Passos:

```
1. Para cada sítio s_i ∈ S faça
    1.1 flag ← 1
    1.2 Para cada grupo g_i ∈ s_i faça
        1.2.1 i ← 0
        1.2.2 j ← 0
        1.2.3 parent ← "PARENT "
        1.2.4 Para cada tarefa t_i ∈ g_i faça
            1.2.4.1 Se flag = 1
            1.2.4.2 então i ← i + 1
            1.2.4.3     predecessor [i] ← t_i
            1.2.4.4     parent ← parent || t_i || " "
            1.2.4.5 senão j ← j + 1
            1.2.4.6     sucessor [j] ← t_i
        1.2.5 Se flag = 0 então
            1.2.5.1 child ← "CHILD "
            1.2.5.2 grava parent no arquivo F
            1.2.5.3 Para ( x=1; x< j; x++ ) faça
                1.2.5.3.1 child ← child || sucessor[x]
                1.2.5.3.2 predecessor [x] ← sucessor[x]
                1.2.5.3.3 parent ← parent || sucessor[x]
            1.2.5.4 grava child no arquivo F
        1.2.6 senão Flag = 0
```

Figura 22 - Algoritmo *AlteraDag*

IV.4.3 - Algoritmo para Escalonamento das Tarefas

O último componente da estratégia é o algoritmo para o escalonamento de tarefas. Uma vez definidos os grupos e a estrutura do DAG para a execução do workflow, cada grupo é associado a um sítio do Grid. O escalonamento é realizado dinamicamente e feito individualmente para cada tarefa baseado na sua identificação. O algoritmo assume uma alocação específica de espaço em disco em cada sítio para o armazenamento dos arquivos das tarefas. As informações sobre os sítios, grupos, e tarefas são mantidas na base de dados de controle que é atualizada pelo algoritmo durante a execução do workflow.

Dada a identificação de uma tarefa, o algoritmo de escalonamento verifica qual é o grupo ao qual a tarefa pertence e obtém a identificação do sítio pela associação do grupo ao sítio. Uma vez definido o sítio para execução da tarefa, a base de controle é modificada para atualizar o espaço disponível no sítio para armazenamento dos dados. Quando o espaço disponível é insuficiente para armazenar todos os arquivos necessários de uma tarefa, arquivos previamente replicados para o sítio são excluídos. A figura 23 mostra o algoritmo de escalonamento de tarefas.

Algoritmo: *EscalonaTarefa*
Entrada: t_i - Tarefa a ser submetida
D - Espaço em disco necessário para processar t_i
$f_1(t_i) \rightarrow$ *Grupo associado a tarefa* t_i
$f_2(g_i) \rightarrow$ *Sítio associado ao grupo* g_i
$f_3(s_i) \rightarrow$ espaço disponível no sítio s_i

Saída: *Solução* - Identificação do Sítio escolhido para executar a tarefa.
Banco de Dados de Controle atualizado.
Passos:
1. $g_i \leftarrow f_1(t_i)$
2. *Solução* $\leftarrow f_2(g_i)$
3. **Se** $D > f_3$(*Solução*)
   então
   determina_arquivos_exclusão
   executa exclusão_arquivo
4. **Atualiza** espaço disponível no sítio *S no Banco de Dados de Controle*
5. **Retorne** *Solução*

Figura 23 - Algoritmo *EscalonaTarefa*

IV.5 - Experimentos

O ambiente de Grid foi modelado como um conjunto de sítios, cada um possuindo um número fixo de processadores e uma quantidade determinada de espaço em disco. Foi assumido que em cada sítio, todos os processadores possuem o mesmo desempenho e podem acessar todo o espaço em disco daquele sítio. Em uma máquina separada, externa ao Grid, o gerenciador do workflow controla o escalonamento das tarefas para os sítios e um serviço de localização de réplicas (RLS) mantém as informações sobre as réplicas espalhadas pelo Grid. O modelo é baseado no ambiente usado nos estudos de Ranganathan e Foster (2002, 2003).

Os sítios, tarefas e arquivos que compõem o workflow são modelados em três arquivos de entrada. Os resultados da simulação são salvos em um arquivo e informam para cada tarefa onde ela foi executada e a sua duração. Os resultados também informam o tempo total de execução do workflow, o número total de transferências de arquivos e a quantidade de espaço em disco para armazenamento dos dados.

No workflow, cada arquivo possui uma identificação única e é caracterizado pelo seu tamanho. Cada tarefa é caracterizada por um identificador único e pela lista de arquivos que ela processa. Para que uma tarefa possa ser executada, o conjunto de arquivos de entrada tem que estar disponível no sítio selecionado para a execução. Foi assumido (como na situação real) que todos os arquivos de entrada do workflow estão localizados em um único servidor de arquivos.

A transferência dos resultados das tarefas não foi considerada nos experimentos, visto que nesta aplicação, o custo desta transferência é insignificante se comparado com o tempo total de execução e não impõe nenhuma carga adicional no Grid. A figura 24 ilustra o ambiente de Grid modelado no simulador.

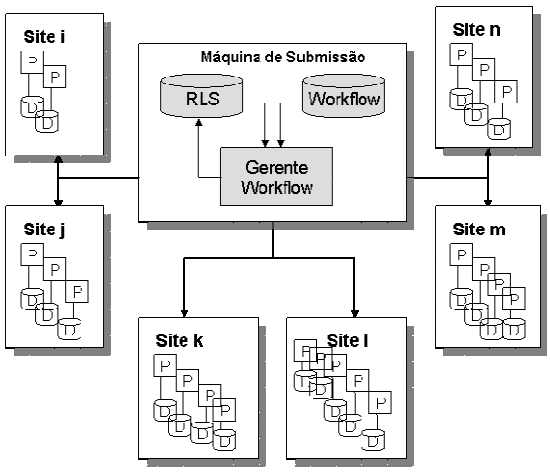


Figura 24 - Arquitetura do simulador de Grid

O comportamento deste Grid foi modelado como um simulador de eventos discreto. Alguns experimentos iniciais foram feitos no ChicaSim (RAGANATHAN, FOSTER, 2003), construído com o sistema de simulação Parsec (BAGRODIA et al.,

1998). Embora este ambiente de simulação forneça inúmeros recursos, o seu desempenho na execução do workflow *Coadd* foi muito lento. Assim, re-codificamos o modelo do simulador em Perl de forma a atender aos algoritmos e as necessidades de execução do workflow empregado pelos sistemas Condor/DagMan e VDS.

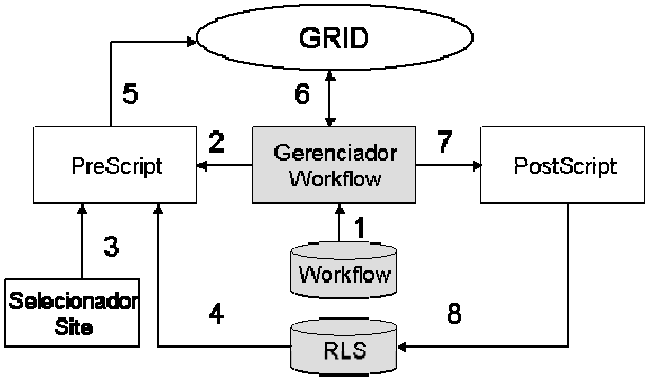


Figura 25 - Arquitetura do Gerenciador do Workflow

Da mesma forma que no sistema Condor/DagMan, procedimentos para o pré e pós-processamento estão associados a cada tarefa. O "*PreScript*" e o "*PostScript*" são responsáveis respectivamente, por transferirem os arquivos de entrada para o sítio de execução e excluírem estes arquivos após o processamento de uma tarefa. O número de "PreScript" executando simultaneamente na máquina de submissão é controlado pelo parâmetro MAXPRE que serve como um conveniente regulador para o número de transferências de arquivos que o gerente do workflow pode impor ao Grid. A figura 25 detalha a arquitetura empregada no simulador para o gerenciamento do workflow.

A execução de uma tarefa é simulada em oito passos:

1. O *Gerenciador do Workflow* recupera a identificação da tarefa e inicia a execução do pré-processamento da tarefa.
2. O *PreScript* recebe a identificação da tarefa, a lista dos arquivos de entrada e é responsável por duas decisões: escolher o sítio de execução da tarefa e transferir para este sítio os arquivos necessários.
3. Para decidir onde executar uma tarefa o *PreScript* invoca o *Selecionador de Sítio*. O simulador implementa um conjunto de algoritmos para o escalonamento de tarefas. Antes de iniciar a simulação, um dos algoritmos disponíveis deve ser informado.

4. Uma vez informado o sítio de execução, o *PreScript* consulta o serviço de localização de réplicas (RLS) para determinar quais arquivos de entrada precisam ser transferidos.
5. Se existir espaço disponível no sítio de execução, os arquivos são transferidos. Do contrário, existe a necessidade de se excluir arquivos deste sítio. O critério adotado pelo simulador é o de exclusão dos menos usados recentemente.
6. A tarefa é então escalonada para um sítio, espera na fila local por um processador livre e executa por um determinado período de tempo.
7. Após o processamento da tarefa, o *Gerenciador do Workflow* inicia a execução do pós-processamento.
8. O *PostScript* recebe a identificação da tarefa e a lista dos arquivos de saída produzidos e é responsável por registrar ou excluir estes arquivos no RLS, dependendo da política de replicação adotada.

Métricas como o tempo médio para transferência de dados e para submissão de tarefas foram retiradas de execuções no Grid3 e usadas para calibrar vários parâmetros de simulação. A taxa de transferência de dados assumida foi de 1MB/segundo o que corresponde a média do tempo de transferência entre dez sítios utilizando o GridFTP *default*.

Cada arquivo de entrada está inicialmente localizado em um sítio não utilizado para execução das tarefas e foi modelado com cinco megabytes de tamanho. Cada tarefa tem um tempo de execução de quinze minutos. Os experimentos foram executados em Grids com 5, 10, 15, 20 e 25 localidades. Os recursos disponíveis em cada local são mostrados na tabela 3. Os valores foram extraídos dos sítios do Grid3. A primeira coluna indica o número da localidade e a segunda e terceira colunas mostram o espaço em disco e o número de processadores existentes em cada localidade. Foi assumida na simulação uma alocação correspondente a 10% dos recursos disponíveis em cada localidade.

O *Gerente do Workflow* inicia o pré-processamento de cada tarefa com um intervalo de um segundo. Quando o número de instâncias de *PreScript* executando alcança o máximo especificado no parâmetro MAXPRE, nenhum novo *PreScript* pode

ser iniciado. De forma semelhante, quando uma tarefa é submetida, ela tem que esperar na fila caso não haja nenhum processador disponível no sítio. Foi assumido que o grupo de processadores em cada sítio permanece disponível e é de uso exclusivo do workflow durante a sua execução. O tempo total para executar uma tarefa é computado somando-se o tempo na fila na máquina de submissão, o tempo para transferir os arquivos de entrada, o tempo na fila remota e o tempo de processamento da tarefa.

Tabela 3 - Recursos disponíveis nas localidades

	Grid3		Simulador	
Local	Disco (GB)	Processador	Disco (GB)	Processador
1	958	720	95	72
2	865	516	86	51
3	200	400	20	40
4	2.170	350	217	35
5	1.627	312	162	31
6	1.130	296	113	29
7	647	272	65	27
8	1.365	158	139	16
9	591	136	59	13
10	976	100	97	10
11	1.712	82	171	8
12	2.172	80	217	8
13	879	80	88	8
14	1.190	76	119	8
15	1.723	64	172	7
16	569	60	57	6
17	1.190	55	119	6
18	243	42	24	4
19	1.712	42	172	4
20	109	40	11	4
21	53	40	6	4
22	976	32	98	3
23	1.311	20	131	2
24	55	19	5	2
25	1.294	12	130	2

Foram analisadas no experimento quatro estratégias para o processamento do workflow:

1. *Aleatória*. O local de execução é escolhido aleatoriamente. Os arquivos de entrada são transferidos para o local de execução mas não ficam disponíveis

para outras tarefas. Uma vez terminada a tarefa, os arquivos transferidos são excluídos.

2. *Agrupamento*. O local de execução é determinado pela identificação da tarefa. Cada tarefa está associada a um grupo como conseqüência do algoritmo *SPCL* e cada grupo é associado a uma localidade. O número de grupos associados a cada localidade é obtido pela divisão do número total de grupos pelo número de localidades disponível no experimento. Os arquivos de entrada são replicados para o local de execução.
3. *Aleatória_Replicação*. O local de execução é escolhido aleatoriamente. Os arquivos de entrada são replicados para o local de execução.
4. *Dado_Presente* - O local escolhido é aquele que possuir o maior número de arquivos de entrada para a tarefa. Caso nenhum local possua algum arquivo de entrada ou haja empate entre as localidades, a escolha é feita aleatoriamente.

As tarefas foram escalonadas aleatoriamente nas estratégias 1, 3 e 4. Na estratégia 2, as tarefas pertencentes a grupos diferentes não foram escalonadas para execução no mesmo local simultaneamente. No capítulo VI são apresentados e analisados os resultados obtidos na simulação da execução do workflow.

V - PLANEJAMENTO DE WORKFLOWS COM UMA ESTRATÉGIA OPORTUNISTA

V.1 - Introdução

Um ambiente de Grid pode ser composto por dezenas de localidades autônomas agregando milhares de processadores e um volume enorme de espaço de armazenamento. Tais ambientes podem ser extremamente dinâmicos fazendo com que recursos fiquem repentinamente sobrecarregados de trabalho ou até mesmo indisponíveis, enquanto outros de certa forma ociosos. Consequentemente, neste tipo de ambiente, fazer o planejamento das tarefas de um workflow *a piori* pode produzir um mau escalonamento porque no momento em que uma determinada tarefa estiver pronta para executar, o recurso associado previamente pode estar indisponível ou sobrecarregado.

Uma outra abordagem para o planejamento é adiar a decisão do escalonamento, fazendo a associação aos recursos dinamicamente no momento da execução da tarefa. Assim, caso um recurso não esteja disponível, ele não será selecionado para processar a tarefa. No entanto, uma vez que muitos recursos podem estar disponíveis, algum critério para a seleção deve ser empregado pelo sistema de planejamento de workflow.

O objetivo deste capítulo é descrever a implementação de uma estratégia oportunista para a execução de workflows em Grid. O restante do capítulo está dividido da seguinte forma: na seção V.2 é explicado o comportamento dos componentes da estratégia bem como os algoritmos implementados. Em seguida, a seção V.3 descreve os experimentos realizados e por fim a seção V.4 mostra o ambiente computacional de Grid utilizado nos experimentos.

V.2 - Estratégia Oportunista

A estratégia oportunista para a execução de workflows apresenta dois componentes: um algoritmo para escalonamento de tarefas (Selecionador de Sítio) e um gerenciador ou monitor de tarefas submetidas. A figura 26 ilustra os componentes da estratégia e suas interações com os módulos de planejamento e execução.

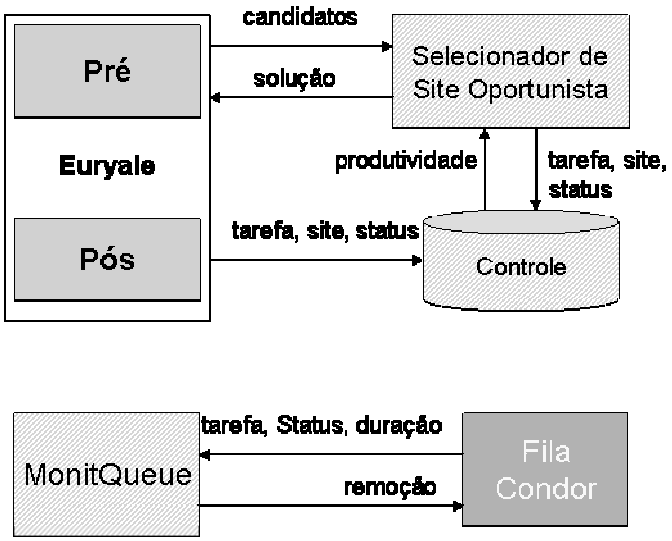


Figura 26 - Componentes da arquitetura oportunista.

V.2.1 - Algoritmo Oportunista

O algoritmo *Oportunista* de escalonamento adota uma abordagem de observação do Grid e explora a idéia de escalonar uma tarefa para um sítio onde provavelmente será mais rápida a sua execução. A principal vantagem do algoritmo é tirar proveito das mudanças de desempenho e disponibilidade dos recursos no Grid durante a execução do workflow, sem a necessidade de investigar a disponibilidade dos recursos remotos. O algoritmo é auto-ajustável, ou seja, sítios que apresentam um bom desempenho na execução recebem mais tarefas para processar enquanto que aqueles cujo desempenho é fraco recebem uma carga de trabalho menor.

Como já mencionado no capítulo III, algumas extensões foram promovidas no sistema Euryale para implementar e avaliar o algoritmo oportunista. O banco de dados

de controle tem a função de guardar as informações sobre as tarefas submetidas e o desempenho dos sítios. Como o banco de controle é atualizado antes e após a execução de cada tarefa, as rotinas de pré e pós-processamento do Euryale também foram modificadas para executarem estas atualizações. A figura 27 mostra uma visão das tabelas do banco de controle utilizadas pela estratégia.

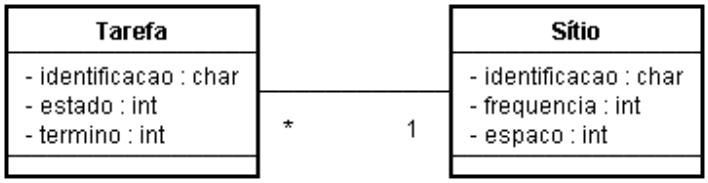


Figura 27 - Visão do esquema conceitual do banco de controle utilizado pela estratégia Oportunista

A tabela **Sítio** armazena a identificação e o número de tarefas associadas a cada localidade onde será executado o workflow. Ela é carregada previamente com as identificações existentes no catálogo de sítios do Euryale. O número de tarefas escalonadas para cada sítio (freqüência) é usado pelo algoritmo para fazer o balanceamento de carga entre os sítios enquanto nenhuma tarefa foi concluída. A tabela **Tarefa** é inicializada vazia e a medida em que o workflow é processado, um registro é inserido toda vez que uma tarefa é submetida ou concluída.

O objetivo do selecionador de sítio é escolher o local de execução de uma tarefa baseado no desempenho de cada sítio durante a execução de tarefas prévias do mesmo workflow. O desempenho é calculado computando-se a razão do número de tarefas submetidas pelo número de tarefas concluídas em cada sítio. Enquanto nenhuma tarefa tiver sido concluída, o algoritmo emprega um critério circular (Round-Robin) para a escolha. Caso uma tarefa esteja sendo replanejada em função da ocorrência de alguma falha ou por ação do monitor de fila, o algoritmo garante que o sítio previamente escolhido não será selecionado novamente. A figura 28 apresenta o algoritmo *oportunista* para seleção de sítios.

Algoritmo: *Oportunista*
Entrada: Tarefa T_i a ser submetida.
Conjunto S_e $\{s_i\}$ de Sítios disponíveis informados pelo Euryale.
Conjunto S_o $\{s_i\}$ de Sítios informados no *Banco de Dados de Controle*
$f_1(S_i) \rightarrow$ Número de tarefas submetidas para o sítio S_i
$f_2(S_i) \rightarrow$ Número de tarefas concluídas para o sítio S_i
$f_3(max) \rightarrow$ Sítio S_i
$f_4(min) \rightarrow$ Sítio S_i
$f_5 \rightarrow \{t_i\}$; Conjunto de tarefas submetidas

Saída: *Solução* - Identificação do sítio escolhido para executar a tarefa.
Banco de Dados de Controle atualizado.
Passos:
1. $flag \leftarrow 0$
2. $min \leftarrow high\ value$
3. $max \leftarrow low\ value$
4. **Para cada** sítio $s_i \in S_o$ **faça**
   4.1 **Se** $s_i \in S_e$ **então**
      4.1.2 $T_s \leftarrow f_1(s_i)$
      4.1.3 **Se** $T_s < min$ **então**
         4.1.3.1 $min \leftarrow T_s$
      4.1.3 $T_c \leftarrow f_2(s_i)$
   4.2 **Se** $T_c > 0$ **então**
      4.2.1 $R_i \leftarrow (T_c / T_s)$
      4.2.2 $flag \leftarrow 1$
      4.2.3 **Se** $R_i > max$ **então**
         4.2.3.1 $max \leftarrow R_i$
5. **Se** $flag = 1$ **então**
   3.1 $Solução \leftarrow f_3(max)$
   senão
   5.2 $Solução \leftarrow f_4(min)$
6. $T_o \leftarrow f_5$
7. **Se** $T_i \in T_o$ **então**
   7.1 **atualiza** o valor do sítio para a tarefa T_i no *banco de dados de controle*
   senão
   7.2 **inclui** tupla (T_i, *solução*) no *banco de dados de controle*.
8. **Retorne** *Solução*

Figura 28 - Algoritmo *oportunista* para seleção de sítios

V.2.2 - Monitor de Fila

O segundo componente da arquitetura é um monitor da fila de submissão das tarefas do workflow. Euryale utiliza o sistema Condor-DagMan para gerenciar a execução do workflow. DagMan gerencia a ordem de execução das tarefas do workflow e submete cada tarefa para o Condor. O sistema Condor mantém na mesma máquina em que o DagMan está executando, uma fila das tarefas submetidas informando o estado corrente, a duração e a localização de cada tarefa. Condor também disponibiliza um conjunto de comandos para monitorar o estado das tarefas.

O objetivo do componente *MonitQueue* é monitorar a fila de tarefas do Condor e eliminar aquelas tarefas do workflow que não estejam apresentando o desempenho esperado pelo usuário. Quando uma tarefa é eliminada, ela é automaticamente re-planejada pelo Euryale. Portanto, o pré-processamento da tarefa é novamente realizado e consequentemente o selecionador de sítio é chamado para definir um novo sítio de execução. Na implementação atual, o *MonitQueue* é iniciado pelo usuário que deve informar os tempos limites que uma tarefa pode permanecer na fila em estado inativo (*idle*) ou executando (*Running*). Entretanto, extensões podem ser promovidas na arquitetura para que o banco de controle armazene as informações necessárias e permita que o *MonitQueue* automaticamente defina os seus parâmetros de execução.

A motivação para implementar o componente *MonitQueue* se deve ao fato de tarefas submetidas ficarem freqüentemente um longo período de tempo na fila de um sítio esperando para serem executadas. O comportamento *default* do sistema Euryale é somente cancelar uma tarefa caso esta fique em espera por mais de duas horas. Este tempo de espera pode ser muito alto para workflows cujas tarefas tenham um tempo de processamento relativamente curto, acarretando uma baixa performance de execução.

Uma boa estratégia de execução deve ser capaz de reconhecer os sítios que não estão conseguindo processar as tarefas escalonadas e não mais utilizá-los, pelo menos durante um período de tempo. A figura 29 apresenta o algoritmo *MonitQueue* para o monitoramento das tarefas submetidas.

Algoritmo: *MonitQueue*

Entrada: Tempo limite T_q para uma tarefa na fila.
Tempo limite T_r para uma tarefa executando.
$f(Q) \rightarrow \{t_i\}$, tal que $t_i = (id, status, duração)$

Passos:

1. **Enquanto** 1 **faça**
   1.1 $Q \leftarrow f(Q)$
   1.2 **Para cada** $t_i \in Q$ **faça**
      1.2.1 **Se** $status_i$ = "I" e $duração_i > T_q$
         então $Q \leftarrow Q - \{t_i\}$
      1.2.2 **Se** $status_i$ = "R" e $duração_i > T_r$
         então $Q \leftarrow Q - \{t_i\}$

Figura 29 - Algoritmo *MonitQueue* para monitoramento de tarefas submetidas

V. 3 - Experimentos

A estratégia oportunista teve o seu desempenho avaliado na execução de três workflows diferentes. Estes workflows utilizam os padrões básicos para a modelagem de componentes que o VDS implementa e que o sistema DagMan é capaz de gerenciar. São eles: seqüencial, divisão paralela e sincronização. As principais características destes workflows são descritas a seguir.

V.3.1 - Workflow 1 – Sequenciamento de Tarefas

Muitas aplicações científicas podem ser caracterizadas por possuírem conjuntos de dados de entrada e dados derivados que necessitam ser processados em vários estágios por um conjunto de programas. Estes "*batch-pipelined*" (THAIN et al.,2003) workflows são compostos de várias e independentes seqüências de programas ("*pipeline"*) onde cada seqüência contém programas que comunicam com seus antecessores e sucessores através de arquivos de dados. O **workflow 1** implementa esta característica. A figura 30 mostra a estrutura dos seus componentes e a figura 31 ilustra o DAG correspondente gerado para os experimentos.

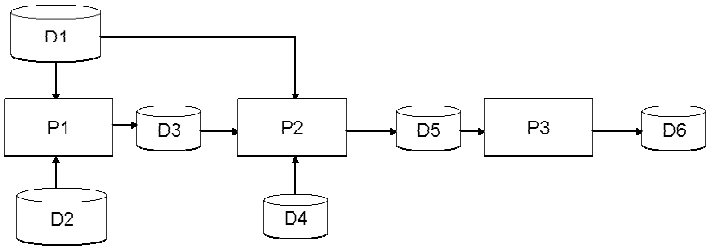


Figura 30 - Componentes do Workflow 1

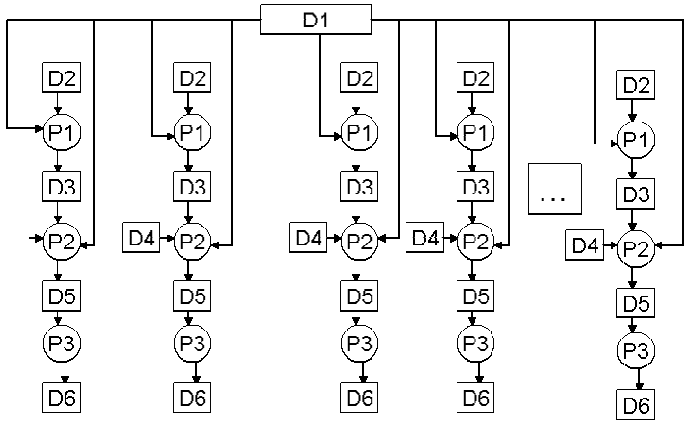


Figura 31 - DAG gerado para execução do workflow 1

O conjunto de dados D1 possui um único arquivo que serve de entrada para o primeiro e segundo programas. O programa P1 também tem como entrada um arquivo do conjunto de dados D2. O programa P2 processa um arquivo D3 que é produzido por P1 e tem como entrada mais 2 arquivos para o processamento: O arquivo único do conjunto de dados D1 e um arquivo do conjunto de dados D4. O terceiro e último programa da seqüência, processa o arquivo de saída produzido por P2 e gera um arquivo no conjunto de dados D6.

O **workflow 1** foi definido com cem tarefas em cada nível. As primeiras cem tarefas (primeiro nível), têm a duração individual de cinco minutos, enquanto que as outras duzentas tarefas têm a duração de um minuto. O tamanho dos arquivos de entrada e saída de todo o workflow é de um megabyte.

V.3.2 - Workflow 2 – Tarefas Independentes

Um segundo modelo de aplicação científica muito comum é o de *bag-of-task*. Muitos programas de bioinformática, como por exemplo, o Blast e o FastA, podem ser processados em Grid com este modelo. O segundo workflow implementa a execução paralela do programa Blast. A figura 32 mostra a estrutura dos seus componentes e a figura 33 ilustra o DAG correspondente gerado para os experimentos.

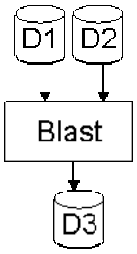


Figura 32 - Componentes do Workflow 2

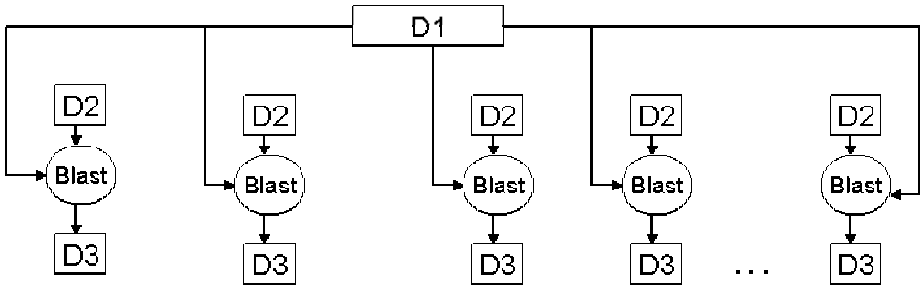


Figura 33 - DAG gerado para execução do workflow 2

O Blast é constituído por uma família de programas que tem por objetivo achar regiões de similaridade entre seqüências. O Blast compara seqüências de nucleotídeos ou proteínas com um banco de dados de seqüências e calcula a significância estatística dos acertos. Nos experimentos realizados, foi utilizado o programa *Blastp* para comparar as seqüências de proteínas do banco de dados *pdbaa*. O banco de dados *pdbaa* utilizado continha 23.913 seqüências e foi fragmentado em 1.200 arquivos de entrada, cada um com aproximadamente 20 seqüências. O **workflow 2** foi definido com 1.200 tarefas independentes que processaram um arquivo de entrada contra o banco *pdbaa* que foi replicado previamente em todos os sítios utilizados no experimento.

V.3.3 - Workflow 3 – Divisão Paralela e Sincronização

O terceiro workflow utilizado combina as características de processamento independente e sequenciamento de tarefas investigadas nos dois workflows anteriores e acrescenta mais dois padrões de modelagem: divisão paralela e sincronização. A figura 34 mostra a estrutura dos seus componentes e a figura 35 ilustra o DAG correspondente gerado para os experimentos.

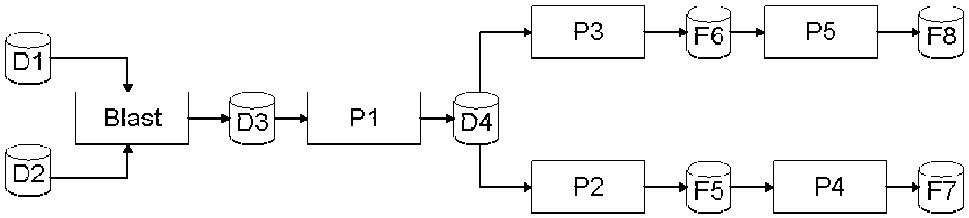


Figura 34 - Componentes do Workflow 3

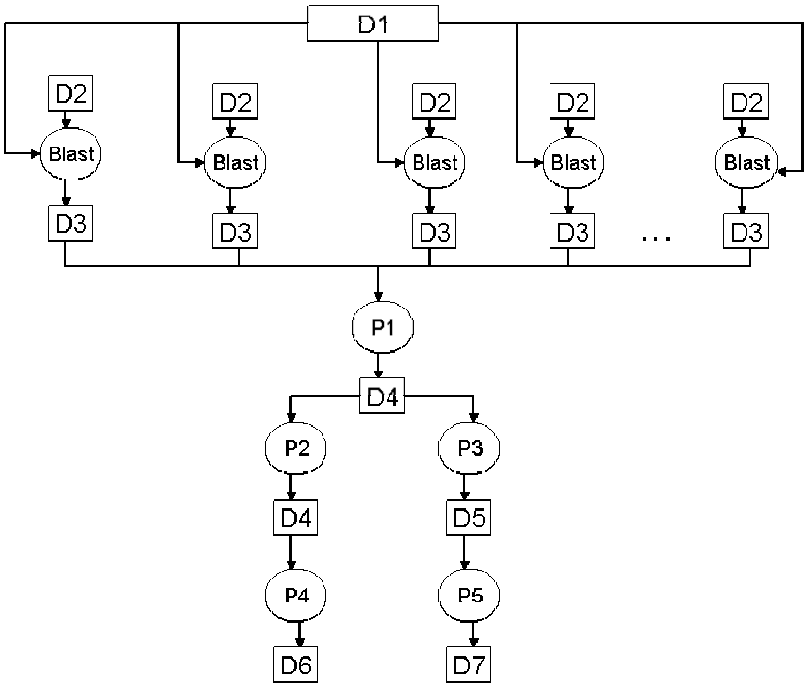


Figura 35 - DAG gerado para execução do workflow 3

O primeiro nível do workflow executa o programa Blast nos mesmos moldes da execução do **workflow 2**. Entretanto no **workflow 3**, O banco de dados *pdbaa* foi fragmentado em 90 arquivos de entrada, cada um com aproximadamente 2.660

seqüências. Estes 90 arquivos de entrada foram processados por 90 tarefas contra o *pdbaa* previamente replicado em todos os sítios utilizados no experimento. O programa P1 tem como entrada os noventa arquivos produzidos pelo Blast e gera um arquivo único D4 obtido pela cópia dos arquivos de entrada. Este arquivo D4 é processado pelos programas P2 e P3 que geram como saída os arquivos D4 e D5 respectivamente. No último estágio do workflow, os arquivos D4 e D5 são processados pelos programas P4 e P5 respectivamente gerando o resultado final do workflow.

V.4 - Ambiente Computacional

A avaliação de desempenho da estratégia oportunista foi realizada no *Open Sciences Grid* - OSG (OPEN SCIENCES GRID CONSORTIUM ,2005). O OSG é uma infra-estrutura de computação em Grid para apoiar a computação científica através da colaboração de pesquisadores, desenvolvedores de sistema, e provedores de recursos para processamento e armazenamento de dados. O consórcio OSG foi formado em 2004 como uma continuação do projeto Grid3. O consórcio constrói e opera este ambiente agregando recursos e pesquisadores de universidades e laboratórios americanos e internacionais.

Os sistemas em desenvolvimento nos Estados Unidos para os experimentos realizados no *Large Hadron Collider* (LHC) no CERN, Suíça, estão sendo construídos e operados pelo OSG. Entretanto, outros projetos nas áreas de física, astrofísica, ciência das ondas gravitacionais e biologia também contribuem e usufruem deste Grid.

O OSG inclui dois ambientes: integração e produção. As novas tecnologias e aplicações são testadas no Grid de integração enquanto que o Grid de produção fornece aos usuários um ambiente com suporte e mais estável. A figura 36 ilustra as instituições participantes do *Open Sciences Grid* na América do Norte, América do Sul e Ásia.

Os recursos e serviços de cada sítio são registrados e têm o seu status exibido através do catálogo GridCat de acesso público. De acordo com o GridCat, o número total de sítios no OSG é de 59, disponibilizando um total de 19.838 processadores e

aproximadamente 69 Terabytes para armazenamento. Nos experimentos realizados, foi utilizado um subconjunto formado por doze localidades. A figura 37 ilustra o número total de processadores existentes em cada um dos locais usados.

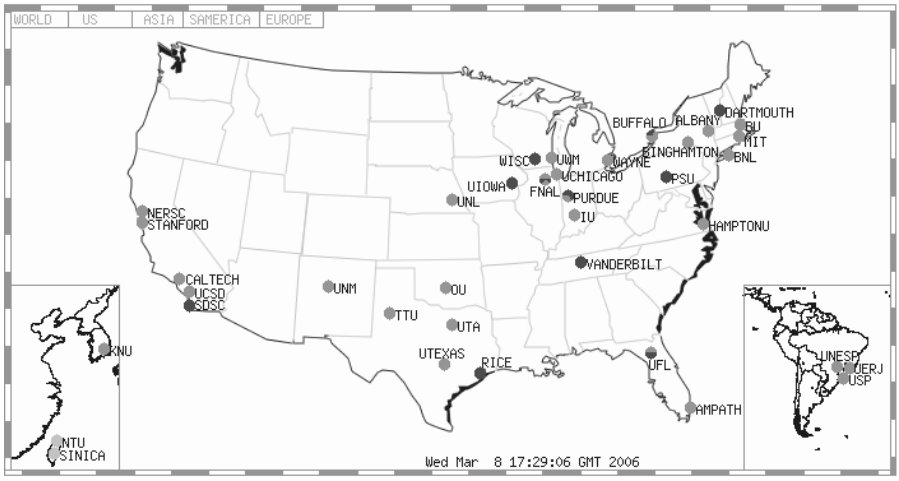


Figura 36 - Instituições participantes do OSG (http://osg-cat.grid.iu.edu)

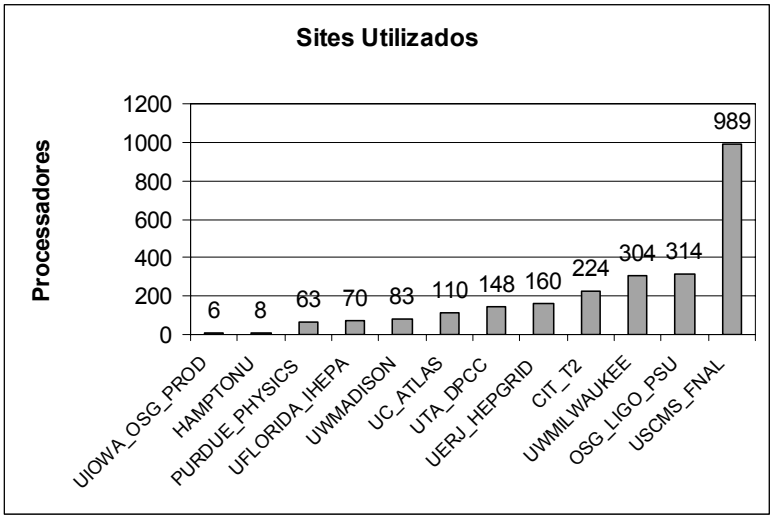


Figura 37 - Processadores existentes por localidade utilizada nos experimentos

Os usuários do OSG são membros de Organizações Virtuais (VO). Cada Organização Virtual inclui pelo menos uma organização membro do consórcio. Cada sítio pode especificar uma política de uso de seus recursos de acordo com a VO. Por exemplo, o sítio da Universidade de Wisconsin em Milwaukee, disponibiliza 100% dos seus recursos para a VO iVDGL. Já o sítio UTA_DPCC no Texas, 80% dos recursos podem ser usados pela VO *atlas* e os 20% restantes pela VO *dosar*.

Durante a execução dos experimentos, foram utilizadas duas máquinas localizadas na Universidade de Chicago. Na primeira máquina (*terminable*) foram instalados todos os catálogos do sistema VDS e ela foi usada para a submissão dos workflows. A outra máquina (*ept*) foi usada para armazenar os arquivos de entrada para todos os experimentos.

V.4.1 - Algoritmos analisados

O sistema de planejamento *Euryale* disponibiliza estratégias para seleção de sítios de forma *aleatória* e *circular*. Para melhor avaliar o algoritmo *Oportunista*, mais duas estratégias foram implementadas nesta tese: *Ultimo_Sítio_Usado* e *Dado_Presente.*

A idéia geral destes algoritmos é:

1. *Aleatório_Ponderado* - O local de execução é escolhido da lista formada pelos sítios existentes no catálogo de sítios. A escolha é ponderada de acordo com o número de processadores informados no catálogo.

2. *Circular* - A escolha do sítio de execução é feita de forma circular.

3. *Ultimo_Sítio_Usado*- O sítio para a execução escolhido é aquele que finalizou por último uma tarefa. Enquanto nenhuma tarefa tiver terminado, a escolha do sítio é feita de forma circular.

4. *Dado_Presente* - O sítio escolhido é aquele que possuir o maior número de arquivos de entrada para a tarefa. Caso nenhum sítio possua algum arquivo de entrada ou haja empate entre sítios, a escolha é feita de forma aleatória.

5. *Oportunista* - O sítio é selecionado de acordo com o desempenho dos sítios nas tarefas do workflow. O desempenho é medido dividindo-se o número de tarefas

concluídas pelo número de tarefas submetidas em cada sítio. Enquanto nenhuma tarefa houver terminado, a escolha do sítio é feita de forma circular.

No próximo capítulo são apresentados e analisados os resultados experimentais obtidos na tese.

VI - ANÁLISE DE DESEMPENHO

VI. 1 - Introdução

O objetivo deste capítulo é analisar o desempenho da estratégia por agrupamento de tarefas para o processamento de workflows espaciais, e da estratégia oportunista para escalonamento de tarefas. Na seção VI.2, é feita a análise do desempenho do workflow SDSS/Coadd no ambiente de simulação. Duas métricas foram analisadas nos experimentos: o número total de transferências de arquivos e o tempo total de execução do workflow. O número de transferências de arquivos é importante pelo seu impacto na utilização dos recursos do ambiente como um todo e também na performance dos algoritmos. O tempo total de execução é importante do ponto de vista dos usuários que querem executar as suas aplicações em um menor período de tempo possível.

As seções VI.3, VI.4 e VI.5 analisam o desempenho dos workflows 1, 2 e 3 em um ambiente de Grid real. Além do tempo total de execução do workflow, também é mostrado o ganho obtido com cada algoritmo. O ganho é calculado como a razão entre o tempo de execução com 1 processador e o tempo de execução no Grid.

VI. 2 - SDSS/Coadd

O ambiente de simulação foi modelado como um subconjunto de vinte e cinco sítios do Grid3. Cada simulação com o tamanho do Grid igual a N (N=5, 10, 15, 20, 25) consistiu dos primeiros N sítios listados na tabela 3. Os resultados aqui apresentados são baseados em sessenta execuções do simulador representando a combinação de cinco tamanhos de Grid, quatro estratégias de escalonamento e três níveis diferentes (MAXPRE=20, 40, 60) de pré-processamento concorrente.

Nós começamos analisando o impacto das quatro estratégias de execução no número total de transferências de arquivos. Como pode ser observado na figura 38, executar o workflow SDSS/Coadd de acordo com as estratégias *aleatória* e *aleatória_replicação* é extremamente caro em termos de movimentação de dados requerendo quase 5.5 milhões de transferências. O reuso de arquivos através da replicação empregada na estratégia *aleatória_replicação* resulta em uma redução pequena no número de transferências. A estratégia por *agrupamento* reduz para aproximadamente 555.000 o número de transferências quando o tamanho do Grid é de cinco ou dez sítios. A associação dos grupos de tarefas aos sítios não levou em consideração os recursos disponíveis em cada sítio. Portanto quando executamos a simulação com 15, 20 e 25 sítios no Grid, sítios com pouca disponibilidade de espaço em disco foram utilizados acarretando um aumento de exclusões de arquivos e por conseqüência, um aumento no número de transferências.

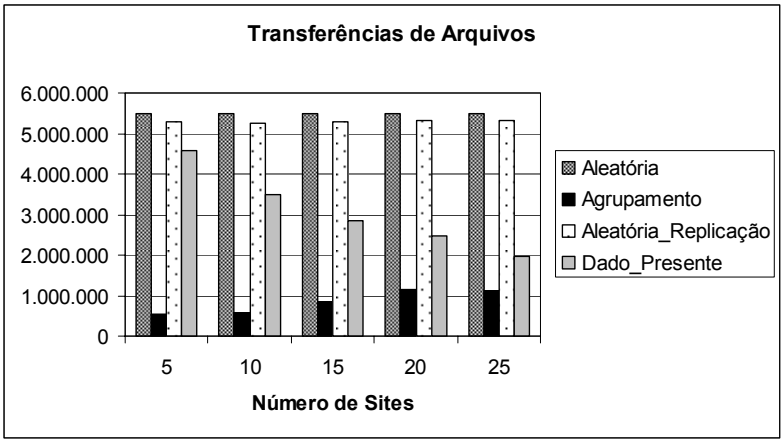


Figura 38 - Número de transferências de arquivos por estratégia de execução e tamanho do Grid

A estratégia *dado_presente* também utiliza a replicação dos arquivos de entrada e como o escalonamento de tarefas leva em consideração o número de réplicas disponível para seleção do sítio, esta estratégia consegue reduzir o número de transferências. Na medida em que mais espaço é acrescido, mais arquivos podem permanecer disponíveis fazendo com que o escalonamento das tarefas tire proveito destes dados reduzindo o número de transferências.

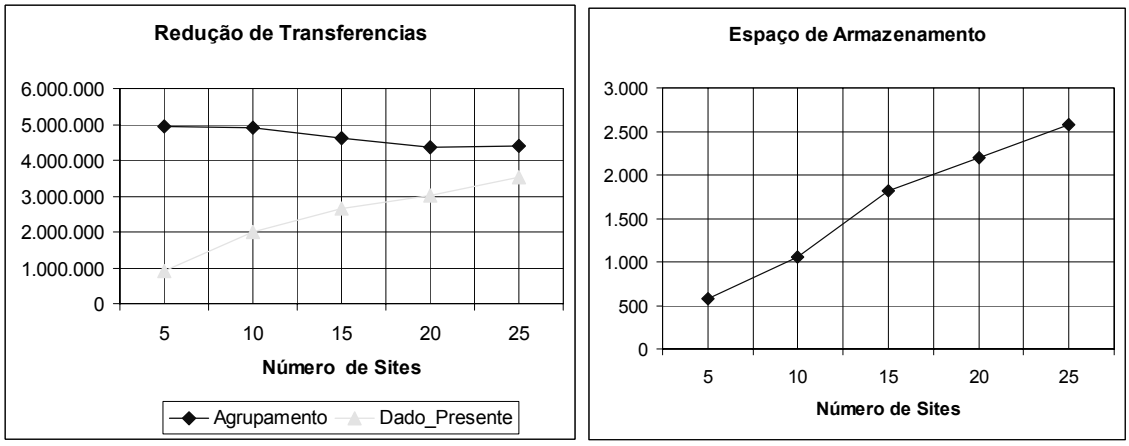


Figura 39 - Redução de transferências por **e**stratégias de execução e espaço de armazenamento no Grid

A figura 39 mostra o desempenho comparativo entre as abordagens *agrupamento* e *dado_presente* em relação à redução de transferências obtidas. Esta redução é computada pela subtração do número total de transferências (5.440.000) pelo número de transferências realizado por cada uma das estratégias. Como também pode ser observado na figura, à medida que o número de sítios no Grid aumenta, o espaço de armazenamento como um todo também aumenta e a estratégia *dado_presente* tira proveito da disponibilidade dos dados reduzindo o número de transferências em aproximadamente 3.500.000 na configuração do Grid com 25 sítios. O acréscimo de sítios não traz benefícios à estratégia por *agrupamento*, porque sítios com menor capacidade de armazenamento receberam o mesmo número de grupos de tarefas que os sítios com mais capacidade de armazenamento. Conseqüentemente estes sítios menores tiveram que executar mais exclusões de arquivos.

As figuras 40, 41 e 42 mostram o desempenho das quatro estratégias com a configuração de número máximo de *scripts* de pré-processamento igual a vinte, quarenta e sessenta, respectivamente. As figuras mostram resultados similares para a estratégia *agrupamento* com 20, 40 e 60 pré-scripts executando simultaneamente.

O agrupamento de tarefas permite o reuso de arquivos, reduz o número de transferências destes arquivos e conseqüentemente o tempo de execução do pré-processamento. Transferir menos arquivos para uma tarefa pode trazer benefícios ao desempenho se no momento em que estas transferências terminam, existe processador disponível para a execução da tarefa. Aumentando o número de pré-scripts causa um

número maior de tarefas aptas a serem escalonadas e executadas, mas tendo que esperar por processadores livres porque o tempo de execução da tarefa é maior que o tempo para transferir poucos arquivos.

Outro ponto a ser observado, é o decréscimo no desempenho quando o número de sítios no Grid aumenta. Como já mencionado anteriormente, o acréscimo de sítios não foi balanceado com o número de grupos associados a cada sítio. Desta forma, sítios com poucos recursos receberam carga de trabalho semelhante aos sítios com mais recursos.

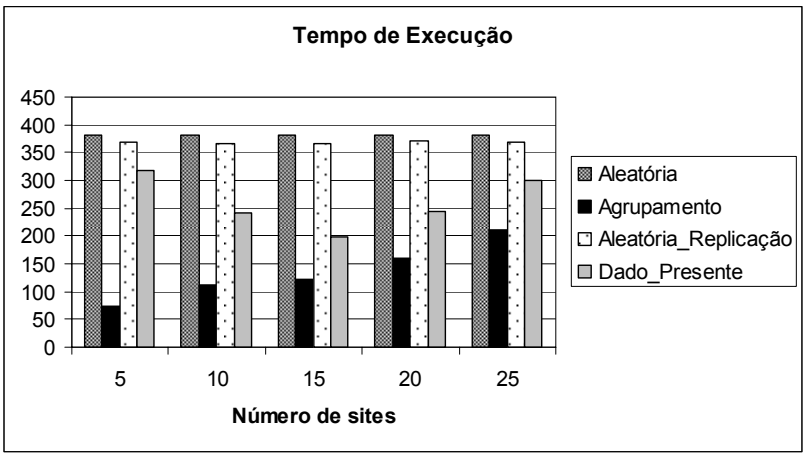


Figura 40 - Desempenho com MAXPRE=20

O aumento no número de pré-script melhora o desempenho das estratégias *aleatória* e *aleatória_replicação*. Para estas duas estratégias, o custo de transferir mais arquivos é refletido na performance quando o número de pré-script é 20. Quando mais arquivos são transferidos em paralelo, o tempo de execução das duas estratégias melhora. Entretanto, ter um número grande de pré-script executando em uma única máquina pode não ser viável se estes pré-processamentos realizarem a transferência de um conjunto grande de arquivos. O número simultâneo de requisições ao servidor de arquivos pode causar problemas a este serviço. O aumento no tamanho do Grid não acarretou em melhoria na performance nestas duas estratégias. Ao contrário, o desempenho é pior com o número de pré-script igual a 40 e 60 quando o Grid é composto por 25 sítios do que quando o tamanho é de 5, 10, 15 e 20 sítios.

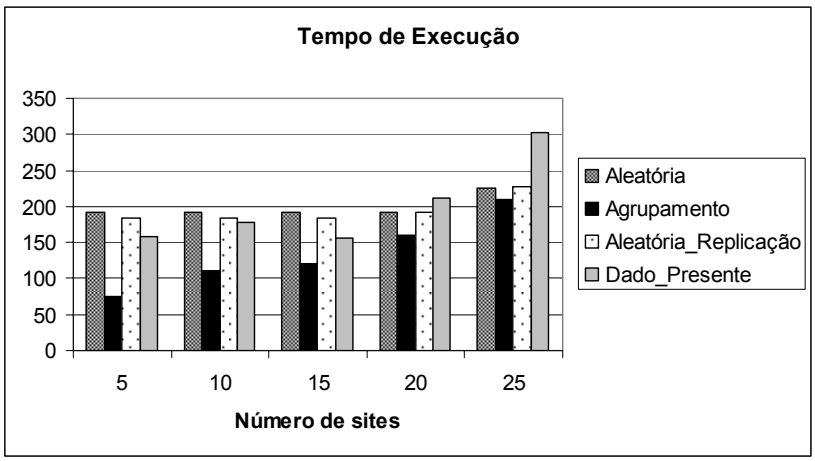


Figura 41 - Desempenho com MAXPRE=40

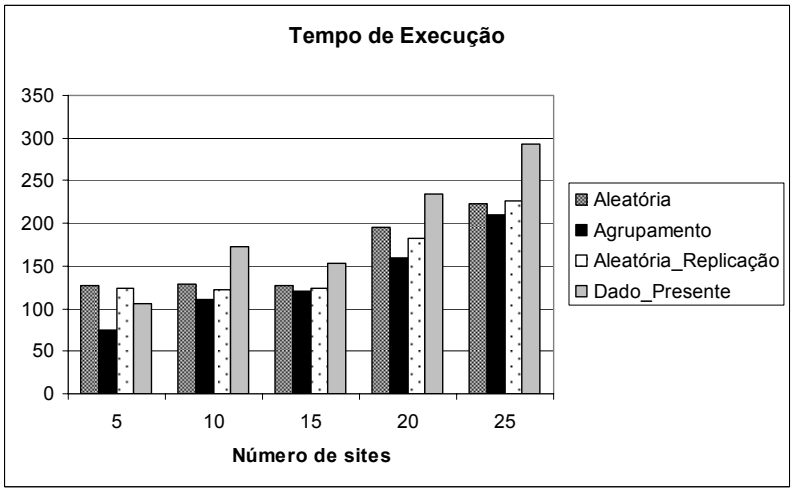


Figura 42 - Desempenho com MAXPRE=60

Com MAXPRE igual 20, a estratégia *dado_presente* apresenta benefícios até que o número de sítios atinge a 15. Depois deste ponto, a inclusão de novos sítios no Grid não trouxe benefícios com nenhum número de pré-scripts. O aumento no número de pré-script para 60 só trouxe melhoria quando o número de sítios era 5. Para todas as estratégias, mostram que a inclusão de sítios com poucos recursos não traz nenhum benefício ao desempenho do workflow se o algoritmo de escalonamento de tarefas não considera os recursos disponíveis em cada sítio.

Nas próximas seções são analisados os resultados dos experimentos realizados para avaliação da estratégia Oportunista. Nos gráficos apresentados, utilizamos a seguinte legenda para designar as diferentes estratégias: *oportunista* (OP),

aleatório_ponderado (RP) *ultimo_sítio_usado* (US) e *circular* (CC) e *dado_presente* (DP)

VI. 3 - Workflow1 – Sequenciamento de Tarefas

A tabela 4 mostra os tempos médios de execução e de transferência de arquivos por tipo de tarefa do ***workflow1.*** A figura 43 mostra o tempo médio por tarefa do workflow. Os resultados aqui apresentados são baseados em vinte execuções realizadas para cada estratégia, totalizando 30.000 execuções no OSG. Os parâmetros utilizados para o Condor/DaGMan foram: MAXPRE=20 e MAXPOST=20.

Tabela 4 - Tempo médio em segundos para transferência de dados e execução por tipo de tarefa.

	Tarefas		
	T1	T2	T3
Quantidade	100	100	100
Transferência de dados	17	27	13
Execução	300	120	60

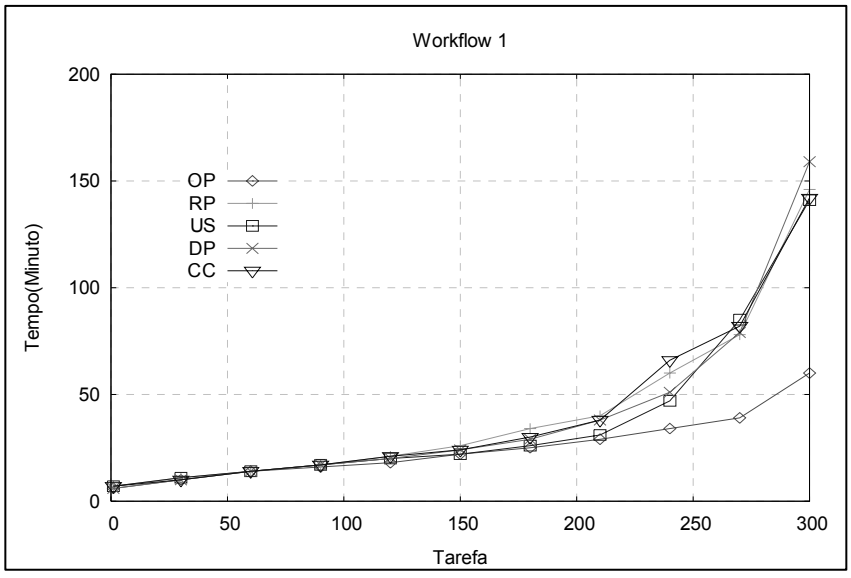


Figura 43 - Tempo médio de cada tarefa do workflow1 de acordo com a abordagem de escalonamento

Como pode ser observado na figura 43, o desempenho das cinco estratégias é similar durante a execução das cem primeiras tarefas. As abordagens *oportunista*,

ultimo_sítio_usado e *circular* adotam a mesma estratégia de escalonamento enquanto nenhuma tarefa for concluída. A estratégia *dado_presente* adota estratégia similar a *aleatória_ponderada* se nenhuma localidade possuir os arquivos de entrada para a tarefa. Como não existem dependências entre as tarefas neste primeiro nível do workflow, Condor/DagMan pode submetê-las para o Grid assim que termina o pré-processamento de cada uma delas. Já que o tempo de transferência dos arquivos de entrada é muito pequeno, o pré-processamento de cada tarefa é realizado rapidamente, fazendo com que a maioria das tarefas neste primeiro nível do workflow seja escalonada antes que alguma tarefa conclua o seu processamento.

A medida em que as tarefas do primeiro e segundo níveis do workflow vão terminando, as abordagens *oportunista*, *ultimo_sítio_usado* e *dado_presente* passam a escalonar as tarefas de maneira distinta. As abordagens *oportunista* e *ultimo_sítio_usado* começam a tirar proveito do caráter observador de suas estratégias. No primeiro caso, o escalonamento das tarefas passa a ser feito de acordo com a taxa (tarefas concluídas/tarefas submetidas) enquanto que a estratégia *ultimo_sítio_usado* passa a escalonar as tarefas para a localidade do Grid que terminou de executar a última tarefa. Estas duas abordagens passam a oferecer um melhor desempenho que as demais. Entretanto o desempenho alcançado pela estratégia *oportunista* supera em aproximadamente 150% a estratégia *ultimo_sítio_usado* a partir das últimas setenta e cinco tarefas.

A estratégia *oportunista* se beneficia dos aspectos dinâmicos do Grid. Se uma localidade começa a oferecer um desempenho ruim, o número de tarefas escalonadas para esta localidade diminui. Da mesma forma, se uma localidade apresenta um bom desempenho, mais tarefas são escalonadas para serem lá executadas. A estratégia *ultimo_sítio_usado* pode não apresentar um bom desempenho quando as tarefas são escalonadas para uma localidade e esperam um longo período na fila remota antes de executarem. Quando isto acontece, a próxima tarefa a ser escalonada provavelmente apresentará a mesma performance da anterior já que será escalonada para a mesma localidade. Este tipo de problema não ocorre com a estratégia oportunista porque uma tarefa é cancelada e re-planejada pelo componente *MonitQueue* se passado o período especificado, ela não começar a sua execução.

O escalonamento circular fornece um bom balanceamento de carga entre as localidades do Grid, mas como o desempenho de cada localidade pode variar, escalonar o mesmo número de tarefas para cada localidade pode não trazer benefícios no desempenho. A estratégia *aleatória_ponderada* também não apresenta um bom desempenho porque escalonar mais tarefas para as localidades com mais recursos existentes não garante necessariamente uma melhor performance já que as tarefas podem ter que esperar nas filas remotas. Este tipo de estratégia parece ser mais indicado para o caso de uso de ambientes de Grid onde é possível fazer a reserva de recursos.

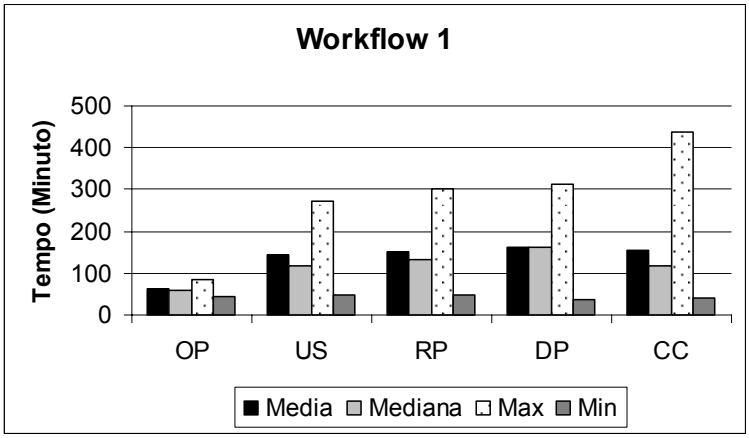


Figura 44 - Estatísticas básicas por estratégia de execução para o workflow1

As figuras 44 e 45 mostram respectivamente algumas estatísticas básicas e o ganho obtido por estratégia de execução. Como pode ser observado, o valor mínimo obtido na execução do workflow é praticamente o mesmo em todas as abordagens. Isto ocorre porque eventualmente, todas as localidades utilizadas para o processamento do workflow podem estar apresentando um bom desempenho em função de estarem no momento da execução com recursos disponíveis para o processamento. Neste caso, pouco importa a estratégia de escalonamento utilizada. Porém como na maioria das vezes o desempenho das localidades difere, os demais valores para os tempos médio, máximo e mediana também diferem de acordo com a estratégia.

A execução do ***workflow1*** no Grid alcançou um ganho de 25 vezes em relação à execução em uma única localidade. Este ganho foi cerca de 150% superior às demais abordagens.

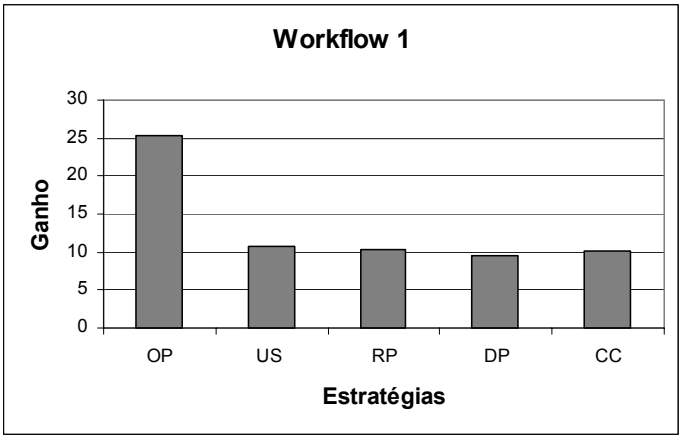


Figura 45 - Ganho por estratégia de execução

VI. 4 - Workflow2 – Tarefas Independentes

A tabela 5 mostra os tempos médios de execução e de transferência de arquivos por tipo de tarefa do ***workflow2.*** A figura 46 mostra o tempo médio por tarefa do workflow. O ***workflow2*** é constituído por um único tipo de tarefa, o programa Blast, que é executado 1.200 vezes. Os resultados aqui apresentados são baseados em dez execuções realizadas para cada estratégia, totalizando 48.000 execuções no OSG. Os parâmetros utilizados para o Condor/DaGMan foram: MAXPRE=20 e MAXPOST=20.

Tabela 5 - Tempo médio em segundos para transferência de dados e execução por tipo de tarefa.

	Tarefa
	T1
Quantidade	1.200
Transferência de dados	3
Execução	25

Por se tratar de um workflow sem compartilhamento de dados, não foram realizadas medições para a estratégia *dado_presente*. Como pode ser observado na

figura 46, o desempenho das quatro estratégias é similar durante a execução das cem primeiras tarefas. A partir deste ponto, as abordagens *oportunista* e *ultimo_sítio_usado* começam a se destacar das demais. Da mesma forma que o ocorrido na execução do ***workflow1***, as tarefas previamente submetidas começam a finalizar suas execuções possibilitando a estas duas estratégias fazer o escalonamento em função do desempenho das localidades. A estratégia *oportunista* continua apresentando um desempenho melhor que as demais sendo superior em aproximadamente 50%, 75% e 200% em relação às abordagens *ultimo_sítio_usado, aleatória_ponderada* e *circular* respectivamente. Porém, diferentemente do resultado do ***workflow1***, existe uma nítida diferença no desempenho das demais abordagens.

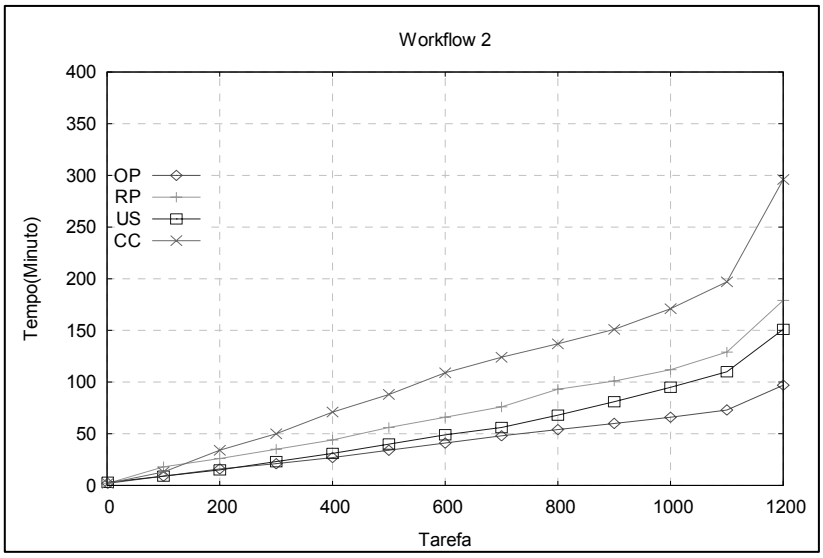


Figura 46 - Tempo médio de cada tarefa do workflow2 de acordo com a estratégia de escalonamento

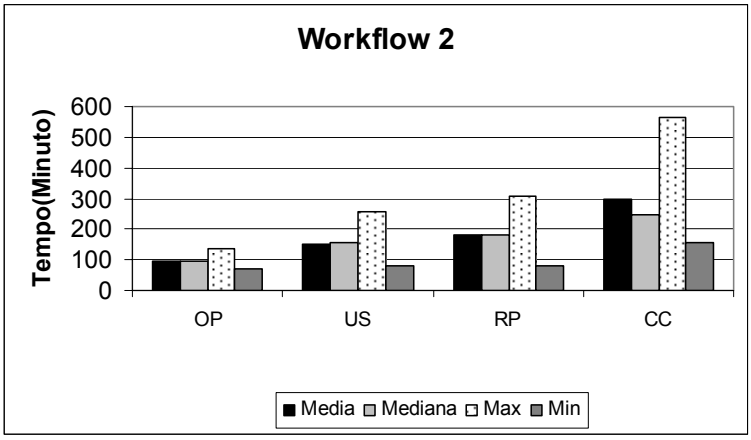


Figura 47 - Estatísticas básicas por estratégia de execução para o workflow2

A figura 47 mostra as estatísticas básicas da execução do ***workflow2*** enquanto que a figura 48 mostra o ganho obtido pela execução no Grid de acordo com as quatro abordagens de escalonamento. O ***workflow2*** apesar de ser formado por tarefas muito rápidas, é composto de um número grande de tarefas. Conseqüentemente a sua execução seqüencial é muito demorada. A sua execução no Grid proporciona ganhos significativos para todas as abordagens.

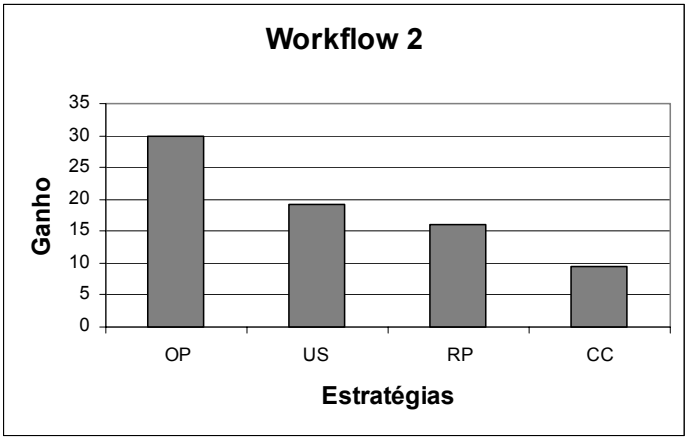


Figura 48 - Ganho por estratégia de execução

VI. 5 - Workflow3 – Divisão Paralela e Sincronização

A tabela 6 mostra os tempos médios de execução e de transferência de arquivos por tipo de tarefa do ***workflow3.*** A figura 49 mostra o tempo médio por tarefa do workflow. Os resultados aqui apresentados são baseados em dez execuções realizadas para cada estratégia, totalizando 4.750 execuções no OSG. Os parâmetros utilizados para o Condor/DaGMan foram: MAXPRE=5 e MAXPOST=20.

A redução no número de pré-scripts simultâneos foi realizada objetivando diminuir o número de tarefas aptas a serem submetidas para execução. Como o número de tarefas no primeiro nível do workflow é pequeno e estas tarefas são rápidas, esta redução beneficia tanto a estratégia *oportunista* como a *ultimo_sítio_usado,* sem trazer nenhum prejuízo as abordagens *aleatória_ponderada, dado_presente* e *circular.*

Diferentemente dos workflows anteriores, o ***workflow3*** apresenta as seguintes particularidades:

- As tarefas T2, T3, T4, T5 e T6 não são paralizáveis, ou seja, são executadas uma única vez;
- As tarefas T3 e T4 bem como T5 e T6 podem executar em paralelo;
- O tempo de transferência de dados nas tarefas T2, T3, T4, T5 e T6 supera o tempo de execução das tarefas.

Tabela 6 - Tempo médio em segundos para transferência de dados e execução por tipo de tarefa.

	Tarefa					
	T1	T2	T3	T4	T5	T6
Quantidade	90	1	1	1	1	1
Transferência de dados	6	765	1265	2018	1669	1726
Execução	122	542	562	839	433	766

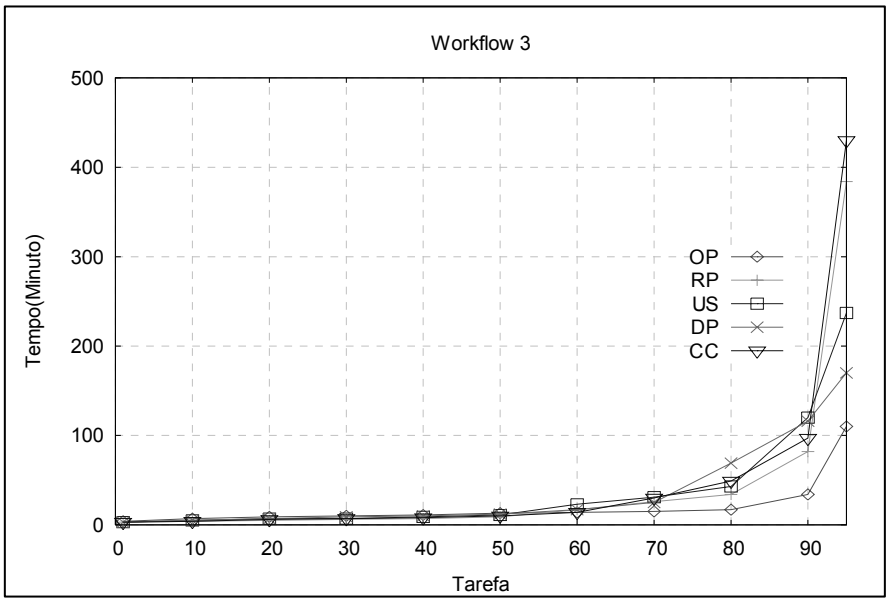


Figura 49 - Tempo médio de cada tarefa do workflow3 de acordo com a estratégia de escalonamento

Como pode ser observado na figura 49, o desempenho das cinco abordagens é praticamente o mesmo durante as primeiras setenta tarefas. Ao final da execução do primeiro nível do workflow, a estratégia oportunista começa a se destacar em relação à demais. A partir deste ponto, a estratégia *dado_presente* começa a se beneficiar da

localidade dos dados. Como o volume de dados intermediários produzidos durante a execução do workflow é considerável, esta estratégia desempenha melhor que as demais abordagens que não são dependentes da localização dos dados. Entretanto, mesmo sob estas condições, a estratégia *oportunista* supera a *dado_presente* em cerca de 50%.

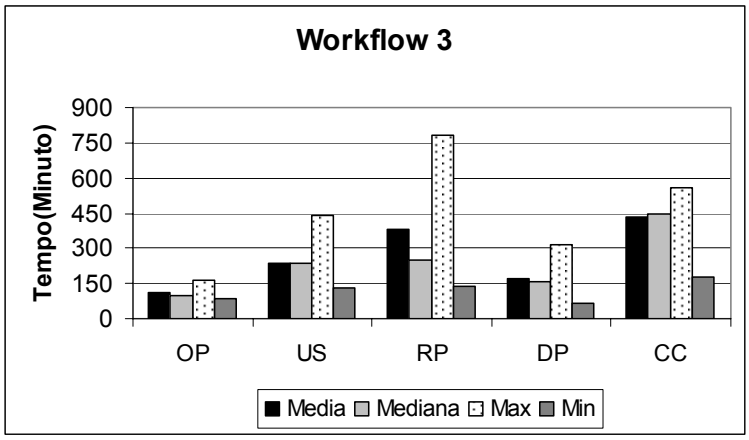


Figura 50 - Estatísticas básicas por estratégia de execução para o *workflow3*

A figura 50 mostra as estatísticas básicas da execução do ***workflow3***. O menor tempo de execução foi alcançado pela estratégia *dado_presente* e a maior com a estratégia *aleatório_ponderada*. Entretanto, os menores valores para a média e mediana continuam sendo atingidos pela estratégia *oportunista*.

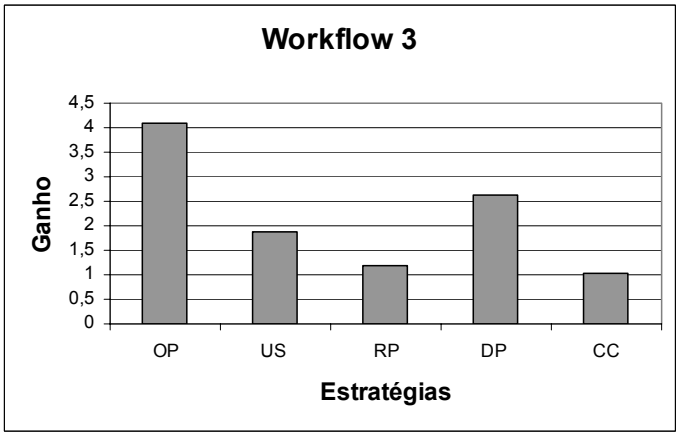


Figura 51 - Ganho por estratégia de execução

A execução seqüencial do ***workflow3*** é beneficiada pelo fato dele possuir menos tarefas e dos dados intermediários não precisarem ser transferidos entre localidades

distintas. Desta forma, o ganho obtido com a execução do workflow3 no Grid é menor que nos outros dois experimentos. A figura 51 mostra o ganho obtido de acordo com a estratégia de execução. Como pode ser notado, o maior ganho é atingido pela estratégia *oportunista* que é de cerca de quatro vezes, seguido pela estratégia *dado_presente* que apresenta um ganho de aproximadamente duas vezes e meio. A estratégia *ultimo_sítio_usado* apresenta um ganho inferior a dois e as demais abordagens praticamente não se beneficiam da execução em Grid.

V. 6 - Considerações quanto aos resultados

A utilização de um grande número de localidades para executar um workflow pode não trazer benefícios ao desempenho. É melhor utilizar um número menor de localidades que forneçam um bom tempo de resposta do que aumentar o grau de paralelismo e escalonar tarefas que permanecerão um longo tempo em uma fila remota.

Os resultados obtidos nos experimentos evidenciam que as estratégias *circular* e *aleatória_ponderada* não se mostraram adequadas para nenhum dos tipos de workflows analisados nos experimentos desta tese. Quando aplicadas ao processamento de um workflow espacial, elas não tiram proveito das informações da abrangência espacial das tarefas; quando aplicadas aos demais workflows implementados na tese, não conseguem detectar o comportamento das localidades no processamento das tarefas submetidas. Estes tipos de algoritmos parecem estar limitados a Grids que permitem a realização de reserva de recursos.

A estratégia *ultimo_sítio_usado* pode não fornecer bom desempenho em workflows que contenham um padrão de sincronização como é o caso do workflow3. Neste caso, a tarefa de sincronização só pode ser iniciada após o término do conjunto de tarefas predecessoras. Caso a última tarefa predecessora tenha executado em uma localidade com baixo desempenho, ou seja, a tarefa tenha permanecido um longo tempo na fila remota, são grandes as chances da tarefa de sincronização também apresentar um desempenho ruim visto que será escalonada para o mesmo sítio.

A estratégia *dado_presente*, como era de se esperar, consegue trazer benefícios em workflows onde grandes volumes de dados são processados pelas tarefas. Entretanto, os resultados mostram que em situações de alto compartilhamento de arquivos como no workflow SDSS/Coadd, a estratégia de agrupamento por proximidade espacial conseguiu ser mais eficiente. No caso do workflow 3, apesar de existirem tarefas cujos tempos de transferência de dados são superiores ao tempo de execução da tarefa, a estratégia oportunista foi mais eficiente que a *dado_presente,* fornecendo um ganho no desempenho em cerca de 60%.

A estratégia oportunista reduz a possibilidade de se ter tarefas sendo escalonadas para a execução em localidades com baixo desempenho. Ela é auto-ajustável sendo capaz de aumentar a carga de trabalho para locais que estão provendo bom desempenho e da mesma forma reduzir a carga para os locais cujo desempenho degrada.

VII – CONCLUSÕES

O processamento de workflows em Grid é ainda um problema em aberto e vem sendo alvo de pesquisas pela comunidade científica. Características específicas encontradas em muitas classes de aplicações como o compartilhamento de dados e o número elevado de transferências de arquivos, e as características do próprio ambiente tais quais a disponibilidade e o desempenho dos recursos, constituem fatores que dificultam o planejamento de execução de um workflow. Apresentamos nesta tese uma solução para o escalonamento dinâmico de tarefas de workflows científicos em ambientes de Grid, por meio de duas novas estratégias: a estratégia de agrupamento por proximidade espacial e a estratégia oportunista. Os resultados experimentais mostraram um desempenho superior da solução proposta em relação às abordagens usuais de escalonamento dinâmico, mostrando a adequação da solução.

Uma das vantagens da solução proposta nesta tese é o desenvolvimento dos algoritmos no contexto de sistemas clássicos para execução de workflows, tais como o Globus, Condor/DagMan e VDS, largamente utilizados na comunidade de Grid. Além disto, a solução está disponível para o uso por outras pessoas.

O custo da solução para o escalonamento dinâmico é desprezível, pois envolve um número pequeno de trocas de mensagens, e por isto não chegou a ser computado. Uma vez que os algoritmos utilizam informações já coletadas e disponibilizadas pelos sistemas Condor e VDS, somente é necessário atualizar a base de controle do Euryale^{+} durante a execução do workflow. A estratégia de agrupamento por proximidade espacial requer que seja feito um projeto de geração de grupos o que representa um custo adicional à solução. Entretanto, este custo adicional pode ser recompensado pelos ganhos obtidos durante a execução, caso o workflow possua uma carga elevada de tarefas, ou venha a ser executado repetidamente.

A estratégia de agrupamento por proximidade espacial é voltada ao processamento de workflows com características espaciais e tem por principal objetivo

reduzir o número de transferências de arquivos durante a execução do workflow. Os resultados obtidos nos experimentos, com a eliminação de 50% a 90% no número de transferências de arquivos em relação à melhor estratégia analisada, mostram a eficácia da estratégia. Esta redução pode melhorar o desempenho do workflow e também beneficiar o Grid como um todo pela redução de tráfego de arquivos entre os sítios. Devido aos requisitos da aplicação SDSS/Coadd, como o elevado espaço em disco necessário para o seu processamento e a dificuldade para a alocação de recursos necessários para os diversos testes, optou-se por realizar os experimentos num ambiente simulado. Entretanto, foi possível realizar dois testes em um ambiente real que validaram experimentalmente os resultados da simulação (MEYER et al., 2005a). Os experimentos foram realizados com amostras do workflow, que foi executado segundo as estratégias de agrupamento por proximidade espacial e aleatória. Os resultados confirmaram o desempenho superior da solução proposta, o que nos encoraja a validá-la em outras aplicações.

A estratégia oportunista propõe uma heurística para o planejamento de workflows baseada na observação do comportamento do Grid. O escalonamento é dinâmico e local, não considera estimativas de desempenho das tarefas nos recursos do Grid e não necessita investigar a disponibilidade dos recursos remotos de cada localidade que participa do Grid. Ela é auto-ajustável sendo capaz de aumentar a carga de trabalho para locais que estão provendo bom desempenho e da mesma forma reduzir a carga para os locais cujo desempenho degrada. A estratégia oportunista procura tirar proveito do desempenho dos melhores sítios durante a execução para diminuir o tempo total de processamento do workflow.

Nesta tese foi possível analisar o comportamento da estratégia oportunista no *Open Sciences Grid* e na execução de workflows com diferentes padrões de composição de seus componentes. Esse ambiente de Grid é um dos maiores ambientes operacionais disponíveis contando com cerca de 60 sítios, cada um contendo diversos processadores. Os resultados alcançados nos três experimentos evidenciam a eficiência do algoritmo que mostrou ganhos no desempenho de até 150% em relação às melhores estratégias analisadas. Os resultados experimentais do aumento de desempenho da estratégia oportunista obtida em um ambiente real de Grid é também uma contribuição importante

fornecida por esta tese. Muitos trabalhos encontrados na literatura analisam os algoritmos de forma isolada, em Grids muito restritos (poucos sítios) ou baseados em simulações.

Para ampliar os ótimos resultados alcançados, algumas extensões podem ser promovidas nas duas estratégias. É conveniente refinar as decisões para a geração de grupos a partir de novos resultados que podem surgir com uma avaliação da estratégia de agrupamento por proximidade espacial sobre um Grid real. Além disso, seria interessante integrar a estratégia oportunista para o escalonamento de grupos de tarefas. Uma possível forma de integração seria a associação dos grupos de tarefas a sítios virtuais e durante a execução do workflow proceder com o mapeamento do sítio virtual para o real de acordo com a estratégia oportunista.

Com relação à estratégia oportunista, uma vez que um Grid pode ser um ambiente extremamente dinâmico, a disponibilidade dos seus recursos pode variar constantemente. Conseqüentemente, o desempenho das localidades também pode variar ao longo da execução do workflow. Por isso, a estratégia oportunista vai eliminando ou reduzindo a carga de trabalho das localidades com baixo desempenho. Quando a duração de um workflow é relativamente pequena, este comportamento não traz problemas como mostrado nos experimentos. Porém, se a execução do workflow for longa, é desejável que exista um mecanismo que possibilite a re-introdução de sítios descartados devido à indisponibilidade ou ao baixo desempenho durante uma fase do workflow.

Um outro aspecto que também pode ser considerado pela estratégia oportunista é a janela de tempo que o algoritmo deve utilizar para computar a razão entre tarefas concluídas e submetidas. Na versão atual, esta janela foi sempre definida a partir do início da execução do workflow. Porém em workflows de longa duração pode ser interessante reduzir o período de tempo de medição de forma a fazer com que a decisão seja feita com base em dados mais recentes e consiga desta forma, aproveitar melhor o desempenho das localidades.

Além disso, se um workflow possuir um conjunto pequeno de tarefas e estas tarefas tiverem um tempo de execução elevado, ou se o workflow não possuir dependências de execução entre as tarefas, então poderá acontecer de o pré-processamento das tarefas terminar antes que alguma tarefa tenha finalizado o seu processamento. Neste caso, as tarefas terão sido escalonadas de forma circular e o caráter de observação do comportamento do Grid da estratégia oportunista não terá sido aproveitado. Uma forma de evitar esta situação é diminuir o número de pré-scripts simultâneos para o workflow. Por exemplo, no caso do Condor/DagMan existe um parâmetro que permite controlar este número. Neste caso, a vazão de planejamento das tarefas do workflow será menor possibilitando que tarefas submetidas terminem sua execução. Uma segunda alternativa seria adiar a seleção do sítio para execução da tarefa dentro do algoritmo. Este adiamento poderia ser implementado através de uma espera que aguardaria pela conclusão de tarefas para então computar a escolha.

Além dos algoritmos implementados nas duas estratégias, também foram desenvolvidos um gerador de workflow abstrato, constituído de uma meta linguagem de definição de workflow (WGL) e um interpretador (GenWF), e um simulador para execução de workflows. A WGL é uma linguagem com uma sintaxe simples que permite ao usuário facilmente definir um workflow na estrutura de um DAG. O GenWF interpreta a WGL e gera como resultado a especificação em VDL do workflow abstrato e a descrição do DAG no formato de leitura definido pelo Condor/DagMan. A vantagem destas duas ferramentas é permitir a definição de workflows com diferentes formas, tamanhos e tempos de execução de seus componentes de uma forma rápida e concisa. Estas características podem ser modeladas facilmente através de parâmetros para produzir um DAG que consuma pequeno ou grande volume de dados e gere conjuntos de arquivos com diferentes números de elementos para serem processados por programas rápidos ou demorados. Uma outra vantagem do gerador de workflow abstrato é que uma vez definido, o DAG que representa o workflow está pronto para ser processado no Grid por qualquer dos sistemas de planejamento do VDS ou pode ser adaptado para ser processado por qualquer sistema de gerência de workflow que processe DAG. As soluções existentes hoje assumem que o workflow será descrito manualmente. Nos experimentos realizados, em que cerca de mil e duzentas tarefas precisam ser definidas esse tipo de solução se mostra inviável.

O simulador de execução do workflow permite que sejam realizados estudos de desempenho e consumo de recursos de acordo com diversas estratégias de execução. Os resultados da simulação são registrados em um arquivo e apresentam além do tempo total de execução do workflow o número total de transferências de arquivos e o espaço em disco consumido em cada sítio. Esses resultados permitem uma análise minuciosa e direcionada do comportamento do workflow. O simulador apresenta funcionalidades similares aos sistemas Condor/DagMan e VDS, aceitando a mesma definição de workflow que é fornecida ao VDS. Isto possibilita que uma mesma definição de DAG, gerada através da WGL ou não, possa ser executada ou simulada sem nenhuma alteração. O simulador trabalha com o escalonamento dinâmico de cada tarefa e permite a execução de acordo com todos os algoritmos de escalonamento analisados nesta tese. É possível definir o número de rotinas de pré-processamento concorrente e pode ser configurado para trabalhar em ambientes com ou sem reserva de recursos. A configuração dos recursos existentes nos sítios é fornecida de acordo com o modelo do usuário.

O selecionador de sítio na arquitetura de planejamento dinâmico é o principal componente para o escalonamento eficiente do workflow. Idealmente, o selecionador de sítio deve ser capaz de gerenciar uma família de estratégias de escalonamento que sejam genéricas ou que atendam as especificidades de aplicações e escolher a que melhor se adequa ao workflow. Como trabalho futuro, pretendemos estender o comportamento do Euryale^{+} com a geração de heurísticas que auxiliem esta tomada de decisão.

Existem grandes possibilidades de pesquisa na área de escalonamento de workflows em Grid. Como apontado por ANDRIEUX et al. (2003), o uso de metodologias *fuzzy* pode ser interessante na medida em que utilizam conceitos e técnicas para representar e inferir conhecimento que é impreciso e incerto, tal como o ambiente de Grid. Técnicas de inteligência artificial para o planejamento de workflows e o uso de ontologias para gerar definições abstratas e concretas de workflows também apresentam um grande potencial e começam a ser utilizadas em alguns trabalhos. Apesar de alguns autores acreditarem que na medida em que a técnica de reserva de recursos melhore, o uso do planejamento estático irá aumentar, acreditamos que o

planejamento dinâmico é mais coerente com a natureza do Grid que tende a se manter dinâmica.

Por fim, acredita-se que a solução desenvolvida nesta tese contribui para que as aplicações científicas tirem um maior proveito dos Grids, ao mesmo tempo em que facilita o uso deste ambiente por parte do cientista.

REFERÊNCIAS BIBLIOGRÁFICAS

AALST, V., HOFSTEDE, A., KIEPUSZEWSKI, B., BARROS, A., 2003, "Workflow patterns", *Distributed and Parallel Databases,* v. 14, n. 3, pp. 5-51.

ALHUSAINI, A., PRASANNA, V., RAGHAVENDRA,C., 1999, "A Unified Resource Scheduling Framework for Heterogeneous Computing Environments", In: *Proceedings of the Eighth Heterogeneous Computing Workshop*, IEEE Computing Society, pp. 156-165, Washington, USA.

ALLEN, G., et al, 2003, "Enabling Applications on the Grid: A GridLab Overview", *International Journal of High Performance Computing Applications: Special issue on Grid Computing: Infrastructure and Applications,* v. 17, n. 3, pp. 449-466, November.

ALLEN, G., et al, 2005, "The Grid Application Toolkit: Towards Generic and Easy Application Programming Interfaces for the Grid", In: *Proceedings of the IEEE*, v. 93, n. 3, pp. 534-550.

ALTINTAS, I., BERCKLEY, C., JAEGER, E., JONES, M. et al, 2004, "Kepler: An Extensible System for Design and Execution of Scientific Workflows", In: *Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM04)*, pp. 423-424, Santorini, Greece, June.

ALTINTAS, I., BIRNBAUM, I., BALDRIDGE, K., LUDAESCHER, B., 2005, "A Framework for Design and Reuse of Grid Wrokflows", In: *International Workshop on Scientific Applications on Grid Computing*, LNCS 3458, pp. 120-133.

ANDRIEUX, A., et al., 2003, *Open Issues in Grid Scheduling*, UKeS-2004-03.

ALTSCHUL S.F., GISH W., MILLER W., MYERS E.W., LIPMAN D.J., 1990, "Basic local alignment search tool", *Journal of Molecular Biology*, n. 215, pp. 215-403, October.

BAGRODIA, R., et al., 1998, "Parsec: A Parallel Simulation Environment for Complex Systems" *IEEE Computer*, v. 31, n. 10, pp. 77-85, October.

BHOWMICK,S., SINGH,D., LAUD, A., 2003, "Data Management in Metaboloinformatics:Issues and Challenges", *Lecture Notes in Computer Scince*, v. 2736, pp. 392-402.

CAMERON, D, et al., 2003a, "Evaluating Scheduling and Replica Optimisation Strategies in OptorSim", In: *Proceedings. of 4th International Workshop on Grid Computing (Grid2003)*, pp. 52-59, Phoenix, USA, November.

CAMERON, D., et al., 2003b, "Evaluation of an Economic-Based File Replication Strategy for a Data Grid", In: *International. Workshop on Agent Based Cluster and Grid*

Computing at Int. Symposium on Cluster Computing and the Grid (CCGrid2003), Tokyo, Japan, May.

CASANOVA, H., et al., 2000, "The AppLeS Parameter Sweep Template: User-Level Middleware for the Grid", *Scientific Programming*, v. 8, n. 3, pp. 111-126.

CHAKRABARTI, A., DHEEPAK, R.A., SENGUPTA, S., 2004, "Integration of Scheduling and Replication in Data Grids", *Lecture Notes in Computer Science*, v. 3296, pp. 375-385, January.

CAO, J., JARVIS, S., SAINI, S., NUDD, G., 2003, "GridFlow: Workflow Management for Grid Computing", In: *Proceedings of the 3rd. International Symposium on Cluster Computing and Grid*", pp.198-206, Tokyo, Japan May.

COOPER, K., DASGUPATA, A., KENNEDY, K., et al., 2004, "New Grid Scheduling and Rescheduling Methods in the GrADS Project", *International Journal of Parallel Programming*, v. 33, n. 2-3, pp.209-229.

CONDOR TEAM, 2003 "*http://www.cs.wisc.edu/condor/manual/v6.4/ref.html"*

CZAJKOWSKIE, K., FITZGERALD, S., FOSTER, I., KESSELMAN, C., 2001, "Grid Information Services for Distributed Resource Sharing", In: "*Proccedings of the 10th IEEE International Symposium on High Performance Distributed Computing*", pp. 181-193, August.

DEELMAN. E., et al., 2003, "Mapping Abstract Workflows onto Grid Environments", *Journal of Grid Computing*, v. 1, n. 1, pp. 25-39.

DEELMAN, E, BLYTHE, J., GIL, Y., KESSELMAN, C., et al., 2004, "Pegasus: Mapping Scientific Workflows onto the Grid", *Lecture Notes in Computer Science*, v.3165, pp. 11-20.

DEELMAN, E., SINGH, G., SU, M., BLYTHE, J., et al., 2005, "Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems" *Scientific Programming*, v. 13, n. 3, pp. 219-237.

DUMITRESCU, C., FOSTER, I., 2005, "Experiences in Running Workloads over Grid3", In: *GCC 2005*, *Lecture Notes in computer Science*, v. 3795, pp.274-286.

FOSTER, I., 2005, "Globus Toolkit Version 4: Software for Services Oriented Systems", In: *International Conference on Network and parallel Computing*, *Lecture Notes in Computer Science*, v. 3779, pp. 2-13.

FOSTER,I., KESSELMAN, C., 1999, "Computational Grids", In: Foster, Kesselman, *The Grid: Blueprint for a New Computing Infrastructure,* first edition, chapter 2*,* Morgan-Kaufman.

FOSTER, I. et al., 2004, "The Grid2003 Production Grid: Principles and Practice", In: *Proceedings of the 13th International Symposium on High Performance Distributed Computing,* pp.236-245.

FOSTER,I., VOECKLER, J., WILDE, M., ZHAO, Y., 2002, "Chimera: A Virtual Data System for Representing, Querying and Automating Data Derivation", In: *Proceedings of the 14th Conference on Scientific and Statistical Database Management,* pp. 37-46, Edinburgh, Scotland, July.

FOSTER, I, VOECKLER, J., WILDE, M., ZHAO, Y., 2003, "The Virtual Data Grid: A New Model and Architecture for Data-Intensive Collaboration", In: *Proceedings of the Conference on Innovative Data System Research – CIDR-2003,* Asilomar, CA, USA, January.

FREY, J., TANNENBAUM, T., FOSTER, I., LIVNY, M., TUECKE, S., 2001, "Condor-G: A Computation Management Agent for Multi-Institutional Grids", In: *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10)*, pp. 237-246, San Francisco, California.

GOBLE, C., PETTIFER, S., STEVENS, R., GREENHALGH, C., 2003, "Knowledge Integration: In silico Experiments in Bioinformatics", In: *The Grid: Blueprint for a New Computing Infrastructure*, Second Edition eds. Ian Foster and Carl Kesselman, Morgan Kaufman, November.

GOODALE, T., TAYLOR, I., WANG, I., 2005, "Integrating Cactus Simulations within Triana Workflows", In: *Proceedings of 13th Annual Mardi Gras Conference - Frontiers of Grid Applications and Technologies*, Louisiana State University, pp. 47-53, February.

GREENWOOD, M., WROE, C., STEVENS, R., GOBLE, C., ADDIS, M., 2002, "Are bioinformaticians doing e-Business?" In: *Proceedings Euroweb 2002: The Web and the GRID - from e-science to e-business*, pp. 3-26, Oxford, UK.

KOCHUT, K., ARNOLD, J., SETH, A., et al., 2003, "IntelliGEN: A Distributed Workflow System for Discovering Protein-Protein Interactions", *International Journal on Distributed and Parallel Databases*, v. 13, n. 1, pp. 43-72.

LASZEWSKI, G., HATEGAN, M., 2005, "Workflow Concepts of Java CoG Kit", *Journal of Grid Computing*, v. 3, n. 3, pp. 239-258, September.

MANDAL, A. et al, 2005, "Scheduling Strategies for Mapping Application Workflows onto the Grid", In: *14th IEEE Symposium on High Performance Distributed Computing (HPDC 2005)*, IEEE Computer Society Press, pp. 125-134.

MEDEIROS, C., VOSSEN, G., WESKE, G., 1995, "WASA: A Workflow-Based Architecture to Support Scientific Database Applications", In: Proceedings of the 6th *DEXAConference* 1995, pp. 574-583, London, England, September.

MEYER, L., MATTOSO, M., 2004, "*Parallel Strategies for Processing Scientific Workflows*", COPPE-SISTEMAS ES-646/04.

MEYER, L., MATTOSO, M., FOSTER, I., et al., 2005a, "*Planning Spatial Workflow to Optimize Grid Performance*", GriPhyN 2005-10.

MEYER, L., MATTOSO, M., ROSSLE, S., 2005b, et al., "Parallelism in Bioinformatics Workflows", In: *Vecpar 2004, Lecture Notes in Computer Science*, v. 3402/2005, pp. 583-596.

MEYER, L., MATTOSO, M., FOSTER, I., et al., 2006a, "Planning Spatial Workflow to Optimize Grid Performance", In: *ACM Symposium of Applied Computing*, v. 2, pp. 786-790, Dijon, França, April.

MEYER, L., MATTOSO, M., FOSTER, I., et al., 2006b, "An Opportunistic Algorithm for Scheduling Workflows on Grids", to appear In: *Proceedings of VECPAR'06*, Rio de Janeiro, Brazil, July.

OINN, T., ADDIS, M., FERRIS, J. et al, 2004, "*Taverna: a Tool for the Composition and Enactment of Bioinformatis Workflow*", In: *BIOINFORMATICS*, v. 20, n. 17 , pp. 3045-3054, Oxford University Press.

OPEN SCIENCES GRID CONSORTIUM , 2005, "Open Sciences Grid Management Plan", disponível em http://osg-docdb.opensciencegrid.org/0003/000314/001/2005-11-OSGmanagementPlan-v10.pdf.

PROTEIN DATA BANK, www.rcsb.org/pdb/

RANGANATHAN,K., FOSTER,I., 2001, "Identifying Dynamic Replication Strategies for a High-Performance Data Grid", In: *Proceedings of the International Workshop on Grid Computing*, pp.75-86, Denver, USA.

RANGANATHAN, K., FOSTER, I., 2002, "Decoupling Computation and Data Scheduling in Distributed Data Intensive Applications", In: *Proceedings of the 11th International Symposium for High Performance Distributed Computing (HPDC-11)*, pp. 352-361, Edinburgh, Scotland.

RANGANATHAN, K., FOSTER, I., 2003a, "Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids", *Journal of Grid Computing,* v.1 ,n.1, pp. 53-62 .

RANGANATHAN,K., FOSTER, I., 2003b, "Computation Scheduling and Data Replication Algorithms for Data Grids", In: *Grid Resource Management: State of the Art and Future Trend*', Kluwer Academic Publishers.

SDSS PROJECT, 2006, http://www.sdss.org

SHAN, H., OLIKER, L., SMITH, W., BISWAS, R., 2004, "Scheduling in Heterogeneous Grid Environments: The Effects of Data Migration", In: *Proceedings of International Conference on Advanced Computing and Communication*, Gujarat, India.

SINGH, G., KESSELMAN, C., DEELMAN, E., 2005, "Optimizing Grid-Based Workflow Execution", *Journal of Grid Computing,* v. 3, n.3-4, pp. 201-219.

STREIT, A., et al, 2005, "UNICORE - From Projects Results to Production Grids", *Grid Computing: New Frontiers of High Performance Computing*, pp. 357-376, Elsevier.

SULAKHE, D., et al., 2005, "GNARE: An Environment for Grid-Based High-Throughput Genome Analysis", *Journal of Clinical Monitoring and Computing*, v. 19, n. 4-5, pp. 361-369, October.

TAYLOR, I., SHIELDS, M., WANG, I., RANA, O., 2003, "Triana Applications within Grid Computing and Peer to Peer Environments**"**. *Journal of Grid Computing*, v. 1, n.2, pp. 199-217. Kluwer Academic Press.

THAIN, C., et al., 2003, "Pipeline and Batch Sharing in Grid Workloads", In: *Proceedings of the Twelfth IEEE Symposum on High Performance Distributed Computing,* pp.152-161, Seattle, Washington, USA.

THAIN, D., TANNENBAUM, T., LIVNY, M., 2005, "Distributed computing in practice: the Condor experience ", *Concurrency and Computation: Practice and Experience*, v. 17, n. 2-4, pp. 323-356, February-April

VARGAS, P., Dutra, I., GEYER, C., 2004, "Application Partitioning and Hierarchical Management in Grid Environments", In: *1st International Middleware Doctoral Symposium*, pp. 314-318, Toronto, Canadá, October.

VARGAS, P., et al., 2005, "Hierarchical Submission in a Grid Environment", In: *Proceedings of the 3rd Workshop on Middleware for Grid Computing*, pp. 1-6, Grenoble, France, December.

VDS USER GUIDE, http://vds.uchicago.edu/twiki/bin/view/VDSWeb/VDSDocs

WIECZOREK, M., PRODAN, R., FAHRINGER, T., 2005, "Scheduling of Scientific Workflows in the ASKALON Grid Environment", *SIGMOD Record*, v. 34, n.3, pp.56-62, September.

WOLSKI, R., SPRING, N., HAYES, J., 1999, "The Network Weather Service: A Dsitributed Resource Performance Forecasting Service for Metacomputing", *Future Generation Computer Systems*, pp. 757-768

WORKFLOW MANAGEMENT COALITION, 1995, "The Workflow Reference Model", TC00-1003, http://www.wfmc.org/.

YU, J., BUYYA, R, 2005, "A Taxonomy of Scientific Workflow Systems for Grid Computing", *SIGMOD Record*, v. 34, n.3, pp. 44-49, September.