Silvio Luiz Stanzani

ESTRATÉGIAS DE ESCALONAMENTO DE WORKFLOWS COM TAREFAS PARALELAS E SEQUENCIAIS EM GRADES COMPUTACIONAIS

São Paulo
2014

Silvio Luiz Stanzani

ESTRATÉGIAS DE ESCALONAMENTO DE WORKFLOWS COM TAREFAS PARALELAS E SEQUENCIAIS EM GRADES COMPUTACIONAIS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Ciências

Área de Concentração:
Engenharia de Computação

Orientador:
Profa. Dra. Liria Matsumoto Sato

São Paulo
2014

Silvio Luiz Stanzani

ESTRATÉGIAS DE ESCALONAMENTO DE WORKFLOWS COM TAREFAS PARALELAS E SEQUENCIAIS EM GRADES COMPUTACIONAIS

Tese apresentada à Escola Politécnica da Universidade de São Paulo para obtenção do título de Doutor em Ciências

Área de Concentração:
Engenharia de Computação

Orientador:
Profa. Dra. Liria Matsumoto Sato

São Paulo
2014

RESUMO

A demanda por alto desempenho é um desafio enfrentado por diversas aplicações científicas. Nesse sentido, ambientes para processamento distribuído, tais como, *clusters* e grades computacionais, têm sido desenvolvidos para prover suporte ao uso de diversos recursos simultaneamente para uma mesma aplicação.

Aplicações computacionalmente intensivas são organizadas em *workflows* e executadas com suporte de *middlewares* para abstrair a complexidade de uso de tais ambientes. Em ambientes de grade computacional, a execução de *workflows* contendo tarefas sequenciais e tarefas com paralelismo interno, obtendo bom desempenho, é um desafio, devido à heterogeneidade e comportamento dinâmico do ambiente. Nesse sentido, o escalonamento de *workflows* em ambientes de grade computacional é essencial.

O problema de escalonamento de tarefas, em sua forma geral, é NP-Completo, dessa forma, o estudo do escalonamento de *workflows* em ambientes de grade computacional é fundamental para aprimorar a execução de aplicações computacionalmente intensivas.

O objetivo dessa tese é propor estratégias de escalonamento de *workflows*, que exploram os seguintes aspectos:

- Avaliação da possibilidade de executar cada tarefa com paralelismo interno usando recursos de múltiplos *clusters*;
- Adaptação de planos de escalonamento no momento da submissão de novos *workflows*.

Foram desenvolvidas duas estratégias: a primeira é uma estratégia para escalonamento estático de *workflows*, que considera o ambiente dedicado a execução de um *workflow*. A segunda foi desenvolvida para ser utilizada em conjunto com a primeira, para melhorar o tempo de resposta de múltiplos *workflows* que podem ser submetidos em diferentes momentos. As estratégias propostas foram avaliadas em um ambiente de simulação.

Palavras-chave: *multi-cluster*, escalonamento de *workflows*;

ABSTRACT

The demand for high performance is a common problem in many scientific applications. In this sense, distributed processing environments such as cluster, grid computing and multi-cluster environments have been developed to provide support for the use of several resources simultaneously for the same application.

Computationally intensive applications are structured as workflows and executed with the support of middleware to abstract the complexity of using such environments. In grid computing environments the execution of workflows containing sequential and parallel tasks, with good performance is a challenge due to the heterogeneity and dynamic behavior of the environment. In this sense, the scheduling of workflows on grid computing environments is essential.

The task scheduling problem in its general form is NP-Complete, in this sense, the study concerning workflow scheduling in grid computing environments is fundamental to improve the performance of computationally intensive applications.

The aim of this thesis is to propose strategies for scheduling workflows that exploit the following aspects:

- Explore the possibility of performing single parallel tasks using multiple clusters;
- Adaptation plans escalation in accordance with the submission of new workflows.

Two strategies were developed: the first one is a strategy for static scheduling of workflows, which considers a dedicated environment to the execution of a workflow. The second one was developed to use in conjunction with the first one, in order to improve the response time of multiple workflows that can be submitted at different times. The proposed strategies were evaluated in a simulation environment.

Key-words: multi-cluster, workflow scheduling;

AGRADECIMENTOS

Gostaria de agradecer em primeiro lugar a Profa. Liria pela orientação, atenção, e amizade.

Ao Dr. Marco Netto, pela colaboração no artigo publicado no WAINA 2013, pela avaliação na banca de qualificação e pelas orientações valiosas na reta final do trabalho.

À Érika, minha mulher, pela paciência, dedicação, carinho, amor e por trazer muita paz e sabedoria nos momentos difíceis.

Ao professor Dr. Alfredo Goldman pelas contribuições na banca de qualificação.

Aos Colegas de pós-graduação Charles, Kakugawa, Calebe, Augusto, Artur, Francisco, Nilton, Darlon, Humberto, Dênis, Fernando Luz, Daniel Lins, Renato, Alan Koch, e demais que não mencionei, pela amizade e colaborações em pesquisa.

Aos Coordenadores de projeto prof. Kazuo, prof. Saraiva, prof. Pedro Correa e prof. Ismael, pelas oportunidades a mim proporcionadas de trabalhar em diversos projetos de pesquisa.

À Infraestrutura da Escola Politécnica da Universidade de São Paulo pelo suporte à realização da pesquisa e suporte financeiro para apresentação de artigos.

LISTA DE ILUSTRAÇÕES

Figura 1. Um Exemplo de um Ambiente de Grade Computacional. 24
Figura 2. Exemplo de um *Workflow* com Tarefas com Paralelismo Interno e Tarefas Sequenciais, Composto por Seis Níveis. 30
Figura 3. Exemplo de um *Workflow* com o Caminho Crítico em Destaque (Tarefas: 1, 5, 9, 13, 14). 32
Figura 4. Tempo de Execução de Dois Planos de Escalonamento (A e B) de um *Workflow* composto por Tarefas com Paralelismo Interno. 39
Figura 5. Exemplo de Escalonamento de Tarefas por Nível. 59
Figura 6. Cálculo do *Rank* das Tarefas do Workflow. 69
Figura 7. Alocação e Mapeamento Realizado pela Estratégia MCA-HEFT. 71
Figura 8. Alocação e Mapeamento Realizado pela Estratégia M-HEFT. 72
Figura 9. GAPs Gerados Pelo Plano de Escalonamento. 74
Figura 10. Modelo Entidade-Relacionamento do Simulador. 78
Figura 11. Modelo de Componentes do Simulador. 79
Figura 12. Submissão de Dois *Workflows* em Momentos Diferentes. 81
Figura 13. Arquivo de Descrição das Dependências e Tarefas do *Workflow* 1. 82
Figura 14. Arquivo de Descrição das Dependências e Tarefas do *Workflow* 2. 82
Figura 15. Arquivo Eventos Para Configuração de Submissão de *Workflows*. 83
Figura 16. Execução das Tarefas do *Workflow* em um *Cluster* de Referência. 84
Figura 17. Custo Computacional e Matriz de Comunicação Entre os Recursos que Executam a Tarefa. 86
Figura 18. Uso das Estratégias Propostas em um Ambiente de Grade Computacional Implementado com Suporte do *Middleware* Globus. 88
Figura 19. Execução de *Workflows* do Formato *Fork-join* Utilizando M-HEFT e MCA-HEFT com diferentes valores de MCA. 92
Figura 20. Execução de *Workflows* de outros Formatos Utilizando M-HEFT e MCA-HEFT com diferentes valores de MCA. 92
Figura 21.Execução de *Workflows* do Primeiro Grupo Utilizando MCA-HEFT com DMCA-HEFT e MCA-HEFT. 96
Figura 22. Execução de *Workflows* do Segundo Grupo Utilizando MCA-HEFT

com DMCA-HEFT e MCA-HEFT. 96

LISTAS DE TABELAS

Tabela 1. Cálculo do MCACCR e MCAMC das Tarefas do Workflow. 70
Tabela 2. Cálculo do MCAET das Tarefas do *Workflow* Usando de 1 à 6 Recursos. 70
Tabela 3. Tempo Gasto com Comunicação Durante a Execução dos *Workflows* Utilizando a Estratégia MCA-HEFT. 93

LISTA DE ABREVIATURAS E SIGLAS

WFMCE	Workflow Management System for Multi-cluster Environment
BRAMS	Brazilian Regional Atmospheric Modeling System
LHC	Large Hadron Collider
WLCG	Worldwide LHC Computing Grid
DAG	Directed Acyclic Graph
ScWFMS	Scientific Workflow Management Systems
WAN	Wide Access Network
VO	Virtual Organization
HEFT	Heterogeneous Earliest Finish Time
CPOP	Critical Path On a Processor
CPA	Critical Path and Area
HCPA	Heterogeneous Critical Path and Area
MCPA	Modified Critical Path and Area
iCASLB	iterative Coupled processor Alocation and Scheduling Algorithm with look-ahead and backfill
EFT	Earliest Finish Time
M-HEFT	Modified Heterogeneous Earliest Finish Time
P-HEFT	Parallel Heterogeneous Earliest Finish Time
AHEFT	Adaptive Heterogeneous Earliest Finish Time

MCA-HEFT	Multi-cluster Allocation-HEFT
DMCA-HEFT	Dynamic Multi-Cluster Allocation-HEFT
MB	Mega Byte
CCR	Computation to Communication Ratio
MDS	Monitoring and Discovering Services
GSI	Grid Security Infrastructure
CA	Certificate Authority
EPR	End Point Reference
MIPS	Million Instructions Per Second

SUMÁRIO

1. Introdução 16

1.1. Objetivos 18

1.2. Justificativa 20

1.3. Estrutura da Tese 21

2. Execução e Escalonamento de *Workflows* em Ambientes de Processamento Distribuído 22

2.1. Definições 22

2.1.1. Ambientes Distribuídos para Execução de Tarefas 22

2.1.1.1. Ambientes de Grade Computacional Compostos por Clusters 23

2.1.1.2. Recursos 25

2.1.2. Tarefa 26

2.1.3. Conjuntos de Tarefas 28

2.1.4. Workflow 29

2.1.5. Níveis da Tarefa em um Worfklow 30

2.1.6. Caminho Crítico 31

2.2. Escalonamento de Tarefas 32

2.2.1. Escalonamento de Tarefas com Paralelismo Interno 33

2.2.2. Escalonamento de Workflows 34

2.2.2.1. Escalonamento Estático de Workflows 35

2.2.2.2. Escalonamento de Workflows em Ambientes Dinâmicos 37

2.2.2.3. Mapeamento de Tarefas 37

2.3. Conclusão 39

3. Estratégias de Escalonamento de Workflows 41

3.1. Estratégias de Escalonamento Estático de Workflows 41

3.1.1. Estratégias de Escalonamento Inter-tarefa 42

3.1.1.1. HEFT (Heterogeneous Earliest Finish Time) 42
3.1.1.2. HEFT com lookahead 43
3.1.1.3. S-HEFT (Stochastic HEFT) 43
3.1.1.4. CPOP (Critical Path On a Processor) 44
3.1.1.5. DAGMap 44
3.1.1.6. PCH (Path Clustering Heuristic) 45
3.1.1.7. BFTT+OLB (Best File Transfer Time+Opportunistic Load Balancing) 45
3.1.2. Estratégias de Escalonamento Intra-tarefa 46
3.1.2.1. M-HEFT (Modified Heterogeneous Earliest Finish Time) 46
3.1.2.2. MHEFT-RSV (M-HEFT-ReSerVation) 47
3.1.2.3. CPA (Critical Path and Area) 47
3.1.2.4. MCPA (Modified Critical Path and Area) 48
3.1.2.5. HCPA (Heterogeneous Critical Path and Area) 49
3.1.2.6. iCASLB (an iterative Coupled processor Alocation and Scheduling Algorithm with look-ahead and backfill) 49
3.2. Estratégias de Escalonamento de Workflows em Ambientes Dinâmicos 50
3.2.1. P-HEFT 50
3.2.2. AHEFT 50
3.2.3. Planner-guided dynamic scheduling 51
3.2.4. Re-escalonamento Híbrido em Ambientes Multi-Cluster 51
3.2.5. Re-escalonamento em Ambientes Multi-Usuário 51
3.2.6. BTS (Balanced Time Scheduling) 52
3.2.7. OWM (Online Workflow Management) 52
3.2.8. MWGS (Multiple Workflow Grid Scheduling) 52
3.3. Estratégias de Escalonamento Dinâmico de Tarefas Independentes com Paralelismo Interno 53

3.3.1. Estratégias Baseadas na Alteração de Reserva de Recursos 53
3.3.2. Estratégias que Utilizam Técnicas de Backfilling 54
3.3.3. Meta-Escalonamento Baseado na Largura de Banda 54
3.4. Análise 54
4. Proposta de Estratégias de Escalonamento de *Workflows* em Ambientes de Grade Computacional 57
4.1. Definição do Problema 57
4.2. Modelo de Aplicações 57
4.3. Estratégia MCA-HEFT (Multi-Cluster Allocation-HEFT) 58
4.3.1. Definição de Prioridade 60
4.3.2. Seleção de Processadores 61
4.3.2.1. Alocação de Cluster 61
4.3.2.2. Alocação de Quantidade de Recursos 63
4.3.2.3. Mapeamento de Tarefas 65
4.3.3. Sequencia de Passos da Estratégia MCA-HEFT 66
4.3.4. Exemplo de Escalonamento Utilizando a Estratégia MCA-HEFT 68
4.4. Estratégia DMCA-HEFT (Dynamic Multi-Cluster Allocation-HEFT) 72
4.4.1. Sequencia de Passos da Estratégia DMCA-HEFT 74
5. Implementação das Estratégias Propostas em um Simulador 76
5.1. Implementação Utilizando Simulador 76
5.2. Estrutura do Simulador 77
5.2.1. Modelo de Dados do Simulador 77
5.2.2. Modelo de Componentes do Simulador 78
5.3. Especificação da simulação 80
5.3.1. Especificação das tarefas 83
5.4. Implementação das Estratégias 86
6. Avaliação das Estratégias Propostas 89

6.1. Caracterização do Ambiente de Testes 89
6.2. Caracterização da Carga de Entrada 90
6.3. Descrição dos Experimentos 90
6.3.1. Experimento de Avaliação da Estratégia MCA-HEFT 91
6.3.2. Experimento de Avaliação da Estratégia DMCA-HEFT 95
6.4. Discussão 98
7. Conclusão e Trabalhos Futuros 99
7.1. Contribuições 99
7.2. Trabalhos Futuros 100
Referências Bibliográficas 101
Apêndice A – Descrição do Ambiente de Testes 109
Apêndice B – Descrição da Carga de Entrada 111

1. Introdução

Muitas descobertas científicas em diversas áreas do conhecimento são obtidas através da pesquisa experimental. Um desafio comum à maioria das aplicações utilizadas para prover suporte à pesquisa experimental é a demanda por computação de alto desempenho.

Um exemplo de aplicação de alto desempenho na área de climatologia é o BRAMS (*Brazilian Regional Atmospheric Modeling System*) (FREITAS et al., 2007), que é um modelo numérico padrão para a região tropical utilizado para simulação de circulação atmosférica. Um desafio relacionado com o BRAMS é a criação de modelos com alto nível de detalhamento (ALMEIDA; BAUER; FAZENDA, 2011) (FAZENDA et al., 2011).

Na área de física de altas energias os projetos de aceleradores de partículas possuem grande demanda por alto desempenho. O LHC (*Large Hadron Collider*)[1] é um dos maiores projetos de aceleradores de partículas, e apresenta dois grandes desafios: a manipulação de grandes quantidades de dados e execução de aplicações computacionalmente intensivas. Nesse contexto, o WLCG (*Worldwide LHC Computing Grid*) é uma infra-estrutura de grade computacional criada para prover suporte ao gerenciamento dos dados gerados e para execução das aplicações do LHC. O WLCG é utilizado também como base para o desenvolvimento de novas tecnologias na área de grades computacionais (BIRD, 2011; ROBERTSON, 2012), que são desenvolvidas para atender os desafios apresentados pelas aplicações do LHC.

O suporte crescente de tecnologias de comunicação e informação para pesquisa científica na área de biologia culminou com o surgimento de um campo chamado Bioinformática. Nesse campo, muitos desafios, tais como, buscas por sequencias genômicas (LIN et al., 2008) e modelagem de distribuição de espécies (MUÑOZ et al., 2011) possuem grande demanda computacional.

Na indústria também é possível verificar a necessidade de suporte à execução de aplicações de alto-desempenho em diversas áreas, tais como, automobilística (WENTZEL, 2009), aviação (MACCHIONI, 2011) e exploração

[1] Site do projeto LHC: http://lhc.web.cern.ch/lhc

de petróleo (BRAZZEL et al., 2012)(NISHIMOTO et al., 2003).

Para atender os desafios relacionados com a execução de aplicações computacionalmente intensivas, vêm sendo desenvolvidos ambientes para processamento distribuído e paralelo, com o objetivo de prover suporte ao uso de diversos recursos simultaneamente por uma mesma aplicação.

As primeiras iniciativas para desenvolvimento de ambientes de processamento distribuídos paralelos foram os *clusters* de computadores (BUYYA, 1999), que são baseados na agregação de computadores homogêneos por meio de uma rede local. A viabilidade de montagem de redes de longo alcance que interligam diferentes domínios administrativos culminou com o surgimento do modelo de grades computacionais, que são infraestruturas computacionais formadas pela agregação de recursos heterogêneos, montados em diversos domínios administrativos (FOSTER, 2001). Uma forma que tem sido muito utilizada para montar uma grade computacional é por meio da agregação de múltiplos *clusters*. Ambientes compostos por múltiplos *clusters* são identificados como *multi-cluster* (BARRETO; ÁVILA; NAVAUX, 2000; BUYYA; BAKER, 2001) e têm sido reconhecidos como infraestruturas eficientes para execução de tarefas com paralelismo interno baseado no paradigma de passagem de mensagem. A execução de tarefas em tais ambientes é realizada com suporte de *middlewares* para abstrair a complexidade de uso de tais ambientes.

Aplicações computacionalmente intensivas são em geral compostas por diversas tarefas interdependentes. Nesse sentido, as aplicações são organizadas em *workflows* de acordo com as dependências de dados entre as tarefas, que podem ser sequenciais ou possuir paralelismo interno. Uma tarefa sequencial apresenta a possibilidade de ser executada em apenas um recurso. Uma tarefa com paralelismo interno, pode ser executada em um ou mais recursos e apresenta diferentes tempos de execução de acordo com o subconjunto de recursos utilizado. Para que a execução de *workflows* que representam aplicações computacionalmente intensivas seja realizada com desempenho adequado em ambientes de grade computacional o escalonamento de tarefas é essencial.

O escalonamento de *workflows* em ambientes de grade computacional é um problema NP-completo (PINEDO, 2008). Nesse sentido, diversas estratégias

têm sido desenvolvidas e aperfeiçoadas para melhorar o desempenho da execução de *workflows* em tais ambientes (YU; BUYYA; RAMAMOHANARAO, 2008)(LOPEZ; HEYMANN; SENAR, 2006).

Um aspecto essencial levado em conta pelas estratégias de escalonamento é que em um ambiente de grade computacional composto por múltiplos *clusters* a execução de uma tarefa com paralelismo interno pode ser realizada utilizando recursos de múltiplos *clusters*. Tal aspecto é amplamente estudado por iniciativas de pesquisa na área de co-alocação (*resource co-allocation*) (NETTO; BUYYA, 2010).

A utilização de recursos de múltiplos *clusters* para redução do tempo de resposta de uma tarefa com paralelismo interno apresenta as seguintes possibilidades:

- Alocação para uma aplicação de uma quantidade de recursos que não é possível encontrar em um único *cluster*;
- Redução do tempo de resposta utilizando recursos de múltiplos *clusters*.

As estratégias de escalonamento de *workflows* em geral limitam a execução de uma tarefa com paralelismo interno sempre em um único *cluster* para evitar a sobrecarga de comunicação (N'TAKPE; SUTER; CASANOVA, 2007), e deixam de explorar características importantes dos ambientes de grade computacional compostas por múltiplos *clusters*.

A utilização de técnicas de co-alocação em conjunto com estratégias de escalonamento de *workflows* apresenta potencial de trazer ganhos de desempenho para uma variedade grande de aplicações computacionalmente intensivas, além de permitir uma utilização mais eficiente dos recursos disponibilizados em ambientes de grade computacional composta por múltiplos *clusters*.

1.1. Objetivos

O problema abordado nesta tese é o escalonamento de múltiplos *workflows* compostos por tarefas sequenciais e tarefas com paralelismo interno em ambientes de grade computacional compostos por múltiplos *clusters*. Cada

tarefa com paralelismo interno pode ser executada em múltiplos recursos. O escopo do problema apresentado é definido de acordo com os seguintes aspectos:

- O primeiro aspecto é que o escalonamento é realizado em um ambiente dinâmico no qual os *workflows* não são conhecidos antecipadamente, e são submetidos em diferentes momentos;
- O segundo aspecto é que cada tarefa com paralelismo interno pode ser executada utilizando um ou mais recursos computacionais, atribuídos pelo escalonador considerando uma quantidade mínima especificada pelo usuário;
- O terceiro aspecto é que uma tarefa com paralelismo interno pode ser executada utilizando recursos de múltiplos *clusters*.

Para contribuir com o avanço do conhecimento no contexto do problema apresentado, esta tese foi desenvolvida considerando a questão a seguir:

"Mapear cada tarefa com paralelismo interno em recursos de múltiplos clusters, considerando o seu grau de paralelismo, pode melhorar o desempenho da execução de múltiplos workflows em ambientes de grade computacional?"

Para responder essa pergunta definiram-se neste trabalho os seguintes objetivos:

- Definir métricas que relacionam as características das tarefas com paralelismo interno e sequenciais de um *workflow*, com a estrutura do *workflow* e características dos recursos para produzir planos de escalonamento, e para decidir se o desempenho de uma tarefa com paralelismo interno pode ser melhorado ao ser mapeada para múltiplos *clusters*;
- Propor estratégias para realizar o escalonamento dos *workflows* e adaptar tais planos de escalonamento quando ocorrerem novas submissões de *workflows*.

Para atender tais objetivos são apresentadas propostas de estratégias de escalonamento de *workflows* que consideram os aspectos mencionados. Para avaliar a proposta foi realizada a implementação das estratégias em um ambiente de simulação e as respectivas análises.

1.2. Justificativa

O uso de estratégias de escalonamento para planejar a execução de aplicações em ambientes de processamento distribuído, tais como, *clusters*, grades computacionais e ambientes *multi-cluster*, é essencial para utilizar de maneira eficiente os recursos oferecidos e obter ganhos de desempenho ao executar aplicações computacionalmente intensivas.

Em ambientes de grade computacional compostos por múltiplos *clusters* os *middlewares* provêm acesso à execução de cada tarefa nos diferentes *clusters* de acordo com um plano de escalonamento gerado previamente. Para gerar um plano de escalonamento adequado no cenário apresentado é importante levar dois aspectos em consideração:

- O primeiro é que é necessário utilizar métricas que forneçam informação precisa a respeito dos recursos disponíveis, e também da carga em execução em todos os *clusters*. É comum que cada *cluster* implemente sua própria estratégia de escalonamento. Nesse sentido, as tarefas são submetidas para um cluster, que utiliza um escalonador local para decidir em quais recursos a tarefa será executada;
- O segundo aspecto é a concorrência de uso de recursos gerada pelas submissões que podem ser realizadas por diversos usuários e em diferentes momentos. Para controlar tal concorrência por recursos, em geral os *middlewares* implementam filas de execução.

As estratégias propostas nesta tese foram desenvolvidas levando em conta que as aplicações são especificadas com *workflows* seguindo o padrão *fork-join* (LI et al., 2003), que é um padrão que aparece em diversas aplicações computacionalmente intensivas. Tal padrão possui duas características:

- A primeira é que é possível identificar, de maneira simples, conjuntos de tarefas que podem ser executadas concorrentemente.
- A segunda é que espaços vagos são gerados pelos planos de escalonamento pois em diversos momentos uma tarefa entra em execução para em seguida liberar a execução de diversas tarefas que serão executadas simultaneamente.

As estratégias que estão sendo propostas nesta tese se baseiam em um modelo de informação que representa todos os recursos de um ambiente de grade computacional de maneira centralizada, que podem ser obtidas antecipadamente e durante a execução dos *workflows*. Tais estratégias podem complementar outras existentes, podem servir de base de informação para outros sistemas de escalonamento, e podem ser utilizadas por *middlewares* de grade computacional para realizar o escalonamento de tarefas.

As estratégias propostas realizam uma alocação de recursos que permite executar *workflows* de maneira eficiente, pois utiliza o grau de paralelismo de cada tarefa e o paralelismo inter-tarefa dos worfklows para decidir a quantidade de recursos para cada tarefa. Além disso, as estratégias refazem o plano de escalonamento quando novos *workflows* são submetidos para que o tempo de resposta da execução de diversos *workflows* seja o menor possível.

1.3. Estrutura da Tese

Esta tese está organizada da seguinte forma: na Seção 2, são apresentados os conceitos relacionados à execução e escalonamento de workflows. Na Seção 3, são apresentadas iniciativas de pesquisa que implementam estratégias de escalonamento de *workflows*. Na Seção 4, são apresentadas as estratégias de escalonamento propostas. Na Seção 5, é apresentada a implementação das estratégias propostas no ambiente de simulação. Na Seção 6, é apresentada a avaliação das estratégias MCA-HEFT e DMCA-HEFT. Na Seção 7, é apresentada a conclusão e trabalhos futuros. Na Seção 0, são apresentadas as referências bibliográficas.

2. Execução e Escalonamento de *Workflows* em Ambientes de Processamento Distribuído

O uso de *workflows* para execução de aplicações computacionalmente intensivas é essencial para facilitar o desenvolvimento científico em diversas áreas devido a dois aspectos. O primeiro aspecto é a possibilidade de automação da execução de aplicações abstraindo a complexidade de uso do ambiente. O segundo aspecto é a possibilidade de melhorar o desempenho de tais aplicações utilizando diversos recursos simultaneamente para uma mesma aplicação em ambientes de processamento paralelo e distribuído. Para que a execução seja eficiente em tais ambientes, é essencial realizar um escalonamento adequado dos *workflows*. Nessa seção, são apresentados os conceitos relacionados com execução e escalonamento de *workflows* utilizados nesta tese.

2.1. Definições

Na presente seção serão apresentadas as definições utilizadas nesta tese.

2.1.1. Ambientes Distribuídos para Execução de Tarefas

Um aspecto essencial para execução de aplicações computacionalmente intensivas com bom desempenho é a utilização de múltiplos recursos simultaneamente para execução de uma mesma tarefa. As primeiras iniciativas que surgiram para agregar recursos distribuídos foram os *clusters*, que agregam uma grande quantidade de recursos computacionais usando uma rede local (BUYYA, 1999). A disponibilidade de supercomputadores em diversos domínios administrativos e a crescente necessidade de execução de aplicações computacionalmente intensivas motivou o desenvolvimento de infraestruturas, que agregam recursos de alto-desempenho por meio de redes de longo alcance WAN (Wide Access Network), chamadas grades

computacionais (FOSTER, 2001).

As grades computacionais permitem agregar quaisquer tipos de recursos, tais como, bancos de dados, programas, arquivos, dispositivos móveis, sensores, recursos computacionais, e até mesmo *clusters* de computadores, considerando que os recursos podem ser montados em múltiplos domínios administrativos. Esse conjunto de recursos agregados são identificados como organizações virtuais *VOs* (*Virtual Organizations)* (HEY; TREFETHEN, 2005).

Devido à heterogeneidade do ambiente de grade computacional e integração de recursos entre múltiplos domínios administrativos, a utilização de uma grade computacional é complexa. Nesse sentido, foram desenvolvidos *middlewares* para prover uma interface padrão de acesso aos recursos abstraindo a complexidade de uso de tais ambientes (COMITO; TALIA; TRUNFIO, 2005).

A seção a seguir descreve a estrutura de ambientes de grade computacionais montados a partir da agregação de múltiplos *clusters*.

2.1.1.1. Ambientes de Grade Computacional Compostos por Clusters

Uma forma de montar um ambiente de grade computacional com foco na obtenção de desempenho é agregar recursos para computação paralela, tais como supercomputadores e *clusters* (BUYYA; BAKER, 2001)(NANDAGOPAL; GOKULNATH; UTHARIARAJ, 2010).

Os componentes da estrutura de uma grade computacional montada a partir da agregação de múltiplos *clusters* são gerenciadores de recursos locais e um *middleware*. Os recursos dos *clusters* são controlados por gerenciadores de recursos locais, tais como, Condor (COUVARES et al., 2007), OpenPBS [2] (Portable Batch Systems) , Slurm (BALLE; PALERMO, 2008), entre outros, e um *middleware* que é responsável por prover acesso centralizado aos gerenciadores de recursos locais dos múltiplos *clusters*.

A Figura 1 mostra um exemplo de uma grade computacional montada utilizando *clusters* e servidores. Nesse cenário, o ambiente está conectando

[2] http://www.mcs.anl.gov/research/projects/openpbs/

duas organizações virtuais.

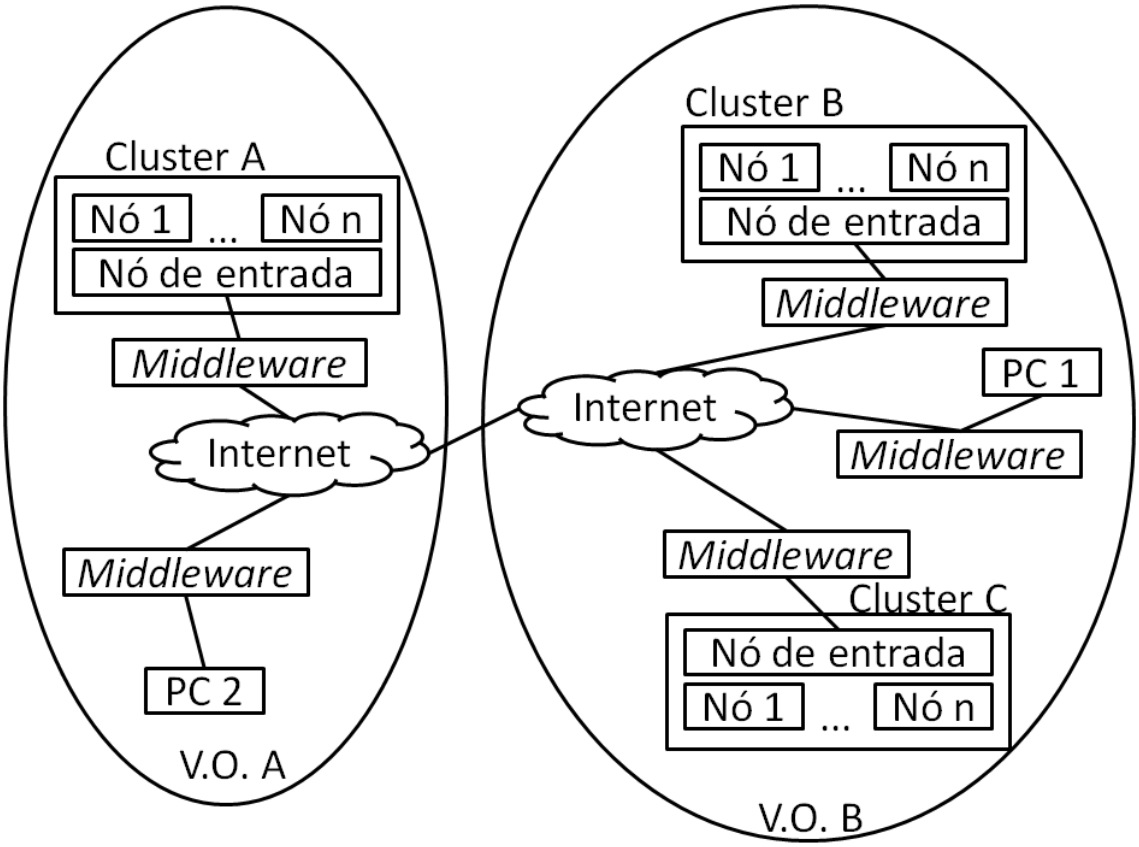

Figura 1. Um Exemplo de um Ambiente de Grade Computacional.

Diversos ambientes de grade computacional que agregam *clusters* têm sido desenvolvidos em diversos países, tais como, o Distributed ASCI (Advanced School for Computing and Imaging) Supercomputer 3 (DAS-3) [3] e o Distributed ASCI Supercomputer 4 (DAS-4) [4] localizados na Holanda, O Grid Computing for Particle Physics (GridPP) [5] localizado no Reino Unido e O Grid'5K [6] localizado na França.

Um ambiente de grade computacional que agrega múltiplos *clusters* é adequado para execução de tarefas com paralelismo interno de maneira escalável. Nesse contexto, o protocolo MPI (Message-Passing Interface) (GROPP; LUSK; SKJELLUM, 1994) tem sido bastante utilizado para paralelizar aplicações. Os *frameworks* que implementam o protocolo MPI, tais como,

[3] DAS-3 Site do projeto: http://www.cs.vu.nl/das3/index.shtml
[4] DAS-4 Site do projeto: http:// http://www.cs.vu.nl/das4/
[5] GridPP Site do projeto: http://www.gridpp.ac.uk
[6] Grid5K Site do projeto: http://www.grid5000.fr

OpenMPI (GRAHAM et al., 2006) e LAM/MPI (SQUYRES; LUMSDAINE, 2003) são em geral implementados para prover suporte ao desenvolvimento de aplicações que serão executadas em um único *cluster*.

Uma abordagem que vem sendo pesquisada é a possibilidade de executar uma tarefa com paralelismo interno usando recursos de múltiplos *clusters*. Nesse sentido, os *frameworks* que implementam o protocolo MPI, tais como, MPICH-VMI (PANT; JAFRI, 2004), MPICH Madeleine (AUMAGE; MERCIER, 2003), PACX-MPI (KELLER et al., 2003), MPICH-G2 (KARONIS; TOONEN; FOSTER, 2003) e o GridMPI (MASSETTO et al., 2011), proveem suporte para realizar encaminhamento de mensagens entre redes heterogêneas.

As principais motivações para executar uma tarefa com paralelismo interno em múltiplos *clusters* são: (i) aplicações que precisam de uma quantidade de recursos que podem não estar disponíveis em um único *cluster*. (ii) usuários que buscam reduzir o tempo de resposta utilizando recursos de múltiplos *clusters* (iii) usuários podem ter tempo de resposta menor utilizando fragmentos de múltiplas filas de escalonamento. Um problema é a possível sobrecarga causada pela comunicação *inter-cluster*. A maior parte dos estudos mostra que 25% do tempo total de execução é uma sobrecarga tolerável. Tais estudos consideram tarefas com paralelismo interno de maneira independente diferente do que acontece em um *workflow* (BUCUR; EPEMA, 2007; ERNEMANN et al., 2002; JONES et al., 2005; MOHAMED; EPEMA, 2005; NETTO; BUYYA, 2010, 2008; SNELL et al., 2000).

2.1.1.2. Recursos

No contexto desta tese está sendo considerado um ambiente heterogêneo composto por diversos *clusters*.

Um *cluster* c é composto por um conjunto de recursos homogêneos $R = \{r_1, r_2, \dots, r_n\}$, que possui uma capacidade em MIPS (Million Instructions Per Second). Em cada *cluster* uma rede interna interliga todos os recursos.

Um dos recursos do *cluster* é chamado de nó de entrada e tem como objetivo prover acesso entre os recursos desse *cluster* e recursos de outros *clusters*. Uma rede de conexão *inter-cluster* interliga todos os nós de entrada.

A rede interna de cada *cluster* possui uma largura de banda e uma latência, ambos em MB/s. A rede conexão inter-*cluster* também possui uma largura de banda e uma latência.

Nesse ambiente uma tarefa pode ser executada em qualquer subconjunto de recursos montados em um *cluster* ou em múltiplos *clusters*.

2.1.2. Tarefa

Uma tarefa representa uma aplicação que deve ser executada em um ou mais recursos computacionais.

De acordo com o modelo de programação utilizado, uma tarefa pode não apresentar paralelismo ou pode apresentar paralelismo que pode ser implícito ou explícito. O paralelismo implícito consiste em partes do código que podem ser trivialmente divididas para serem executadas em mais de um recurso simultaneamente. O paralelismo explícito consiste na utilização de técnicas de programação paralela para dividir a computação de uma tarefa em diferentes partes para ser executada em múltiplos recursos simultaneamente. A utilização de tais técnicas de paralelização não deve alterar a computação que será realizada.

De acordo com o paralelismo apresentado ou ausência de paralelismo as tarefas podem ser de dois tipos: sequenciais ou com paralelismo interno.

- As tarefas sequenciais não apresentam paralelismo explícito nem paralelismo implícito. Dessa maneira, uma tarefa sequencial é executada em um único recurso.
- As tarefas com paralelismo interno apresentam paralelismo implícito ou explícito. Dessa maneira, podem ser executadas utilizando múltiplos recursos e podem dividir a computação entre os recursos.

As tarefas com paralelismo interno podem ser rígidas, moldáveis ou maleáveis (FEITELSON et al., 1997):

- Tarefas rígidas: São executadas em um número fixo de recursos;
- Tarefas moldáveis: Podem ser executadas com qualquer número de

recursos, que são reservados para a tarefa até o fim da execução;

- Tarefas maleáveis: podem ter a quantidade de recursos definida para execução modificada durante a execução.

No contexto desta tese uma tarefa possui uma quantidade de processamento e uma quantidade de dados trafegada ao longo da execução da tarefa. No caso de tarefas sequenciais o tráfego de dados é realizado por um único recurso e não gera custo de comunicação na rede. No caso das tarefas com paralelismo interno o tráfego de dados é realizado entre os recursos que executam a tarefa e geram custo na rede.

- A quantidade de processamento é medida em milhões de instruções (MI).
- A quantidade de dados trafegada é medida em MegaBytes (MB).

A computação que uma tarefa com paralelismo interno deve realizar é dividida conceitualmente em duas partes: uma parte da computação que pode ser dividida para ser executada em diversos recursos, que é chamada porção paralela e uma parte que deve ser necessariamente executada em um recurso, que é chamada porção sequencial. Em alguns casos é possível que toda a computação da tarefa possa ser dividida entre os recursos.

A eficiência de uma tarefa com paralelismo interno é uma medida do grau de aproveitamento dos recursos. Com base nessa medida é possível estimar a quantidade de recursos adequada para a execução da tarefa com bom nível de desempenho. A eficiência de uma tarefa com paralelismo interno é limitada pela porção paralela.

A estimativa da porção paralela de uma tarefa pode ser definida a partir do *speedup* (S_p) da tarefa. De acordo com a lei de Amdahl o *speedup* de uma tarefa com paralelismo interno é limitado pela porcentagem sequencial do código, independente da quantidade de recursos usados.

O *speedup* de uma tarefa é definido de acordo com a equação (1), onde T_1 é o tempo de execução utilizando 1 recurso e T_p é o tempo de execução utilizando p recursos.

$$S_p = \frac{T_1}{T_p} \quad (1)$$

O cálculo da porção paralela é baseado no *speedup* máximo, que é obtido considerando uma quantidade específica de recursos. Para obter o *speedup* de todas as tarefas de um workflow, pode ser utilizado um *cluster* de referência.

A porção paralela do código pf_t é estimada usando a equação (2), que é baseada no *speedup* obtido, considerando uma quantidade específica de recursos.

$$pf_t = \frac{P * (1 - S_p)}{S_p * (1 - P)} \quad (2)$$

2.1.3. Conjuntos de Tarefas

Uma aplicação pode ser composta por uma ou mais tarefas, sendo que um conjunto de tarefas pode conter tarefas de dois tipos:

- Tarefas independentes: é identificado pelo termo (*Bag of Tasks*) (CIRNE et al., 2003), e a execução das tarefas pode ser realizada em qualquer ordem.
- Tarefas interdependentes: a execução das tarefas deve seguir uma ordem definida pela relação de precedência entre as tarefas.

Nesta tese abordamos o problema de escalonamento de tarefas interdependentes.

A comunicação entre as tarefas para definir as precedências possui um custo associado. Tal custo pode provocar um grande impacto no desempenho da execução dos *workflows*. Nesse sentido, diversas estratégias de escalonamento buscam minimizar o tempo dessas transferências.

As estratégias propostas nesta tese realizam o escalonamento de *workflows*

sem levar em conta os custos de transferência entre as tarefas. Porém, poderiam ser utilizadas em conjunto com estratégias de escalonamento, que tem como objetivo minimizar custos de transferência entre as tarefas.

2.1.4. Workflow

O termo *workflow* surgiu para identificar as tecnologias para automação de processos de negócio, no qual documentos, informações e tarefas são passadas de um colaborador para outro, de acordo com um fluxo pré-determinado e de acordo com um conjunto de regras (WORKFLOW MANAGEMENT COALITION, 1999).

No contexto de sistemas distribuídos o conceito de *workflow* foi adotado como paradigma para estruturar e executar aplicações científicas complexas. Tais aplicações envolvem o uso de diversas ferramentas de simulação e análise, que devem ser executadas repetidamente e de maneira coordenada (YU; BUYYA, 2005). Devido à natureza científica das aplicações foi adotado o termo *workflow* científico. Nessa tese é adotado o termo *workflow* no lugar de *workflows* científicos.

Um *workflow* pode ser representado como um grafo dirigido que contém ciclos (*Directed Cyclic Graph* - DCG*)* ou por um grafo dirigido acíclico (*Directed Acyclic Graph* - DAG*)*. A diferença entre as duas representações é que na representação de *workflow* utilizando grafos com ciclos a sequência de execução do *workflow* pode ter diversos caminhos possíveis, enquanto que a representação de um *workflow* utilizando grafos acíclicos apresenta apenas uma sequência de execução.

No contexto de aplicações científicas os *workflows* são comumente representados como um DAG, que é definido da seguinte maneira $DAG = (N, E)$, onde $\mathrm{N} = \{\mathrm{n_i} \mid \mathrm{i} = 1, \ldots, \mathrm{N}\}$ é o conjunto de vértices representando as tarefas do *workflow* (*job)*, e $\mathrm{E} = \{\mathrm{e_{ij}} \mid (\mathrm{i}, \mathrm{j}) \in \{1, \ldots, \mathrm{N}\}\, \mathrm{x}\, \{1, \ldots, \mathrm{N}\}\}$ é o conjunto de arcos que representam as dependências de dados entre as tarefas (SINGH; VOUK, 1996). As estratégias propostas nesta tese consideram que os *workflows* submetidos são representados por um DAG.

Um *workflow* é composto por diversas tarefas que devem ser executadas de

acordo com as dependências de arquivos entre as tarefas. Nesse sentido, é necessário um componente para controlar a execução do *workflow* como uma camada acima do *middleware*. Tais componentes são chamados Sistemas Gerenciadores de *Workflows* Científicos ScWFMS (*Scientific Workflow Management Systems*) (YU; BUYYA, 2005) (DEELMAN et al., 2003).

2.1.5. Níveis da Tarefa em um Worfklow

Em um *workflow* do tipo *fork-join* as tarefas são organizadas em níveis. De tal forma que tarefas no mesmo nível podem ser executadas simultaneamente. Cada nível do *workflow* pode ser composto por tarefas sequenciais e tarefas com paralelismo interno, conforme mostrado na Figura 2.

O objetivo de identificar os níveis do *workflow* é realizar uma divisão dos recursos para cada tarefa por nível de maneira independente. Dessa forma, todos os recursos disponíveis podem ser alocados para as tarefas de cada nível.

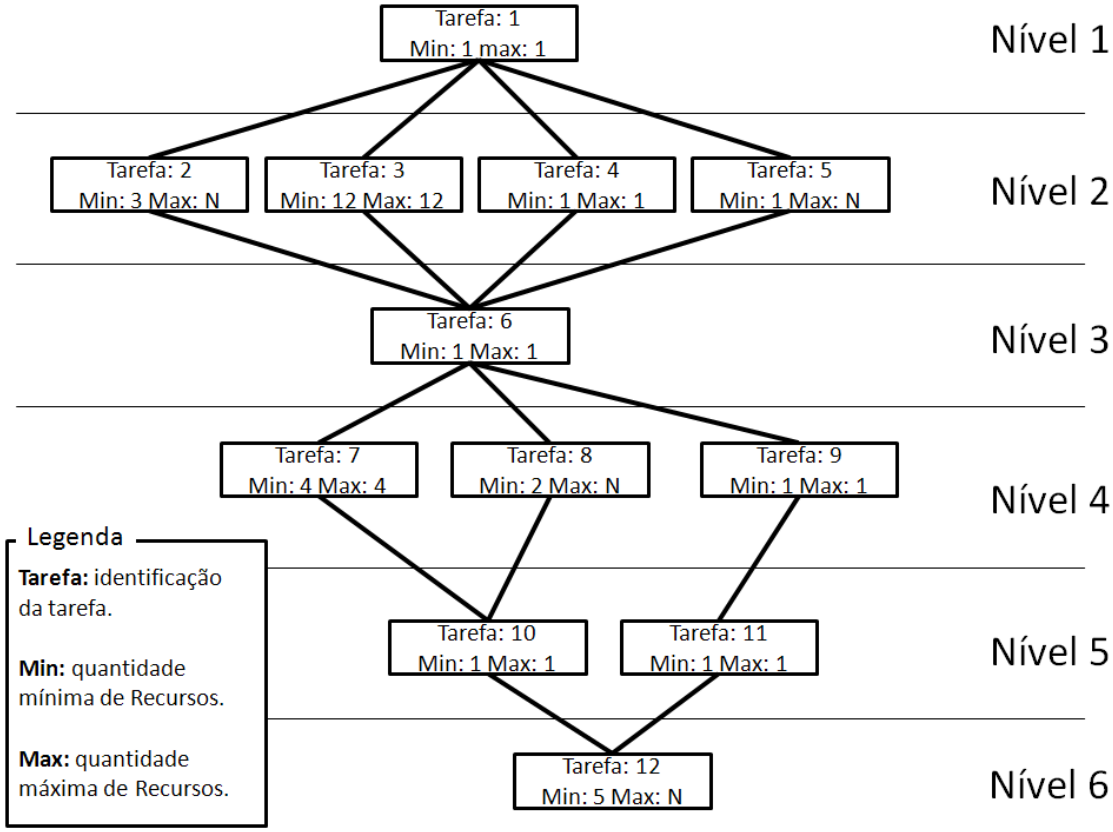

Figura 2. Exemplo de um *Workflow* com Tarefas com Paralelismo Interno e Tarefas Sequenciais, Composto por Seis Níveis.

2.1.6. Caminho Crítico

O caminho crítico é uma medida usada para estimar o tempo de execução mínimo de um workflow, que é o caminho crítico do workflow, e o impacto do tempo de execução de cada tarefa no tempo total de execução do workflow, que é o caminho crítico de uma tarefa.

- Caminho crítico de um workflow: é o caminho que possui maior custo entre a primeira tarefa e a última tarefa. O tempo total de execução de um *workflow* será igual ou maior do que o tempo para executar cada tarefa do caminho crítico no melhor conjunto de recursos.
- Caminho crítico de uma tarefa: é definido como o caminho que passa pela tarefa que possui maior custo, que não é necessariamente o caminho crítico do workflow. O cálculo do caminho crítico de uma tarefa pode ser realizado de duas formas. A primeira forma é realizar o cálculo do caminho de uma tarefa t considerando desde a primeira tarefa até a tarefa t. A segunda forma é realizar o cálculo do caminho a partir da tarefa t até a última tarefa do workflow. Nessa tese foi adotada a segunda forma.

A Figura 3 mostra um exemplo de um *workflow* com 14 tarefas com o caminho crítico do *workflow* em destaque. O caminho composto pelas tarefas 1, 5, 9,13 e 14 representa o caminho crítico do workflow. Cada tarefa está representada na figura por meio de três valores: o primeiro é a identificação da tarefa, o segundo é o custo computacional em milhões de instruções (MI) e o terceiro é a quantidade de dados trafegada internamente em MegaBytes (MB).

A tarefa 8 não está no caminho crítico do workflow, no entanto, é possível verificar que há dois caminhos possíveis entre a tarefa 8 e a última tarefa. O primeiro é o caminho: 8, 11 e 14, e o segundo é 8, 12 e 14. O segundo caminho possui maior custo de execução, dessa forma, o custo acumulado do caminho crítico da tarefa 8 representa o impacto dessa tarefa no tempo total do workflow.

No contexto desta tese, adotamos o caminho crítico da tarefa nas

estratégias de escalonamento propostas.

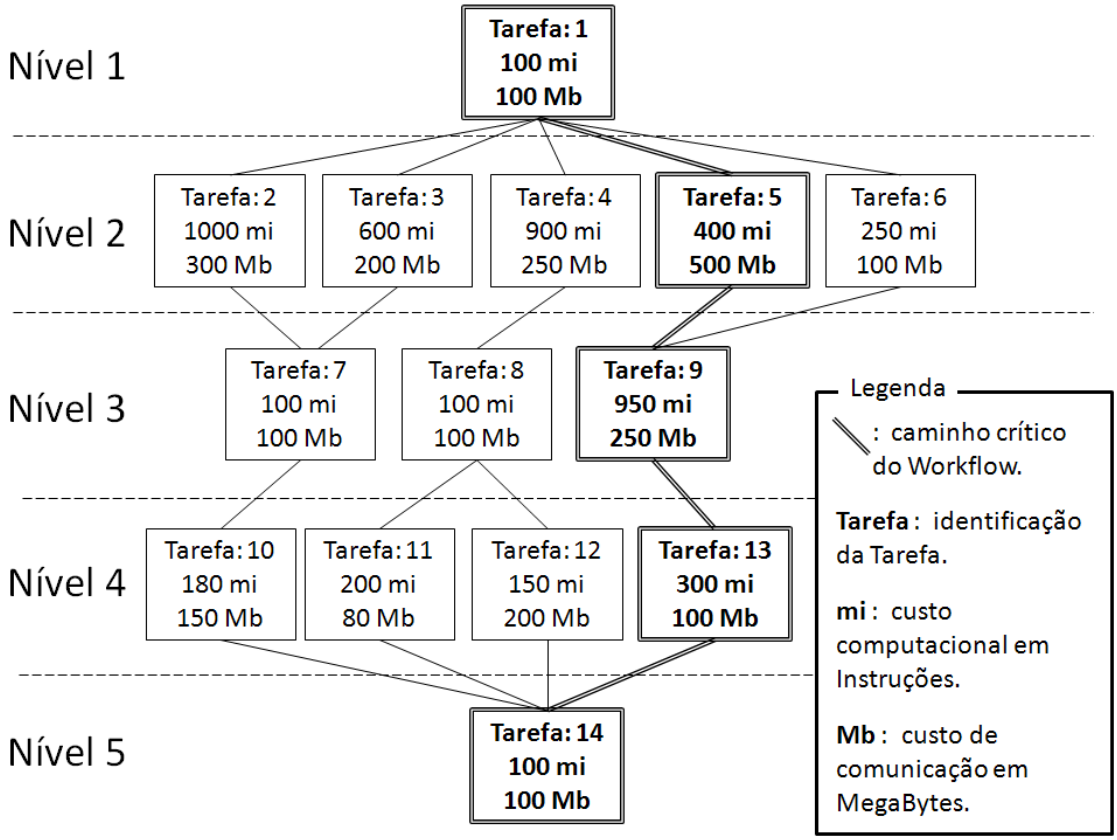

Figura 3. Exemplo de um *Workflow* com o Caminho Crítico em Destaque (Tarefas: 1, 5, 9, 13, 14).

2.2. Escalonamento de Tarefas

A utilização de técnicas de escalonamento é essencial para que aplicações computacionalmente intensivas sejam executadas com bom desempenho em ambientes de processamento distribuído. O escalonamento de tarefas é um processo que consiste em fazer um mapeamento de um conjunto de tarefas para um conjunto de recursos computacionais, com o objetivo de otimizar algum critério (CASAVANT; KUHL, 1988), que pode ser por exemplo executar o conjunto de tarefas no menor tempo possível, executar com o menor custo possível, executar as tarefas dentro de um prazo limite, entre outras.

As estratégias para realizar o escalonamento de tarefas abordam o problema de acordo com três aspectos: o conjunto de tarefas a ser escalonado, o ambiente computacional no qual as tarefas serão executadas, e os objetivos

a serem minimizados (PINEDO, 2008).

Para definir o objetivo que deve ser minimizado, diversas métricas podem ser utilizadas. Duas métricas comumente utilizadas são: o *makespan* e o tempo de resposta. O *makespan* corresponde ao intervalo de tempo desde o horário de início da execução da primeira tarefa até o horário do término da execução da última tarefa. O tempo de resposta é definido com o tempo desde o momento em que o usuário submete uma requisição até o momento que a resposta é recebida.

O *makespan* é utilizado quando é considerado um cenário no qual o conjunto de tarefas e o estado dos recursos são conhecidos no início do escalonamento. Nesse sentido, o *makespan* mede o tempo de execução do conjunto de tarefas submetidas.

O tempo de resposta é utilizado quando é considerado um cenário no qual o escalonador entra em execução e aguarda as submissões das tarefas, que podem ocorrer em diferentes momentos. Nesse sentido, o tempo de resposta mede o tempo de execução de cada tarefa.

Em um ambiente de produção o tempo de execução de cada tarefa pode variar de acordo com diversos aspectos, tais como, variação de desempenho e falhas. Nesse sentido, o tempo de resposta médio é uma métrica que permite medir a eficiência da estratégia de escalonamento considerando diversos eventos.

2.2.1. Escalonamento de Tarefas com Paralelismo Interno

O escalonamento de tarefas com paralelismo interno independentes em ambientes de grade computacional é bastante investigado no contexto de duas áreas de pesquisa: co-alocação de recursos e balanceamento de carga.

As técnicas de co-alocação de recursos tem como objetivo prover acesso simultâneo a recursos controlados por múltiplos domínios administrativos para execução de uma única tarefa com paralelismo interno (NETTO; BUYYA, 2010).

O cenário considerado pelas técnicas de co-alocação são ambientes compostos por múltiplas filas de execução, que recebem submissões de

tarefas em diferentes momentos. Para cada tarefa é definido um conjunto de recursos mínimo para execução, que é informado pelos usuários no momento da submissão. Nesse sentido, as tarefas aguardam na fila para serem executadas, e quando estão disponíveis para serem executadas, devem aguardar os recursos necessários para sua execução.

As técnicas utilizadas para otimizar o escalonamento de tarefas com paralelismo interno independentes em tais ambientes investigam, entre outros aspectos, a utilização de diferentes políticas de prioridade em uma fila de tarefas e de reserva de recursos.

O escalonamento de tarefas com paralelismo interno em ambientes de processamento distribuído pode gerar planos de escalonamento, no qual os recursos podem ficar ociosos em alguns momentos. A técnica de *backfilling* consiste em identificar a existência desses espaços ociosos e adiantar tarefas da fila de execução para esses espaços (SRINIVASAN et al., 2002).

O balanceamento de carga é uma técnica de escalonamento que consiste em fazer um mapeamento de tarefas em um conjunto de recursos buscando homogeneizar o uso dos recursos. Uma técnica muito usada é a transferência de carga entre recursos para aumentar o desempenho das aplicações (LI; LAN, 2005) (RUDOLPH; SLIVKIN-ALLALOUF; UPFAL, 1991).

2.2.2. Escalonamento de Workflows

Nesta tese são definidos conjuntos de tarefas de entrada como *workflows*, que são caracterizados como um conjunto de tarefas que possuem dependências de dados entre si. O ambiente é composto por recursos heterogêneos. O objetivo a ser minimizado é o makespan, quando é avaliado um *workflow* isoladamente no conjunto de *workflows*, e o tempo de resposta, quando é avaliado o comportamento dinâmico do sistema.

Em ambientes de grade computacionais compostos por múltiplos *clusters*, diversas características inerentes a infraestrutura influenciam o tempo de execução de um workflow, tais como, poder computacional dos recursos, largura de banda e latência de comunicação entre os recursos de um mesmo *cluster* e entre recursos de diferentes *clusters*. Tais ambientes são utilizados

por diversos usuários que submetem *workflows* que não são conhecidos previamente pelo escalonador e em diferentes momentos. Nesse sentido, o ambiente apresenta um comportamento dinâmico que deve ser levado em conta pelo escalonador.

Com base nas características dos recursos, das tarefas e do objetivo a ser minimizado é necessário definir um algoritmo para mapear o conjunto de tarefas para os recursos, de maneira que sejam produzidos planos de escalonamento que apresentem bom desempenho. Esse problema é considerado NP-Completo em termos de complexidade (PINEDO, 2008). Nesse sentido, diversas estratégias têm sido propostas considerando diferentes critérios e políticas.

Uma classificação feita entre as estratégias de escalonamento de *workflow* com relação ao processo de escalonamento são (WIECZOREK; HOHEISEL; PRODAN, 2008): escalonamento *just in-time*, antecipado ou híbrido.

O escalonamento antecipado é também chamado de escalonamento estático. Esse modo considera que os *workflows* a serem escalonados são conhecidos e o plano de escalonamento é montado antes de entrarem em execução.

O escalonamento *just in-time* é também chamado de escalonamento dinâmico. Nesse modo, o escalonamento é realizado somente para as tarefas que estão prontas para entrar em execução, dessa forma, o escalonamento é adiado o máximo possível.

O modo híbrido é uma combinação dos dois modos. Primeiro o escalonador realiza um escalonamento antecipado com base nas informações disponíveis e altera o plano em decorrência de diversos eventos, tais como, mudanças de carga dos recursos, mudanças de disponibilidade ou submissão de novos *workflows*.

2.2.2.1. Escalonamento Estático de Workflows

O escalonamento estático considera que os *workflows* que serão escalonados são conhecidos e que o ambiente é dedicado à execução desses *workflows*.

De modo geral existem quatro classes de algoritmos de escalonamento de *workflows* baseados em heurística: *individual task scheduling*, *list scheduling*, *clustering* e *duplication based scheduling*, além dos algoritmos baseados em meta-heurística (YU; BUYYA; RAMAMOHANARAO, 2008).

Os algoritmos baseados em meta-heurísticas são métodos para resolver, de maneira genérica, problemas de otimização. De forma geral, utilizam combinação de escolhas aleatórias de soluções e baseados nessas escolhas realizam iterações para escolha de melhores resultados. A cada iteração o algoritmo descobre informações sobre o problema e utiliza essa informação para buscar melhores soluções.

A classe de algoritmo *individual task scheduling* leva em conta, para o escalonamento de *workflows*, apenas as dependências de tarefas.

As classes *clustering* e *duplication based scheduling* foram desenvolvidas com foco em diminuir o tempo de transferência de arquivos entre tarefas. A classe *clustering* tem como objetivo agrupar tarefas que possuem um custo alto de comunicação entre si, em um conjunto de recursos que possuam grande capacidade para realizar as transferências de arquivos, enquanto que na classe *duplication* a tarefa é duplicada em recursos ociosos para aumentar a chance de obter melhor desempenho.

A classe *list-scheduling* é um dos modelos mais utilizados de escalonamento estático de *workflows*. Tal classe realiza o escalonamento em dois passos. No primeiro, é associada uma prioridade de escalonamento de cada tarefa, chamada *rank*. No segundo passo as tarefas são mapeadas para os melhores recursos em ordem de prioridade.

O *rank* de uma tarefa é um valor atribuído à cada tarefa do *workflow* que indica a ordem seguida para realizar o escalonamento de cada tarefa. Tal valor indica o peso do tempo de execução da tarefa comparado com o tempo de execução do *workflow* inteiro. Quanto maior o valor do *rank* maior o peso computacional da tarefa em relação ao workflow. Nesse sentido, as tarefas são escalonadas por ordem decrescente de *rank.*

O cálculo do *rank* é feito dessa maneira para que as tarefas que estejam no caminho crítico do *workflow* sejam alocadas para uma quantidade maior de recursos e escalonadas para os melhores recursos, pois ao minimizar o tempo

de execução das tarefas no caminho crítico, o *makespan* do *workflow* é minimizado.

2.2.2.2. Escalonamento de Workflows em Ambientes Dinâmicos

Os recursos em ambientes de grades computacionais e ambientes *multi-cluster* são caracterizados por um comportamento dinâmico, pois são utilizados por múltiplos usuários, apresentam falhas, heterogeneidade, entre outros fatores. Dessa forma, o escalonamento de tarefas em tais ambientes é feito sem que se tenha conhecimento a respeito de quando os *workflows* serão submetidos, da disponibilidade dos recursos e do desempenho dos recursos.

Existem duas abordagens para realizar o escalonamento em ambientes dinâmicos. A primeira abordagem é tomar a decisão de escalonamento somente quando a tarefa estiver pronta para execução. A segunda abordagem consiste em realizar um escalonamento considerando que o ambiente é estático, e refazer esse mapeamento durante a execução do *workflow* (WIECZOREK; HOHEISEL; PRODAN, 2008).

O processo de refazer o plano de escalonamento chamado re-escalonamento, pode ser iniciado por diferentes eventos durante a execução do workflow, tais como, eventos de mudanças de disponibilidade ou desempenho de recursos, submissão de tarefas fora do controle do escalonador, submissão de novos *workflows* ou quando um conjunto de tarefas de um ou mais *workflows* têm suas dependências atendidas e podem ser executadas.

A seção a seguir descreve o processo de mapeamento de tarefas, que é parte do processo de escalonamento de *workflows*.

2.2.2.3. Mapeamento de Tarefas

O mapeamento de tarefas deve levar em conta as características das tarefas, que podem ser sequenciais, rígidas, moldáveis ou maleáveis. As tarefas sequenciais são sempre mapeadas para um recurso, as tarefas rígidas

são mapeadas para uma quantidade fixa de recursos, as tarefas moldáveis podem ser mapeadas para uma quantidade de recursos e manter essa quantidade até o fim da execução e as tarefas maleáveis pode utilizar uma quantidade variável de recursos durante a execução da tarefa.

Os *workflows* podem possuir ao longo dos seus níveis conjuntos de tarefas sequenciais que podem ser executadas simultaneamente. Tal característica é chamada paralelismo inter-tarefa. As estratégias exploram essa possibilidade mapeando cada tarefa de um mesmo nível para diferentes recursos.

Em um conjunto de tarefas de um mesmo nível de um *workflow* podem estar presentes tarefas sequenciais e tarefas com paralelismo interno (WIECZOREK; HOHEISEL; PRODAN, 2008). Nesse contexto, existe a possibilidade de executar diversas tarefas simultaneamente, sendo que cada tarefa pode ser mapeada para uma quantidade diferente de recursos. Tal característica é chamada paralelismo intra-tarefa. As estratégias de escalonamento exploram o paralelismo intra-tarefa realizando um balanceamento entre a quantidade de recursos alocada para as tarefas presentes em um mesmo nível (N'TAKPE; SUTER; CASANOVA, 2007) (STANZANI; SATO, 2012).

O balanceamento de alocação de quantidade de recursos entre tarefas com paralelismo interno em um mesmo nível pode contribuir para que o tempo total de execução do *workflow* seja minimizado (CASANOVA; DESPREZ; SUTER, 2010). A Figura 4 mostra dois exemplos de escalonamento de um mesmo *workflow* composto por tarefas moldáveis. Nesse exemplo, as tarefas B, C e D podem ser executadas simultaneamente com quaisquer quantidades de recursos, limitadas à quantidade total de recursos do ambiente. No segundo plano de escalonamento a tarefa D é executada em um tempo menor do que no primeiro plano. O segundo plano apresenta um tempo de execução maior para a tarefa B, porém apresenta um tempo menor de execução do *workflow* completo.

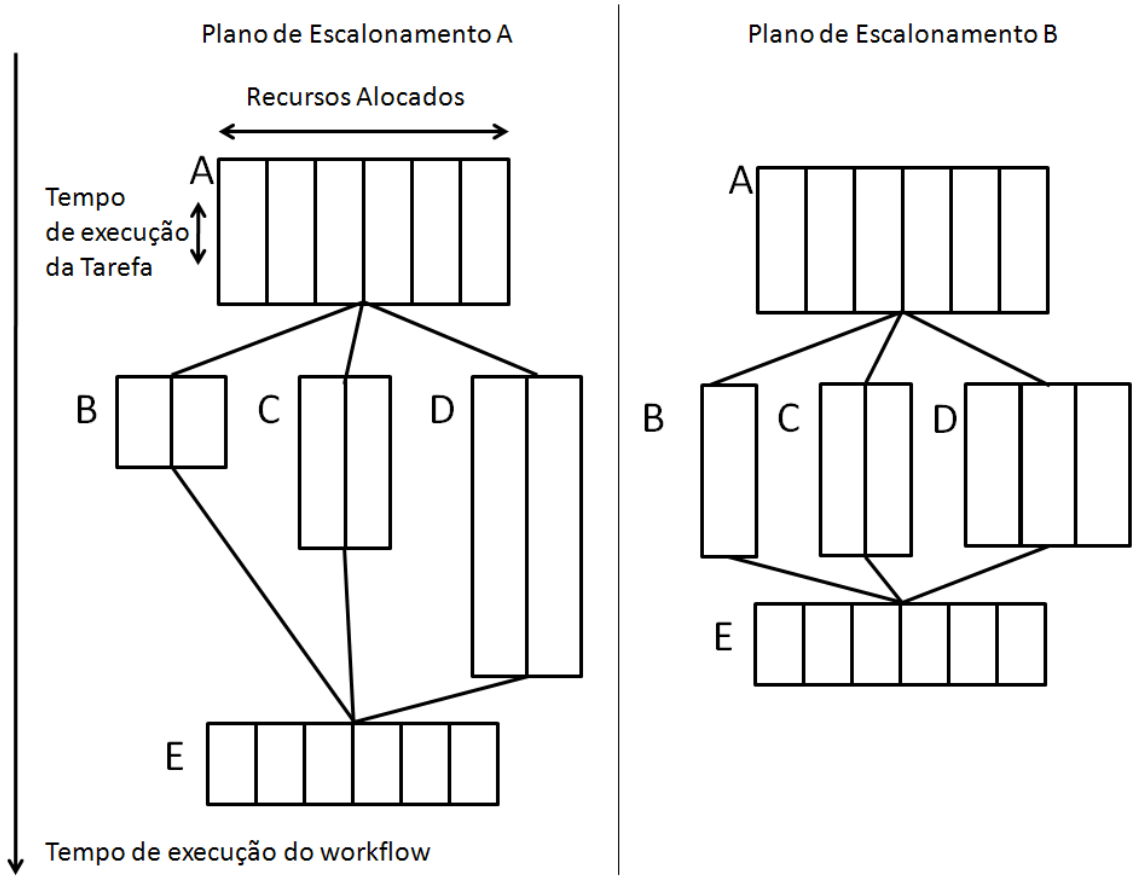

Figura 4. Tempo de Execução de Dois Planos de Escalonamento (A e B) de um *Workflow* composto por Tarefas com Paralelismo Interno.

A quantidade de recursos que deve ser alocada para cada tarefa é definida de acordo com as características da tarefa. Um critério para definir tal alocação é considerar a eficiência da tarefa em conjunto com os custos de comunicação, e de computação, para estimar o tempo de execução de uma tarefa com paralelismo interno de acordo com a quantidade de recursos alocada para a tarefa.

2.3. Conclusão

Para que a execução de múltiplos *workflows* computacionalmente intensivos em ambientes de grade computacional seja realizada com desempenho adequado, é necessário utilizar estratégias de escalonamento que realizam o escalonamento estático e seja capaz de adequar tal escalonamento na ocorrência de submissão de novos *workflows*.

O caminho crítico de cada tarefa e o balanceamento de distribuição de recursos entre tarefas sequenciais e tarefas com paralelismo de um mesmo nível do *workflow* são aspectos essenciais, que são explorados de diferentes formas pelas estratégias de escalonamento de *workflows* e estratégias de escalonamento de tarefas independentes. O capítulo a seguir mostra diversas estratégias que exploram tais aspectos.

3. Estratégias de Escalonamento de Workflows

As estratégias propostas nesta tese exploram três problemas. O primeiro problema é o escalonamento estático de *workflows* considerando o paralelismo intra-tarefa. O segundo problema é o escalonamento de *workflows* em ambientes dinâmicos. O terceiro problema é verificar a possibilidade de executar cada tarefa com paralelismo interno em recursos de múltiplos *clusters*.

Nessa seção serão descritas estratégias de escalonamento de *workflows* que abordam os problemas explorados pelas estratégias propostas nesta tese.

Na Seção 3.1 serão descritas as estratégias de escalonamento de estático de *workflows* que exploram o primeiro problema. Boa parte das estratégias que consideram o paralelismo inter-tarefa foram desenvolvidas a partir das estratégias que consideram apenas o paralelismo inter-tarefa. Nesse sentido, serão apresentadas estratégias que consideram o paralelismo inter-tarefa e estratégias que consideram o paralelismo intra-tarefa.

Na Seção 3.2 serão descritas as estratégias que exploram o segundo problema.

As estratégias de escalonamento de *workflows* não exploram o terceiro problema, devido a possíveis sobrecargas de rede. Porém, tal possibilidade é largamente investigada por estratégias de escalonamento de tarefas com paralelismo interno independentes em ambientes dinâmicos. Tais estratégias são descritas na Seção 3.3.

Na Seção 3.4 é mostrada uma análise das estratégias apresentadas.

3.1. Estratégias de Escalonamento Estático de Workflows

As seções a seguir mostram as iniciativas de trabalhos que desenvolvem estratégias de escalonamento estático de *workflows*. Primeiramente, na Seção 3.1.1 serão apresentadas as estratégias que realizam o escalonamento de *workflows*, que apresentam paralelismo inter-tarefa. Em seguida, na Seção 3.1.2 serão apresentadas as estratégias que realizam o escalonamento de *workflows* que apresentam paralelismo intra-tarefa.

3.1.1. Estratégias de Escalonamento Inter-tarefa

As estratégias apresentadas nesta seção exploram o paralelismo inter-tarefa apresentados pelos *workflows*. Nesse contexto, as estratégias consideram que as tarefas dos *workflows* são sequenciais, e sempre são mapeadas para um único recurso.

3.1.1.1. HEFT (Heterogeneous Earliest Finish Time)

A estratégia HEFT (*Heterogeneous Earliest Finish Time*) (TOPCUOUGLU; HARIRI; WU, 2002) (TOPCUOGLU; HARIRI; WU, 1999) tem como objetivo produzir planos de escalonamento para *workflows* em ambientes de processamento distribuído composto por recursos heterogêneos, aliando simplicidade e obtenção de bons resultados. O HEFT é baseado no modelo *list-scheduling,* dessa forma, o plano de escalonamento do HEFT é gerado seguindo dois passos: o primeiro passo chamado cálculo de *rank*, tem como objetivo associar uma prioridade de escalonamento para cada tarefa, o segundo passo chamado seleção de processadores, tem como objetivo mapear cada tarefa para o recurso que apresenta o menor tempo de término (*Earliest Finish Time*) seguindo a ordem de escalonamento definida pela prioridade de cada tarefa.

A definição de prioridade é realizada seguindo dois passos:

- O primeiro passo consiste na definição de pesos da tarefa e de cada comunicação entre tarefas dependentes do workflow.
    - O peso de cada tarefa é o custo médio de computação da tarefa em todas as máquinas;
    - O peso de cada comunicação entre tarefas dependentes é a média dos custos de comunicação sobre todos os pares de recursos.

O cálculo dos pesos é feito dessa forma para obter um custo mais preciso

da execução de cada tarefa considerando que os recursos e infraestrutura de comunicação são heterogêneos.

- O segundo passo consiste na definição de um *rank* a partir desses pesos, de acordo com a equação (3).

$$rank_u(t_i) = \overline{w_i} + \max_{t_j \, suc(t_i)} (\overline{c_{i,j}} + rank_u(t_j)) \quad (3)$$

Onde, w_i é a média do tempo de execução da tarefa t_i sobre todos os recursos, $suc(t_i)$ é o conjunto de sucessores imediatos de t_i, e $c_{i,j}$ é a média dos custos de comunicação entre t_i e t_j. Dessa forma, o $rank_u$ de uma tarefa t_i é o peso de t_i mais o valor máximo resultando do peso de cada filho de t_i.

3.1.1.2. HEFT com lookahead

Um mecanismo que pode ser utilizado por estratégias estáticas é chamado *lookahead*, que consiste em fazer um plano de escalonamento e refinar esse plano de acordo com o tempo estimado das tarefas dependentes da tarefa que está sendo escalonada. Uma variação do HEFT com *lookahead* (BITTENCOURT; SAKELLARIOU; MADEIRA, 2010) foi desenvolvida incluindo uma rotina antes do mapeamento da tarefa. Tal rotina tem como objetivo medir o custo do mapeamento das tarefas dependentes da tarefa que está sendo mapeada, considerando o mapeamento dessa tarefa em todos os recursos disponíveis. O recurso que apresenta o menor custo de execução das tarefas dependentes é escolhido.

3.1.1.3. S-HEFT (Stochastic HEFT)

Uma das dificuldades de utilização do HEFT é a necessidade do algoritmo de conhecer o tempo de execução da tarefa e de transferência de arquivos entre as tarefas dependentes. Para minimizar esse problema um algoritmo

chamado S-HEFT (Stochastic HEFT) (TANG et al., 2011), propõe uma abordagem baseada em distribuição estocástica. A definição da prioridade das tarefas e o mapeamento das tarefas são baseados em uma distribuição estocástica do tempo de execução e transferência de arquivos. Uma dificuldade da utilização desse algoritmo é modelar o comportamento estocástico das tarefas.

*3.1.1.4. CPOP (*Critical Path On a Processor*)*

A estratégia CPOP (*Critical Path On a Processor*) (TOPCUOUGLU; HARIRI; WU, 2002) (TOPCUOGLU; HARIRI; WU, 1999) é similar ao HEFT, apresentando duas diferenças:

- A primeira diferença é o cálculo dos pesos. O cálculo do peso de cada tarefa é feito somando o custo computacional da tarefa t até a última tarefa do *workflow* pelo caminho de maior custo, e da tarefa t até a primeira tarefa do *workflow* também pelo caminho de maior custo. O cálculo do custo de transferência de dados entre tarefas dependentes, também é feito somando o custo de transferência de dados da tarefa t até a última tarefa do *workflow* pelo caminho de maior custo, e da tarefa t até a primeira tarefa do *workflow* também pelo caminho de maior custo.
- A segunda diferença é que antes de iniciar o escalonamento dos *workflows* um recurso é escolhido para executar todas as tarefas que estão no caminho crítico do workflow. Nesse sentido, as tarefas que estão no caminho crítico do *workflow* são mapeadas para esse recurso.

3.1.1.5. DAGMap

A estratégia DAGMap é um exemplo de estratégia que combina duas classes de escalonamento: o *cluster scheduling* e o *list scheduling* (CAO et al., 2008). A estratégia DAGMap realiza um escalonamento em três fases: a fase de priorização, a fase de agrupamento e a fase de mapeamento. A seguir cada

fase é descrita:

- Na fase de priorização, um *rank* é atribuído para cada tarefa, seguindo o mesmo critério da estratégia CPOP, para identificar a ordem utilizada para escalonar as tarefas;
- Na fase de agrupamento, um laço é realizado por todas as tarefas do *workflow* pela ordem de *rank*. Um grupo é criado contendo a primeira tarefa pela ordem de *rank*. A partir daí, as próximas tarefas são adicionadas a esse grupo, ou um novo grupo é criado caso a tarefa seguinte possua uma relação de dependência com a tarefa anterior. Dessa forma, as tarefas do *workflow* são divididas em grupos de tarefas independentes.
- Na fase de mapeamento as tarefas são mapeadas por ordem de grupo e por ordem de *rank*.

Essa estratégia é mais complexa que a HEFT e CPOP, pois envolve uma fase de agrupamento. Uma avaliação comparativa entre o HEFT, CPOP e DAGMap utilizando diversos *workflows* aleatórios mostrou que o DAGMap apresenta um tempo menor de execução.

3.1.1.6. PCH (Path Clustering Heuristic)

O Algoritmo PCH (Path Clustering Heuristic) (BITTENCOURT et al., 2005) é outro exemplo que combina duas classes de escalonamentos: o *list-scheduling* e o *cluster-scheduling*. Na fase de seleção de processadores é utilizada uma política de inserção, que consiste em procurar espaços vagos entre tarefas já mapeadas para mapear as tarefas, para melhorar a eficiência de uso dos recursos. Outra estratégia que realiza a mesma combinação é a estratégia *Performance Effective Task Scheduling* (PETS) (ILAVARASAN; THAMBIDURAI; MAHILMANNAN, 2005).

3.1.1.7. BFTT+OLB (Best File Transfer Time+Opportunistic Load Balancing)

O algoritmo para escalonamento de *workflows* e tarefas independentes chamado BFTT+OLB (*Best File Transfer Time*)+(*Opportunistic Load Balacing*) tem como objetivo otimizar as transferências de arquivos das tarefas em conjunto com uma estratégia de balanceamento de carga (MACHTANS; SATO; DEPPMAN, 2009).

3.1.2. Estratégias de Escalonamento Intra-tarefa

As estratégias apresentadas nesta seção exploram o paralelismo intra-tarefa apresentados pelos *workflows*. Nesse contexto, as estratégias consideram que os *workflows* podem ser compostos por tarefas sequenciais e por tarefas com paralelismo interno.

*3.1.2.1. M-HEFT (*Modified Heterogeneous Earliest Finish Time*)*

A estratégia M-HEFT (*Modified Heterogeneous Earliest Finish Time*) (SUTER; DESPREZ; CASANOVA, 2004) é uma extensão do HEFT para escalonamento de *workflows* compostos por tarefas sequenciais e tarefas com paralelismo interno, com modificações nas fases de cálculo do *rank* e de seleção de processadores.

Na fase de cálculo do *rank* do M-HEFT, o custo computacional de cada tarefa é calculado da mesma maneira que no HEFT. Com base no valor do *rank* é definida uma quantidade de recursos para cada tarefa.

Na fase de seleção de processadores é escolhido o *cluster* que possua a quantidade de recursos alocada para a tarefa, e dentro do *cluster* o subconjunto de recursos que apresentam o menor tempo de execução.

Um problema que pode ocorrer com o M-HEFT é que para alguns *workflows* podem ser produzido planos de escalonamento no qual algumas tarefas são mapeadas para uma grande quantidade de recursos para que seja obtida uma diminuição muito pequena no tempo total de execução. Nesse sentido, é importante que a quantidade de recursos que será alocada para cada tarefa seja balanceada com as outras tarefas do workflow. Nesse contexto, foi

proposta uma variação no algoritmo M-HEFT para que sejam utilizadas variáveis de controle para limitar o mapeamento de recursos para uma tarefa (N'TAKPE; SUTER; CASANOVA, 2007). Nesse contexto, foram criadas três variáveis: M-HEFT-IMP, M-HEFT-EFF e M-HEFT-MAX. A variável M-HEFT-MAX define uma quantidade máxima de recursos que pode ser alocada para uma tarefa em termos de porcentagem dos recursos do *cluster* escolhido. A variável M-HEFT-IMP define a porcentagem mínima de diminuição do tempo de execução para que um novo recurso seja adicionado à tarefa. A variável M-HEFT-EFF define a porcentagem mínima de ganho de desempenho para que um novo recurso seja adicionado à tarefa.

3.1.2.2. MHEFT-RSV (M-HEFT-ReSerVation)

Uma variante do HEFT chamada MHEFT-RSV (M-HEFT-ReSerVation) (ZHANG; LUO; DONG, 2013) foi desenvolvida com o objetivo de realizar o escalonamento em um ambiente no qual os recursos podem ser utilizados por diversos usuários. Nesse contexto, os recursos não são dedicados aos *workflows* do conjunto de entrada, porém o momento em que os recursos estarão livres é conhecido. Nesse sentido, a estratégia MHEFT-RSV faz o mapeamento das tarefas para os recursos que estarão disponíveis no momento estimado que a tarefa estará pronta para execução.

*3.1.2.3. CPA (*Critical Path and Area*)*

O CPA (*Critical Path and Area*) (RADULESCU; VAN GEMUND, 2001) é uma estratégia que tem como objetivo escalonar as tarefas de *workflows* em ambientes compostos por recursos homogêneos, balanceando o tempo de execução das tarefas no caminho crítico do *workflow* com o tempo médio de uso dos recursos.

O tempo de execução das tarefas chamado T_{CP} no caminho crítico é especificado como o maior tempo de execução entre todas as tarefas do caminho crítico, considerando o custo acumulado, desde a tarefa t até a última

tarefa do workflow.

O tempo de uso dos recursos chamado T_A é especificado como a somatória de tempo de execução das tarefas mapeadas em cada recurso, multiplicado pela quantidade de recursos utilizada pela tarefa, dividido pela quantidade de recursos total utilizada.

A estratégia inicia alocando um recurso para cada tarefa. Em seguida entra em um laço que é executado enquanto a seguinte condição for verdadeira: $T_{CP} > T_A$.

- O primeiro passo é identificar a tarefa do caminho crítico que pode obter maior ganho de desempenho ao receber mais um recurso.
- O segundo passo é alocar um recurso para essa tarefa.
- O terceiro passo é atualizar o conjunto de tarefas que compõe o caminho crítico.

Após finalizar a alocação da quantidade de recursos para cada tarefa, o CPA mapeia cada tarefa do *workflow* para o primeiro conjunto livre de recursos composto pela quantidade de recursos alocada para a tarefa.

*3.1.2.4. MCPA (*Modified Critical Path and Area*)*

O MCPA (*Modified Critical Path and Area*) (BANSAL; KUMAR; SINGH, 2006) é uma modificação da estratégia CPA, que tem como objetivo melhorar a alocação da quantidade de recursos para cada tarefa, para que na fase de mapeamento seja possível explorar o paralelismo inter-tarefa apresentado pelos *workflows*.

Na fase de alocação o MCPA aloca novos recursos considerando duas condições:

- A primeira condição é escolher a tarefa que pode obter maior ganho de desempenho ao receber mais um recurso.
- A segunda condição é que a quantidade de recursos alocada para tarefas do caminho crítico do *workflow* no mesmo nível, não podem ser maior que a quantidade total de recursos disponível.

Utilizando essas condições na fase de alocação de quantidade de recursos o MCPA consegue realizar a alocação de quantidade de recursos de maneira balanceada com o paralelismo inter-tarefa.

*3.1.2.5. HCPA (*Heterogeneous Critical Path and Area*)*

Para atender os requisitos de escalonamento em ambientes heterogêneos foi desenvolvida uma extensão do CPA chamada HCPA (*Heterogeneous Critical Path and Area*) (DUTOT et al., 2009) (N'TAKPE; SUTER; CASANOVA, 2007) (CASANOVA; DESPREZ; SUTER, 2010). O diferencial do HCPA é que é a estimativa de tempo de execução de cada tarefa dos *workflows* é definida a partir de uma execução prévia de cada tarefa de todos os *workflows*, separadamente, em um mesmo *cluster* homogêneo, que é chamado *cluster* de referência. Nesse sentido, o tempo de execução de cada tarefa em qualquer recurso do ambiente de grade computacional pode ser estimado, a partir da relação entre a capacidade do recurso escolhido e a capacidade de um dos recursos do *cluster* de referência.

O HCPA foi utilizado como base para implementar uma estratégia chamada Bi-CPA (DESPREZ; SUTER, 2010), que tem como objetivo considerar mais de uma função objetivo a ser otimizada. O Bi-CPA realiza o escalonamento de tarefas otimizando o *Makespan* e o consumo de energia.

*3.1.2.6. iCASLB (*an iterative Coupled processor Alocation and Scheduling Algorithm with look-ahead and backfill*)*

O iCASLB (*an iterative Coupled processor Alocation and Scheduling Algorithm with look-ahead and backfill*) (VYDYANATHAN et al., 2006) é um algoritmo iterativo que tem como objetivo minimizar o *makespan* dado o tempo de execução e *speedup* das tarefas, utilizando técnicas de *lookahead* e *backfilling*. O iCASLB é mais complexo que o CPA mas pode gerar escalonamentos com *makespan* menor.

3.2. Estratégias de Escalonamento de Workflows em Ambientes Dinâmicos

A presente seção mostra iniciativas de pesquisa de estratégias de escalonamento de *workflows* para ambiente dinâmicos, que consideram o paralelismo inter-tarefa ou o paralelismo intra-tarefa.

3.2.1. P-HEFT

A estratégia P-HEFT (BARBOSA; MOREIRA, 2011) é baseada na classe de escalonamento *individual task scheduling* e foi desenvolvida para ser utilizada em conjunto com as estratégias M-HEFT e HPTS (*Heterogeneous Parallel Task Scheduler*). Tal estratégia monitora a execução das tarefas dos *workflows* e quando um conjunto de tarefas torna-se pronto para ser executado a estratégia M-HEFT ou HPTS é utilizada para realizar o escalonamento estático dessas tarefas.

3.2.2. AHEFT

O AHEFT (*Adaptative HEFT*) é uma abordagem adaptativa do HEFT (LEE et al., 2007) para realizar escalonamento dinâmico de *workflows*. A estratégia é composta por dois componentes *planner* e *executor.* O componente *planner* tem como objetivo fazer escalonamento estático e re-escalonamento dos *workflows*. O escalonamento estático é feito assim que o *workflow* é submetido para execução. O re-escalonamento é realizado considerando duas características do ambiente: disponibilidade de recursos e variação de tempo de execução da tarefa. A disponibilidade de recursos consiste na adição de novos recursos, ou falha em um recurso. A variação de desempenho ocorre devido a imprecisões de estimativas de tempo de execução ou variações de carga nos *clusters* que impactem no desempenho. O *executor* é responsável por executar o *workflow* nos recursos definidos pelo *planner* e monitorar a execução para alimentar as bases usadas pelo *planner* para realizar o re-escalonamento. Dois aspectos importantes dessa estratégia são: a

colaboração entre os componentes *planner* e *executor* para realizar o escalonamento dinâmico, e a colaboração entre as diversas execuções de *workflows* simultâneas.

3.2.3. Planner-guided dynamic scheduling

Uma variante do HEFT chamada *Planner-guided dynamic scheduling* foi proposta para realizar o escalonamento de múltiplos *workflows* em ambientes dinâmicos (YU; SHI, 2008). O diferencial dessa variante é que a definição de *rank* é feita de maneira única para todos os *workflows* que estão sendo submetidos.

3.2.4. Re-escalonamento Híbrido em Ambientes Multi-Cluster

Outra estratégia dinâmica para escalonamento de *workflows* baseado na classe de escalonamento *individual task scheduling* foi proposta, considerando uma estimativa de tempo de execução das tarefas baseado no tempo que a tarefa ficará na fila de execução dos *clusters* (ZHANG; KOELBEL; COOPER, 2009). A execução das tarefas é monitorada e um re-escalonamento é realizado quando o tempo estimado de execução torna-se muito diferente do tempo real.

3.2.5. Re-escalonamento em Ambientes Multi-Usuário

Uma proposta de realizar o escalonamento de *workflows* em ambientes *multi-cluster* dinâmicos considera que os usuários locais têm maior prioridade que os usuários que submetem *workflows* (HE et al., 2005). A estratégia determina o custo computacional da tarefa e escalona a tarefa para os *clusters* menos carregados. Durante a execução é realizada uma monitoração das tarefas submetidas de outros domínios administrativos e das tarefas submetidas localmente. No caso da carga local aumentar além de um nível

mínimo pré-configurado, é feito um re-escalonamento para dar maior prioridade à carga local.

*3.2.6. BTS (*Balanced Time Scheduling*)*

O algoritmo BTS (*Balanced Time Scheduling*) (BYUN et al., 2011) faz uma estimativa da quantidade de recursos que deve ser alocada para um *workflow* para que seja executado em um tempo limite definido pelo usuário. O objetivo é mapear as tarefas do *workflow* para a menor quantidade possível de recursos sem ultrapassar o tempo limite de execução do workflow. O BTS explora também a possibilidade de alterar a quantidade de recursos alocada para cada tarefa maleável durante a sua execução.

3.2.7. OWM (Online Workflow Management*)*

A estratégia *Online Workflow Management* (OWM) (HSU; HUANG; WANG, 2011) foi proposta para realizar o escalonamento dinâmico de tarefas considerando dois aspectos. O primeiro aspecto é que a estratégia decide a quantidade de recursos que deve ser alocada para uma tarefa com paralelismo interno baseado no custo do caminho crítico. O segundo aspecto é que a estratégia considera a possibilidade de adiantar tarefas para preencher espaços vagos gerados pelas dependências de tarefas.

O OWM considera a possibilidade de executar tarefas com paralelismo interno em múltiplos *clusters*, sem avaliar se essa decisão pode diminuir o tempo de execução da tarefa com paralelismo interno.

*3.2.8. MWGS (*Multiple Workflow Grid Scheduling*)*

Duas estratégias chamadas MWGS4 (*Multiple Workflow Grid Scheduling 4 stages*) e MWGS2 (*Multiple Workflow Grid Scheduling 2 stages*) (HIRALES-CARBAJAL et al., 2012) foram propostas com o objetivo de realizar o

escalonamento dinâmico de *workflows* considerando que as estimativas de tempo de execução são imprecisas ou não são conhecidas.

3.3. Estratégias de Escalonamento Dinâmico de Tarefas Independentes com Paralelismo Interno

A presente seção mostra estratégias de escalonamento de tarefas com paralelismo interno independentes em ambientes *multi-cluster* dinâmico.

3.3.1. Estratégias Baseadas na Alteração de Reserva de Recursos

Em ambientes *multi-cluster* a execução de tarefas é controlada por meio de filas. Um problema que pode ocorrer é que quando uma tarefa da fila entrar em execução pode não haver a quantidade de recursos que a tarefa necessita. Um mecanismo para minimizar esse problema é realizar reserva avançada de recursos (SNELL et al., 2000).

Uma desvantagem de utilizar reserva avançada é que uma vez que a tarefa termina sua execução, os recursos continuam reservados pelo tempo definido pelo usuário. Nesse sentido, uma política para minimizar esse problema é gerenciar a execução de tarefas utilizando uma fila de espera por recursos (MOHAMED; EPEMA, 2005). As tarefas nessa fila aguardam por um período de tempo determinado. Caso obtenham os recursos dentro desse período, entram em execução, caso contrário, a tarefa volta ao fim da fila e a quantidade de recursos alocada para a tarefa é diminuída para que a chance de entrar em execução aumente.

Uma política para refazer o mapeamento de tarefas permitindo alterar as reservas avançadas foi proposta (NETTO; BUYYA, 2008) com o objetivo de reduzir a ociosidade de recursos provocada pelas reservas avançadas fixas.

Uma estratégia dinâmica de escalonamento de tarefas com paralelismo interno foi proposta para realizar migração de tarefas da fila entre *clusters*, considerando a carga corrente dos *clusters* (YIGITBASI et al., 2011). Diversas métricas são utilizadas para medir a carga do *cluster*, para trazer uma maior

precisão da estimativa, tais como, uso médio de CPU, utilização de disco e utilização dos *clusters*.

3.3.2. Estratégias que Utilizam Técnicas de Backfilling

Outra maneira de alterar a ordem de execução de tarefas provocando pouco impacto nas tarefas em execução foram implementadas com base em técnicas de *backfilling* (SRINIVASAN; KRISHNAMOORTHY; SADAYAPPAN, 2003),(ZHANG et al., 2000), que é um processo que consiste em alterar a ordem de execução de tarefas, com o objetivo de diminuir o tempo de resposta, sem gerar sobrecarga significativa nas demais tarefas.

Um aspecto importante para executar tarefas com paralelismo interno em múltiplos *clusters* é a sobrecarga de comunicação gerada pela comunicação *inter-cluster*. O impacto de executar tarefas com paralelismo interno em múltiplos *clusters* foi avaliado considerando a política *First-Come-First-Serve* (FCFS) em conjunto com técnicas de *backfilling* (ERNEMANN et al., 2002), mostrando que uma sobrecarga de até 25% de comunicação é tolerável para se obter bons resultados.

3.3.3. Meta-Escalonamento Baseado na Largura de Banda

Uma estratégia de escalonamento de tarefas com paralelismo interno foi proposta com base na alocação de banda de rede para as tarefas (JONES et al., 2005). Dessa forma, a utilização de recursos de múltiplos *clusters* é realizada quando existe uma quantidade mínima de banda de rede necessária para executar a aplicação.

3.4. Análise

Nessa seção foram apresentados os esforços de pesquisa para o escalonamento de *workflows* em ambientes de grade computacional e para o

escalonamento de tarefas com paralelismo interno em ambientes dinâmicos.

Diversas iniciativas de pesquisa com objetivo de implementar ScWFMS utilizam as estratégias apresentadas nesse capítulo, por exemplo, o Askalon (FAHRINGER et al., 2005) utiliza uma estratégia baseada na estratégia HEFT, o Pegasus (DEELMAN et al., 2005) utiliza uma estratégia baseada no modelo *cluster scheduling*, e o GriPhyn (DEELMAN et al., 2002) utiliza uma estratégia que seleciona os recursos para as tarefas de acordo com o custo de transferência de dados de entrada.

As estratégias de escalonamento estático de *workflows* estático parte da premissa que o escalonador possui conhecimento total dos *workflows* e que os recursos são dedicados à execução desses workflows. Dessa forma, tais estratégias utilizam as características das tarefas e do ambiente para produzir planos de escalonamento próximos do resultado ótimo.

Em ambientes dinâmicos as estratégias de escalonamento estático possuem um desempenho ruim devido às alterações de desempenho que acontecem de maneira imprevisível. Nesse sentido, a maior parte das estratégias que atacam o problema de escalonamento de *workflows* em ambientes dinâmicos, utilizam as estratégias estáticas para realizar um plano de escalonamento e implementam rotinas de re-escalonamento para adaptar tais planos às alterações de desempenho.

As estratégias baseadas no modelo *list-scheduling* são adequadas para realizar o escalonamento de tarefas em ambientes dinâmicos, pois conseguem definir o impacto da decisão de escalonamento de uma tarefa em todas as tarefas do workflow.

As estratégias de escalonamento de tarefas com paralelismo interno independentes em ambientes dinâmicos apresentam bons resultados, pois levam em conta o custo de comunicação das tarefas e a carga dos *clusters* para tomar decisões de escalonamento, além de utilizar informações sobre reserva de recursos para otimizar o tempo de execução. Com base nessas informações, algumas estratégias realizam o mapeamento de tarefas com paralelismo em múltiplos *clusters*, diminuindo o tempo de execução.

A partir dessas observações é possível verificar que as estratégias de escalonamento de *workflows* em ambientes dinâmicos não levam em conta a possibilidade de executar cada tarefa com paralelismo interno em múltiplos

clusters para decisão de escalonamento. Porém a utilização de técnicas de co-alocação em conjunto com uma estratégia de escalonamento de *workflows* em ambientes dinâmicos tem potencial de melhorar o desempenho de uma grande variedade de *workflows* compostos por tarefas sequenciais e tarefas com paralelismo interno.

4. Proposta de Estratégias de Escalonamento de *Workflows* em Ambientes de Grade Computacional

Esta seção apresenta duas estratégias chamadas MCA-HEFT (Multi-Cluster Allocation-HEFT) e DMCA-HEFT (Dynamic Multi-Cluster Allocation-HEFT) propostas para realizar o escalonamento de *workflows* compostos por tarefas sequenciais e tarefas com paralelismo interno em ambientes de grade computacional compostos por múltiplos *clusters*.

Na Seção 4.1 é apresentada a definição do problema de escalonamento de *workflows*. Na Seção 4.2 é mostrado o modelo de aplicações. Na Seção 4.3 é mostrada a estratégia MCA-HEFT e na Seção 4.4 é mostrada a estratégia DMCA-HEFT.

4.1. Definição do Problema

O problema de escalonamento de *workflows* compostos por tarefas sequenciais e tarefas com paralelismo interno em ambientes de grade computacional composta por múltiplos *clusters* é um problema NP-Completo (PINEDO, 2008).

O cenário de execução de *workflows*, aqui considerados, possui as seguintes características: os *workflows* podem apresentar paralelismo intra-tarefa, as tarefas com paralelismo interno podem ser executadas em múltiplos *cluster*, os *workflows* não são conhecidos antes da sua submissão e nem o momento que serão submetidos pelos usuários.

4.2. Modelo de Aplicações

Conforme apresentado na Seção 2.1.4 um *workflow* é composto por um conjunto de tarefas interdependentes. Neste trabalho cada tarefa t é definida pelo seguinte conjunto de valores $[comp_t, comm_t, mincores_t, Type_t, pf_t]$:

1. $comp_t$ é o tamanho computacional da tarefa t especificado em

número de instruções (MIPS);

2. $comm_t$ é a quantidade de dados que devem ser transferidos internamente durante a execução da tarefa t que possui paralelismo interno, entre os recursos utilizados para executar a tarefa em MB (Mega Bytes);
3. $mincores_t$ representa o número mínimo de recursos requisitado para executar a tarefa t;
4. $type_t$ é usado para identificar se a tarefa t é seqüencial, rígida ou moldável:
    a. $type_t = 1$ indica que é uma tarefa sequencial;
    b. $type_t = mincores_t \neq 1$ indica que é uma tarefa rígida;
    c. $type_t > mincores_t$ indica que é uma tarefa moldável;
5. pf_t é a porção paralela da tarefa t.

4.3. Estratégia MCA-HEFT (Multi-Cluster Allocation-HEFT)

A estratégia MCA-HEFT (STANZANI; SATO; NETTO, 2013) tem como objetivo realizar o escalonamento estático de *workflows* compostos por tarefas sequenciais e tarefas com paralelismo interno verificando para cada tarefa com paralelismo interno a possibilidade de ser executada concorrentemente em recursos montados em múltiplos *clusters*. A Figura 5 mostra um exemplo de uso da estratégia MCA-HEFT.

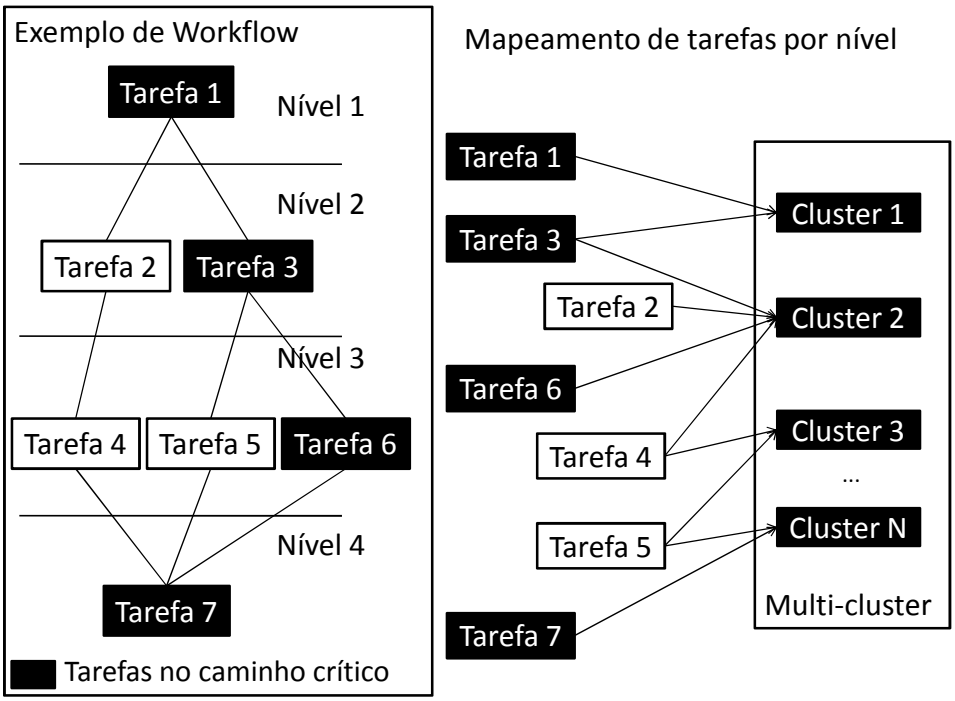

Figura 5. Exemplo de Escalonamento de Tarefas por Nível.

A estratégia MCA-HEFT é baseada na estratégia HEFT (TOPCUOUGLU; HARIRI; WU, 2002), e possui as seguintes características:

- Considera que os *workflows* podem ser compostos por tarefas sequenciais ou com paralelismo interno;
- Utiliza a informação sobre a porção paralela das tarefas para definir a alocação de recursos;
- Explora a possibilidade de alocar recursos de múltiplos *clusters* para execução de tarefas com paralelismo interno.

A estratégia MCA-HEFT realiza o escalonamento de tarefas em duas fases chamadas: definição de prioridade e seleção de processadores.

- A função definição de prioridade tem como objetivo definir a ordem que será aplicada para escalonar as tarefas;
- Na fase de seleção de processador é realizada a alocação de uma quantidade de recursos para cada tarefa e em seguida o mapeamento de cada tarefa para um conjunto de recursos que apresente o menor tempo de execução.

A seguir será descrita cada uma das fases, e em seguida será mostrada a sequencia de passos da estratégia MCA-HEFT.

4.3.1. Definição de Prioridade

A definição da prioridade de escalonamento de cada tarefa do *workflow* é feita por nível e *rank.* Dentro de um nível o escalonamento das tarefas é feito por ordem decrescente de *rank*. Essa ordem é adotada, pois permite mapear as tarefas que podem diminuir o tempo total da execução do *workflow* nos melhores recursos.

O MCA-HEFT define o valor do *rank* de uma tarefa com base no custo computacional da tarefa e com base no custo de comunicação interno da tarefa.

O cálculo do *rank* de uma tarefa $\boldsymbol{t}$ no MCA-HEFT é realizado por meio da soma dos custos de computação e de comunicação ($\boldsymbol{comp_t\ e\ comm_t}$), com os custos ($\boldsymbol{comp_t\ e\ comm_t}$) das tarefas sucessivas com o maior *rank* até a primeira tarefa do *workflow* que não possuir tarefas sucessivas, de acordo com a Equação (4).

$$rank(t) = (tempocomp_t + tempocomm_t) + \max\,(rank(succ(t))). \quad (4)$$

Se $succ(t)$ não possui tarefas retorna $(tempocomp_t + tempocomm_t)$.

Onde,

- O custo de computação é medido da seguinte maneira: $tempocomp_t = comp_t / Min(\ avgcomp_c\ )$.
- O custo de comunicação é medido da seguinte maneira: $tempocomm_t = comm_t / Min(\ avgbandwidth_c\ )$.
- O $avgbandwidth_c$ representa a largura de banda média da rede.

- $\mathrm{succ(t)}$ representa as tarefas sucessivas da tarefa t até a última tarefa do workflow.

4.3.2. Seleção de Processadores

A seleção de processadores é realizada em três passos:

1. Alocação de *cluster*: é definido para cada tarefa se deverá ser mapeado para apenas um *cluster* ou se poderá ser mapeada para múltiplos *clusters*.
2. Alocação da quantidade de recursos: é definida a quantidade de recursos que será alocada para cada tarefa.
3. Mapeamento de tarefas: é feito o mapeamento de cada tarefa para os recursos. Esse mapeamento pode ser feito por uma das seguintes formas:
    - A tarefa é mapeada para um *cluster* que possua a quantidade de recursos definida.
    - A tarefa é mapeada para múltiplos *clusters* para receber a quantidade de recursos definida.
    - A quantidade de recursos da tarefa definida no passo 2 é alterada de acordo com a quantidade de recursos disponíveis de um dos *clusters*.

4.3.2.1. Alocação de Cluster

O passo de alocação de *cluster* da estratégia MCA-HEFT é executado para cada tarefa para definir se a tarefa pode ser mapeada em múltiplos *clusters* ou se deve ser mapeada para apenas um *cluster*. O critério para definir se uma tarefa pode ser executada em múltiplos *clusters* é baseado em dois aspectos: quantidade de operações de comunicação realizada, e na qualidade da

interconexão de rede entre os *clusters*. Quanto mais comunicação a tarefa realiza, maior será a perda de desempenho ao ser executada em uma infra-estrutura de rede de baixo desempenho, utilizando múltiplos *clusters*. Quanto melhor a qualidade da interconexão de rede entre os *clusters*, mais valerá a pena executar as tarefas em múltiplos *clusters*.

Para medir a quantidade de comunicação realizada por uma tarefa foi criada uma métrica chamada $MCACCR$. O objetivo dessa métrica é medir a quantidade de comunicação em comparação com a quantidade de computação da porção paralela realizada pela tarefa. A métrica $MCACCR$ é definida com base na CCR (proporção computação para comunicação) e na porção paralela pf_t. O CCR é uma estimativa da quantidade de computação realizada junto com comunicação, e o pf_t é usado para estimar o CCR da porção paralela da tarefa. O $MCACCR$ de uma tarefa t é definido de acordo com a Equação (5).

$$MCACCR(t) = \left(\frac{comp_t}{comm_t}\right) * pf_t. \quad (5)$$

Para avaliar a qualidade da interconexão de rede entre os *clusters* foi definida uma variável chamada MCA, essa variável define o valor mínimo do $MCACCR$ de uma tarefa para que a tarefa seja mapeada para múltiplos *clusters*. Nesse sentido, ao definir um valor baixo de MCA, muitas tarefas serão mapeadas para múltiplos *clusters*, e ao definir um valor alto de MCA, poucas tarefas serão mapeadas para múltiplos *clusters*. Dessa forma, o MCA é definido empiricamente com base na qualidade da rede interconexão entre *clusters*, no caso de uma rede com baixa qualidade é definido um MCA com valor alto para inibir a execução de tarefas em múltiplos *clusters*, e no caso de uma rede com boa qualidade é definido um MCA baixo para que mais tarefas sejam mapeadas em múltiplos *clusters*.

4.3.2.2. Alocação de Quantidade de Recursos

O passo de alocação de quantidade de recursos da estratégia MCA-HEFT tem como objetivo definir a quantidade de recursos que será alocada para cada tarefa.

O processo de alocação é realizado por ordem de níveis e *rank*. Para cada nível é considerado que todos os recursos do ambiente *multi-cluster* estão disponíveis. A partir daí os recursos são divididos entre as tarefas.

O processo de alocação é iniciado alocando a quantidade mínima de recursos exigida por cada tarefa $mincores_t$.

Para cada tarefa é definida uma razão máxima da quantidade total de recursos. Essa razão máxima é definida de maneira proporcional ao *rank* da tarefa em relação à somatória dos *ranks* das tarefas no mesmo nível. Essa porcentagem é definida pela função $MCAMC(t)$. O cálculo é mostrado na Equação (6) onde $totalnodes_c$ representa o número total de recursos do *cluster* c e $totalTaskbyLevel$ representa o número de tarefas no nível da tarefa t.

$$MCAMC(t) = \left(\frac{rank(t)}{\sum_{j=1}^{totalTaskbyLevel} rank(j)}\right) * \sum_{1}^{clusters} totalnodes_c. \quad (6)$$

Para cada tarefa é iniciado um laço para realizar a alocação de um novo recurso. O laço se repete até que a quantidade de recursos disponíveis já tenha sido alocada, ou que a tarefa possa ter novos recursos alocados, de acordo com os seguintes critérios:

1. O tempo de execução da tarefa deve ser minimizado em pelo menos uma porcentagem de tempo, comparando com o tempo de execução sem esse novo recurso.
2. A quantidade de recursos alocada deve ser menor que o percentual máximo do total de recursos definida por $MCAMC(t)$, e menor que MCAMAX.

3. A razão do tempo gasto com computação pelo tempo gasto com comunicação da tarefa não pode ser menor do que 10%.

A variável MCAMAX define a quantidade máxima de recursos que pode ser alocada para cada tarefa. A quantidade de recursos considerada é a soma do total de recursos de todos os *clusters*.

A variável MCAIMP é uma porcentagem que é usada para definir que para alocar um novo recurso para a tarefa, o custo computacional precisar ser diminuído por pelo menos uma porcentagem igual à MCAIMP. O custo computacional de uma tarefa é estimado de acordo com o algoritmo (7) em termos da quantidade de instruções, e o tempo de execução é definido na Equação (8) em segundos, de acordo com a quantidade estimada de instruções.

$$compCost(t, nodes) = comp_t$$
$$for\ n = 0\ até\ n = nodes$$
$$compCost(t, nodes) = compCost(t, nodes) * (1 - pf_t) + compCost(t, nodes) * \frac{pf_t}{nodes} \quad (7)$$

$$MCAET\ (t, nodes) = \frac{compCost(t, nodes)}{Min(\ avgcomp_c\ )} \quad (8)$$

O $avgcomp_c$ é a média de poder computacional de cada um dos recursos do *cluster* c especificado em MIPS e é calculado de acordo com a Equação (9), onde $avgcomp_r$ é o poder computacional de um recurso r.

$$avgcomp_c = \sum_{r=1}^{totalnodes_c} avgcomp_r\ /totalnodes_c. \quad (9)$$

O poder computacional considerado $avgcomp_c$ é o poder computacional mais baixo entre todos os *clusters*. A alocação de quantidade de recursos é aumentada no caso em que $MCAET\ (t, nodes + 1)$ seja mais baixo que

$MCAET\ (t, nodes)$ por pelo menos uma porcentagem igual à MCAIMP.

Depois de alocar uma quantidade de recursos para cada tarefa é feito o mapeamento das tarefas para a quantidade de recursos alocada. Esse processo é descrito na seção seguinte.

4.3.2.3. Mapeamento de Tarefas

O mapeamento é realizado por ordem de nível e de *rank*. As tarefas são mapeadas para o *cluster* que possuem uma quantidade de recursos que é maior ou igual à quantidade de recursos alocada para a tarefa. Os *clusters* são verificados em ordem decrescente de carga. A carga do *cluster* é definida pela Equação (10), onde, $usednodes_c$ é a quantidade de recursos do *cluster* c que já foi alocada.

$$load(c) = \frac{avgcomp_c * usednodes_c}{totalnodes_c}. \quad (10)$$

No caso de nenhum *cluster* possuir mais recursos livres do que os recursos alocados pela tarefa, a estratégia MCA-HEFT pode mudar a quantidade de recursos alocados para a tarefa ou pode mapear a tarefa para múltiplos *clusters*. O critério para tomar essa decisão foi definido na Seção 4.3.2.

Depois que a tarefa é mapeada para um ou mais *clusters*, é feita a escolha dos recursos menos carregados para executar a tarefa. A carga de cada recurso é armazenada na variável $nodeload_{c,r}$ onde c é o *cluster* e r o recurso do *cluster* c.

A variável $allocSET$ representa o conjunto de recursos para os quais a tarefa t foi mapeada, e $alloc_t$ representa a quantidade de recursos alocados para a tarefa. Depois que o mapeamento é finalizado, a carga de cada recurso do conjunto $allocSET$ é atualizada, executando $nodeload_{c,r} = nodeload_{c,r} + compCost(t)$, e o número de recursos ocupados do *cluster* c é atualizada executando $usednodes_c = usednodes_c + alloc_t$.

4.3.3. Sequencia de Passos da Estratégia MCA-HEFT

Os passos da estratégia MCA-HEFT foram divididos em três funções: **Cálculo_do_Rank(), Alocação_de_Recursos() e Mapeamento_de_Tarefas()**. A sequencia de passos da estratégia MCA-HEFT é mostrada no Algoritmo 4.

O algoritmo 1 realiza o cálculo do *rank.*

Algoritmo 1. Cálculo_do_Rank().

1: **for** all t ∈ Workflow

2: $tempocomp_t = comp_t / Min(\ avgcomp_c\ ).$

3: $tempocomm_t = comm_t / Min(\ avgbandwidth_c\ ).$

4: $\boldsymbol{if}\ succ(t) == \emptyset$

5: $rank(t) = (tempocomp_t + tempocomm_t)$

6: $\boldsymbol{else}$

7: $rank(t) = (tempocomp_t + tempocomm_t) + \max\ (rank(succ(t))).$

O algoritmo 2 realiza a alocação de recursos. Na linha 3 é feito o cálculo do MCACCR. Na linha 4 é feito o cálculo do MCAMC. Nas linhas de 5 a 8 é verificado se a tarefa pode ser mapeada para múltiplos *clusters*. Nas linhas de 9 a 11 é feita a alocação de recursos.

Algoritmo 2. Alocação_de_Recursos().

1: **for** each level l ∈ Levels

2: **for** all t ∈ l

3: $MCACCR(t) = \left(\frac{comp_t}{comm_t}\right) * pf_t$

4: $MCAMC(t) = \left(\frac{rank(t)}{\sum_{j=1}^{totalTaskbyLevel} rank(j)}\right) * \sum_{1}^{clusters} totalnodes_c$

5: $\boldsymbol{if} MCACCR(t) < MCA$

6: $multiCluster(t) = 0$

7: ***else***

8: $multiCluster(t) = 1$

9: $alloc_t = mincores_t$

10: ***while*** $(\text{alloc}_t + \mathbf{1} < \text{MCAMC}_t)$ **and** $(\text{alloc}_t + \mathbf{1} < MCAMAX)$ **and** $(MCAET\ (t, alloc_t + 1) > MCAET\ (t, alloc_t) - \left(MCAET\ (t, alloc_t) * \frac{\text{MCAIMP}}{100}\right))$

11: $alloc_t = alloc_t + 1$

O algoritmo 3 realiza o mapeamento das tarefas para os recursos menos carregados do *cluster* com menor carga. Na linha 3 o *cluster* com menor carga é selecionado. Nas linhas de 4 a 6 é verificado se a quantidade de recursos livres no *cluster* é menor do que a quantidade exigida pela tarefa. Em caso positivo e caso a tarefa tenha sido mapeada para apenas um cluster, a quantidade de recursos alocada para a tarefa é alterada para a quantidade de recursos livres do cluster. Nas linhas de 8 a 12 é realizado o mapeamento de cada recurso alocado para a tarefa para o recurso menos carregado do *cluster*. Caso o *cluster* não possua a quantidade de recursos alocada para a tarefa, e a tarefa tenha sido alocada para múltiplos *clusters*, o próximo *cluster* menos carregado é selecionado para mapear os recursos da tarefa.

Algoritmo 3. Mapeamento_de_Tarefas().

```
1:  for each level l ∈ Levels
2:      for all t ∈ l
3:          c = min(load(cluster))
4:          if alloc_t > totalnodes_c − usednodes_c
5:              if multiCluster(t) < 1
6:                  alloc_t = totalnodes_c − usednodes_c
7:          while alloc_t > 0
8:              while usednodes_c ≤ totalnodes_c
9:                  alloc_t = alloc_t − 1
10:                 allocSET = allocSET + min(nodeload_{c,r})
11:                 nodeload_{c,r} = nodeload_{c,r} + compCost(t, alloc_t)
12:                 usednodes_c = usednodes_c + 1
13:             c = min(load(cluster))
```

Algoritmo 4. Estratégia MCA-HEFT.

```
1:  Cálculo_do_Rank()
2:  Alocação_de_Recursos()
3:  Mapeamento_de_Tarefas()
```

4.3.4. Exemplo de Escalonamento Utilizando a Estratégia MCA-HEFT

A presente seção apresenta todos os passos realizados para produzir um plano de escalonamento para um *workflow* utilizando a estratégia MCA-HEFT e utilizando a estratégia M-HEFT. A comparação é realizada com o M-HEFT por ser uma heurística muito utilizada para escalonamento de *workflows* compostos por tarefas sequenciais e tarefas com paralelismo interno.

O *workflow* utilizado é composto por oito tarefas. O custo computacional e

custo de comunicação interna da tarefa são mostrados na Figura 6. Todas as tarefas possuem porção paralela 0,5.

O ambiente é composto por dois *clusters* com seis recursos homogêneos cada um. O MCA adotado é 1 e o MCAIMP adotado é 5%. A capacidade de cada recurso do ambiente é 20 e a largura de banda é 100 Mb.

O primeiro passo é a definição de *rank* que é feito da mesma maneira no MCA-HEFT e no M-HEFT conforme mostrado na Figura 6.

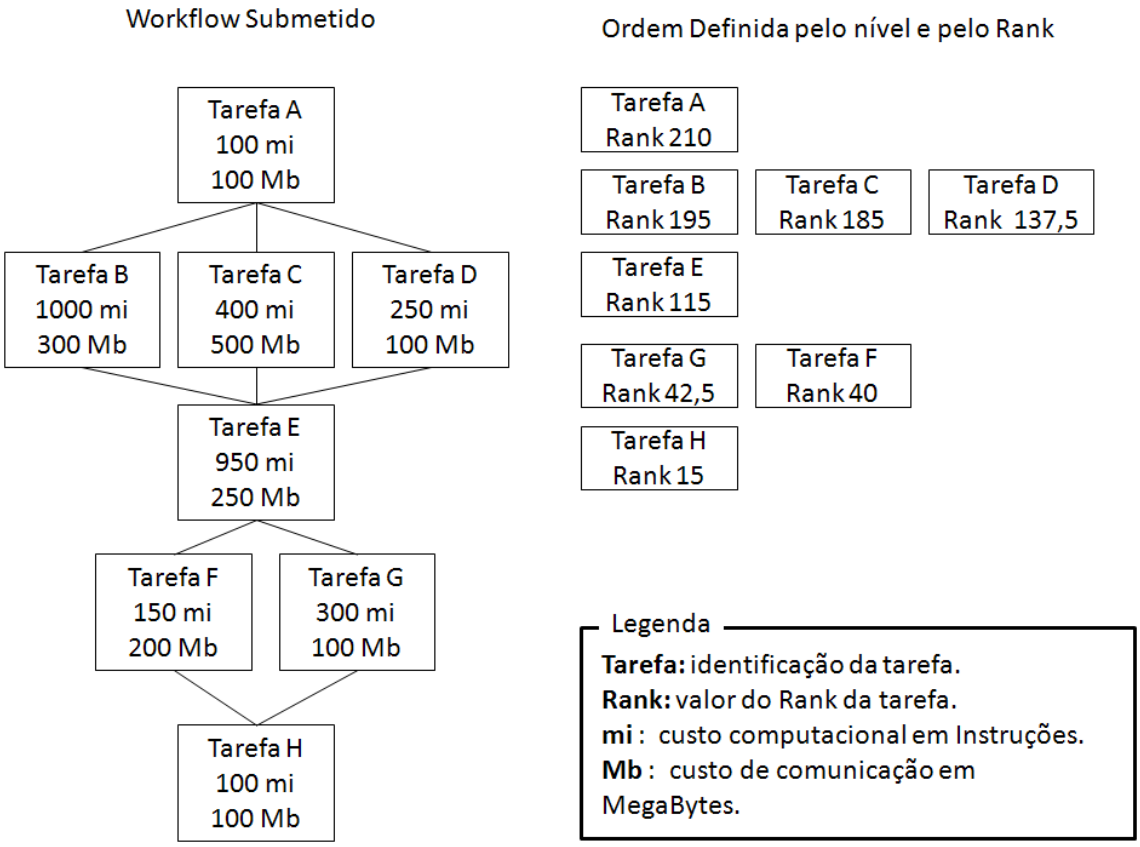

Figura 6. Cálculo do *Rank* das Tarefas do Workflow.

Em seguida é mostrada a alocação e mapeamento das tarefas nos recursos utilizando as estratégias MCA-HEFT e M-HEFT.

A Tabela 1 mostra o valor do MCACCR e MCAMC de cada tarefa do workflow. Com base nesses valores a estratégia decide se cada tarefa deve ser mapeada para múltiplos *clusters* ou um *cluster* e a alocação máxima permitida para cada tarefa. Para cada tarefa é realizado o cálculo do MCAET, que é mostrado na Tabela 2, para definir a quantidade de recursos alocada para cada tarefa.

Tabela 1. Cálculo do MCACCR e MCAMC das Tarefas do Workflow.

Tarefa	MCACCR	MCAMC	MCACCR>MCA	Alocação máxima
A	1	1	1 *Cluster*	6
B	3,3	0.5	Múltiplos *clusters*	5
C	0,8	0.13	1 *Cluster*	4
D	2,5	0.37	Múltiplos *clusters*	3
E	3,8	1	Múltiplos *clusters*	6
F	0,75	0.2	1 *Cluster*	6
G	3	0.8	Múltiplos *clusters*	6
H	1	1	1 *Cluster*	6

Tabela 2. Cálculo do MCAET das Tarefas do *Workflow* Usando de 1 à 6 Recursos.

	MCAET()					
Tarefa	**1**	**2**	**3**	**4**	**5**	**6**
A	100	50	25	12,5	6,25	3,125
B	1000	500	250	125	62,5	
C	400	200	100	50		
D	250	125	65,5			
E	950	475	237,5			
F	150	75	37,5	16,75	8,37	4,16
G	300	150	75	37,5	16,75	8,37
H	100	50	25	12,5	6,25	3,125

A Figura 7 mostra a alocação de recursos e em seguida o mapeamento realizado pela estratégia MCA-HEFT. No nível dois foram alocados cinco recursos para a tarefa B, quatro recursos para a tarefa C e três recursos para a

tarefa D. Dessa forma, a tarefa D precisou ser mapeada para recursos de dois *clusters* para receber a alocação definida.

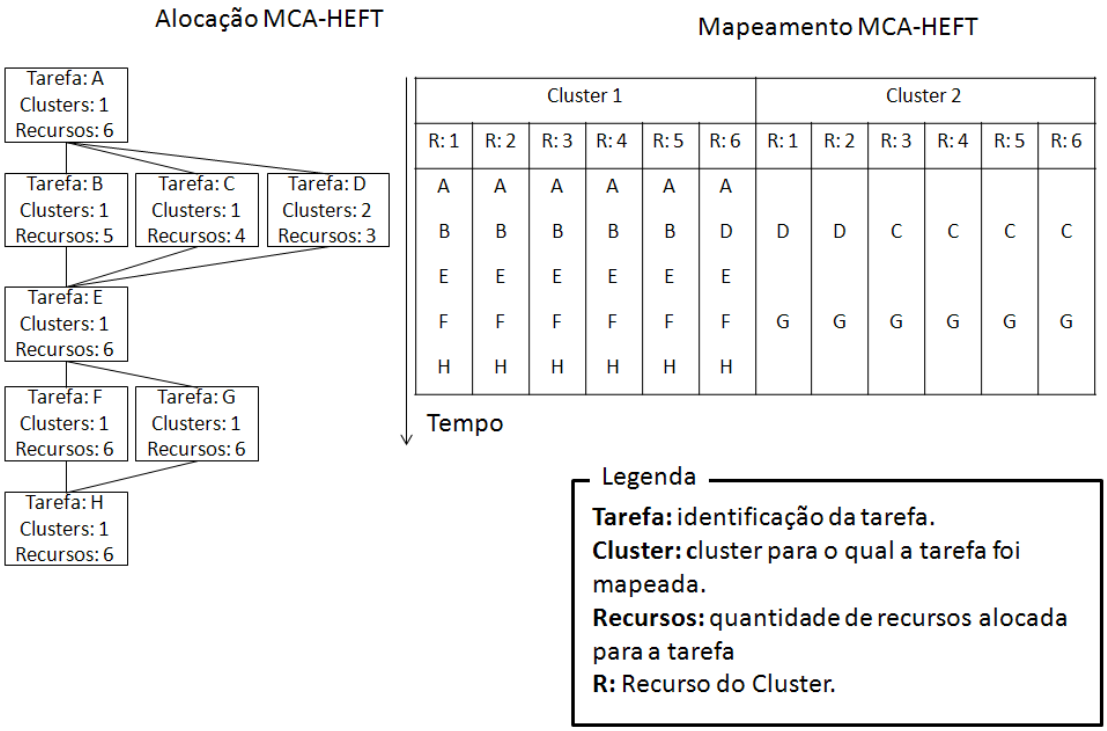

Figura 7. Alocação e Mapeamento Realizado pela Estratégia MCA-HEFT.

A alocação e mapeamento realizados pela estratégia M-HEFT é mostrada na Figura 8. A estratégia M-HEFT alocou no máximo três recursos para cada tarefa, pois o valor máximo definido foi 50% do total de recursos do *cluster* escolhido.

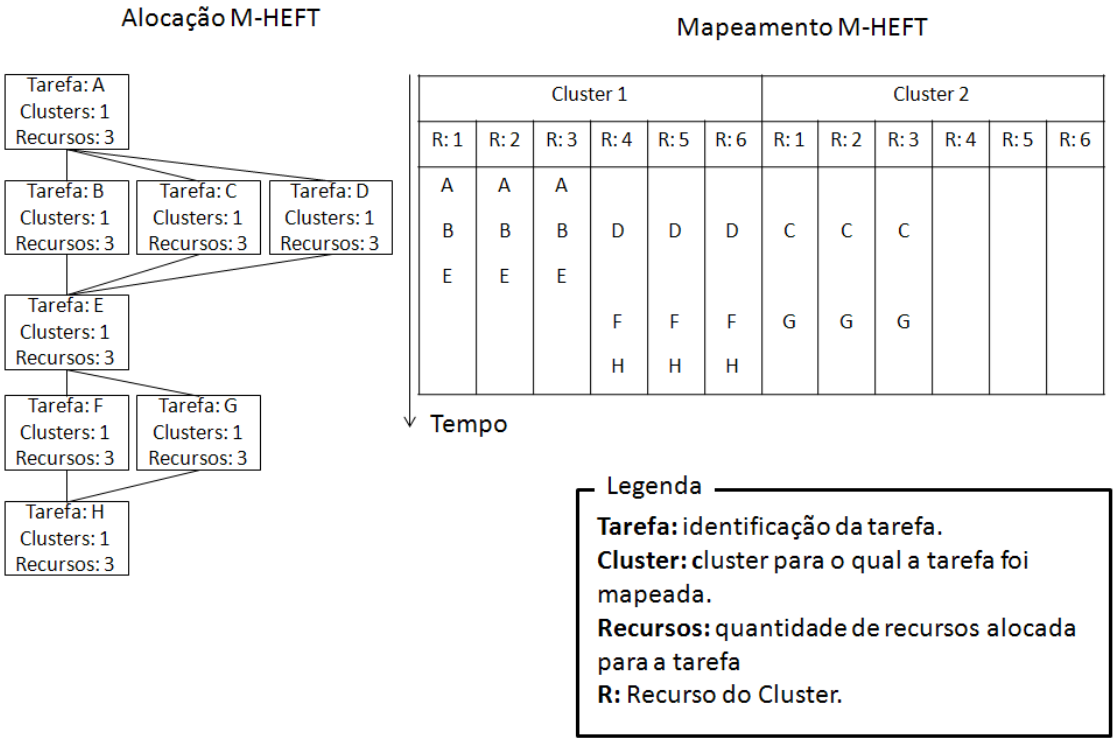

Figura 8. Alocação e Mapeamento Realizado pela Estratégia M-HEFT.

A seção a seguir descreve a estratégia DMCA-HEFT.

4.4. Estratégia DMCA-HEFT (Dynamic Multi-Cluster Allocation-HEFT)

A estratégia DMCA-HEFT (STANZANI, 2013) tem como objetivo realizar o escalonamento de *workflows* em ambientes dinâmicos, que são caracterizados como sendo ambientes nos quais os *workflows* podem ser submetidos em diferentes momentos, sem que o escalonador os conheça antecipadamente.

Um plano de escalonamento gerado pelas estratégias MCA-HEFT e M-HEFT considera a execução de apenas um único *workflow* e que o ambiente será dedicado à execução desse workflow. Dessa forma, quando diversos *workflows* são submetidos em diferentes momentos, uma fila de submissão é criada, dessa forma, quando a última tarefa do *workflow* em execução é finalizada a primeira tarefa do próximo *workflow* entra em execução.

Os planos de escalonamento gerados pelas estratégias MCA-HEFT e M-HEFT possuem intervalos de tempos nos quais os recursos ficam ociosos. Tais intervalos ocorrem entre os diferentes níveis do workflow, devido as

dependências de dados entre as tarefas. Tais intervalos são identificados como GAPs e são definidos como o intervalo de tempo entre o fim da execução de uma tarefa e o início da execução da próxima tarefa mapeada para esse recurso.

A estratégia DMCA-HEFT identifica os GAPs a partir dos planos de escalonamento gerados para os *workflows* em execução, e aplica a técnica de *backfilling*, adiantando a execução de tarefas de novos *workflows* submetidos para esses GAPs. Ao adiantar uma tarefa o tempo de execução dos *workflows* em execução pode sofrer um aumento que é compensado pela diminuição do tempo de execução de todos os *workflows*.

A Figura 9 mostra um exemplo de escalonamento gerado para um *workflow* composto por cinco tarefas em um ambiente com dois *clusters* com seis recursos cada um. De acordo com o plano de escalonamento gerado é possível perceber que diversos GAPs de diferentes tamanhos foram gerados. Nesse exemplo, três tarefas do *workflow* submetido foram adiantadas para GAPs identificados no plano de escalonamento do *workflow* que está em execução.

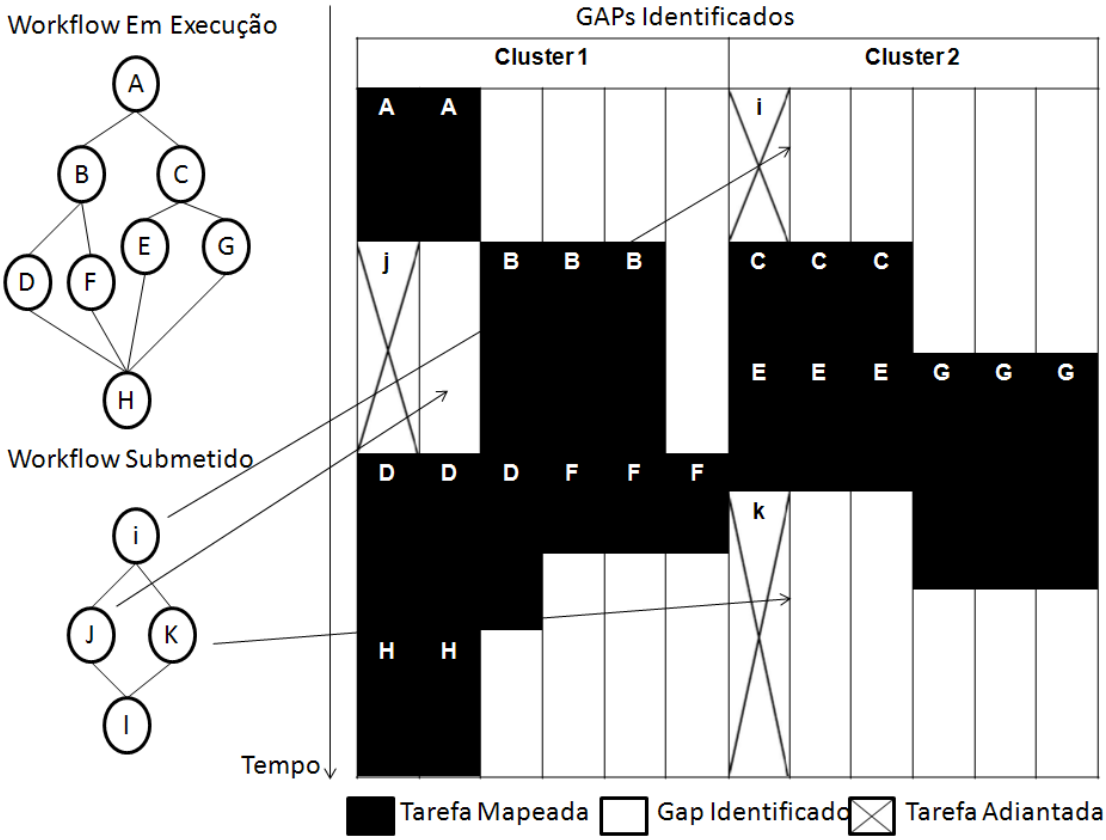

Figura 9. GAPs Gerados Pelo Plano de Escalonamento.

O DMCA-HEFT pode ser utilizado em conjunto com as estratégias MCA-HEFT e M-HEFT.

4.4.1. Sequencia de Passos da Estratégia DMCA-HEFT

A estratégia DMCA-HEFT é executada quando um novo *workflow* é submetido. Nesse momento a rotina realiza três passos: o primeiro é identificar todos os GAPs presentes nos planos de escalonamento dos *workflows* em execução. O segundo passo é verificar para cada tarefa do *workflow* quais podem ser adiantadas para um dos GAPs. O terceiro passo é adiantar a tarefa para o GAP encontrado.

A identificação de GAPs nos planos de escalonamento é realizada executando os seguintes passos:

- Para cada recurso do ambiente é realizado um laço por todas as tarefas mapeadas para esse recurso identificando a hora de fim

estimada de cada tarefa e a hora de início da próxima tarefa mapeada para o recurso.

- Caso a diferença seja maior que zero esse intervalo é registrado como um GAP identificado pela tarefa anterior ao GAP, tarefa posterior ao GAP, e nível da tarefa anterior ao GAP.

O processo de adiantar uma tarefa para um GAP é realizado executando os seguintes passos:

- Para cada tarefa do *workflow* submetido é realizado um laço por todos os GAPs identificados, buscando um conjunto de GAPs em um mesmo nível que possua um tamanho de intervalo de tempo maior que o tempo estimado da tarefa.
- Caso o conjunto de GAPs seja encontrado a tarefa é adiantada para esse GAP. O processo é feito incluindo duas novas dependências relacionadas com a tarefa adiantada: uma da tarefa anterior ao GAP para a tarefa adiantada e outra da tarefa adiantada para a próxima tarefa do GAP. Isso é realizado para garantir que a tarefa adiantada não será executada concorrentemente com outras tarefas nos recursos selecionados.

Depois que uma tarefa é adiantada os GAPs de níveis anteriores são desconsiderados para as próximas tarefas desse workflow. Isso é feito para impedir que uma tarefa seja adiantada sem que as tarefas antecessoras tenham sido adiantadas.

A rotina termina quando todas as tarefas do *workflow* foram adiantadas ou quando uma das tarefas do *workflow* não puder ser adiantada.

Quando a rotina termina uma dependência é criada entre a última tarefa do *workflow* em execução e a primeira tarefa que não foi adiantada, e a estratégia MCA-HEFT ou M-HEFT é aplicada nas tarefas restantes.

5. Implementação das Estratégias Propostas em um Simulador

Nesta seção é apresentada a implementação das estratégias propostas na Seção 4. Será mostrado o modelo de componentes do simulador, a representação de dados do simulador, o ambiente de simulação, a utilização da estratégia proposta em um ambiente de grade computacional e a metodologia utilizada para especificar *workflows* no simulador.

5.1. Implementação Utilizando Simulador

As estratégias foram implementadas e avaliadas em um simulador. A avaliação foi realizada utilizando simulação por dois motivos. O primeiro motivo é por ser uma técnica eficiente para estudar estratégias de escalonamento de maneira repetitiva em um ambiente controlado. Pois o simulador permite que diferentes configurações de ambientes com as mesmas características de um ambiente real sejam montadas. O segundo motivo é que é possível avaliar estratégias de escalonamento considerando cenários com grandes quantidades de recursos utilizados de maneira dedicada.

O simulador foi desenvolvido utilizando o SimGrid Toolkit versão 3.7.1 (CASANOVA; LEGRAND; QUINSON, 2008), usando o framework MSG. O framework MSG permite especificar ambientes de execução e tarefas. A partir dessas especificações é possível aplicar diferentes estratégias de escalonamento de tarefas nos recursos, executar as tarefas e medir o tempo de execução.

O framework SimDag[7] também faz parte do SimGrid e é voltado a simulação de tarefas especificadas como *workflows*, porém, no SimDag não é possível definir que uma tarefa pode ser executada em múltiplos recursos. Nesse sentido, foi utilizado o framework MSG adicionando um controle de precedência entre as tarefas por meio de troca de mensagens. Uma vantagem de usar o framework MSG é a possibilidade de representar tarefas com paralelismo interno utilizando diretivas MPI.

[7] http://simgrid.gforge.inria.fr/simgrid/latest/doc/group__SD__API.html

Os recursos que compõem o ambiente de execução são computadores e redes de interconexão. O desempenho dos recursos é definido de acordo com três aspectos: capacidade de processamento medido em MIPS, largura de banda de rede e latência de rede. Tais aspectos podem ser definidos de maneira fixa, ou sofrendo variações ao longo do tempo.

Os computadores podem ser organizados em *clusters*, que são definidos como um conjunto de computadores e uma rede de interconexão entre os computadores desse *cluster*.

A submissão dos *workflows* é definida em um arquivo que especifica cada submissão de *workflow* como um evento ao longo do tempo.

A seguir, é apresentada a estrutura do simulador.

5.2. Estrutura do Simulador

O simulador foi implementado utilizando um modelo de dados para representar o ambiente e as tarefas, e um conjunto de componentes que controla a execução dos *workflows*. Na Seção 5.2.1 é descrito o modelo de dados e na Seção 5.2.2 é descrito o modelo de componentes.

5.2.1. Modelo de Dados do Simulador

O modelo de dados definido para representar as tarefas, o ambiente de grade computacional composto por múltiplos *clusters* e os planos de escalonamento criados pelas estratégias são mostrados na Figura 10.

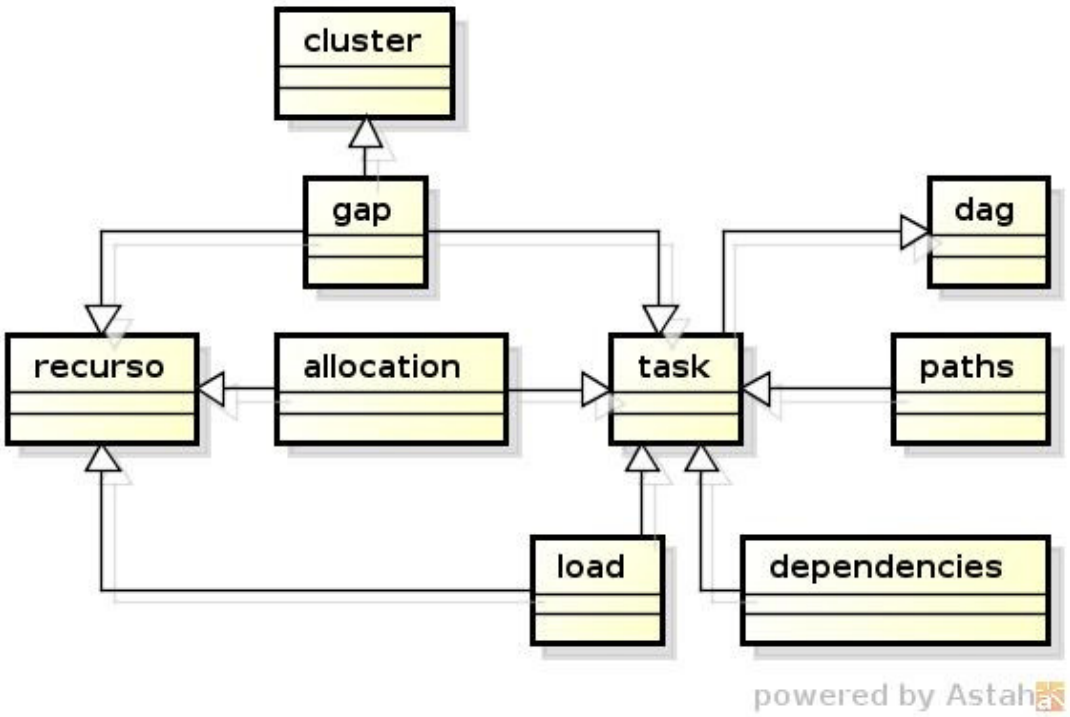

Figura 10. Modelo Entidade-Relacionamento do Simulador.

Na tabela **task** são armazenadas as tarefas de todos os *workflows* submetidos. Na tabela **DAG** são armazenados os *workflows*. As dependências entre as tarefas são armazenadas na tabela **dependencies**. Os caminhos existentes entre a primeira e a última tarefa do *workflow* são armazenados na tabela **path**. Na tabela **allocation** é armazenado o número de recursos que foram alocados para cada tarefa. Na tabela **load** é armazenado o mapeamento de cada tarefa nos recursos. Na tabela **GAP** são armazenados os espaços ociosos de tempo presentes em todos os recursos. Na tabela **cluster** são armazenadas informações sobre os *clusters*. Na tabela **recurso** são armazenadas informações sobre os recursos de cada *cluster*.

5.2.2. Modelo de Componentes do Simulador

O modelo de componentes do simulador é mostrado na Figura 11 e é explicado a seguir.

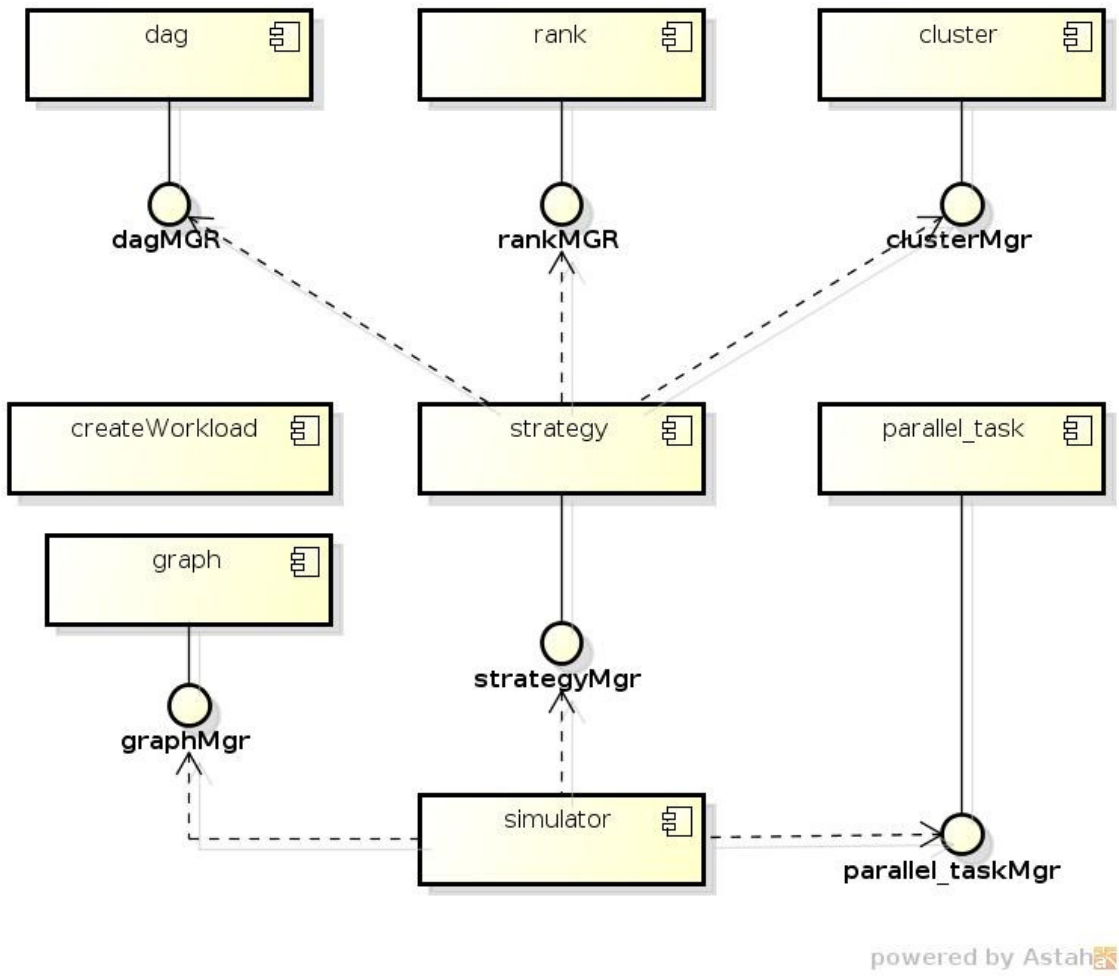

Figura 11. Modelo de Componentes do Simulador.

O componente **simulator** inicia a simulação e monitora dois eventos: *execute* e *scheduling*. O evento *scheduling* indica que um novo *workflow* foi submetido e informa a estrutura do *workflow* submetido para que seja feito o escalonamento. O evento *execute* indica que a tarefa entrará em execução.

O componente **graph** provê suporte para obter todos os caminhos possíveis entre a primeira e última tarefa do workflow. A partir desses caminhos é feito o cálculo do caminho crítico de cada tarefa do workflow. Tal componente foi implementado com suporte da biblioteca igraph (CSÁRDI; NEPUSZ, 2006).

O componente **cluster** provê suporte para monitoração da carga dos recursos do *cluster*.

O componente **parallel_task** provê suporte à execução de tarefas nos recursos.

O componente **CreateWorkload** foi desenvolvido com suporte do gerador

sintético de *workflows* (Synthetic DAG generation) (CASANOVA; DESPREZ; SUTER, 2010). A geração de *workflows* é definida de acordo com os seguintes parâmetros: número de níveis do *workflow* e número mínimo e máximo de tarefas por nível. Baseado nesses parâmetros o gerador define valores aleatórios para custo computacional, custo de comunicação e porção paralela de cada tarefa com paralelismo interno do workflow.

O componente **strategy** realiza o escalonamento de um *workflow* submetido utilizando uma das seguintes estratégias MCA-HEFT, M-HEFT, DMCA-HEFT em conjunto com MCA-HEFT ou DMCA-HEFT em conjunto com M-HEFT.

O componente **rank** realiza o cálculo do *rank* de cada tarefa.

O componente **dag** retorna informações sobre os *workflows*, tarefas, e dependências entre as tarefas.

5.3. Especificação da simulação

Os *workflows* são especificados utilizando dois arquivos chamados **tarefas** e **dependências**. No arquivo **tarefas** cada tarefa de cada *workflow* é especificada indicando:

- Custo computacional medido em instruções;
- Custo de comunicação intra-tarefa medido em bytes;
- Tipo da tarefa (Sequencial ou com Paralelismo Interno);
- Porção paralela.

O arquivo **dependências** registra a relação de dependências entre todas as tarefas dos *workflows*.

Um arquivo XML chamado **plataforma** especifica a plataforma computacional utilizada na simulação.

Um arquivo chamado **eventos** lista os momentos nos quais cada *workflow* será submetido.

O simulador utiliza os arquivos **tarefas**, **dependências**, **plataforma** e **evento** para realizar a simulação. Ao final da simulação, um arquivo é gerado contendo a descrição detalhada da execução.

Na Figura 12 é mostrado um exemplo de submissão de dois *workflows*.

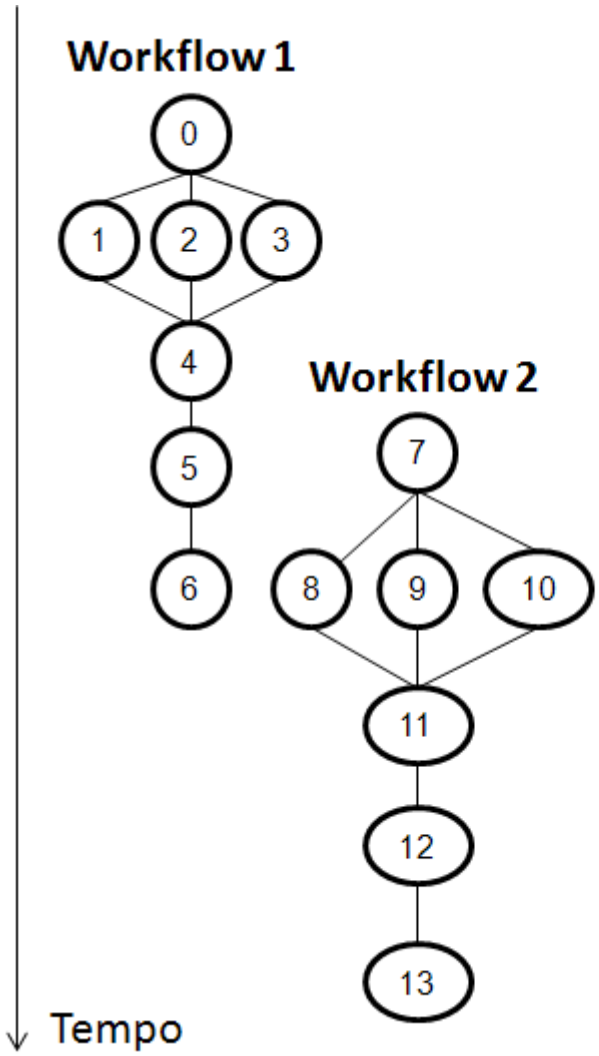

Figura 12. Submissão de Dois *Workflows* em Momentos Diferentes.

Os arquivos **tarefas** e **dependências** do *workflow* 1 é mostrado na Figura 13.

Workflow 1

Dependências	
Pai	Filho
0	1
0	2
0	3
1	4
2	4
3	4
4	5
5	6

Tarefas				
tarefa	$comp_t$	$comm_t$	$Type_t$	pf_t
0	400020	0	1	0.1
1	81003060111	41200000024	0	0.3
2	400008	41200000024	0	0.3
3	81003060545	41200000024	0	0.3
4	101003060794	22200000037	0	0.4
5	96003060477	41200000024	0	0.3
6	101003060866	0	1	0.1

Figura 13. Arquivo de Descrição das Dependências e Tarefas do *Workflow* 1.

Os arquivos **tarefas** e **dependências** do *workflow* 2 é mostrado na Figura 14.

Workflow 2

Dependências	
Pai	Filho
7	8
7	9
7	10
8	11
9	11
10	11
11	12
12	13

Tarefas				
tarefa	$comp_t$	$comm_t$	$Type_t$	pf_t
7	400084	0	1	1
8	96003060510	41200000020	0	0.1
9	96003060111	15200000042	0	0.9
10	81003060545	75200000027	0	0.8
11	400084	0	1	1
12	500084	0	1	1
13	600084	0	1	1

Figura 14. Arquivo de Descrição das Dependências e Tarefas do *Workflow* 2.

O primeiro *workflow* foi submetido no início e o segundo *workflow* 16 segundos depois. A sequência de eventos de submissão está especificada no

arquivo **evento** e é mostrada na Figura 15. O arquivo **evento** especifica dois tipos de eventos *scheduling* e *execute*. O evento *scheduling* indica que um *workflow* foi submetido, nesse momento um plano de escalonamento é gerado para o *workflow* submetido. O evento *execute* indica que a tarefa do *workflow* submetido entrará em execução, nesse momento a tarefa é colocada em execução no recurso definido no plano de escalonamento.

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM "http://simgrid.gforge.inria.fr/simgrid.dtd">
<platform version="3">
<process host="multimedia-consortium-1.das3.nl" function="scheduling" start_time="1">
<argument value="7"/><argument value="0"/><argument value="300"/><argument value="0"/></process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="1.1"><argument value="0"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="1.1"><argument value="1"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="1.1"><argument value="2"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="1.1"><argument value="3"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="1.1"><argument value="4"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="1.1"><argument value="5"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="1.1"><argument value="6"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="scheduling" start_time="16">
<argument value="7"/> <argument value="1"/><argument value="300"/><argument value="0"/></process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="16.1"><argument value="7"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="16.1"><argument value="8"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="16.1"><argument value="9"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="16.1"><argument value="10"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="16.1"><argument value="11"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="16.1"><argument value="12"/>
</process>
<process host="multimedia-consortium-1.das3.nl" function="execute" start_time="16.1"><argument value="13"/>
</process>
</platform>
```

Figura 15. Arquivo Eventos Para Configuração de Submissão de *Workflows*.

A seção a seguir descreve o procedimento adotado para definir as características de custo computacional e custo de comunicação intra-tarefa das tarefas.

5.3.1. Especificação das tarefas

Para especificar cada tarefa de cada *workflow* que será utilizada na simulação é necessário conhecer o tempo de execução sequencial de cada

tarefa em um *cluster* de referência, o tempo de execução das tarefas variando a quantidade de recursos, o desempenho dos recursos que compõem o *cluster* em termos de instruções e a capacidade da rede.

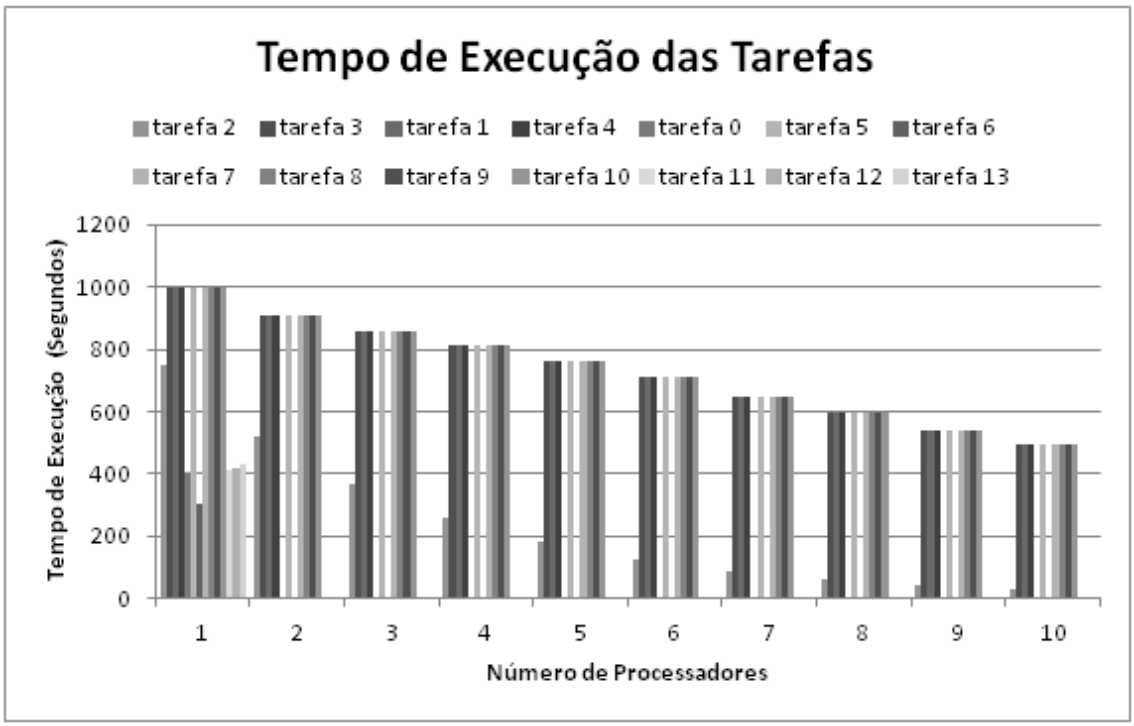

Figura 16. Execução das Tarefas do *Workflow* em um *Cluster* de Referência.

A definição do custo computacional e custo de comunicação intra-tarefa são baseadas no tempo de execução de uma tarefa em um *cluster* de referência. A Figura 16 mostra o tempo de execução das tarefas dos *workflows* mostrados na Figura 12 em um *cluster* de referência.

Para obter uma estimativa do custo computacional é necessário dividir o tempo de execução sequencial de cada tarefa em segundos pelo MIPS do recurso utilizado. A porção paralela de cada tarefa é obtida a partir do *speedup* utilizando a equação (2).

Para obter a estimativa de custo de comunicação intra-tarefa é necessário medir o tráfego entre os recursos e a quantidade de recursos utilizados.

O tempo de processamento em cada recurso utilizado para executar uma tarefa com paralelismo interno é calculado com base no custo computacional e custo de comunicação intra-tarefa, de acordo com a equação (11). O custo em cada recurso diminui a medida que novos recursos são atribuídos à execução

da tarefa, de acordo com a porção paralela da tarefa.

$$CustoComputacionalNó_t = comp_t/alloc_t \quad (11)$$

A comunicação intra-tarefa é representada como uma matriz na qual cada posição representa uma comunicação entre processos. Dessa forma, a quantidade de comunicação é dividida em partes iguais e atribuída a cada posição da matriz de acordo com a equação (12).

$$CustoComunicaçãoNó_t = comm_t/(alloc_t * alloc_t) \quad (12)$$

Uma tarefa t será utilizada para mostrar como calcular o custo computacional em cada recurso e o custo de comunicação entre os recursos que executarão a tarefa.

A tarefa t possui as seguintes características:

- Instruções: 10920000;
- Custo de comunicação: 250 MB;
- Porção paralela: 0,4.

A Figura 17 mostra a divisão de custo computacional e de comunicação aplicando as equações 11 e 12, considerando a execução em dois recursos e em três recursos.

Execução Paralela utilizando 2 recursos:
Instruções computadas em cada recurso: 6552000

	Nó 0	Nó 1
Nó 0	Comunicação: 0,0 Comm: 62,5 Mb	Comunicação: 0,1 Comm: 62,5 Mb
Nó 1	Comunicação: 1,0 Comm: 62,5 Mb	Comunicação: 1,1 Comm: 62,5 Mb

Legenda

Comunicação: Indica o nó de origem e o nó de destino para envio dos dados.
Comm: Quantidade de dados a ser trafegado entre o nó de origem e nó de destino.

Execução Paralela utilizando 3 recursos:
Instruções computadas em cada recurso: 3931200

	Nó 0	Nó 1	Nó 2
Nó 0	Comunicação: 0,0 Comm: 27,7 Mb	Comunicação: 0,1 Comm: 27,7 Mb	Comunicação: 0,2 Comm: 27,7 Mb
Nó 1	Comunicação: 1,0 Comm: 27,7 Mb	Comunicação: 1,1 Comm: 27,7 Mb	Comunicação: 1,2 Comm: 27,7 Mb
Nó 2	Comunicação: 2,0 Comm: 27,7 Mb	Comunicação: 2,1 Comm: 27,7 Mb	Comunicação: 2,2 Comm: 27,7 Mb

Figura 17. Custo Computacional e Matriz de Comunicação Entre os Recursos que Executam a Tarefa.

5.4. Implementação das Estratégias

As estratégias propostas nesta tese utilizam um modelo de dados para representar os *workflows* submetidos e o estado dos recursos do ambiente para produzir planos de escalonamento de *workflows*. As informações sobre os *workflows* submetidos são mostradas na Figura 12, e as informações sobre o estado dos recursos são obtidas utilizando a API do simulador SimGrid. Tal modelo é adequado para produzir planos de escalonamento que serão executados no ambiente de simulação do SimGrid.

Para que as estratégias propostas produzam planos de escalonamento que possam ser executados em ambientes de produção é necessário atender três requisitos:

- O primeiro requisito é que a informação sobre os *workflows* submetidos seja transformada para o formato utilizado pelas estratégias mostrado na

Figura 13;

- O segundo requisito é alterar o componente **cluster** que realiza a consulta sobre o estado dos recursos, para utilizar o mecanismo disponibilizado pelo *middleware* utilizado no ambiente de grade computacional;
- O terceiro requisito é gerar um plano de escalonamento no formato utilizado pelo ScWFMS.

Para mostrar como tais requisitos podem ser atendidos será utilizado um ambiente de exemplo implementado utilizando o *Middleware* Globus (FOSTER, 2005), que é um dos *Middlewares* mais utilizados.

Diversos ScWFMS podem ser utilizados para controlar a execução de *workflows* como uma camada acima do Globus, tais como, Pegasus (DEELMAN et al., 2005), Askalon (FAHRINGER et al., 2005), GridFlow (CAO et al., 2003) e o GCSE (PAULA, 2009) . Nesse exemplo, será considerado o GCSE, por prover suporte à execução de tarefas com paralelismo interno em múltiplos *clusters*, e porque implementa o escalonamento de *workflows* em um componente separado do componente de execução de *workflows*.

A Figura 18 mostra o ambiente de exemplo que é composto por dois domínios administrativos A e B, sendo que o domínio A possui dois *clusters* e o domínio B possui um *cluster*. O *Middleware* Globus (FOSTER, 2005) é utilizado para prover acesso aos recursos, o GCSE controla a execução dos *workflows* em uma camada acima do ambiente computacional, o gateway é utilizado para prover suporte à execução de tarefas com paralelismo interno em múltiplos *cluster* (MASSETTO et al., 2011), e um componente de escalonamento produz planos de escalonamento para o ScWFMS.

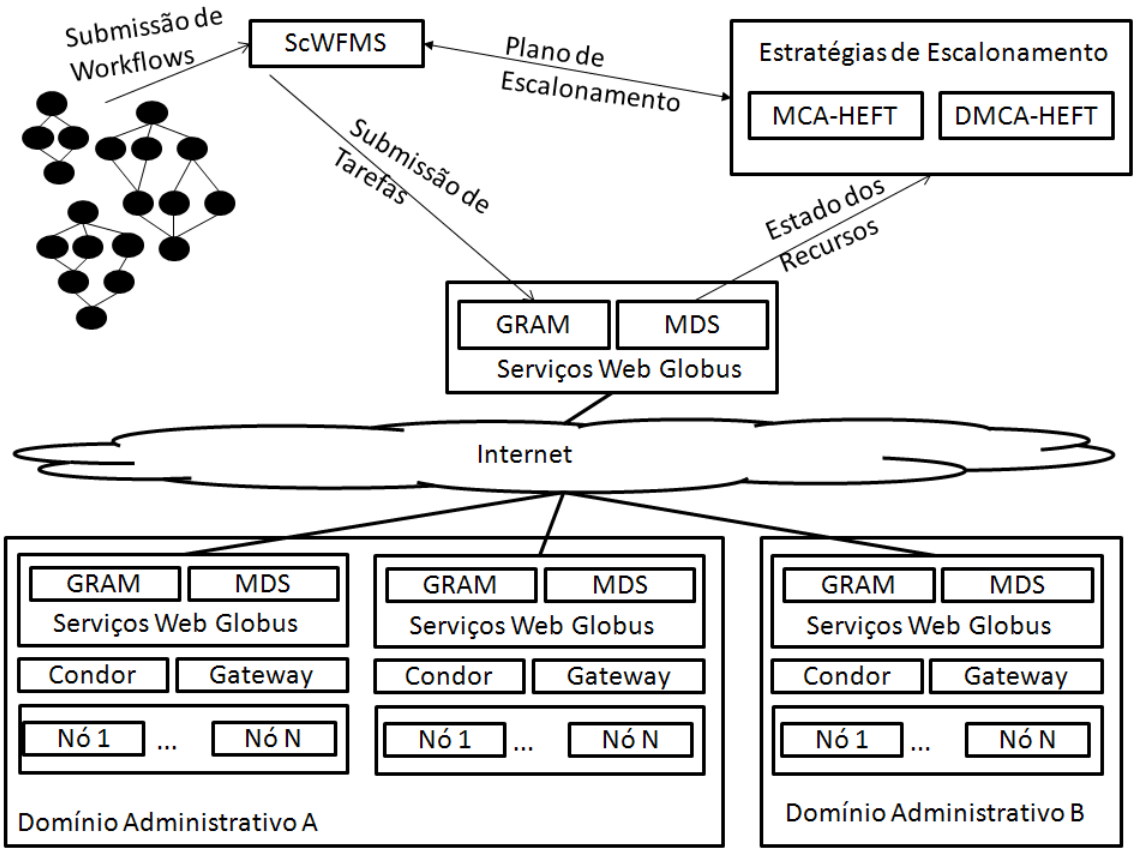

Figura 18. Uso das Estratégias Propostas em um Ambiente de Grade Computacional Implementado com Suporte do *Middleware* Globus.

Para atender o primeiro requisito o GCSE oferece o componente **GCSEJobRequest** que provê informações sobre todos as tarefas submetidas.

Para atender o segundo requisito, o componente **cluster** deve ser adaptado pode obter informações sobre os recursos a partir de componentes de informação do *middleware* de grade computacional, que é chamado MDS (*Monitoring and Discovering Services*) (HE et al., 2005) (SCHOPF et al., 2006). A partir desse componente é possível descobrir os recursos disponibilizados por cada *cluster*, bem como, a carga de cada recurso.

Para atender o terceiro requisito é necessário adaptar o componente **Co-Allocation Manager** do GCSE, que é o componente responsável por realizar o escalonamento dos *workflows* submetidos ao GCSE, para que seja utilizado as estratégias MCA-HEFT ou DMCA-HEFT.

6. Avaliação das Estratégias Propostas

Nesta seção é mostrada a avaliação de desempenho das estratégias MCA-HEFT e DMCA-HEFT.

A estratégia MCA-HEFT é avaliada comparando o tempo de execução quando aplicados os planos de escalonamento gerados pelas estratégias MCA-HEFT e M-HEFT. Foi escolhida a estratégia M-HEFT porque é uma estratégia baseada na estratégia HEFT, que apresenta bons resultados para escalonamento de *workflows* com tarefas sequenciais e com paralelismo interno.

A estratégia DMCA-HEFT é avaliada comparando o tempo de execução quando aplicados os planos de escalonamento gerados, pela estratégia DMCA-HEFT em conjunto com a estratégia MCA-HEFT e pela estratégia MCA-HEFT.

Na Seção 6.1 é apresentada a caracterização dos ambientes de testes, na Seção 6.2 é apresentada a caracterização da carga de entrada, na Seção 6.3 são descritos os experimentos realizados no ambiente de simulação e as respectivas análises, e na Seção 6.4 é apresentada uma discussão com relação à avaliação de desempenho realizada.

6.1. Caracterização do Ambiente de Testes

O ambiente de testes criado para realizar a avaliação de desempenho é composto por quatro *clusters*, compostos por quarenta recursos cada um, que estão montados em domínios administrativos distintos.

Cada *cluster* possui um *switch* para prover suporte à comunicação entre os recursos internos, e um roteador para prover suporte à comunicação entre os nós de entrada de cada *cluster*. A largura de banda do *switch* de cada *cluster* é 1 GB/s e a latência de rede é 50 µs. A largura de banda do roteador é 100 MB/s e a latência de rede é 500 µs.

A descrição completa do ambiente de testes usado na simulação está no Apêndice A.

6.2. Caracterização da Carga de Entrada

Para cada experimento foi definida uma carga de entrada composta por um conjunto de *workflows* gerados aleatoriamente, utilizando o componente **CreateWorkload** com os seguintes parâmetros:

- Cada *workflow* pode possuir formato *fork-join* ou pode possuir outro formato aleatório;
- Cada *workflow* contém um conjunto entre 4 e 30 tarefas;
- Cada tarefa do *workflow* possui um valor aleatório de porção paralela;
- O custo computacional de cada tarefa do *workflow* é igual a um dos seguintes valores em MIPS: 31400000000000, 2900000000000, 3000000000000, 3400000000000, 3500000000000, 31000000000000 ou 32000000000000;
- O custo de comunicação interna de cada tarefa do *workflow* é igual a um dos seguintes valores: 1 MB, 29 GB, 93 GB, 316 GB, 745 GB.

Os *workflows* foram definidos com base nesses critérios para avaliar o desempenho das estratégias, considerando diferentes combinações de custo computacional e de custo de comunicação.

A descrição completa de cada *workflow* está presente na Seção 0 Apêndice B.

6.3. Descrição dos Experimentos

A avaliação das estratégias propostas foi realizada utilizando dois experimentos.

O primeiro experimento foi definido para avaliar o desempenho da estratégia MCA-HEFT e possui dois objetivos. O primeiro objetivo é medir o tempo de execução dos *workflows* aplicando os planos de escalonamento gerados pelas estratégias M-HEFT e MCA-HEFT. O segundo objetivo é avaliar o impacto da métrica MCACCR para decidir quais tarefas podem ser mapeadas

para múltiplos *clusters*. O resultado e as respectivas análises desse experimento são mostrados na seção 6.3.1.

O segundo experimento foi definido para avaliar o desempenho da estratégia DMCA-HEFT e tem como objetivo realizar uma comparação do tempo de execução de múltiplos *workflows* submetidos em diferentes momentos, utilizando planos de escalonamento gerados pelas estratégias DMCA-HEFT em conjunto com a estratégia MCA-HEFT e pela estratégia MCA-HEFT. O resultado e as respectivas análises desse experimento são mostrados na seção 6.3.2.

6.3.1. Experimento de Avaliação da Estratégia MCA-HEFT

Esse experimento foi realizado de acordo com a sequência a seguir:

- Foram criados dez *workflows* utilizando o componente **CreateWorkload**, sendo cinco *workflows* com formato *fork-join e* cinco *workflows* com outros formatos.
- Para cada workflow foi gerado um plano de escalonamento utilizando a estratégia M-HEFT e três planos de escalonamento utilizando a estratégia MCA-HEFT. Cada plano foi gerado com um dos seguintes valores de MCA: 1, 1000, 17. Foram escolhidos tais valores para medir o tempo gasto com comunicação executando diferentes quantidades de tarefas em múltiplos *clusters*.
- Para cada workflow foram realizadas execuções utilizando os planos de escalonamento gerados. Para cada execução foi medido o tempo de execução e o tempo gasto com comunicação.

O resultado da execução dos *workflows* do formato *fork-join* é mostrado no gráfico da Figura 19 e o tempo de execução dos *workflows* com outros formatos é mostrado no gráfico da Figura 20. O tempo de comunicação de todos os *workflows* é mostrado na Tabela 3.

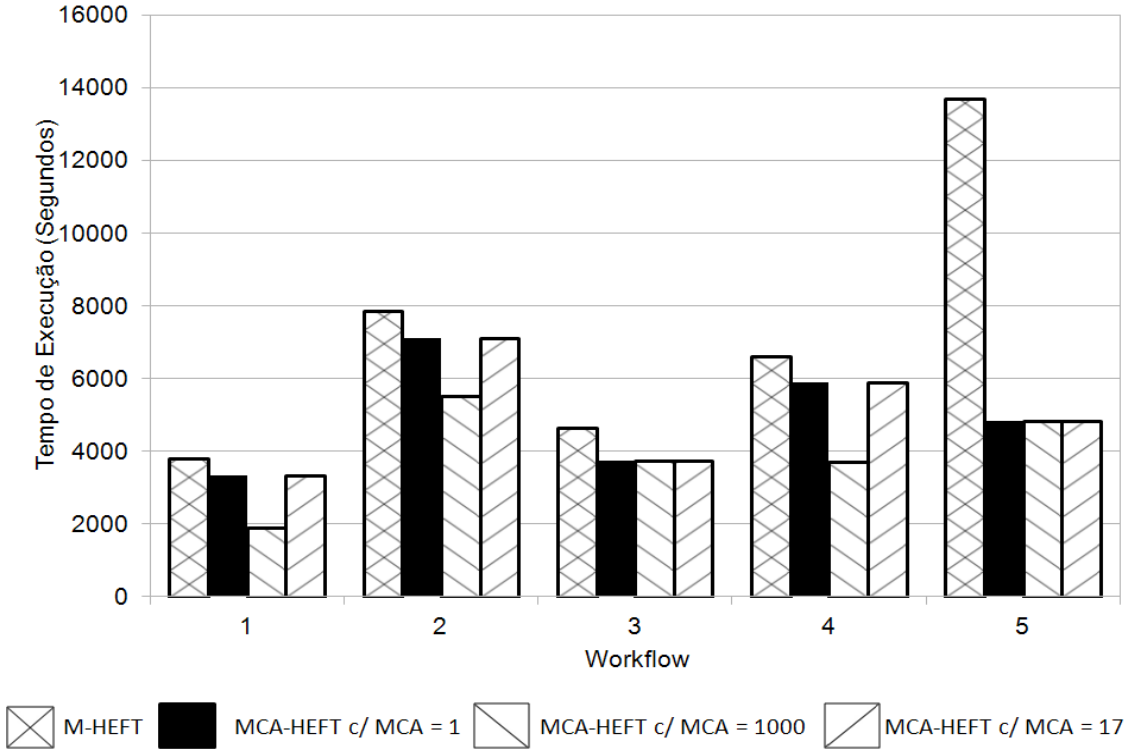

Figura 19. Execução de *Workflows* do Formato *Fork-join* Utilizando M-HEFT e MCA-HEFT com diferentes valores de MCA.

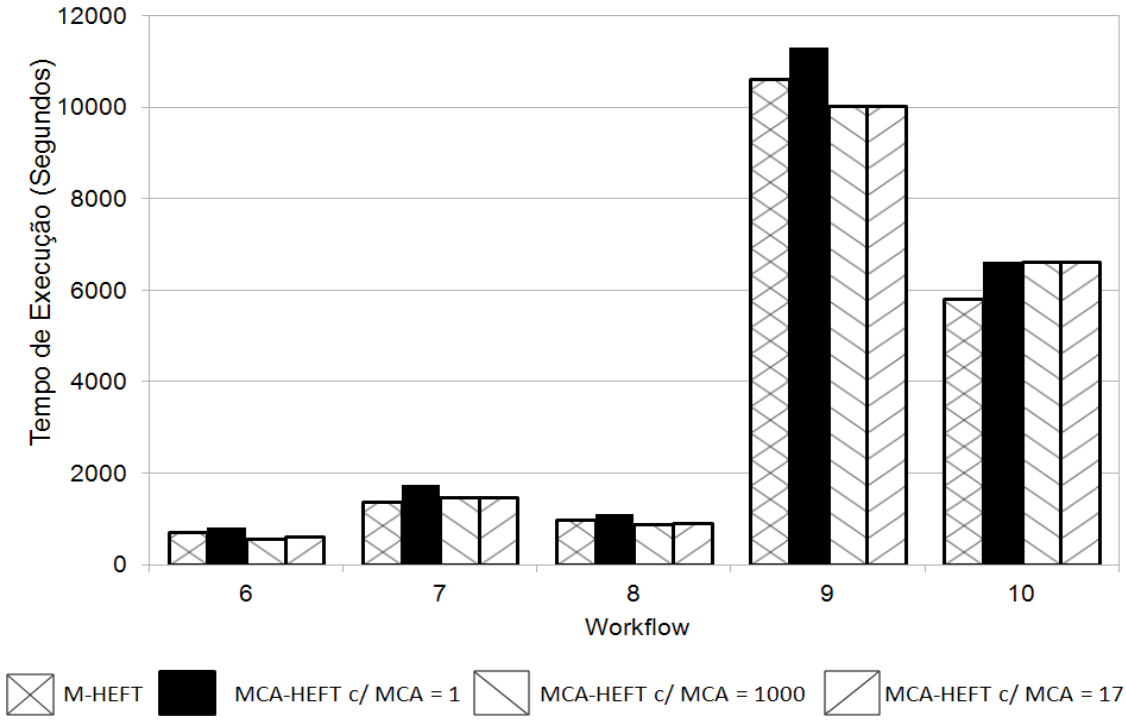

Figura 20. Execução de *Workflows* de outros Formatos Utilizando M-HEFT e MCA-HEFT com diferentes valores de MCA.

Tabela 3. Tempo Gasto com Comunicação Durante a Execução dos *Workflows* Utilizando a Estratégia MCA-HEFT.

Fork-Join					
MCA	**1**	**2**	**3**	**4**	**5**
1	0	694	725	0	2560
1000	265	326	725	398	2560
17	0	694	725	0	2560
Outros Formatos					
MCA	**6**	**7**	**8**	**9**	**10**
1	361	519	3260	725	422
1000	753	519	3480	725	422
17	621	519	3459	725	422

Os resultados desse experimento mostram que a execução de *workflows* do tipo *fork-join* apresenta menor tempo de execução, quando são aplicados os planos de escalonamento gerados pela estratégia MCA-HEFT. Isso acontece por dois motivos:

- O primeiro motivo é que a estratégia MCA-HEFT faz uma alocação de quantidade de recursos maior para cada tarefa do que a estratégia M-HEFT. Nesse sentido, a utilização da estratégia MCA-HEFT faz com que a execução de tarefas no caminho crítico tenha um tempo de execução menor, do que ao utilizar a estratégia M-HEFT;
- O segundo motivo é que a estratégia MCA-HEFT faz uma priorização de escalonamento de tarefas por ordem de nível e de *rank*. Tal priorização faz com que seja realizado um balanceamento do uso de recursos pelas tarefas em um mesmo nível de acordo com o *rank* de cada tarefa. Tal balanceamento permite que o tempo de execução de todas as tarefas em um mesmo nível seja parecido.

A execução de *workflows* de outros formatos apresenta tempo de execução maior, quando é aplicado um plano de escalonamento gerado pela estratégia MCA-HEFT, e em alguns casos, tempo de execução similar. Isso acontece, pois a estratégia MCA-HEFT realiza a alocação por ordem de nível e *rank*, porém para outros formatos de workflow, algumas tarefas independentes podem aparecer em níveis diferentes. Tais tarefas não são identificadas pela estratégia MCA-HEFT como tarefas que podem ser executadas simultaneamente. Nesses casos, a quantidade de recursos alocada para cada tarefa é similar a quantidade alocada pela estratégia M-HEFT.

O custo de comunicação da execução dos *workflows* foi medida nesse experimento, para avaliar o uso de diferentes valores de MCA no escalonamento de *workflows*. A partir dessa medição pode-se verificar quatro aspectos com relação à utilização de diferentes valores de MCA:

- O primeiro aspecto é que nos *workflows* 2 e 6, o custo de rede é maior quando é utilizado um valor baixo de MCA. Isso acontece porque mais tarefas são mapeadas para múltiplos *clusters* e são executadas em uma infraestrutura de comunicação com menor capacidade;
- O segundo aspecto é que nos *workflows* 3, 5, 7, 9 e 10, o custo de rede é independente do valor do MCA. Isso porque nesses *workflows,* poucas tarefas foram mapeadas para múltiplos *clusters* e não houve impacto grande por conta da execução de tarefas em múltiplos *clusters*;
- O terceiro aspecto é que nos *workflows* 1, 4 e 8 foi verificado que ao utilizar um valor alto de MCA poucas tarefas foram mapeadas para múltiplos *clusters*, porém o custo de rede foi mais alto do que utilizando valor 1, o que fez com que tarefas fossem mapeadas para múltiplos *clusters*. Isso ocorreu porque houve um aumento do uso da infraestrutura de rede interna dos *clusters*;
- O quarto aspecto é que a utilização de um mesmo valor de MCA para geração de planos de escalonamento, apresenta impacto no custo de comunicação diferente para cada workflow. Isso ocorre pois a ocupação de rede no momento da execução da tarefa não é levada em conta pela

estratégia MCA-HEFT.

6.3.2. Experimento de Avaliação da Estratégia DMCA-HEFT

Esse experimento foi realizado de acordo com a sequência a seguir:

- Foi definido dois grupos de cinco conjuntos de *workflows* no formato *fork-join,* utilizando o componente **CreateWorkload**:
    - O primeiro grupo é composto por conjuntos de *workflows*, nos quais, o primeiro *workflow* submetido possui muitas tarefas por nível*,* e os *workflows* que são submetidos depois são compostos por *workflows* com poucas tarefas em um nível;
    - O segundo grupo é composto por conjuntos de *workflows* que possuem muitas tarefas por nível;
- Para cada conjunto de *workflows*, foi gerado um plano de escalonamento utilizando a estratégia DMCA-HEFT em conjunto com a estratégia MCA-HEFT, e um plano de escalonamento utilizando a estratégia MCA-HEFT. O valor de MCA escolhido para realizar os experimentos foi 17. Foi escolhido esse valor, pois apresentou os melhores tempos de execução dos *workflows* desse experimento.
- Execução de cada conjunto de *workflows* utilizando os planos de escalonamento gerados. Para cada execução foi medido o tempo de execução do conjunto de *workflows*.

Os resultados da execução dos *workflows* do primeiro grupo são mostrados no gráfico da Figura 21. Os resultados da execução dos *workflows* do segundo grupo são mostrados no gráfico da Figura 22.

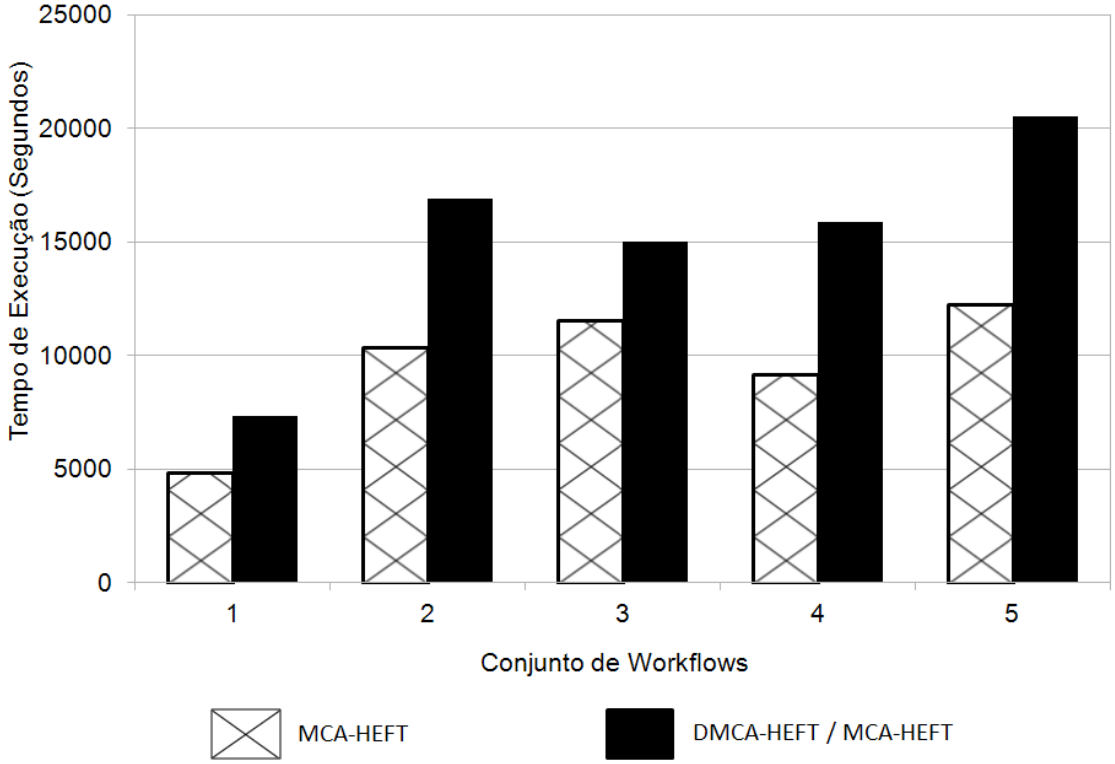

Figura 21.Execução de *Workflows* do Primeiro Grupo Utilizando MCA-HEFT com DMCA-HEFT e MCA-HEFT.

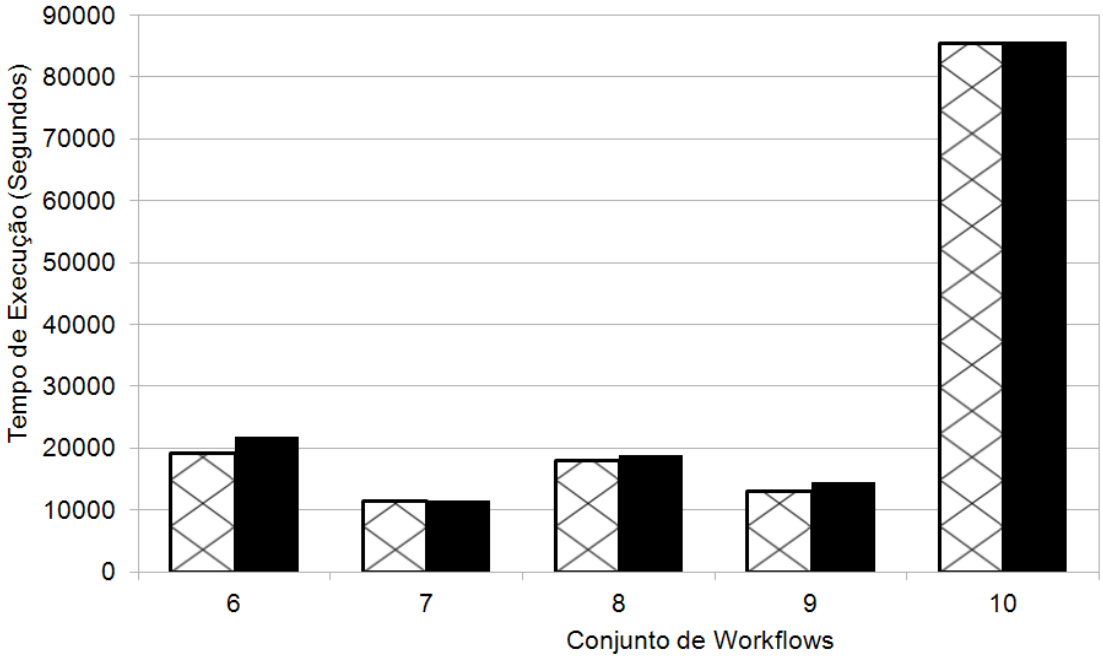

Figura 22. Execução de *Workflows* do Segundo Grupo Utilizando MCA-HEFT com DMCA-HEFT e MCA-HEFT.

A execução dos conjuntos de *workflows* do primeiro grupo apresentou

tempo de execução menor quando foi utilizado o plano de escalonamento gerado pela estratégia DMCA-HEFT em conjunto com a estratégia MCA-HEFT, do que quando foi utilizado o plano de escalonamento gerado pela estratégia MCA-HEFT. Isso aconteceu devido a duas razões:

- Quando é utilizada a estratégia MCA-HEFT cada *workflow* só inicia sua execução após o fim do *workflow* anterior. Nesse sentido, os GAPs que aparecem nos *workflows* não são aproveitados pelos próximos *workflows* submetidos;
- Quando é utilizada a estratégia DMCA-HEFT em conjunto com a estratégia MCA-HEFT, as tarefas dos *workflows* podem ser adiantadas para GAPs gerados pelos *workflows* que estão em execução. O escalonamento dos conjuntos de *workflows* do primeiro grupo gerou muitos GAPs que foram aproveitados. Tais GAPs geraram pouco atraso no tempo de execução dos *workflows* anteriores e alta diminuição no tempo de execução dos *workflows* que tiveram tarefas adiantadas.

A execução dos conjuntos de *workflows* do segundo grupo apresentou tempo de execução similar quando foi utilizado o plano de escalonamento gerado pela estratégia DMCA-HEFT em conjunto com a estratégia MCA-HEFT, do que quando foi utilizado o plano de escalonamento gerado pela estratégia MCA-HEFT. Isso aconteceu devido a duas razões:

- Os GAPs que são gerados pelas tarefas em execução possuem tamanho menor, devido a grande quantidade de tarefas em um mesmo nível que são alocadas para muitos recursos. Nesse sentido, apenas tarefas com baixo custo computacional conseguem ser adiantadas para os GAPs dos *workflows* em execução;
- Em alguns casos as tarefas que são adiantadas estão em um nível do *workflow* que possui outras tarefas com custo computacional muito mais alto do que o da tarefa que foi adiantada. Nesse contexto, a tarefa provoca pouco atraso no *workflow* para o qual foi adiantada, e nenhuma diminuição no tempo de execução do *workflow* ao qual a tarefa pertence.

6.4. Discussão

Os resultados mostram que a estratégia MCA-HEFT é adequada para realizar planos de escalonamento de *workflows* no formato *fork-join,* e não apresenta uma perda significativa de desempenho quando são utilizados outros formatos de *workflows*.

Um aspecto importante identificado na estratégia MCA-HEFT é com relação à definição do valor de MCA que precisa ser feito de maneira automática pela estratégia, pois tal valor possui impacto no tempo de execução e é dependente de diversos fatores, tais como, ocupação da rede no momento que a tarefa será executada, custo computacional, custo de comunicação e porção paralela da tarefa.

Com relação à utilização do DMCA-HEFT verifica-se que ao adiantar tarefas é possível obter menor tempo de execução em diversos casos, aproveitando os GAPs que são gerados pelas dependências de *workflows*. Quando a estratégia DMCA-HEFT não é utilizada muitos GAPs acabam não sendo utilizados, e o desempenho é impactado pelo aumento na ociosidade dos recursos.

Um aspecto que precisa ser levado em conta pela estratégia DMCA-HEFT é o impacto de adiantar uma tarefa de um workflow, no tempo de execução do *workflow* da tarefa, e no tempo de execução do *workflow* ao qual a tarefa foi adiantada. Isso é importante para que esse mapeamento seja realizado, somente quando for possível obter uma diminuição significativa no tempo de execução.

7. Conclusão e Trabalhos Futuros

O suporte à execução de aplicações com grande demanda computacional é uma necessidade identificada em diversas áreas de ciência. Em geral as aplicações são estruturadas como *workflows* compostos por tarefas sequenciais e com paralelismo interno e executadas em ambientes de processamento distribuído, tais como grades computacionais e *clusters*. Nesse contexto, o estudo referente ao escalonamento de *workflows* é essencial para identificar estratégias que possam obter o melhor desempenho possível em tais ambientes.

A abordagem adotada neste trabalho para melhorar a execução de um *workflow* composto por tarefas sequenciais e tarefas com paralelismo interno foi avaliar a vantagem de utilizar recursos de múltiplos *clusters*, para melhorar o desempenho da execução de cada tarefa com paralelismo interno de um workflow, com base nas características das tarefas do workflow.

Outro desafio para realizar o escalonamento de *workflows* em ambientes de grades computacionais compostos por múltiplos *clusters* é o comportamento dinâmico de tais ambientes. Nesta tese esse problema foi abordado verificando a possibilidade de escalonar tarefas de novos *workflows* em recursos ociosos, identificados nos planos de escalonamento dos *workflows* em execução.

A utilização dessas abordagens no desenvolvimento de novas estratégias de escalonamento de *workflows* se mostrou eficiente para obter ganhos de desempenho em diversos cenários. A seção a seguir apresenta as contribuições dessa tese.

7.1. Contribuições

As contribuições dessa tese são:

- Criação de métricas baseadas nas características das tarefas com paralelismo interno para definir a alocação de recursos e para definir o mapeamento para um *cluster* ou para múltiplos *clusters*;
- Proposição de uma estratégia de escalonamento chamada Multi-Cluster Allocation-HEFT (MCA-HEFT), que considera a execução de

cada tarefa com paralelismo interno de um *workflow* em recursos de múltiplos *clusters*;

- Proposição de uma estratégia chamada Dynamic Multi-Cluster Allocation-HEFT (DMCA-HEFT) que realiza o escalonamento de múltiplos *workflows* em ambientes *multi-cluster* dinâmicos, e utiliza a estratégia MCA-HEFT para mapear as tarefas.

7.2. Trabalhos Futuros

A seguir são descritos alguns trabalhos futuros que podem ser desenvolvidos na área de escalonamento de *workflows* compostos por tarefas sequenciais e tarefas com paralelismo interno, baseado nas estratégias propostas:

- Escalonamento de múltiplos *workflows* em ambiente dinâmicos, considerando diferentes aspectos de um ambiente dinâmico, tais como, ocorrência de falhas nos recursos, variações de desempenho nos recursos, usuários com diferentes níveis de prioridade, custo de energia, entre outros;
- Escalonamento de múltiplos *workflows* utilizando as estratégias desenvolvidas em conjunto com estratégias que consideram o custo de comunicação entre as tarefas dos *workflows*;
- Utilização de ferramentas para realizar predição de desempenho em conjunto com as estratégias desenvolvidas;
- Utilização das estratégias desenvolvidas para escalonamento de *workflows* em ambientes de nuvem computacional;
- Análise comparativa das estratégias desenvolvidas com outras estratégias de escalonamento;
- Escrita de artigo para ser submetido em um periódico detalhando a estratégia dinâmica.

Referências Bibliográficas

ALMEIDA, E. S.; BAUER, M. A.; FAZENDA, A. L. **Evaluation of HPC architectures for BRAMS numerical weather model**. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED PROCESSING TECHNIQUES AND APPLICATIONS. 2011

AUMAGE, O.; MERCIER, G. **MPICH/MADIII: a cluster of clusters enabled MPI implementation**Cluster Computing and the Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on. **Anais**...2003

BALLE, S.; PALERMO, D. Enhancing an Open Source Resource Manager with Multi-core/Multi-threaded Support. In: FRACHTENBERG, E.; SCHWIEGELSHOHN, U. (Eds.). **Job Scheduling Strategies for Parallel Processing**. Lecture Notes in Computer Science. [s.l.] Springer Berlin Heidelberg, 2008. v. 4942p. 37–50.

BANSAL, S.; KUMAR, P.; SINGH, K. An improved two-step algorithm for task and data parallel scheduling in distributed memory machines. **Parallel Computing**, v. 32, n. 10, p. 759–774, nov. 2006.

BARBOSA, J. G.; MOREIRA, B. Dynamic scheduling of a batch of parallel task jobs on heterogeneous clusters. **Parallel Computing**, v. 37, n. 8, p. 428–438, ago. 2011.

BARRETO, M.; ÁVILA, R. B.; NAVAUX, P. O. A. **The MultiCluster Model to the Integrated Use of Multiple Workstation Clusters**Proceedings of the 15 IPDPS 2000 Workshops on Parallel and Distributed Processing. **Anais**...: IPDPS '00.London, UK, UK: Springer-Verlag, 2000.

BIRD, I. Computing for the Large Hadron Collider. **Annual Review of Nuclear and Particle Science**, v. 61, n. 1, p. 99–118, 2011.

BITTENCOURT, L. F. et al. **A path clustering heuristic for scheduling taks graphs onto a grid**3rd ACM International Workshop on Middleware for Grid Computing, Grenoble, France. **Anais**...nov. 2005

BITTENCOURT, L. F.; SAKELLARIOU, R.; MADEIRA, E. R. M. **DAG Scheduling Using a Lookahead Variant of the Heterogeneous Earliest Finish Time Algorithm**Proceedings of the 2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing. **Anais**...: PDP '10.Washington, DC, USA: IEEE Computer Society, 2010.

BRAZZEL, OWEN et al. **Running Schlumberger Simulators in a PBS Professional Computing Environment: Integration White Paper and How-To Guide**. [s.l: s.n.].

BUCUR, A. I. D.; EPEMA, D. H. J. Scheduling Policies for Processor Coallocation in Multicluster Systems. **IEEE Transactions on Parallel and Distributed Systems**, v. 18, n. 7, p. 958 –972, jul. 2007.

BUYYA, R. **High Performance Cluster Computing: Architectures and Systems, Vol. 1**. 1. ed. [s.l.] Prentice Hall, 1999.

BUYYA, R.; BAKER, M. **Emerging Technologies for MultiCluster/Grid Computing**Proceedings of the 3rd IEEE International Conference on Cluster Computing. **Anais**...: CLUSTER '01.Washington, DC, USA: IEEE Computer Society, 2001.

BYUN, E.-K. et al. BTS: Resource capacity estimate for time-targeted science workflows. **Journal of Parallel and Distributed Computing**, v. 71, n. 6, p. 848 – 862, 2011.

CAO, H. et al. **DAGMap: Efficient scheduling for DAG grid workflow job**2008 9th IEEE/ACM International Conference on Grid Computing. **Anais**...2008

CAO, J. et al. **GridFlow: workflow management for grid computing**2003

CASANOVA, H.; DESPREZ, F.; SUTER, F. On cluster resource allocation for multiple parallel task graphs. **Journal of Parallel and Distributed Computing**, v. 70, n. 12, p. 1193–1203, dez. 2010.

CASANOVA, H.; LEGRAND, A.; QUINSON, M. **SimGrid: A Generic Framework for Large-Scale Distributed Experiments**Tenth International Conference on Computer Modeling and Simulation, 2008. UKSIM 2008. **Anais**...cambridge, UK: IEEE, abr. 2008

CASAVANT, T. L.; KUHL, J. G. A taxonomy of scheduling in general-purpose distributed computing systems. **IEEE Trans. Softw. Eng.**, v. 14, n. 2, p. 141–154, 1988.

CIRNE, W. et al. **Running Bag-of-Tasks applications on computational grids: the MyGrid approach**2003 International Conference on Parallel Processing, 2003. Proceedings. **Anais**... In: 2003 INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 2003. PROCEEDINGS. 2003

COMITO, C.; TALIA, D.; TRUNFIO, P. Grid services: principles, implementations and use. **Int. J. Web Grid Serv.**, v. 1, n. 1, p. 48–68, ago. 2005.

COUVARES, P. et al. Workflow Management in Condor. In: TAYLOR, I. et al. (Eds.). **Workflows for e-Science**. [s.l.] Springer London, 2007. p. 357–375.

CSÁRDI, G.; NEPUSZ, T. The igraph software package for complex network research. **InterJournal Complex Systems**, p. 1695, 2006.

DEELMAN, E. et al. **GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists**Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing. **Anais**...: HPDC '02.Washington, DC, USA: IEEE Computer Society, 2002.

DEELMAN, E. et al. Mapping Abstract Complex Workflows onto Grid Environments. **Journal of Grid Computing**, v. 1, n. 1, p. 25–39, 2003.

DEELMAN, E. et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. **Sci. Program.**, v. 13, n. 3, p. 219–237, jul. 2005.

DESPREZ, F.; SUTER, F. A Bi-criteria Algorithm for Scheduling Parallel Task Graphs on Clusters. **Cluster Computing and the Grid, IEEE International Symposium on**, v. 0, p. 243–252, 2010.

DUTOT, P. F. et al. Scheduling Parallel Task Graphs on (Almost) Homogeneous Multicluster Platforms. **IEEE Transactions on Parallel and Distributed Systems**, v. 20, n. 7, p. 940–952, 2009.

ERNEMANN, C. et al. **On Advantages of Grid Computing for Parallel Job Scheduling**2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, 2002. **Anais**...2002

FAHRINGER, T. et al. **ASKALON: a Grid application development and computing environment**Grid Computing, 2005. The 6th IEEE/ACM International Workshop on. **Anais**...nov. 2005

FAZENDA, A. L. et al. Challenges and solutions to improve the scalability of an operational regional meteorological forecasting model. **Int. J. High Perform. Syst. Archit.**, v. 3, n. 2/3, p. 87–97, maio 2011.

FEITELSON, D. G. et al. **Theory and Practice in Parallel Job Scheduling**Proceedings of the Job Scheduling Strategies for Parallel Processing. **Anais**...: IPPS '97.London, UK, UK: Springer-Verlag, 1997.

FOSTER, I. T. **The Anatomy of the Grid: Enabling Scalable Virtual Organizations** Springer-Verlag, 2001.

FOSTER, I. T. **Globus Toolkit Version 4: Software for Service-Oriented Systems.** (H. Jin, D. A. Reed, W. Jiang, Eds.): Lecture Notes in Computer Science.Springer, 2005

FREITAS, E. et al. Interactions of an urban heat island and sea-breeze circulations during winter over the metropolitan area of São Paulo, Brazil. **Boundary-Layer Meteorology**, v. 122, n. 1, p. 43–65, 2007.

GRAHAM, R. L. et al. **Open MPI: A High-Performance, Heterogeneous MPI**Cluster Computing, 2006 IEEE International Conference on. **Anais**...set. 2006

GROPP, W.; LUSK, E.; SKJELLUM, A. **Using MPI: portable parallel programming with the message-passing interface**. Cambridge, MA, USA: MIT Press, 1994.

HE, K. et al. Building Grid Monitoring System Based on Globus Toolkit: Architecture and Implementation. In: ZHANG, J.; HE, J.-H.; FU, Y. (Eds.). **Computational and Information Science**. Lecture Notes in Computer Science. [s.l.] Springer Berlin / Heidelberg, 2005. v. 3314p. 353–358.

HE, L. et al. **Mapping DAG-based applications to multiclusters with background workload**: CCGRID '05.Washington, DC, USA: IEEE Computer Society, 2005.

HEY, T.; TREFETHEN, A. E. Cyberinfrastructure for e-Science. **Science**, v. 308, n. 5723, p. 817–821, 2005.

HIRALES-CARBAJAL, A. et al. Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid. **Journal of Grid Computing**, v. 10, n. 2, p. 325–346, 2012.

HSU, C.-C.; HUANG, K.-C.; WANG, F.-J. Online scheduling of workflow applications in grid environments. **Future Generation Computer Systems**, v. 27, n. 6, p. 860 – 870, 2011.

ILAVARASAN, E.; THAMBIDURAI, P.; MAHILMANNAN, R. **Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System**Proceedings of the The 4th International Symposium on Parallel and Distributed Computing. **Anais**...: ISPDC '05.Washington, DC, USA: IEEE Computer Society, 2005.

JONES, W. M. et al. Characterization of Bandwidth-Aware Meta-Schedulers for Co-Allocating Jobs Across Multiple Clusters. **J. Supercomput.**, v. 34, n. 2, p. 135–163, nov. 2005.

KARONIS, N. T.; TOONEN, B.; FOSTER, I. MPICH-G2: a Grid-enabled implementation of the Message Passing Interface. **J. Parallel Distrib. Comput.**, v. 63, n. 5, p. 551–563, 2003.

KELLER, R. et al. Towards Efficient Execution of MPI Applications on the Grid: Porting and Optimization Issues. **Journal of Grid Computing**, v. 1, n. 2, p. 133–149, 2003.

LEE, J. et al. **Adaptive workflow scheduling strategy in service-based grids**Proceedings of the 2nd international conference on Advances in grid and pervasive computing. **Anais**...: GPC'07.Berlin, Heidelberg: Springer-Verlag, 2007.

LI, Q. et al. **An optimal scheduling algorithm for fork-join task graphs**Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. **Anais**... In: PROCEEDINGS OF THE FOURTH INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED COMPUTING, APPLICATIONS AND TECHNOLOGIES, 2003. PDCAT'2003. 2003

LI, Y.; LAN, Z. A Survey of Load Balancing in Grid Computing. In: ZHANG, J.; HE, J.-H.; FU, Y. (Eds.). **Computational and Information Science**. Lecture Notes in Computer Science. [s.l.] Springer Berlin Heidelberg, 2005. p. 280–285.

LIN, H. et al. **Massively parallel genomic sequence search on the Blue Gene/P architecture**High Performance Computing, Networking, Storage and Analysis, 2008. SC 2008. International Conference for. **Anais**...nov. 2008

LOPEZ, M. M.; HEYMANN, E.; SENAR, M. A. **Analysis of Dynamic Heuristics for Workflow Scheduling on Grid Systems**Proceedings of the Proceedings of The Fifth International Symposium on Parallel and Distributed Computing. **Anais**...: ISPDC '06.Washington, DC, USA: IEEE Computer Society, 2006.

MACCHIONI, M. Simulation Streamlines Aircraft Door Development. 2011.

MACHTANS, E. C.; SATO, L. M.; DEPPMAN, A. **Improvement on Scheduling Dependent Tasks for Grid Applications** 2009

MASSETTO, F. et al. A Message Forward Tool for Integration of Clusters of Clusters Based on MPI Architecture. In: HSU, C.-H.; MALYSHKIN, V. (Eds.). **Methods and Tools of Parallel Programming Multicomputers**. Lecture Notes in Computer Science. [s.l.] Springer Berlin / Heidelberg, 2011. v. 6083p. 105–114.

MOHAMED, H. H.; EPEMA, D. H. J. **Experiences with the KOALA co-allocating scheduler in multiclusters**IEEE International Symposium on Cluster Computing and the Grid, 2005. CCGrid 2005. **Anais**...2005

MUÑOZ, M. E. DE S. et al. openModeller: a generic approach to species' potential distribution modelling. **GeoInformatica**, v. 15, n. 1, p. 111–135, 1 jan. 2011.

N'TAKPE, T.; SUTER, F.; CASANOVA, H. **A Comparison of Scheduling Approaches for Mixed-Parallel Applications on Heterogeneous Platforms**. In: PARALLEL AND DISTRIBUTED COMPUTING, 2007. ISPDC '07. SIXTH INTERNATIONAL SYMPOSIUM ON. 2007

NANDAGOPAL, M.; GOKULNATH, K.; UTHARIARAJ, V. R. **Sender initiated decentralized dynamic load balancing for multi cluster computational grid environment**Proceedings of the 1st Amrita ACM-W Celebration on Women in Computing in India. **Anais**...: A2CWiC '10.New York, NY, USA: ACM, 2010.

NETTO, M. A. S.; BUYYA, R. Resource Co-Allocation in Grid Computing Environments. In: **Handbook of Research on P2P and Grid Systems for Service-Oriented Computing: Models, Methodologies and Applications**. [s.l.] IGI Global, 2010. p. 476–494.

NETTO, M.; BUYYA, R. **Rescheduling co-allocation requests based on flexible advance reservations and processor remapping**2008 9th IEEE/ACM International Conference on Grid Computing. **Anais**...out. 2008

NISHIMOTO, K. et al. Numerical Offshore Tank: Development of Numerical Offshore Tank for Ultra Deep Water Oil Production Systems. In: **In: 22nd International Conference on Offshore Mechanics & Arctic Engineering**. [s.l: s.n.].

PANT, A.; JAFRI, H. **Communicating efficiently on cluster based grids with MPICH-VMI**Cluster Computing, 2004 IEEE International Conference on. **Anais**...2004

PAULA, N. C. DE. **Um ambiente de monitoramento de recursos e escalonamento cooperativo de aplicações paralelas em grades computacionais**. Tese de Doutorado—São Paulo: Universidade de São Paulo, 2009.

PINEDO, M. L. **Scheduling: Theory, Algorithms, and Systems**. 3rd. ed. [s.l.] Springer, 2008.

RADULESCU, A.; VAN GEMUND, A. J. C. **A Low-Cost Approach towards Mixed Task and Data Parallel Scheduling**Proceedings of the International Conference on Parallel Processing. **Anais**...: ICPP '01.Washington, DC, USA: IEEE Computer Society, 2001.

ROBERTSON, L. Computing Services for LHC: From Clusters to Grids. In: BRUN, R.; CARMINATI, F.; GALLI CARMINATI, G. (Eds.). **From the Web to the Grid and Beyond**. The Frontiers Collection. [s.l.] Springer Berlin Heidelberg, 2012. p. 69–89.

RUDOLPH, L.; SLIVKIN-ALLALOUF, M.; UPFAL, E. **A simple load balancing scheme for task allocation in parallel machines**Proceedings of the third annual ACM symposium on Parallel algorithms and architectures. **Anais**...: SPAA '91.New York, NY, USA: ACM, 1991.

SCHOPF, J. M. et al. Monitoring the grid with the Globus Toolkit MDS4. **Journal of Physics: Conference Series**, v. 46, n. 1, p. 521, 2006.

SINGH, M. P.; VOUK, M. A. **Scientific Workflows: Scientific Computing Meets Transactional Workflows**. [s.l: s.n.].

SNELL, Q. et al. **The Performance Impact of Advance Reservation Meta-scheduling**Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing. **Anais**...: IPDPS '00/JSSPP '00.London, UK, UK: Springer-Verlag, 2000.

SQUYRES, J. M.; LUMSDAINE, A. **A Component Architecture for LAM/MPI**Proceedings, 10th European PVM/MPI Users' Group Meeting. **Anais**...: Lecture Notes in Computer Science.Venice, Italy: Springer-Verlag, out. 2003

SRINIVASAN, S. et al. **Characterization of backfilling strategies for parallel job scheduling**International Conference on Parallel Processing Workshops, 2002. Proceedings. **Anais**.2002

SRINIVASAN, S.; KRISHNAMOORTHY, S.; SADAYAPPAN, P. **A robust scheduling technology for moldable scheduling of parallel jobs**2003 IEEE International Conference on Cluster Computing, 2003. **Anais**...2003

STANZANI, S. L. **ESTRATÉGIAS DE ESCALONAMENTO DE WORKFLOWS COM TAREFAS PARALELAS E SEQUENCIAIS EM GRADES COMPUTACIONAIS**. Tese de Doutorado—São Paulo: Universidade de São Paulo, 2013.

STANZANI, S. L.; SATO, L. M. Scheduling DAGs with Parallel Tasks in Multi-Clusters Based on Parallel Efficiency. In: J. (JONG HYUK) PARK, J. et al. (Eds.). **Computer Science and Convergence**. Lecture Notes in Electrical Engineering. [s.l.] Springer Netherlands, 2012. v. 114p. 753–760.

STANZANI, S. L.; SATO, L. M.; NETTO, M. A. S. **Scheduling Workflows in Multi-cluster Environments**2013 27th International Conference on Advanced Information Networking and Applications Workshops (WAINA). **Anais**... In: 2013 27TH INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS (WAINA). 2013

SUTER, F.; DESPREZ, F.; CASANOVA, H. From Heterogeneous Task Scheduling to Heterogeneous Mixed Parallel Scheduling. In: DANELUTTO, M.; VANNESCHI, M.; LAFORENZA, D. (Eds.). **Euro-Par 2004 Parallel Processing**. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004. v. 3149p. 230–237.

TANG, X. et al. A stochastic scheduling algorithm for precedence constrained tasks on Grid. **Future Generation Computer Systems**, v. 27, n. 8, p. 1083 – 1091, 2011.

TOPCUOGLU, H.; HARIRI, S.; WU, M.-Y. **Task scheduling algorithms for heterogeneous processors**1999

TOPCUOUGLU, H.; HARIRI, S.; WU, M. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. **IEEE Trans. Parallel Distrib. Syst.**, v. 13, n. 3, p. 260–274, mar. 2002.

VYDYANATHAN, N. et al. **An Integrated Approach for Processor Allocation and Scheduling of Mixed-Parallel Applications**International Conference on Parallel Processing, 2006. ICPP 2006. **Anais**...IEEE, ago. 2006

WENTZEL, H. Simulation Speeds Modular Truck Development Process. 2009.

WIECZOREK, M.; HOHEISEL, A.; PRODAN, R. Taxonomies of the Multi-Criteria Grid Workflow Scheduling Problem. In: **Grid Middleware and Services**. [s.l.] Springer US, 2008. p. 237–264.

WORKFLOW MANAGEMENT COALITION. **Terminology and Glossary**, 1999. Disponível em: <http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html>

YIGITBASI, N. et al. **Performance Evaluation of Overload Control in Multi-cluster Grids**: GRID '11.Washington, DC, USA: IEEE Computer Society, 2011.

YU, J.; BUYYA, R. A taxonomy of scientific workflow systems for grid computing. **SIGMOD Rec.**, v. 34, n. 3, p. 44–49, 2005.

YU, J.; BUYYA, R.; RAMAMOHANARAO, K. Workflow Scheduling Algorithms for Grid Computing. In: **Metaheuristics for Scheduling in Distributed Computing Environments**. 2005.

YU, Z.; SHI, W. **A Planner-Guided Scheduling Strategy for Multiple Workflow Applications**Parallel Processing - Workshops, 2008. ICPP-W ’08. International Conference on. **Anais**...2008

ZHANG, J.; LUO, J.; DONG, F. Scheduling of scientific workflow in non-dedicated heterogeneous multicluster platform. **Journal of Systems and Software**, v. 86, n. 7, p. 1806 – 1818, 2013.

ZHANG, Y. et al. **Improving parallel job scheduling by combining gang scheduling and backfilling techniques**Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International. **Anais**...2000

ZHANG, Y.; KOELBEL, C.; COOPER, K. **Hybrid Re-scheduling Mechanisms for Workflow Applications on Multi-cluster Grid**Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. **Anais**...: CCGRID ’09.Washington, DC, USA: IEEE Computer Society, 2009.

Apêndice A – Descrição do Ambiente de Testes

Neste apêndice é mostrada a descrição do ambiente utilizado.

```
<?xml version='1.0'?>
<!DOCTYPE platform SYSTEM
"http://simgrid.gforge.inria.fr/simgrid.dtd">

<platform version="3">
<AS id="AS_das3" routing="Dijkstra">

   <cluster id="1" prefix="cluster1-" suffix=".das3.nl" radical="0-
30" power="4.40E9" bw="125000000" lat="1.0E-5" sharing_policy="SHARED"
bb_bw="125000000" bb_lat="1.0E-4"
bb_sharing_policy="SHARED"></cluster>

   <cluster id="2" prefix="cluster2-" suffix=".das3.nl" radical="0-
30" power="4.60E9" bw="125000000" lat="1.0E-5" sharing_policy="SHARED"
bb_bw="125000000" bb_lat="1.0E-4"
bb_sharing_policy="SHARED"></cluster>

   <cluster id="3" prefix="cluster3-" suffix=".das3.nl" radical="0-
30" power="4.60E9" bw="125000000" lat="1.0E-5" sharing_policy="SHARED"
bb_bw="125000000" bb_lat="1.0E-4"
bb_sharing_policy="SHARED"></cluster>

   <cluster id="4" prefix="cluster4-" suffix=".das3.nl" radical="0-
30" power="4.60E9" bw="125000000" lat="1.0E-5" sharing_policy="SHARED"
bb_bw="125000000" bb_lat="1.0E-4"
bb_sharing_policy="SHARED"></cluster>

     <link id="backbone" bandwidth="12500000" latency="5E-5"
bandwidth_file="link.bw" latency_file="link.lat"
sharing_policy="SHARED"/>

   <ASroute src="1" dst="2" gw_src="cluster1-1_router.das3.nl"
            gw_dst="cluster2-2_router.das3.nl">
            <link_ctn id="backbone" />
      </ASroute>
```

```
    <ASroute src="1" dst="3" gw_src="cluster1-1_router.das3.nl"
                gw_dst="cluster3-3_router.das3.nl">
                <link_ctn id="backbone" />
        </ASroute>

    <ASroute src="1" dst="4" gw_src="cluster1-1_router.das3.nl"
                gw_dst="cluster4-4_router.das3.nl">
                <link_ctn id="backbone" />
        </ASroute>

    <ASroute src="2" dst="3" gw_src="cluster2-2_router.das3.nl"
                gw_dst="cluster3-3_router.das3.nl">
                <link_ctn id="backbone" />
        </ASroute>

    <ASroute src="2" dst="4" gw_src="cluster2-2_router.das3.nl"
                gw_dst="cluster4-4_router.das3.nl">
                <link_ctn id="backbone" />
        </ASroute>

    <ASroute src="3" dst="4" gw_src="cluster3-3_router.das3.nl"
                gw_dst="cluster4-4_router.das3.nl">
                <link_ctn id="backbone" />
        </ASroute>

</AS>

</platform>
```

Apêndice B – Descrição da Carga de Entrada

Neste apêndice é mostrada a descrição da carga de entrada utilizada.

A seguir são descritos os workflows do experimento de avaliação de desempenho da estratégia MCA-HEFT.

Workflow 1

0,1	0,32000000000000,340000000000,0,0.9,1
0,2	1,3400000000000,1000000,0,0.6,1
0,3	2,3000000000000,100000000000,0,0.6,1
0,4	3,2900000000000,800000000000,0,0.7,1
0,5	4,2900000000000,800000000000,0,0.7,1
0,6	5,1400000000000,32000000000,0,0.1,1
0,7	6,31000000000000,340000000000,0,0.8,1
0,8	7,3400000000000,1000000,0,0.6,1
0,9	8,3000000000000,100000000000,0,0.6,1
0,10	9,1400000000000,32000000000,0,0.1,1
1,11	10,2900000000000,800000000000,0,0.7,1
2,11	11,32000000000000,340000000000,0,0.9,1
3,11	
4,11	
5,11	
6,11	
7,11	
8,11	
9,11	
10,11	

Workflow 2

0,1	0,1400000000000,32000000000,0,0.1,1
0,2	1,1400000000000,32000000000,0,0.1,1
0,3	2,31000000000000,340000000000,0,0.8,1
0,4	3,31000000000000,340000000000,0,0.8,1
0,5	4,31000000000000,340000000000,0,0.8,1
0,6	5,3000000000000,100000000000,0,0.6,1

0,7	6,1400000000000,32000000000,0,0.1,1
0,8	7,2900000000000,800000000000,0,0.7,1
0,9	8,1400000000000,32000000000,0,0.1,1
0,10	9,1400000000000,32000000000,0,0.1,1
0,11	10,31000000000000,340000000000,0,0.8,1
0,12	11,2900000000000,800000000000,0,0.7,1
1,19	12,32000000000000,340000000000,0,0.9,1
2,13	13,31000000000000,340000000000,0,0.8,1
2,14	14,31000000000000,340000000000,0,0.8,1
13,16	15,32000000000000,340000000000,0,0.9,1
14,18	16,32000000000000,340000000000,0,0.9,1
3,19	17,3000000000000,100000000000,0,0.6,1
4,19	18,32000000000000,340000000000,0,0.9,1
5,19	19,31000000000000,340000000000,0,0.8,1
6,19	
7,19	
8,19	
9,19	
10,19	
11,19	
12,15	
15,17	
16,19	
17,19	
18,19	

Workflow 3

0,1	0,3000000000000,100000000000,0,0.6,1
0,2	1,3000000000000,100000000000,0,0.6,1
0,3	2,3000000000000,100000000000,0,0.6,1
0,4	3,32000000000000,340000000000,0,0.9,1
0,5	4,32000000000000,340000000000,0,0.9,1
0,6	5,3400000000000,1000000,0,0.6,1
0,7	6,1400000000000,32000000000,0,0.1,1
0,8	7,3400000000000,1000000,0,0.6,1
0,9	8,32000000000000,340000000000,0,0.9,1
0,10	9,1400000000000,32000000000,0,0.1,1

0,11	10,32000000000000,340000000000,0,0.9,1
0,12	11,3400000000000,1000000,0,0.6,1
0,13	12,1400000000000,32000000000,0,0.1,1
0,14	13,31000000000000,340000000000,0,0.8,1
0,15	14,3400000000000,1000000,0,0.6,1
0,16	15,3000000000000,100000000000,0,0.6,1
0,17	16,2900000000000,800000000000,0,0.7,1
0,18	17,1400000000000,32000000000,0,0.1,1
0,19	18,32000000000000,340000000000,0,0.9,1
0,20	19,3000000000000,100000000000,0,0.6,1
1,21	20,2900000000000,800000000000,0,0.7,1
2,21	21,31000000000000,340000000000,0,0.8,1
3,21	
4,21	
5,21	
6,21	
7,21	
8,21	
9,21	
10,21	
11,21	
12,21	
13,21	
14,21	
15,21	
16,21	
17,21	
18,21	
19,21	
20,21	

Workflow 4

0,1	0,32000000000000,340000000000,0,0.9,1
0,2	1,3400000000000,1000000,0,0.6,1
0,3	2,3000000000000,100000000000,0,0.6,1
0,4	3,2900000000000,800000000000,0,0.7,1
0,5	4,2900000000000,800000000000,0,0.7,1

0,6	5,1400000000000,32000000000,0,0.1,1
0,7	6,31000000000000,340000000000,0,0.8,1
0,8	7,3400000000000,1000000,0,0.6,1
0,9	8,3000000000000,100000000000,0,0.6,1
0,10	9,1400000000000,32000000000,0,0.1,1
1,11	10,2900000000000,800000000000,0,0.7,1
2,11	11,32000000000000,340000000000,0,0.9,1
3,11	12,3400000000000,1000000,0,0.6,1
4,11	13,3000000000000,100000000000,0,0.6,1
5,11	14,2900000000000,800000000000,0,0.7,1
6,11	15,2900000000000,800000000000,0,0.7,1
7,11	16,1400000000000,32000000000,0,0.1,1
8,11	17,31000000000000,340000000000,0,0.8,1
9,11	18,3400000000000,1000000,0,0.6,1
10,11	19,3000000000000,100000000000,0,0.6,1
11,12	20,1400000000000,32000000000,0,0.1,1
11,13	21,2900000000000,800000000000,0,0.7,1
11,14	22,32000000000000,340000000000,0,0.9,1
11,15	23,3200000000,340000,0,0.9,1
11,16	
11,17	
11,18	
11,19	
11,20	
11,21	
12,22	
13,22	
14,22	
15,22	
16,22	
17,22	
18,22	
19,22	
20,22	
21,22	
22,23	

Workflow 5

0,1	0,3000000000000,100000000000,0,0.6,1
0,2	1,32000000000000,340000000000,0,0.9,1
0,3	2,31000000000000,340000000000,0,0.8,1
0,4	3,1400000000000,32000000000,0,0.1,1
0,5	4,3400000000000,1000000,0,0.6,1
0,6	5,3400000000000,1000000,0,0.6,1
0,7	6,1400000000000,32000000000,0,0.1,1
0,8	7,1400000000000,32000000000,0,0.1,1
0,9	8,32000000000000,340000000000,0,0.9,1
0,10	9,1400000000000,32000000000,0,0.1,1
0,11	10,3000000000000,100000000000,0,0.6,1
0,12	11,31000000000000,340000000000,0,0.8,1
0,13	12,1400000000000,32000000000,0,0.1,1
0,14	13,2900000000000,800000000000,0,0.7,1
0,15	14,3400000000000,1000000,0,0.6,1
0,16	15,1400000000000,32000000000,0,0.1,1
0,17	16,3400000000000,1000000,0,0.6,1
0,18	17,3400000000000,1000000,0,0.6,1
0,19	18,2900000000000,800000000000,0,0.7,1
0,20	19,31000000000000,340000000000,0,0.8,1
1,21	20,32000000000000,340000000000,0,0.9,1
2,21	21,3000000000000,100000000000,0,0.6,1
3,21	
4,21	
5,21	
6,21	
7,21	
8,21	
9,21	
10,21	
11,21	
12,21	
13,21	
14,21	
15,21	
16,21	

17,21	
18,21	
19,21	
20,21	

Workflow 6

0,1	0,1400000000000,32000000000,0,0.1,1
0,2	1,1400000000000,32000000000,0,0.1,1
0,3	2,3000000000000,100000000000,0,0.6,1
1,4	3,31000000000000,340000000000,0,0.8,1
1,5	4,1400000000000,32000000000,0,0.1,1
1,6	5,1400000000000,32000000000,0,0.1,1
1,7	6,1400000000000,32000000000,0,0.1,1
1,8	7,2900000000000,800000000000,0,0.7,1
4,14	8,32000000000000,340000000000,0,0.9,1
5,15	9,3000000000000,100000000000,0,0.6,1
6,16	10,3000000000000,100000000000,0,0.6,1
7,18	11,1400000000000,32000000000,0,0.1,1
8,19	12,1400000000000,32000000000,0,0.1,1
2,9	13,1400000000000,32000000000,0,0.1,1
9,17	14,32000000000000,340000000000,0,0.9,1
3,10	15,32000000000000,340000000000,0,0.9,1
3,11	16,32000000000000,340000000000,0,0.9,1
10,12	17,1400000000000,32000000000,0,0.1,1
11,13	18,1400000000000,32000000000,0,0.1,1
12,20	19,3400000000000,1000000,0,0.6,1
13,20	20,3000000000000,100000000000,0,0.6,1
14,20	
15,20	
16,20	
17,20	
18,20	
19,20	

Workflow 7

0,1	0,2900000000000,800000000000,0,0.7,1
0,2	1,1400000000000,32000000000,0,0.1,1
1,3	2,3000000000000,100000000000,0,0.6,1

1,4	3,3000000000000,100000000000,0,0.6,1
1,5	4,31000000000000,340000000000,0,0.8,1
1,6	5,3000000000000,100000000000,0,0.6,1
1,7	6,31000000000000,340000000000,0,0.8,1
3,13	7,32000000000000,340000000000,0,0.9,1
4,15	8,2900000000000,800000000000,0,0.7,1
5,16	9,1400000000000,32000000000,0,0.1,1
6,17	10,31000000000000,340000000000,0,0.8,1
7,22	11,32000000000000,340000000000,0,0.9,1
2,8	12,31000000000000,340000000000,0,0.8,1
2,9	13,3400000000000,1000000,0,0.6,1
2,10	14,2900000000000,800000000000,0,0.7,1
2,11	15,31000000000000,340000000000,0,0.8,1
2,12	16,2900000000000,800000000000,0,0.7,1
8,14	17,31000000000000,340000000000,0,0.8,1
9,18	18,1400000000000,32000000000,0,0.1,1
10,19	19,32000000000000,340000000000,0,0.9,1
11,20	20,3400000000000,1000000,0,0.6,1
12,21	21,1400000000000,32000000000,0,0.1,1
13,23	22,2900000000000,800000000000,0,0.7,1
14,23	23,2900000000000,800000000000,0,0.7,1
15,23	
16,23	
17,23	
18,23	
19,23	
20,23	
21,23	
22,23	

Workflow 8

0,1	0,1400000000000,32000000000,0,0.1,1
1,2	1,31000000000000,340000000000,0,0.8,1
1,3	2,31000000000000,340000000000,0,0.8,1
2,4	3,1400000000000,32000000000,0,0.1,1
3,5	4,32000000000000,340000000000,0,0.9,1
4,26	5,3000000000000,100000000000,0,0.6,1

5,6	6,3000000000000,100000000000,0,0.6,1
6,7	7,32000000000000,340000000000,0,0.9,1
7,8	8,3000000000000,100000000000,0,0.6,1
7,9	9,1400000000000,32000000000,0,0.1,1
7,10	10,3000000000000,100000000000,0,0.6,1
7,11	11,32000000000000,340000000000,0,0.9,1
7,12	12,31000000000000,340000000000,0,0.8,1
7,13	13,2900000000000,800000000000,0,0.7,1
7,14	14,2900000000000,800000000000,0,0.7,1
8,15	15,1400000000000,32000000000,0,0.1,1
9,16	16,2900000000000,800000000000,0,0.7,1
10,17	17,1400000000000,32000000000,0,0.1,1
11,18	18,31000000000000,340000000000,0,0.8,1
12,19	19,1400000000000,32000000000,0,0.1,1
13,20	20,31000000000000,340000000000,0,0.8,1
14,21	21,3400000000000,1000000,0,0.6,1
15,22	22,1400000000000,32000000000,0,0.1,1
22,25	23,32000000000000,340000000000,0,0.9,1
16,26	24,3000000000000,100000000000,0,0.6,1
17,26	25,2900000000000,800000000000,0,0.7,1
18,26	26,1400000000000,32000000000,0,0.1,1
19,23	
23,24	
20,26	
21,26	
24,26	
25,26	

Workflow 9

0,1	0,2900000000000,800000000000,0,0.7,1
1,2	1,1400000000000,32000000000,0,0.1,1
2,3	2,2900000000000,800000000000,0,0.7,1
3,4	3,1400000000000,32000000000,0,0.1,1
3,5	4,31000000000000,340000000000,0,0.8,1
3,6	5,3000000000000,100000000000,0,0.6,1
3,7	6,3000000000000,100000000000,0,0.6,1
4,8	7,1400000000000,32000000000,0,0.1,1

5,9	8,31000000000000,340000000000,0,0.8,1
6,10	9,3000000000000,100000000000,0,0.6,1
7,11	10,31000000000000,340000000000,0,0.8,1
8,40	11,31000000000000,340000000000,0,0.8,1
9,40	12,3400000000000,1000000,0,0.6,1
10,12	13,31000000000000,340000000000,0,0.8,1
10,13	14,1400000000000,32000000000,0,0.1,1
12,18	15,32000000000000,340000000000,0,0.9,1
13,23	16,1400000000000,32000000000,0,0.1,1
11,14	17,32000000000000,340000000000,0,0.9,1
11,15	18,1400000000000,32000000000,0,0.1,1
11,16	19,32000000000000,340000000000,0,0.9,1
11,17	20,2900000000000,800000000000,0,0.7,1
14,19	21,1400000000000,32000000000,0,0.1,1
15,20	22,32000000000000,340000000000,0,0.9,1
16,21	23,31000000000000,340000000000,0,0.8,1
17,22	24,1400000000000,32000000000,0,0.1,1
18,40	25,3400000000000,1000000,0,0.6,1
19,24	26,3000000000000,100000000000,0,0.6,1
19,25	27,1400000000000,32000000000,0,0.1,1
24,28	28,31000000000000,340000000000,0,0.8,1
25,31	29,31000000000000,340000000000,0,0.8,1
20,40	30,31000000000000,340000000000,0,0.8,1
21,26	31,32000000000000,340000000000,0,0.9,1
26,29	32,1400000000000,32000000000,0,0.1,1
22,27	33,3400000000000,1000000,0,0.6,1
27,30	34,2900000000000,800000000000,0,0.7,1
23,40	35,1400000000000,32000000000,0,0.1,1
28,40	36,3400000000000,1000000,0,0.6,1
29,32	37,1400000000000,32000000000,0,0.1,1
29,33	38,2900000000000,800000000000,0,0.7,1
32,36	39,1400000000000,32000000000,0,0.1,1
33,37	40,31000000000000,340000000000,0,0.8,1
30,34	
34,38	
31,35	

35,39	
36,40	
37,40	
38,40	
39,40	

Workflow 10

0,1	0,1400000000000,32000000000,0,0.1,1
0,2	1,1400000000000,32000000000,0,0.1,1
0,3	2,3000000000000,100000000000,0,0.6,1
1,4	3,31000000000000,340000000000,0,0.8,1
1,5	4,1400000000000,32000000000,0,0.1,1
1,6	5,1400000000000,32000000000,0,0.1,1
1,7	6,1400000000000,32000000000,0,0.1,1
1,8	7,2900000000000,800000000000,0,0.7,1
4,14	8,32000000000000,340000000000,0,0.9,1
5,15	9,3000000000000,100000000000,0,0.6,1
6,16	10,3000000000000,100000000000,0,0.6,1
7,18	11,1400000000000,32000000000,0,0.1,1
8,19	12,1400000000000,32000000000,0,0.1,1
2,9	13,1400000000000,32000000000,0,0.1,1
9,17	14,32000000000000,340000000000,0,0.9,1
3,10	15,32000000000000,340000000000,0,0.9,1
3,11	16,32000000000000,340000000000,0,0.9,1
10,12	17,1400000000000,32000000000,0,0.1,1
11,13	18,1400000000000,32000000000,0,0.1,1
12,20	19,3400000000000,1000000,0,0.6,1
13,20	20,3000000000000,100000000000,0,0.6,1
14,20	21,3000000000000,100000000000,0,0.6,1
15,20	22,1400000000000,32000000000,0,0.1,1
16,20	23,1400000000000,32000000000,0,0.1,1
17,20	24,1400000000000,32000000000,0,0.1,1
18,20	25,32000000000000,340000000000,0,0.9,1
19,20	26,32000000000000,340000000000,0,0.9,1
20,21	27,32000000000000,340000000000,0,0.9,1
21,22	28,1400000000000,32000000000,0,0.1,1
21,23	29,1400000000000,32000000000,0,0.1,1

21,24	30,3400000000000,1000000,0,0.6,1
21,25	31,3000000000000,100000000000,0,0.6,1
21,26	
21,27	
21,28	
21,29	
22,30	
23,30	
24,30	
25,30	
26,30	
27,30	
28,30	
29,30	
30,31	

A seguir são descritos os conjuntos de workflows do experimento de avaliação de desempenho da estratégia DMCA-HEFT.

Conjunto 1:

Workflow 1

0,1	0,3400000000000,1000000,0,15,0.6,1
0,2	1,3000000000000,100000000000,0,15,0.6,1
0,3	2,32000000000000,340000000000,0,15,0.9,1
0,4	3,3400000000000,1000000,0,15,0.6,1
1,5	4,31000000000000,340000000000,0,15,0.8,1
2,5	5,1400000000000,32000000000,0,15,0.1,1
3,5	
4,5	

Workflow 2

6,7	6,32000000000000,340000000000,0,15,0.9,1
6,8	7,2900000000000,800000000000,0,15,0.7,1
6,9	8,3000000000000,100000000000,0,15,0.6,1
6,10	9,1400000000000,32000000000,0,15,0.1,1
7,11	10,1400000000000,32000000000,0,15,0.1,1

8,11	11,2900000000000,800000000000,0,15,0.7,1
9,11	
10,11	

Workflow 3

12,13	12,3400000000000,1000000,0,15,0.6,1
12,14	13,1400000000000,32000000000,0,15,0.1,1
13,19	14,32000000000000,340000000000,0,15,0.9,1
14,15	15,1400000000000,32000000000,0,15,0.1,1
15,16	16,1400000000000,32000000000,0,15,0.1,1
16,17	17,2900000000000,800000000000,0,15,0.7,1
17,18	18,3000000000000,100000000000,0,15,0.6,1
18,19	19,3000000000000,100000000000,0,15,0.6,1

Conjunto 2:

Workflow 1

0,1	0,3000000000000,100000000000,0,15,0.6,1
0,2	1,3400000000000,1000000,0,15,0.6,1
0,3	2,32000000000000,340000000000,0,15,0.9,1
1,6	3,31000000000000,340000000000,0,15,0.8,1
2,4	4,31000000000000,340000000000,0,15,0.8,1
4,5	5,32000000000000,340000000000,0,15,0.9,1
3,6	6,1400000000000,32000000000,0,15,0.1,1
5,6	

Workflow 2

7,8	7,1400000000000,32000000000,0,15,0.1,1
8,9	8,1400000000000,32000000000,0,15,0.1,1
8,10	9,31000000000000,340000000000,0,15,0.8,1
8,11	10,3000000000000,100000000000,0,15,0.6,1
9,12	11,3000000000000,100000000000,0,15,0.6,1
10,13	12,1400000000000,32000000000,0,15,0.1,1
11,14	13,2900000000000,800000000000,0,15,0.7,1
12,15	14,3400000000000,1000000,0,15,0.6,1
13,15	15,1400000000000,32000000000,0,15,0.1,1
14,15	

Workflow 3

16,17	16,3400000000000,1000000,0,15,0.6,1
17,18	17,3000000000000,100000000000,0,15,0.6,1
18,19	18,31000000000000,340000000000,0,15,0.8,1
19,20	19,1400000000000,32000000000,0,15,0.1,1
20,21	20,1400000000000,32000000000,0,15,0.1,1
21,22	21,3400000000000,1000000,0,15,0.6,1
22,23	22,32000000000000,340000000000,0,15,0.9,1
23,24	23,2900000000000,800000000000,0,15,0.7,1
	24,3000000000000,100000000000,0,15,0.6,1

Workflow 4

25,26	25,2900000000000,800000000000,0,15,0.7,1
25,27	26,1400000000000,32000000000,0,15,0.1,1
25,28	27,31000000000000,340000000000,0,15,0.8,1
26,31	28,32000000000000,340000000000,0,15,0.9,1
27,29	29,1400000000000,32000000000,0,15,0.1,1
29,30	30,31000000000000,340000000000,0,15,0.8,1
28,31	31,1400000000000,32000000000,0,15,0.1,1
30,31	

Conjunto 3:

Workflow 1

0,1	0,31000000000000,340000000000,0,15,0.8,1
0,2	1,32000000000000,340000000000,0,15,0.9,1
0,3	2,2900000000000,800000000000,0,15,0.7,1
0,4	3,31000000000000,340000000000,0,15,0.8,1
1,5	4,32000000000000,340000000000,0,15,0.9,1
2,5	5,32000000000000,340000000000,0,15,0.9,1
3,5	
4,5	

Workflow 2

6,7	6,32000000000000,340000000000,0,15,0.9,1
6,8	7,3400000000000,1000000,0,15,0.6,1
6,9	8,2900000000000,800000000000,0,15,0.7,1
6,10	9,1400000000000,32000000000,0,15,0.1,1

7,11	10,32000000000000,340000000000,0,15,0.9,1
8,11	11,1400000000000,32000000000,0,15,0.1,1
9,11	
10,11	

Workflow 3

12,13	12,2900000000000,800000000000,0,15,0.7,1
13,14	13,32000000000000,340000000000,0,15,0.9,1
13,15	14,31000000000000,340000000000,0,15,0.8,1
13,16	15,1400000000000,32000000000,0,15,0.1,1
14,17	16,31000000000000,340000000000,0,15,0.8,1
15,18	17,1400000000000,32000000000,0,15,0.1,1
16,19	18,1400000000000,32000000000,0,15,0.1,1
17,20	19,3400000000000,1000000,0,15,0.6,1
18,20	20,1400000000000,32000000000,0,15,0.1,1
19,20	

Workflow 4

21,22	21,3000000000000,100000000000,0,15,0.6,1
21,23	22,1400000000000,32000000000,0,15,0.1,1
21,24	23,3000000000000,100000000000,0,15,0.6,1
21,25	24,31000000000000,340000000000,0,15,0.8,1
22,26	25,2900000000000,800000000000,0,15,0.7,1
23,26	26,32000000000000,340000000000,0,15,0.9,1
24,26	
25,26	

Workflow 5

27,28	27,2900000000000,800000000000,0,15,0.7,1
27,29	28,1400000000000,32000000000,0,15,0.1,1
27,30	29,1400000000000,32000000000,0,15,0.1,1
27,31	30,32000000000000,340000000000,0,15,0.9,1
28,32	31,1400000000000,32000000000,0,15,0.1,1
29,32	32,1400000000000,32000000000,0,15,0.1,1
30,32	
31,32	

Conjunto 4:

Workflow 1

0,1	0,1400000000000,32000000000,0,15,0.1,1
1,2	1,3400000000000,1000000,0,15,0.6,1
2,3	2,3400000000000,1000000,0,15,0.6,1
3,4	3,3000000000000,100000000000,0,15,0.6,1
4,5	4,2900000000000,800000000000,0,15,0.7,1
5,6	5,2900000000000,800000000000,0,15,0.7,1
6,7	6,3000000000000,100000000000,0,15,0.6,1
7,8	7,3000000000000,100000000000,0,15,0.6,1
	8,31000000000000,340000000000,0,15,0.8,1

Workflow 2

9,10	9,32000000000000,340000000000,0,15,0.9,1
9,11	10,1400000000000,32000000000,0,15,0.1,1
10,12	11,1400000000000,32000000000,0,15,0.1,1
12,13	12,3000000000000,100000000000,0,15,0.6,1
11,16	13,1400000000000,32000000000,0,15,0.1,1
13,14	14,1400000000000,32000000000,0,15,0.1,1
14,15	15,1400000000000,32000000000,0,15,0.1,1
15,16	16,3400000000000,1000000,0,15,0.6,1

Workflow 3

17,18	17,2900000000000,800000000000,0,15,0.7,1
17,19	18,32000000000000,340000000000,0,15,0.9,1
17,20	19,3400000000000,1000000,0,15,0.6,1
18,23	20,32000000000000,340000000000,0,15,0.9,1
19,23	21,2900000000000,800000000000,0,15,0.7,1
20,21	22,2900000000000,800000000000,0,15,0.7,1
21,22	23,3400000000000,1000000,0,15,0.6,1
22,23	

Workflow 4

24,25	24,25
25,26	25,26
25,27	25,27
25,28	25,28
26,29	26,29
27,30	27,30
28,31	28,31
29,32	29,32

30,32	30,32
31,32	31,32

Conjunto 5:

Workflow 1

0,1	0,1400000000000,32000000000,0,15,0.1,1
0,2	1,3400000000000,1000000,0,15,0.6,1
0,3	2,1400000000000,32000000000,0,15,0.1,1
1,4	3,31000000000000,340000000000,0,15,0.8,1
4,5	4,3400000000000,1000000,0,15,0.6,1
2,6	5,3000000000000,100000000000,0,15,0.6,1
3,6	6,31000000000000,340000000000,0,15,0.8,1
5,6	

Workflow 2

7,8	7,1400000000000,32000000000,0,15,0.1,1
8,9	8,3000000000000,100000000000,0,15,0.6,1
9,10	9,3400000000000,1000000,0,15,0.6,1
10,11	10,3400000000000,1000000,0,15,0.6,1
10,12	11,2900000000000,800000000000,0,15,0.7,1
11,13	12,3400000000000,1000000,0,15,0.6,1
12,14	13,31000000000000,340000000000,0,15,0.8,1
13,15	14,2900000000000,800000000000,0,15,0.7,1
14,15	15,2900000000000,800000000000,0,15,0.7,1

Workflow 3

16,17	16,32000000000000,340000000000,0,15,0.9,1
16,18	17,1400000000000,32000000000,0,15,0.1,1
17,19	18,3000000000000,100000000000,0,15,0.6,1
19,20	19,1400000000000,32000000000,0,15,0.1,1
18,23	20,32000000000000,340000000000,0,15,0.9,1
20,21	21,3400000000000,1000000,0,15,0.6,1
21,22	22,32000000000000,340000000000,0,15,0.9,1
22,23	23,1400000000000,32000000000,0,15,0.1,1

Workflow 4

24,25	24,3400000000000,1000000,0,15,0.6,1
25,26	25,32000000000000,340000000000,0,15,0.9,1

26,27	26,2900000000000,800000000000,0,15,0.7,1
27,28	27,2900000000000,800000000000,0,15,0.7,1
27,29	28,3400000000000,1000000,0,15,0.6,1
28,30	29,3400000000000,1000000,0,15,0.6,1
29,31	30,32000000000000,340000000000,0,15,0.9,1
30,32	31,32000000000000,340000000000,0,15,0.9,1
31,32	32,1400000000000,32000000000,0,15,0.1,1

Conjunto 6:

Workflow 1

0,1	0,3000000000000,100000000000,0,15,0.6,1
0,2	1,2900000000000,800000000000,0,15,0.7,1
1,3	2,32000000000000,340000000000,0,15,0.9,1
1,4	3,1400000000000,32000000000,0,15,0.1,1
3,5	4,31000000000000,340000000000,0,15,0.8,1
4,6	5,1400000000000,32000000000,0,15,0.1,1
2,7	6,2900000000000,800000000000,0,15,0.7,1
5,7	7,31000000000000,340000000000,0,15,0.8,1
6,7	

Workflow 2

8,9	8,32000000000000,340000000000,0,15,0.9,1
8,10	9,32000000000000,340000000000,0,15,0.9,1
8,11	10,2900000000000,800000000000,0,15,0.7,1
8,12	11,1400000000000,32000000000,0,15,0.1,1
9,13	12,2900000000000,800000000000,0,15,0.7,1
10,13	13,1400000000000,32000000000,0,15,0.1,1
11,13	
12,13	

Workflow 3

14,15	14,2900000000000,800000000000,0,15,0.7,1
14,16	15,1400000000000,32000000000,0,15,0.1,1
14,17	16,3400000000000,1000000,0,15,0.6,1
15,20	17,31000000000000,340000000000,0,15,0.8,1
16,18	18,31000000000000,340000000000,0,15,0.8,1
18,19	19,31000000000000,340000000000,0,15,0.8,1

17,20	20,3000000000000,100000000000,0,15,0.6,1
19,20	

Workflow 4

21,22	21,3000000000000,100000000000,0,15,0.6,1
21,23	22,1400000000000,32000000000,0,15,0.1,1
22,28	23,31000000000000,340000000000,0,15,0.8,1
23,24	24,2900000000000,800000000000,0,15,0.7,1
23,25	25,2900000000000,800000000000,0,15,0.7,1
24,26	26,1400000000000,32000000000,0,15,0.1,1
25,27	27,3400000000000,1000000,0,15,0.6,1
26,28	28,31000000000000,340000000000,0,15,0.8,1
27,28	

Workflow 5

29,30	29,32000000000000,340000000000,0,15,0.9,1
30,31	30,1400000000000,32000000000,0,15,0.1,1
30,32	31,2900000000000,800000000000,0,15,0.7,1
30,33	32,1400000000000,32000000000,0,15,0.1,1
31,34	33,2900000000000,800000000000,0,15,0.7,1
32,35	34,32000000000000,340000000000,0,15,0.9,1
33,36	35,32000000000000,340000000000,0,15,0.9,1
34,37	36,32000000000000,340000000000,0,15,0.9,1
35,37	37,1400000000000,32000000000,0,15,0.1,1
36,37	

Conjunto 7:

Workflow 1

0,1	0,1400000000000,32000000000,0,15,0.1,1
0,2	1,2900000000000,800000000000,0,15,0.7,1
1,3	2,1400000000000,32000000000,0,15,0.1,1
1,4	3,3000000000000,100000000000,0,15,0.6,1
3,5	4,1400000000000,32000000000,0,15,0.1,1
4,6	5,1400000000000,32000000000,0,15,0.1,1
2,7	6,3000000000000,100000000000,0,15,0.6,1
5,7	7,1400000000000,32000000000,0,15,0.1,1
6,7	

Workflow 2

8,9	8,2900000000000,800000000000,0,15,0.7,1
8,10	9,1400000000000,32000000000,0,15,0.1,1
8,11	10,1400000000000,32000000000,0,15,0.1,1
8,12	11,1400000000000,32000000000,0,15,0.1,1
9,13	12,3400000000000,1000000,0,15,0.6,1
10,13	13,1400000000000,32000000000,0,15,0.1,1
11,13	
12,13	

Workflow 3

14,15	14,2900000000000,800000000000,0,15,0.7,1
15,16	15,3000000000000,100000000000,0,15,0.6,1
15,17	16,31000000000000,340000000000,0,15,0.8,1
16,18	17,31000000000000,340000000000,0,15,0.8,1
17,19	18,3400000000000,1000000,0,15,0.6,1
18,20	19,3000000000000,100000000000,0,15,0.6,1
20,21	20,32000000000000,340000000000,0,15,0.9,1
19,22	21,1400000000000,32000000000,0,15,0.1,1
21,22	22,1400000000000,32000000000,0,15,0.1,1

Workflow 4

23,24	23,1400000000000,32000000000,0,15,0.1,1
23,25	24,32000000000000,340000000000,0,15,0.9,1
23,26	25,3000000000000,100000000000,0,15,0.6,1
23,27	26,32000000000000,340000000000,0,15,0.9,1
24,28	27,1400000000000,32000000000,0,15,0.1,1
25,28	28,32000000000000,340000000000,0,15,0.9,1
26,28	
27,28	

Workflow 5

29,30	29,31000000000000,340000000000,0,15,0.8,1
29,31	30,2900000000000,800000000000,0,15,0.7,1
29,32	31,2900000000000,800000000000,0,15,0.7,1
29,33	32,3400000000000,1000000,0,15,0.6,1
30,34	33,2900000000000,800000000000,0,15,0.7,1
31,34	34,3000000000000,100000000000,0,15,0.6,1
32,34	
33,34	

Conjunto 8:

Workflow 1

0,1	0,31000000000000,340000000000,0,15,0.8,1
0,2	1,32000000000000,340000000000,0,15,0.9,1
1,3	2,3400000000000,1000000,0,15,0.6,1
3,6	3,32000000000000,340000000000,0,15,0.9,1
2,4	4,3400000000000,1000000,0,15,0.6,1
4,5	5,31000000000000,340000000000,0,15,0.8,1
5,7	6,3000000000000,100000000000,0,15,0.6,1
6,7	7,1400000000000,32000000000,0,15,0.1,1

Workflow 2

8,9	8,32000000000000,340000000000,0,15,0.9,1
8,10	9,1400000000000,32000000000,0,15,0.1,1
8,11	10,1400000000000,32000000000,0,15,0.1,1
9,14	11,1400000000000,32000000000,0,15,0.1,1
10,14	12,1400000000000,32000000000,0,15,0.1,1
11,12	13,1400000000000,32000000000,0,15,0.1,1
12,13	14,1400000000000,32000000000,0,15,0.1,1
13,14	

Workflow 3

15,16	15,32000000000000,340000000000,0,15,0.9,1
16,17	16,3000000000000,100000000000,0,15,0.6,1
16,18	17,1400000000000,32000000000,0,15,0.1,1
17,19	18,1400000000000,32000000000,0,15,0.1,1
18,20	19,32000000000000,340000000000,0,15,0.9,1
19,23	20,31000000000000,340000000000,0,15,0.8,1
20,21	21,1400000000000,32000000000,0,15,0.1,1
21,22	22,32000000000000,340000000000,0,15,0.9,1
22,23	23,31000000000000,340000000000,0,15,0.8,1

Workflow 4

24,25	24,1400000000000,32000000000,0,15,0.1,1
24,26	25,32000000000000,340000000000,0,15,0.9,1
24,27	26,31000000000000,340000000000,0,15,0.8,1
25,30	27,32000000000000,340000000000,0,15,0.9,1
26,28	28,3000000000000,100000000000,0,15,0.6,1

28,29	29,32000000000000,340000000000,0,15,0.9,1
27,30	30,31000000000000,340000000000,0,15,0.8,1
29,30	

Conjunto 9:

Workflow 1

0,1	0,1400000000000,32000000000,0,15,0.1,1
0,2	1,3400000000000,1000000,0,15,0.6,1
0,3	2,2900000000000,800000000000,0,15,0.7,1
0,4	3,1400000000000,32000000000,0,15,0.1,1
1,5	4,32000000000000,340000000000,0,15,0.9,1
5,7	5,1400000000000,32000000000,0,15,0.1,1
2,6	6,1400000000000,32000000000,0,15,0.1,1
6,8	7,1400000000000,32000000000,0,15,0.1,1
3,17	8,1400000000000,32000000000,0,15,0.1,1
4,17	9,3400000000000,1000000,0,15,0.6,1
7,9	10,1400000000000,32000000000,0,15,0.1,1
7,10	11,3000000000000,100000000000,0,15,0.6,1
7,11	12,3400000000000,1000000,0,15,0.6,1
9,13	13,32000000000000,340000000000,0,15,0.9,1
10,15	14,2900000000000,800000000000,0,15,0.7,1
11,16	15,1400000000000,32000000000,0,15,0.1,1
8,12	16,1400000000000,32000000000,0,15,0.1,1
12,14	17,2900000000000,800000000000,0,15,0.7,1
13,17	
14,17	
15,17	
16,17	

Workflow 2

18,19	18,3000000000000,100000000000,0,15,0.6,1
18,20	19,1400000000000,32000000000,0,15,0.1,1
18,21	20,2900000000000,800000000000,0,15,0.7,1
18,22	21,31000000000000,340000000000,0,15,0.8,1
18,23	22,3000000000000,100000000000,0,15,0.6,1
18,24	23,2900000000000,800000000000,0,15,0.7,1

18,25	24,1400000000000,32000000000,0,15,0.1,1
18,26	25,1400000000000,32000000000,0,15,0.1,1
18,27	26,2900000000000,800000000000,0,15,0.7,1
19,30	27,1400000000000,32000000000,0,15,0.1,1
20,30	28,1400000000000,32000000000,0,15,0.1,1
21,30	29,3400000000000,1000000,0,15,0.6,1
22,30	30,1400000000000,32000000000,0,15,0.1,1
23,30	
24,30	
25,30	
26,30	
27,28	
28,29	
29,30	

Workflow 3

31,32	31,2900000000000,800000000000,0,15,0.7,1
31,33	32,3000000000000,100000000000,0,15,0.6,1
31,34	33,32000000000000,340000000000,0,15,0.9,1
31,35	34,1400000000000,32000000000,0,15,0.1,1
31,36	35,2900000000000,800000000000,0,15,0.7,1
31,37	36,3400000000000,1000000,0,15,0.6,1
31,38	37,1400000000000,32000000000,0,15,0.1,1
31,39	38,32000000000000,340000000000,0,15,0.9,1
31,40	39,32000000000000,340000000000,0,15,0.9,1
31,41	40,1400000000000,32000000000,0,15,0.1,1
32,42	41,2900000000000,800000000000,0,15,0.7,1
33,42	42,3400000000000,1000000,0,15,0.6,1
34,42	
35,42	
36,42	
37,42	
38,42	
39,42	
40,42	
41,42	

Workflow 4

43,44	43,2900000000000,800000000000,0,15,0.7,1
43,45	44,2900000000000,800000000000,0,15,0.7,1
44,62	45,1400000000000,32000000000,0,15,0.1,1
45,46	46,3400000000000,1000000,0,15,0.6,1
45,47	47,1400000000000,32000000000,0,15,0.1,1
46,48	48,2900000000000,800000000000,0,15,0.7,1
47,49	49,1400000000000,32000000000,0,15,0.1,1
48,50	50,1400000000000,32000000000,0,15,0.1,1
48,51	51,3000000000000,100000000000,0,15,0.6,1
50,53	52,1400000000000,32000000000,0,15,0.1,1
51,54	53,2900000000000,800000000000,0,15,0.7,1
49,52	54,3000000000000,100000000000,0,15,0.6,1
52,55	55,2900000000000,800000000000,0,15,0.7,1
53,62	56,1400000000000,32000000000,0,15,0.1,1
54,56	57,32000000000000,340000000000,0,15,0.9,1
54,57	58,2900000000000,800000000000,0,15,0.7,1
54,58	59,1400000000000,32000000000,0,15,0.1,1
56,59	60,3000000000000,100000000000,0,15,0.6,1
57,60	61,32000000000000,340000000000,0,15,0.9,1
58,61	62,1400000000000,32000000000,0,15,0.1,1
55,62	
59,62	
60,62	
61,62	

Conjunto 10:

Workflow 1

0,1	0,3000000,1000,0,15,0.6,1
1,2	1,14000000000000,320000000000,0,15,0.1,1
1,3	2,32000000000000,340000000000,0,15,0.9,1
1,4	3,32000000000000,340000000000,0,15,0.9,1
2,5	4,2900000000000,800000000000,0,15,0.7,1
3,5	5,31000000000000,340000000000,0,15,0.8,1
4,5	6,14000000000000,32000000000,0,15,0.1,1
5,6	7,34000000,100,0,15,0.6,1

6,7	

Workflow 2

8,9	8,340000000000000,1000000000,0,15,0.6,1
9,10	9,320000000000,3400000000,0,15,0.9,1
10,11	10,34000000000,10000,0,15,0.6,1
11,12	11,14000000000,32000000,0,15,0.1,1
	12,3000000000000,100000000000,0,15,0.6,1