Provisionamento dinâmico de recursos para execução de workflows científicos em nuvens

Ricardo Juliano Mesquita Silva Oda

Dissertação apresentada
ao
Instituto de Matemática e Estatística
da
Universidade de São Paulo
para
obtenção do título
de
Mestre em Ciências

Programa: Ciência da Computação

Orientadora: Prof[a]. Dr[a]. Kelly Rosa Braghetto

Durante o desenvolvimento deste trabalho o autor recebeu auxílio financeiro do CNPq
Processo: 134403/2013-4

São Paulo, agosto de 2016

Provisionamento dinâmico de recursos para execução de workflows científicos em nuvens

Esta é a versão original da dissertação elaborada pelo candidato Ricardo Juliano Mesquita Silva Oda, tal como submetida à Comissão Julgadora.

Resumo

ODA, R. J. M. S. **Provisionamento dinâmico de recursos para execução de workflows científicos em nuvens** 2016. 46 f. Dissertação (Mestrado) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2016.

Workflows científicos têm o papel de automatizar experimentos, análises e processos científicos que manualmente seriam impraticáveis devido ao volume de dados envolvidos. A nuvem é uma plataforma emergente que serve como infraestrutura para executar esse tipo de procedimento. No uso da nuvem, esses sistemas não se adaptam a variações na carga de trabalho por serem passivos às configurações da infraestrutura e, assim, causam possíveis sobrecargas ou subutilizações dos recursos computacionais disponíveis. Este trabalho propõe melhoras na utilização dos recursos das plataformas de computação em nuvem pelos sistemas gerenciadores de workflows científicos durante a execuções de workflows. Um novo mecanismo para gerenciar o provisionamento de forma dinâmica foi proposto, implementado e validado em uma plataforma real de computação em nuvem. O mecanismo proposto revela um modo de utilizar informações do workflow a fim de beneficiar-se de característica da nuvem para melhorar a utilização de recursos. Além disso, um de caso do compartilhamento de processadores entre atividades foi estudado, mostrando que tal compartilhamento causa um impacto negativo nos tempos de execução das tarefas, porém promove o aumento do paralelismo que, por sua vez, pode melhorar a execução do workflow como um todo.

Palavras-chave: workflows científicos, provisionamento de recursos, escalonamento, computação em nuvem.

Abstract

ODA, R. J. M. S. **Dynamic resource provisioning for scientific workflows execution in clouds**. 2016. 46 f. Thesis (Master) - Instituto de Matemática e Estatística, Universidade de São Paulo, São Paulo, 2016.

Scientific workflows have the role of automating scientific experiments, analyses and processes that would be manually impractical due to the amount of data involved. The cloud is an emerging platform which serves as infrastructure to host this type of procedure. This work proposes improvement in the usage of the cloud computing platform resources by the scientific workflow management systems during workflow executions. In the cloud, these systems do not adapt to workload changes, being passive to the infrastructure configuration and, thus, cause potential overloads or under-utilization of available computational resources. A new mechanism to manage the provisioning dynamically was proposed, implemented and validaded in a real cloud computing plataform. The proposed mechanism displays a way to use the workflow data in order to benefit from cloud characteristics to improve the resource utilization. Futhermore, a case of processors sharing among jobs was studied, showing that such sharing causes a negative impact on the jobs execution times, nevertheless the sharing promotes paralelism which in turn can improve the workflow execution as a whole.

Keywords: scientific workflows, resource provisioning, scheduling, cloud computing.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

De acordo com Jim Gray [HTT09], estamos vivenciando uma nova forma de se fazer ciência, fortemente baseada na análise de enormes quantidades de dados. Nesse paradigma, essa quantidade enorme de dados pode ser proveniente de uma captura automatizada ou manual, feita por instrumentos ou até mesmo gerada em simulações. Antes de serem analisados pelos cientistas, esses dados necessitam de um processamento intensivo que, por sua vez, possibilita reduzir o elevado volume de informação, antes incompreensível por um ser humano, a algo que seja legível e interpretável. Nesse caso, a exploração e tratamento dos dados são fatores essenciais do se fazer ciência que amparam a análise tanto empírica como teórica de problemas do mundo científico.

Com essa nova abordagem do se fazer ciência, é natural falarmos do conceito *eScience* (eCiência), que unifica a teoria, experimentação e simulação feita em ciência, e ao mesmo tempo lida com grandes volumes de informação e um elevado poder computacional. Em eScience, muitos cientistas trabalham com os chamados *workflows científicos*, uma classe de *workflow* (fluxo de trabalho) voltado para pesquisa. Desde seu surgimento nos anos de 1980, workflows científicos têm ganhado atenção e, atualmente, tornaram-se um ferramental importante durante a realização de experimentos científicos de grande escala [TDGS07]. Workflows são ferramentas para organização e automação de processos, onde o processamento ocorre seguindo uma sequência pré-definida de passos e regras. No caso de workflows científicos, o processamento é feito de informações de forma a possibilitar a automação de experimentos e análises que manualmente seriam impraticáveis devido à quantidade de dados envolvidos.

Um exemplo de processo científico na forma de workflow é a composição de imagens feita pelo *Montage* [mon]. O Montage é uma ferramenta de código aberto desenvolvida pelo *IRSA* (*NASA/IPAC Infrared Science Archieve*) para agregar imagens em mosaicos personalizados. É aplicado na área de astronomia para criar composições de imagens do espaço a partir de dados coletados por telescópios. As imagens de entrada seguem o fluxo de processamento definido pelo workflow, o qual é composto por diversos passos. Nesse fluxo as imagens são tradadas, alinhadas e por fim agregadas, formando uma composição única.

Sistemas gerenciadores de workflows científicos (SGWCs) são uma classe de ferramentas criadas para facilitar a composição, o gerenciamento e a execução de workflows complexos. Ao utilizá-los, um cientista tem a possibilidade de compor e facilmente executar workflows em ambientes com muitos recursos computacionais. Assim, mesmo sem conhecimentos específicos na área de computação, seja de programação para definição dos roteiros de execução, ou técnicos para configuração dos recursos computacionais, o cientista consegue executar um experimento complexo através de um

workflow científico. Utilizando SGWCs, também há a possibilidade de compartilhar workflows com a comunidade científica e/ou executar workflows compartilhados por outros centros e institutos de pesquisa.

Uma das plataformas de computação em evidência nos últimos tempos é a plataforma de *computação em nuvem* (*cloud computing*). Essa plataforma já é amplamente usada na indústria, por prover grande quantidade de recursos computacionais sob demanda, em pouco tempo e em quantidades precisas. Mas a computação em nuvem também é atrativa para cientistas que necessitam de processamento intensivo de dados. Com ela, pequenos laboratórios não necessitam mais do investimento em recursos como supercomputadores, aglomerados (*clusters*) ou grades computacionais (*grids*): basta alugar os recursos necessários para efetuar o processamento desejado.

Recursos computacionais em nuvens geralmente fazem referência a *máquinas virtuais* (*VMs*). Não há um controle direto sobre os recursos físicos de uma nuvem, já que ela funciona por meio de virtualização. Máquinas virtuais são ambientes virtuais que podem compartilhar o uso de recursos físicos mas possuem um isolamento lógico. Isto é, várias instâncias de máquinas virtuais podem residir em uma única máquina real. Disponibilizados como infraestrutura, esses recursos podem ser alocados e configurados conforme o necessário.

O provisionamento dinâmico de recursos permite que a demanda por poder computacional em processamento de dados seja atendida, isso é um diferencial que a nuvem proporciona através da *elasticidade*. A elasticidade da nuvem representa o poder de adaptação do sistema pela alocação e liberação de recursos computacionais em tempo de execução. Quando há subutilização de recursos, esse controle pode diminuir o desperdício pela liberação de recursos excedentes. Além disso, quando os recursos estiverem sobrecarregados, permite que o poder computacional seja ampliado através da disponibilização de novos recursos.

1.1 Motivação

Para evitar e combater enchentes, é imprescindível a existência de infraestrutura para drenagem adequada da água. Do mesmo modo, infraestrutura tecnológica e recursos computacionais são essenciais durante o atual *dilúvio de dados* (Data Deluge) [HT03, BHS09]. Sejam gerados em experimentos, simulações, por sensores e/ou satélites, os dados podem ser tratados por meio de programas computacionais, no nosso caso: workflows científicos. Esse eminente volume de informação não é compatível com o poder computacional em posse da maioria dos pesquisadores, laboratórios e instituições.

A plataforma de computação em nuvem é uma possível solução para sanar a demanda por recursos computacionais, que é gerada pelo aumento da quantidade de dados analisados em experimentos científicos. Tendo seus dados transferidos para uma nuvem, um workflow científico pode ser executado nessa infraestrutura remota e, ao término de sua execução, o resultado pode ser entregue ao usuário que requisitou o processamento. O ambiente de computação em nuvem possibilita o controle da quantidade e qualidade dos recursos utilizados, que variam dependendo de fatores como: a quantidade de dados que serão processados, da verba do usuário para custear a execução do experimento e do tempo limite para o término da execução.

Os SGWCs nasceram e se desenvolveram utilizando ambientes para computação de alto desempenho como grades (*grids*) e aglomerados (*clusters*). Atualmente, os sistemas gerenciadores de

workflows estão se adaptando, e a maioria já utiliza nuvens como plataforma de computação para execução de workflows. Diversos trabalhos [HMF+08, JDV+09, DSL+08] mostram que a nuvem é uma plataforma viável e atraente para execução de workflows científicos. Contudo a adaptação feita ainda não se beneficia plenamente das vantagens existentes na plataforma de computação em nuvem. Os recursos da nuvem são utilizados da mesma forma como os recursos das grades eram utilizados, porque a elasticidade não é utilizada. Aglomerados virtuais de tamanho estático são criados e configurados a priori para, então, serem utilizados como recursos de aplicações de workflows. Durante execuções de workflows, não ocorrem alterações nos recursos disponíveis, nem no escalonamento das atividades (feito no início da execução de cada workflow). O provisionamento inicial dos recursos é fixo durante toda a execução do workflow. Nisso, não ocorre a alocação dinâmica de recursos, e a elasticidade que a nuvem tem como ponto diferencial é descartada. Ou seja, a nuvem, após a configuração inicial de execução, é utilizada como um aglomerado estático de máquinas.

Há duas principais características que diferenciam a infraestrutura concedida por uma nuvem, daquela proveniente de uma grade computacional. A primeira é a medição da quantidade, qualidade e tempo de utilização dos recursos alocados para cobrança posterior. E a segunda é a elasticidade, que permite tanto a alocação, como a liberação de recursos durante a execução de uma aplicação.

Esse tipo de execução de workflows científicos em nuvens é diferente do que acontece, por exemplo, com a hospedagem de sites. Os próprios provedores de ambientes de computação em nuvem já fornecem servidores *web* horizontalmente escaláveis, onde, um balanceador de carga controla o número de servidores ativos e é responsável pela distribuição do tráfego de acesso entre esses servidores [VRMB11]. Isso tende a minimizar o tempo ocioso das máquinas, pois é feito de acordo com a demanda de acessos do site hospedado.

O *escalonamento* de workflows, processo que distribui as atividades (passos do workflow a serem executados) entre os recursos disponíveis, já existe para plataformas de computação em nuvem. Vários trabalhos [RHRB13, WDL+13, MJDN12] tratam a questão do escalonamento de atividades em plataformas de computação em nuvem. Apesar de algum desses trabalhos lidarem com restrições sobre o custo monetário e prazo de término de execução, nenhum ataca a questão de utilizar-se da elasticidade da nuvem e as informações do workflow para adaptar os recursos disponíveis à execução de um workflow em tempo de execução. Outra observação é que muitos deles são feitos em cima de simulações, por exemplo, utilizado o *CloudSim* [CRB+11].

Elaborar e utilizar mecanismos de provisionamento dinâmico de recursos em sistemas implantados em nuvens ainda é um desafio, principalmente em sistemas singulares como os gerenciadores de workflows científicos. Vários motivos dificultam a implementação e uso da elasticidade em SGWCs. Durante a execução de um workflow, existem variações na demanda por recursos causadas pela estrutura do workflow e pelas diferenças entre as atividades que o compõem. Um workflow pode começar com uma única atividade grande e depois ramificar-se para inúmeras atividades que podem ser executadas em paralelo, resultando em um mudança expressiva na utilização dos recursos. As variações se tornam ainda mais complexas quando são consideradas várias instâncias de workflows. As instâncias podem ser iniciadas, executadas e finalizadas de forma concorrente, gerando oscilações na taxa de requisição e utilização dos recursos e, consequentemente, podem sobrecarregar ou subutilizar os recursos disponíveis no sistema.

Na implementação dos mecanismos, o uso da elasticidade deve ser automatizado, pelo menos em parte, para que os cientistas não necessitem interagir com o controle de recursos. Políticas e

mecanismos devem ser definidos para o controle do aumento e da diminuição dos recursos que atendam a variação da necessidade de recursos do sistema. Além disso, esse controle meticuloso sobre a infraestrutura varia conforme o uso de distintos provedores de computação em nuvem, devido à diversidade de produtos e serviços disponíveis em cada provedor. Todos esses desafios são enfrentados pelos SGWCs atuais, os quais ainda não funcionam de forma inteligente e autônoma na utilização de recursos, pois são passivos com relação à configuração da infraestrutura.

1.2 Objetivos Gerais

Este trabalho aborda o problema de provisionamento dinâmico de recursos para a execução de workflows científicos em ambientes de nuvens. A melhora na utilização de plataformas de computação em nuvem pelos SGWCs é o alvo deste trabalho. Em suma, queremos melhorar a eficiência da utilização de recursos de plataformas computação em nuvem pelos SGWCs durante execuções de workflows científicos, onde não há adaptações às variações na carga de trabalho gerando, assim, possíveis sobrecargas e/ou subutilizações dos recursos computacionais disponíveis.

Para atacar tal questão, primeiramente, estudamos o compartilhamento de processadores entre atividades de workflows científicos. O compartilhamento causa um impacto negativo no tempo de execução das tarefas, mas permite que haja um maior paralelismo e pode melhorar o desempenho geral do workflow como um todo.

Também, estudamos um mecanismo de gerenciamento autônomo e dinâmico do provisionamento e escalonamento do workflow. A principal função desse mecanismo é a de liberar ou alocar recursos dinamicamente, e mapear as atividades em recursos disponíveis. Dessa forma, é possível otimizar a utilização dos recursos.

1.3 Contribuições

As principais contribuições deste trabalho são:

1. A proposta de um novo mecanismo de gerenciamento e escalonamento que explora a elasticidade das nuvens para a execução eficiente de workflows científicos;
2. A implementação desse mecanismo para o SGWC Pegasus e o gerenciador de recursos HTCondor;
3. Experimentos com workflows reais para validar o mecanismo e avaliar os ganhos proporcionados por ele;
4. Experimentos analisando o impacto do compartilhamento de recursos entre atividades durante a execução de workflows.

Os experimentos foram realizados em nuvens comerciais da *Google* e da *Microsoft*, utilizando ferramentas de software livre como o *Pegasus* (gerenciador de workflows científicos) e o *HTCondor* (gerenciador de recursos) como objeto de estudo e alvo das implementações. O *Montage* e *Epigenomics*, aplicações reais e comuns na comunidade cientifica, foram utilizados para execução dos experimentos.

O primeiro dos experimentos, descrito no Capítulo 5, investigou o caso do compartilhamento de recursos entre atividades de um workflow. Os workflows foram configurados para serem executados em vários níveis de particionamento de recursos entre as atividades. E o impacto desse particionamento sobre a execução foi analisado.

O segundo experimento, descrito no Capítulo 6, foi feito validando o mecanismo desenvolvido neste trabalho. O mecanismo de provisionamento de recursos e escalonamento de atividades surgiu do estudo de como utilizar a elasticidade da nuvem para melhorar a execução de workflows científicos em plataformas de computação em nuvem. Ele consegue inferir um número ideal de recursos que devem ser alocados em um dado momento. Utiliza informações de execuções passadas e limita-se à um orçamento que tem como entrada junto aos workflows que gerencia. Para sua validação, o algoritmo desenvolvido foi implementado [pro] possibilitando sua execução junto ao Pegasus [peg].

A análise dos experimentos revela que o provisionamento e escalonamento a partir de novas técnicas podem prover melhor desempenho e utilização de recursos. O mecanismo descrito neste trabalho se destaca por beneficiar-se das características da nuvem e utilizar informações do próprio workflow para melhorar seu desempenho. Validamos a ideia de que a plataforma de computação em nuvem é um ambiente propício para esse tipo de solução, devido à característica de elasticidade que possibilita a implementação de um auto provisionamento de recursos.

1.4 Organização do Texto

O restante deste texto está organizado como descrito a seguir. No Capítulo 2, os fundamentos relacionados ao trabalho são descritos em mais detalhes. São apresentados fundamentos sobre workflows, a plataforma de computação em nuvem e escalonamento. No Capítulo 3, trabalhos relacionados são listados e discutidos. No Capítulo 4, é apresentado como foram ambientados os experimentos deste trabalho. Nos Capítulos 5 e 6, cada experimento é descrito em detalhes, desde sua motivação à análise dos resultados. Por fim, no Capítulo 7, uma conclusão é apresentada ressaltando as contribuições deste trabalho; além disso, possíveis trabalhos futuros são delineados.

Capítulo 2

Fundamentos

2.1 Workflows Científicos

Os chamados *workflows de negócio* são utilizados para automação e gerenciamento de *processos de negócios*. Atualmente, workflows auxiliam o mundo acadêmico: os *workflows científicos* possibilitam a automação de processos e experimentos científicos de diversas áreas.

De modo simplificado, workflows descrevem processos de forma pragmática. Formalmente, *workflows científicos* são definidos como "redes de processos tipicamente utilizadas como '*pipelines* de análises de dados' ou ainda para comparar dados observados ou previstos, e que podem incluir uma vasta gama de componentes, e.g. para consultar bancos de dados, para transformar ou minerar dados, para executar simulações em computadores de alto desempenho, etc." [LAB+06]. Assim, um workflow científico descreve de forma pragmática um experimento ou processo científico. E seus passos são descritos como atividades interligadas de acordo com as precedências que podem existir entre elas e/ou as dependências dos dados necessários para sua execução [YB05].

A automação decorrente dos workflows científicos permite a execução de análises que antes eram impraticáveis manualmente devido à grande carga de dados envolvidos. Desse modo, workflows científicos criam oportunidades para análises computacionalmente intensas e, principalmente, que envolvam grandes volumes de dados.

O ciclo de vida de um workflow científico é composto por quatro fases [LWMB09]:

1. *Projeto*: consiste na construção de um modelo de workflow a partir de uma hipótese a ser testada, ou de algum objetivo experimental específico. Normalmente, modelos de workflows preexistentes são reutilizados ou refinados. Tanto projetos de workflows como produtos (resultados, componentes, subworkflows, etc.) podem ser compartilhados entre pesquisadores através de repositórios.

2. *Instanciação*: é a fase onde os dados de entrada são selecionados e os parâmetros de execução são definidos. Em função de onde e como a execução será efetuada, recursos computacionais são escolhidos e alocados. Além disso, os dados de entrada devem ser transferidos para o ambiente onde ocorrerá a execução.

3. *Execução*: durante a execução, a entrada é consumida e novos dados são gerados. Nessa etapa, é importante haver um monitoramento que permita o acompanhamento de dados intermediários

e informações de proveniência do workflow. Assim, o progresso e possíveis erros de execução podem ser visualizados pelos cientistas.

As informações de proveniência representam os dados necessários para reprodução do experimento. Incluem os dados de entrada, os parâmetros definidos inicialmente, o registro das atividades já executadas, seus respectivos tempos de início e término, etc.

4. *Análise Pós-Execução*: ao fim da execução, os cientistas interpretam os resultados obtidos, examinam os traços de execução e dependência dos dados, efetuam depurações, ou simplesmente observam o desempenho da execução. Deste modo, incoerências podem ser identificadas, como por exemplo: a hipótese inicial pode ser rejeitada, ou um gargalo da execução do workflow pode ser identificado.

   Dependendo do resultado dessa análise, a hipótese inicial ou os objetivos experimentais podem ser revisados ou refinados, dando origem a uma nova iteração do ciclo de vida do workflow.

Para não haver ambiguidade quando nos referirmos a múltiplas instâncias de workflows, definiremos o termo *série de execução de workflows* ou *série de workflows*. Uma *série de execução* (de workflows) é a execução de uma ou mais aplicações de workflow por um ou mais usuários, utilizando uma ou mais instâncias de workflows, em uma ordem qualquer, com ou sem concorrência, e em um ambiente comum de execução. O termo *instância de workflow* será definido na Seção 2.1.1.

O conceito, série de execução de workflows, pode ser empregado quando um grupo de pesquisadores ou uma organização compartilha uma configuração de execução em um ambiente de nuvem, de modo que instâncias de workflows de diferentes usuários disputem os recursos disponíveis. Ou, quando um experimento, feito através de um workflow, precisa ser executado com diversas entradas e parâmetros, assim instâncias distintas de uma mesma aplicação de workflow podem ser executadas em paralelo com diferentes configurações. Podemos citar, também, o uso de *workflow ensembles* [DJMN13], que representam experimentos obtidos a partir da execução de conjuntos de workflows independentes mas relacionados entre si.

Neste trabalho estaremos interessados nas fases de instanciação e, principalmente, de execução de workflows durante execuções de séries de workflows.

2.1.1 Terminologia

Workflows científicos são compostos principalmente por atividades de processamento de dados e pelas conexões que ligam essas atividades. Os termos básicos usados na representação de workflows são os seguintes:

Modelo de workflow Define a organização das atividades que devem ser executadas no workflow para que um determinado objetivo científico seja atingido. Através das conexões, estabelece a ordem (ainda que parcial) na qual as atividades devem ser executadas.

Pode ser representado por um arcabouço formal ou uma linguagem de especificação (ou modelagem) de workflow. Arcabouços formais possibilitam a análise de propriedades dos processos modelados, mas as linguagens de especificação possuem um nível de detalhe mais próximo do que é requerido para a execução do processo.

Estamos interessados nas linguagens de especificação, pois essa é representação utilizada para definir workflows científicos em SWGCs.

Atividade É uma unidade atômica de trabalho, um passo da execução do workflow. Pode ser um acesso ao banco, um filtro de dados, uma transformação, etc. Na literatura também é conhecida como: *tarefa, processo, ação, transição.*

Um workflow pode ter passos manuais, ou semiautomáticos, onde há interação humana no processo. Mas, neste trabalho estamos interessados em workflows compostos somente por atividades automáticas que são definidas, executadas e gerenciadas através de SGWCs. Além disso, consideraremos que as atividades são *caixas pretas*, isto é, a priori não há conhecimentos sobre as características das atividades como, por exemplo, a duração de sua execução em relação à entrada, ou a carga de trabalho que a atividade gera.

Conector Define dependências entre as atividades, podendo representar uma relação de precedência entre atividades, ou a necessidade da transferência de dados de uma atividade para outra. A diversidade de conectores em uma linguagem de especificação define sua representatividade.

Instância de workflow É uma execução particular de um dado workflow com uma determinada entrada e parâmetros, isto é, a instanciação de um modelo de workflow em específico.

Instâncias distintas de um mesmo workflow não necessariamente executarão suas atividades em uma mesma ordem, já que um workflow pode ter em seu modelo ramos alternativos de execução, ou atividades que são executadas somente com a presença de um tipo específico de dado.

2.1.2 Modelagem e Representação de Workflows Científicos

Há uma grande diversidade de aplicações de workflows e, por consequência, uma grande variedade de SGWCs. Não há uma representação padrão para os modelos de workflows científicos, cada SGWC possui sua própria linguagem para definição e representação de workflows, além de diferentes modelos de execução com foco em diversos domínios de aplicações [DGST09, YB05].

Um workflow pode ser modelado sob diferentes perspectivas, duas delas são relevantes para este trabalho:

Fluxo de Controle Define-se a precedência e, por consequência, uma ordem (parcial) entre as atividades por meio de diferentes conectores. O controle é transferido entre atividades de forma que uma atividade é inciada quando as atividades que a precedem forem concluídas.

Fluxo de Dados Retrata a dependência entre os dados e as atividades, e também a forma como os dados são transferidos entre atividades.

Essas perspectivas não são excludentes, a maioria dos SGWCs utilizam linguagens de modelagem híbridas que podem descrever fluxos de controle e de dados. Um exemplo comum de modelagem de workflows é o uso de grafos acíclicos dirigidos (em inglês *directed acyclic graphs* ou DAGs), onde os vértices representam tarefas, e as arestas representam os conectores que podem definir tanto dependências de fluxo de dados como de fluxo de controle. É o modelo utilizado pelo Pegasus [peg, DSS+05]. A Figura 2.1 é um exemplo de DAG com quatro atividades, onde cada atividade possui um arquivo de entrada e pelo menos um arquivo de saída. As dependências do fluxo de dados são representadas pelas arestas dirigidas entre arquivos e atividades.

As construções básicas da estrutura do fluxo de controle de workflows são [vDATHKB03, WfM99]:

- *Sequência:* encadeamento de atividades onde uma atividade é executada ao fim da conclusão de outra atividade.
- *Paralelismo (Divisão-E):* divisão do fluxo em ramos onde as atividades podem ser executadas em paralelo.
- *Sincronização (Junção-E):* encontro de diferentes ramos do fluxo de execução, onde há uma sincronização das atividades.
- *Escolha Exclusiva (Divisão-OU-Exclusivo):* divisão do fluxo em ramos onde somente um deles permanece com o controle de execução. A escolha baseia-se em uma decisão ou dados do controle do workflow.
- *Junção (Junção-OU-Exclusivo):* convergência de ramos alternativos de execução, sem a necessidade de sincronização.
- *Ciclo (ou Iteração):* repetição da execução de um bloco de uma ou mais atividades do workflow.

E as construções básicas do fluxo de dados são [BCD+08]:

- *Processamento:* representa o processamento efetuado por uma atividade, a qual consume dados de entrada para produção de dados de saída.
- *Pipeline:* é formado por processamentos sequenciais de forma que uma atividade consome os dados de saída de sua precedente.
- *Distribuição de dados:* ocorre quando uma atividade produz mais do que um conjunto de dados de saída a partir do processamento de somente uma entrada. Pode ocorrer tanto para distribuição da mesma saída para diferentes atividades, como para divisão dos dados em partições menores a fim de aumentar o paralelismo.
- *Agregação de dados:* ocorre quando uma atividade processa múltiplos dados de entrada para produção de um conjunto de saída. Serve como sincronização para linhas paralelas de execução.
- *Redistribuição de dados:* ocorre quando uma atividade processa múltiplos dados de entrada e produz diversos conjuntos de saída. Pode sincronizar linhas paralelas de execução, além de criar novos ramos de paralelismo.

2.1.3 Sistemas Gerenciadores de Workflows Científicos

Um *sistema gerenciador de workflows científicos* (SGWC) é "um sistema que define por completo, modifica, gerencia, monitora e executa workflows científicos através da execução de tarefas científicas, cuja ordem de execução é ditada por uma representação computacional da lógica do workflow" [LLF+09]. Assim, complexos experimentos e procedimentos científicos são modelados, executados e analisados por meio de SGWCs.

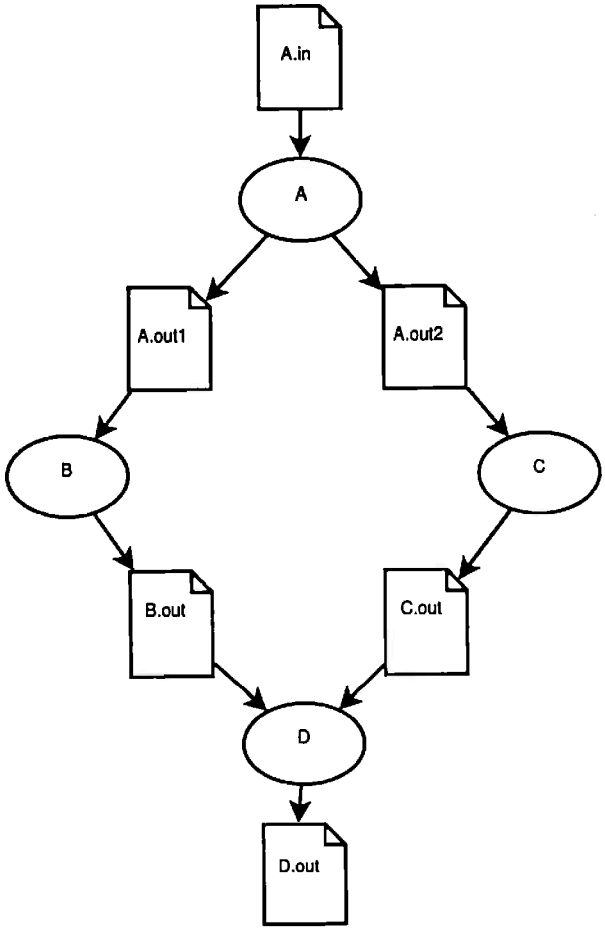

Figura 2.1: *Exemplo de DAG com quatro atividades e seis arquivos. As arestas definem o fluxo de dados, ou seja, as dependências entre as atividades e arquivos de dados. A entrada do workflow é o arquivo* A.in *e a saída o arquivo* D.out*, os outros arquivos são resultados de processamentos intermediários. Baseado no* Diamond Workflow *presente no tutorial do Pegasus [peg].*

Cada SGWC possui suas próprias características e atua em um ou mais domínios específicos de aplicações. Mas podemos citar subsistemas gerais que compõem um SGWC [LLF+09]:

- *Subsistema de Projeto:* possibilita gerar e editar workflows científicos através de uma linguagem de modelagem de workflows. Através desse subsistema é possível haver a interoperabilidade entre diferentes SGWCs, normalmente pelo uso de uma linguagem padrão. Contudo a maioria dos SGWCs possui uma linguagem própria.

- *Subsistema de Visualização e Apresentação:* permite apresentar tanto o workflow como os resultados finais e parciais gerados em sua execução. Utiliza representações digitais em diversos formatos para armazenar essas informações. E assim, facilita a análise e extração do conhecimento criado durante a execução.

- *Subsistema de Execução:* é composto por ferramentas computacionais destinadas ao gerenciamento da execução do workflow. É responsável pela instanciação e execução das atividades de um workflow de acordo com sua representação digital. Além disso, também cuida do escalonamento dessas atividades ao longo dos recursos computacionais disponíveis. E pode comportar, em um mesmo momento, múltiplas execuções concorrentes de diferentes instâncias de workflows.

- *Subsistema de Monitoramento:* monitora o estado das instâncias de workflows durante as execuções. Também provê ferramentas para identificação e tratamento de falhas caso ocorram.

- *Subsistema de Gerenciamento de Atividades:* é endereçado a problemas de distribuição e integração de atividade heterogêneas. Provê o gerenciamento do ambiente necessário para a execução das atividades de acordo com o modelo do workflow.

- *Subsistema de Gerenciamento de Proveniência:* é responsável pelo gerenciamento de dados e metadados de proveniência dos workflows científicos, através de funções como: representação, visualização, armazenamento e consulta.

- *Subsistema de Gerenciamento de Dados:* gerencia os dados consumidos e produzidos pelas atividades durante a execução de workflows. Deve lidar com problemas de heterogeneidade e movimentação de grande volumes de dados.

Neste trabalho estamos interessados principalmente nos subsistemas de execução e monitoramento. Além do monitoramento do estado das atividades, um monitoramento do estado dos recursos computacionais também deverá ocorrer. Assim, o gerenciamento dos recursos e escalonamento das atividades poderá ser dinâmico.

2.1.4 Gerenciadores de Recursos

O problema do gerenciamento de recursos em SGWCs é a atribuição eficiente de recursos para processar as atividades em tempo de execução. Os SGWCs possuem *gerenciadores de recursos* que tem o papel de:

1. Monitorar o estado dos recursos;

2. Localizar recursos elegíveis e disponíveis para execução de atividades;

3. Escalonar as atividades ao longo dos recursos disponíveis;

4. Gerenciar o provisionamento de recursos computacionais.

Gerenciadores de recursos podem ser encontrados, por exemplo, nos subsistemas de execução e de gerenciamento de dados descritos na Seção 2.1.3. No primeiro caso para escalonamento das atividades e alocação de recursos computacionais para o processamento de workflows, e no segundo para gerenciamento da transferência de dados e do provisionamento de recursos de armazenamento.

O gerenciamento de recursos é simples quando o quantidade de recursos é pequena e/ou todos os recursos são locais, pois podem ser controlados por um gerenciador centralizado, contudo o problema fica difícil quando os recursos são distribuídos. Podemos citar duas abordagens para o gerenciamento de recursos distribuídos [DES98]:

Gerenciamento global Essa abordagem utiliza um *gerenciador global de recursos* (*Global Resource Manager* ou GRM) para administrar os recursos distribuídos. Todos os recursos são registrados no GRM com os papéis que podem assumir, isto é, uma referência das atividades que podem ser atendidas é armazenada para cada recurso (um programa que executa uma atividade pode ser implantado em recurso específico). O gerenciador também é responsável por monitorar o estado de cada recurso registrado, ou seja, a atribuição das atividades ao longo dos recursos é monitorada.

Gerenciamento local São utilizados múltiplos *gerenciadores locais de recursos* (*Local Resource Manager* ou LRM) distribuídos ao longo do sistema. Cada LRM armazena toda informação e tem controle total sobre os recursos de um determinado local. Pode haver um GRM central que mantenha somente as informações dos papéis dos recursos e a que LRM eles pertencem. Os SGWCs utilizam individualmente os LRM para atribuição de recursos quando um processamento deve ocorrer, e cada LRM é responsável por monitorar o estado de seus recursos.

O controle sobre os recursos é fácil e eficiente quando há um gerenciamento global devido à localidade da informação sobre todos os recursos. Contudo há um grande impacto no desempenho quando monitora-se o estado de diversos recursos remotos. Por isso, a primeira abordagem acaba não sendo utilizada na prática, já que o uso de múltiplos recursos distribuídos é grande, como ocorre em plataformas de computação em nuvem.

2.2 Plataforma de Computação em Nuvem

O termo *computação em nuvem* engloba diferentes arquiteturas, tecnologias e serviços, mas, em geral refere-se às aplicações entregues como serviços pela Internet e também ao hardware e software presentes nos *data centers* que fornecem esses serviços [AFG+09]. Assim, *plataforma de computação em nuvem* é o ambiente que permite a existência dessas aplicações, e o uso dessa plataforma é nada mais que a utilização de pelo menos um de seus serviços. Chamaremos de *nuvem*, ou *infraestrutura da nuvem*, o hardware e software dos *data centers*.

Uma definição formal, de acordo com o *National Institute of Standards and Technology* (NIST), diz que "computação em nuvem é um modelo para possibilitar o acesso pela rede de forma ubíqua,

conveniente e sob demanda, para um conjunto compartilhado de recursos computacionais configuráveis (e.g., redes, servidores, armazenamento, aplicações, e serviços) que podem ser rapidamente provisionados e liberados com esforço mínimo de gerenciamento ou interação com o provedor de serviços" [MG09].

Tipos distintos de nuvem são desenvolvidos com diferentes finalidades. Conforme são concebidas, as nuvens podem oferecer uma variedade de serviços em diversos níveis de abstração. Os modelos de serviço seguem o formato *XaaS*, que significa *X como Serviço* (do inglês *X as a Service*), onde *X* é o tipo de serviço fornecido. Os serviços da nuvem podem ser classificados nos seguintes modelos [MG09]:

Software como Serviço (SaaS) O consumidor tem o poder de utilizar as aplicações do provedor que estão implantados na nuvem. Tais aplicações podem ser acessadas através da Internet por meio de diferentes dispositivos e interfaces, como navegadores ou programas próprios. O consumidor não gerencia nem controla a infraestrutura da nuvem por trás da aplicação, incluindo rede, servidores, sistemas operacionais, armazenamento ou até mesmo características específicas da aplicação com exceção de configurações limitadas para usuários. Exemplos: Gmail [gma], Twitter [twi].

Plataforma como Serviço (PaaS) O consumidor pode utilizar o ambiente da nuvem para instalar aplicações próprias ou adquiridas através de linguagens, bibliotecas, serviços e ferramentas aceitas pelo provedor. O consumidor não gerencia nem controla a infraestrutura da nuvem por trás da aplicação, incluindo rede, servidores, sistemas operacionais, armazenamento, mas tem poder sobre características específicas da aplicação e possivelmente configurações sobre a hospedagem e sobre ambiente em que a aplicação está implantada. Exemplos: Google app engine [app], Microsoft Azure [azuc].

Infraestrutura como Serviço (IaaS) O consumidor tem a disposição processamento, armazenamento, rede e outros recursos básicos de computação, geralmente através da virtualização da infraestrutura do provedor. Assim, de forma arbitrária, o consumidor pode instalar e utilizar softwares na infraestrutura da nuvem, incluindo sistemas operacionais e aplicações. O consumidor não gerencia nem controla a infraestrutura da nuvem mas administra o armazenamento e os sistemas operacionais, além das aplicações implantadas, e possivelmente configurações limitadas sobre componentes da rede como firewall e portas de acesso. Exemplos: Amazon EC2 [ec2], GoGrid [gog].

O foco deste trabalho será o modelo IaaS, onde recursos virtuais podem ser alocados e sistemas criados de forma *ad-hoc* para atender a necessidade das aplicações de workflows científicos. Recursos computacionais em nuvens geralmente fazem referência a *máquinas virtuais* (em inglês *virtual machines*, ou *VMs*), que comportam sistemas operacionais e servem como alvo de implantação de softwares. Configurações personalizadas (de núcleos de CPU, memória RAM, disco) podem ser atribuídas às VMs conforme a necessidade do usuário. Os SGWCs utilizam VMs como base para implantação de programas para execução de workflows científicos.

As seguintes características da infraestrutura da nuvens destacam-se como úteis para aplicações de workflows científicos:

Auto provisionamento Em plataformas de computação em nuvem, o usuário pode provisionar os recursos que precisar e definir um escalonador para controlar a execução de seus programas. Os recursos são isolados virtualmente para cada usuário. Esse modelo é ideal para aplicações pouco acopladas como workflows, permitindo que uma alocação de recursos atenda diversas utilizações por diferentes programas, o que diminui o *overhead* total de escalonamento e melhora o desempenho da execução de um workflow. Diferente do que ocorre em *grades* (grids) e *aglomerados* (clusters), onde o usuário, após definir o processamento, delega ao sistema a alocação de recursos e escalonamento de atividades. Assim, durante a execução, os recursos são compartilhados entre diferentes usuários, de modo que um escalonador central agenda as tarefas de todos usuários e geralmente utiliza uma política de melhor esforço (*best-effort*), a fim de otimizar o uso dos recursos do sistema [JD11b].

Infinidade de recursos sob demanda Os usuários podem requisitar uma quantidade arbitrária de recursos a qualquer momento, criando a ilusão de que a nuvem possua uma infinidade de recursos disponíveis [AFG+09]. Assim, tanto pequenas como grandes quantidades de recursos computacionais podem ser provisionados sob demanda para execução de workflows, sem a necessidade de aquisições ou planejamento prévio de infraestruturas computacionais.

Pagar pelo uso A forma como a nuvem disponibiliza recursos computacionais permite que sua utilização seja quantificada e, assim, precificada. Para execução de workflows científicos, pagar pelo uso é vantajoso a curto prazo, principalmente para execução de experimentos de laboratórios que não possuem muitos recursos computacionais.

Elasticidade A elasticidade refere-se ao poder de provisionamento dinâmico de recursos que a nuvem provê. Ela tem um papel fundamental neste trabalho, vamos discutir sobre ela em detalhes posteriormente.

Multilocação (*Multi-tenancy*) Os recursos da plataforma de computação em nuvem são multi-inquilinos (*multi-tentants*)), isto é, são compartilhados por diversos usuários que são logicamente isolados. Assim, cada usuário aparenta usar o ambiente isoladamente, sem tomar conhecimento do compartilhamento de recursos. No caso de execução de workflows, podemos considerar que recursos *multi-inquilinos* conseguem ser utilizados por diferentes instâncias de atividades ao mesmo tempo.

Heterogeneidade Os recursos disponibilizados pelas nuvens são heterogêneos, isto é, as VMs a serem disponibilizadas podem possuir uma variedade de configurações (número de núcleos de CPU, memória RAM, disco).

Neste trabalho, a multilocação será considerada no modelo de execução de workflows. Pois, isso permite que execuções de atividades adicionais sejam atribuídas a uma unidade de recurso que esteja sendo subutilizada.

2.2.1 Elasticidade e Provisionamento

Elasticidade é definida como "o grau em que um sistema consegue se adaptar a mudanças na carga de trabalho através do provisionamento e desprovisionamento de recursos de forma automática, de modo que a cada instante do tempo os recursos disponíveis correspondam à demanda

corrente o mais próximo possível" [HKR13]. Assim, a elasticidade presente em plataformas de computação em nuvem é o principal elemento que possibilita a existência do provisionamento dinâmico de recursos durante a execução de workflows.

É comum haver ambiguidade entre os termos: elasticidade, escalabilidade e eficiência. Além disso, outros termos relacionados à elasticidade e ao provisionamento de recursos podem causar confusão. Por isso, definiremos e/ou explicaremos os seguintes termos que são usados ao longo deste texto:

Processo de adaptação (ou adaptação dos recursos) Intervalo em que ocorre um provisionamento dinâmico de recursos.

Provisionamento de recursos (ou alocação de recursos) Neste texto, o provisionamento de recursos, em geral, refere-se tanto à ampliação como à redução de recursos computacionais feito pelos gerenciadores de recursos. Um provisionamento é dinâmico quando ele ocorre durante uma série de execução de workflows. Por outro lado um provisionamento estático ocorre em tempo de configuração do ambiente, antes do início de uma série de execução de workflows.

Dimensão do processo de adaptação Durante o provisionamento, a quantidade de um ou, possivelmente, mais tipos de recursos podem ser ampliados ou reduzidos. Cada tipo de recurso pode ser visto como uma dimensão do processo de adaptação. Caso um recurso seja composto por outros recursos, como é caso da atribuição de núcleos de CPUs e memória RAM a uma VM, podemos considerar dimensões em múltiplos níveis. Geralmente, os recursos são provisionados em unidades discretas como número de núcleos de CPU ou número de VMs.

Recursos disponíveis São os recursos computacionais que foram provisionados (ou alocados), isto é, são recursos que estão aptos a serem utilizados para o processamento de atividades.

Ampliação de recursos (*scale up*) Refere-se ao aumento da quantidade de um ou mais tipos de recursos computacionais. Nesse processo, o sistema passa de um estado de sobrecarga de recursos para um estado ótimo ou de subutilização.

Redução de recursos (*scale down*) Refere-se à diminuição da quantidade de um ou mais tipos de recursos computacionais. Nesse processo, o sistema passa de um estado de subutilização de recursos para um estado ótimo ou de sobrecarga.

Estado de sobrecarga (dos recursos) Estado do sistema em que os recursos disponíveis não atendem à carga atual de trabalho. É propenso a uma ampliação dos recursos disponíveis.

Estado de subutilização (dos recursos) Estado do sistema em que existem recursos excedentes para a carga atual de trabalho. É propenso a uma redução dos recursos disponíveis.

Estado ótimo (dos recursos) Estado em que o sistema possui a quantidade ideal para atender a carga atual de trabalho.

Escalabilidade Refere-se à capacidade de um sistema suportar o aumento da carga de trabalho através do uso de recursos computacionais adicionais. Mas a escalabilidade não considera a velocidade e granularidade da adaptação dos recursos, nem faz referência à precisão e o quão bem a demanda de recursos é atendida pelos recursos alocada.

Eficiência A eficiência refere-se ao rendimento de um sistema, isto é, à quantidade de recursos utilizados para processar uma dada quantidade de trabalho. Um sistema é mais eficiente que outro quando ele consegue processar uma mesma quantidade de trabalho com menos recursos que o outro. Normalmente, uma melhor elasticidade, mais precisa (mais próxima do estado ótimo), aumenta a eficiência de um sistema. Por outro lado, o contrário não é válido, o aumento da eficiência não implica em uma melhor elasticidade. A eficiência pode ser influenciada por outros fatores independentes da elasticidade, como por exemplo: implementações diferentes de uma mesma operação.

Também podemos citar os seguintes aspectos do processo de adaptação [HKR13]:

Velocidade A velocidade de ampliação de recursos de um processo de adaptação é definida pelo tempo que o sistema demora para passar de um estado de sobrecarga de recursos para um estado ótimo ou de subutilização. A velocidade de redução de recursos de um processo de adaptação é definida pelo tempo que o sistema demora para passar de um estado de subutilização de recursos para um estado de sobrecarga ou ótimo.

Precisão A precisão da ampliação/redução dos recursos de um processo de adaptação é definida como o desvio absoluto entre a quantidade de recursos alocados e a quantidade demandada pelo sistema após o processo de adaptação.

2.2.2 Aglomerados Virtuais e Sistemas de Arquivos

Aplicações de workflows geralmente necessitam de vários ciclos de computação para processar suas atividades. Em plataformas de computação em nuvem, esse poder computacional é providenciado pela disponibilização de VMs. O uso simultâneo de vários desses recursos computacionais é necessário para alcançar o desempenho necessário por workflows de grande escala [JD10]. Uma coleção de VMs, chamada de *aglomerado virtual* (em inglês *virtual cluster*) [FFK+06], pode ser configurada e coordenada através de gerenciadores de recursos como o PBS/TORQUE [pbs, tor] ou Condor [htc, CKR+07]. A configuração de aglomerados virtuais pode ser complexa e propensa a erros, por isso existem ferramentas para automatizar esse processo, como o Nimbus Context Broker [nim].

O armazenamento e a transferência de dados entre nós também precisam ser gerenciados. Isso pode ser feito por meio de um sistema de arquivos tradicional ou de um sistema de arquivos compartilhados. No caso de sistemas de arquivos compartilhados, em nuvens, são utilizados sistemas de armazenamento da própria nuvem, como o Amazon S3 [s3], ou podem ser implantados sistemas de arquivos distribuídos (NFS). Outra alternativa, quando o desempenho é um fator importante, seria a implantação de sistemas de arquivos paralelos como o PVFS [pvf] ou GlusterFS [glu]. Quando não há utilização de um sistema de arquivos compartilhados, o gerenciador de recursos (ou algum outro subsistema) coordena a transferência de dados entre nós.

2.3 Escalonamento de Workflows Científicos

A representação de um workflow descreve a precedência das atividades, além da dependência e movimentação dos dados necessários para sua execução, contudo não descreve quais nem como os

recursos computacionais devem ser utilizados. Essa representação abstrata precisa ser convertida para um modelo executável, no qual exista uma atribuição dos recursos computacionais às atividades a serem executadas. No contexto deste trabalho, o *escalonamento* refere-se a esse processo que satisfaz as restrições impostas pelo modelo abstrato do workflow científico e mapeia suas atividades nos recursos computacionais disponíveis visando a satisfazer algum critério específico.

Existem diferentes critérios (ou funções objetivo) estudados em problemas de escalonamento. Em relação à computação de alto desempenho, o critério de otimização mais notório é o *makespan*, que indica o tempo decorrido entre o início da execução do workflow e o término da última atividade executada. Quando usamos o *makespan* como função objetivo do escalonamento, queremos reduzir o tempo total gasto pela execução do workflow. Mas, por exemplo, um outro critério interessante neste trabalho seria o custo gasto na locação de recursos de uma dada nuvem.

Problemas de escalonamento são problemas de otimização combinatória. Dado um conjunto de de recursos computacionais (α) e um conjunto de tarefas (as atividades do workflow) e suas restrições (definidas pelo modelo do workflow)(β), deseja-se encontrar uma atribuição, no tempo (que satisfaça a precedência modelada), das atividades aos recursos de tal forma que, algum critério de otimização (γ) seja otimizado. Essa é a notação $\alpha|\beta|\gamma$ introduzida por *Graham et al.* [GLLK79|. Em workflows científicos, o tempo de execução de atividades geralmente não é conhecido.

2.3.1 Escalonamento Dinâmico

O escalonamento dinâmico ocorre quando um escalonamento em tempo de execução é efetuado devido a alguma alteração nas condições do escalonamento. Não ocorre em workflows mas, por exemplo, quando a precedência das atividades não é conhecida de antemão, nesse caso um escalonamento pode ocorrer sempre que um dado de precedência for obtido. Neste trabalho, o escalonamento dinâmico deverá reagir a alterações nas instâncias de workflows em execução e a variação da utilização dos recursos computacionais disponíveis.

Além da variação na utilização de recursos ao longo da execução de uma instância de workflow, novas instâncias podem surgir ou instâncias, já em execução, podem ser finalizadas. Por isso, a carga de trabalho do sistema estará em constante mudança junto à demanda por recursos computacionais. Ao longo da execução, o provisionamento de recursos computacionais deve acompanhar essas mudanças e reagir alterando o volume de recursos disponíveis, por outro lado o escalonamento dinâmico deve ser capaz de mapear as atividades de todas as instâncias nos recursos que estiverem disponíveis de forma a otimizar algum critério.

Capítulo 3

Trabalhos Relacionados

O atual *dilúvio de dados* (data deluge) [HT03, BHS09] vem comprovando o surgimento de um novo paradigma de como se fazer ciência introduzido por Jim Gray [HTT09], contexto que popularizou a utilização de workflows científicos [CBGK13]. O artigo de *Ludäscher et al.* [LWMB09] apresenta o aumento do interesse e da pesquisa sobre os workflows científicos nos últimos anos, também evidencia as diferenças entre workflows científicos e workflows de negócio.

O uso das nuvens pelos SGWCs como plataformas de computação também é recente. No artigo de *Hoffa et al.* [HMF+08], há uma comparação entre ambientes locais e ambiente remotos virtuais. A comparação mostra que plataformas de computação remotas e virtuais, como a nuvem, são candidatas para execução de workflows científicos por serem plataformas escaláveis, contudo sofrem atrasos devido à comunicação e ao escalonamento remoto. *Juve et al.* [JDV+09] comparam, em termos de desempenho e custo, um ambiente de computação em nuvem com um ambiente tradicional de computação de alto desempenho; a comparação mostra que o desempenho da nuvem varia de acordo com a quantidade de recursos utilizados, e também mostra que a nuvem é um ambiente favorável a execução de workflows científicos. *Deelman et al.* [DSL+08] avalia o custo do uso de nuvens comerciais para execução de workflows; a análise feita varia, ao longo dos testes, tanto o tamanho e complexidade do workflow como a quantidade de recursos provisionados. Concluem que a plataforma de computação em nuvem proporciona quantidades significativas de computação e armazenamento sob demanda, sendo uma nova opção para aplicações científicas; também concluem que o custo de armazenamento é insignificante em comparação ao de processamento, assim, as nuvens aparentemente oferecem uma solução eficaz em relação ao custo para aplicações intensivas em dados.

Rahman et al. [RHRB13] apresentam uma solução para o problema de escalonamento de workflows em ambientes com dinamicidade de recursos. A solução proposta é um algoritmo de escalonamento dinâmico baseado em heurísticas que visam determinar o caminho de execução mais longo (ou caminho crítico) de workflows modelados como DAGs. O desempenho da solução é comparado ao de estratégias existentes de escalonamento através de simulações, mostrando a efetividade da solução proposta quando há variação nos recursos. O foco é voltado a grades computacionais com variação de recursos, apesar disso, há um esboço de solução voltada a nuvens no final do trabalho. Contudo o trabalho visa resolver o problema de variações passivas (disponibilidade) de recursos, sem utilizar a característica de elasticidade da nuvem para alocar ou liberar recursos durante a execução de workflows.

Wang et al. [WDL+13] apresentam um projeto inovador de SGWC para execução em nuvens,

em que o provisionamento de recursos e escalonamento é feito de forma iterativa visando aprimorar o desempenho e custo de execuções de workflows com ciclos. A técnica de *máquinas de vetores de suporte* (*SVM*, pertencente às áreas de *aprendizagem de máquina* e *mineração de dados*) é utilizada em um modelo baseado em previsão, onde o tempo de execução das tarefas é previsto em treinamentos e atualizados iterativamente durante a execução de workflows. O sistema provisiona recursos com base nos tempos estimados de execução das atividades visando minimizar o total custo gasto durante a execução. Um novo algoritmo de escalonamento também é apresentado para aproveitar as características desse provisionamento. O modelo é testado experimentalmente comparando algoritmos de escalonamento e seus desempenhos, porém não há comparação de desempenho com outros SGWCs. Além disso, o foco do trabalho apresentado são workflows com ciclos (iterativos) sem considerar instâncias concorrentes de execução.

Deelman et al. [DJMN13] desenvolvem a ideia de se trabalhar com *ciência hospedada* (*hosted science*) com foco em workflows científicos. Ciência hospeda faz referência à hospedagem de experimentos em recursos remotos como a plataforma de computação em nuvem, e ao uso desses ambientes para pesquisa. Problemas existentes nesse ecossistema são levantados, principalmente com relação a *workflow ensembles*. Em um trabalho anterior dos mesmos autores [MJDN12], três algoritmos são propostos para resolver o problema de escalonar *workflow ensembles* em um ambiente de computação em nuvem com restrições de custo e duração, além de prioridades (de precedência) entre os workflows. Um dos algoritmos de escalonamento apresentados é estático, SPSS (*Static Provisioning Static Scheduling*), enquanto os outros dois são dinâmicos: DPDS (*Dynamic Provisioning Dynamic Scheduling*) e WA-DPDS (*Workflow Aware DPDS*). Não há comparações das soluções com escalonamentos reais feitos por SGWCs, todavia as soluções são testadas e comparadas entre si através de simulações utilizando um simulador desenvolvido e baseado no CloudSim [CRB+11]. Os resultados mostram que os algoritmos dinâmicos produzem resultados melhores por adaptarem-se a mudanças de condições do ambiente como, por exemplo, atrasos no provisionamento de recursos. Além disso, no modelo considera-se que as atividades do workflow têm acesso exclusivo às VMs (as VMs não são multi-inquilino) sem interrupção.

Azarnoosh et al. [ARJ+13] apresentam PRECIP (*Pegasus Repeatable Experiments for the Cloud in Python*), uma ferramenta para gerenciamento de experimentos em nuvens por meio da administração de comunicações e recursos. Através dele é possível implantar e reproduzir experimentos em diferentes ambientes de nuvens. As bibliotecas Boto [bot] e Paramiko [par] são utilizadas para comunicação com nuvens e gerenciamento de conexões SSH com VMs. Boto fornece uma API integrada para os requisição de serviços em diversos tipos de interfaces de gerenciadores de nuvens (Amazon EC2, Eucalyptus, OpenStack e Open Nebula). Paramiko fornece conexões seguras (encriptadas e autenticadas) à VMs através do protocolo SSH2. Apesar do PRECIP possuir apoio a recuperação de falhas, essa ferramenta não toma decisões de forma autônoma sobre o provisionamento de recursos; ela é apenas uma API para configuração e execução de experimentos. Um outro projeto anterior, Wrangler de *Juve* e *Deelman* [JD11c, JD11a], serve para automatizar a configuração e o provisionamento de recursos em nuvens. Contudo o Wrangler necessita que as aplicações, a serem implantadas, sejam descritas estaticamente em arquivos XML e não permite que mudanças sejam feitas facilmente na configuração da nuvem como, por exemplo, a instanciação ou remoção de VMs.

Yazir et al. [YMF+10] tratam o problema do gerenciamento dinâmico e autônomo de recursos físicos da nuvem, isto é, o problema da utilização inteligente de recursos físicos do lado do provedor

de nuvem. Em *data center*, a atribuição de VMs à recursos físicos deve respeitar *acordos de nível de serviço* (SLA) e, ao mesmo tempo, minimizar a subutilização dos recursos físicos. Na solução proposta, a distribuição de recursos físicos entre VMs é iterativa e controlada através de migrações (transferências de VMs entre recursos físicos). Uma arquitetura descentralizada é utilizada, na qual agentes associados a cada nó físico monitoram tanto o estado do nó como o estado das VMs alocadas nele. O monitoramento permite identificar quebras de SLA e problemas de desempenho, assim, migrações de VMs podem ser efetuadas para atender a esses problemas. Para encontrar recursos aptos à migração de VMs, os agentes consultam um oráculo que mantém a lista de todos os recursos disponíveis da infraestrutura. Diferentes critérios sobre o estado dos recursos físicos e da variação da utilização de recursos de uma determinada VM são utilizados para seleção do recurso físico adequado para alocação da VM. A avaliação dos resultado é feita em um simulador desenvolvido na linguagem C. As conclusões mostram que a abordagem feita é promissora em relação à escalabilidade, viabilidade e flexibilidade por ser uma solução distribuída e baseada em critérios que podem ser alterados. O trabalho trata somente do lado do gerenciamento e configuração das VMs em relação aos recursos físicos, supõe que a demanda computacional das VM é conhecida e previsível. A visão do lado provedor da infraestrutura é discutida, diferente deste trabalho em que queremos aprimorar o lado do usuário e sua utilização da nuvem.

O *glideinWMS* [gli, Sfi08] é uma ferramenta que funciona em sobre o gerenciador de recursos *HTCondor*. Esta ferramenta monitora a fila de atividades prontas e em espera para serem executadas, e, à partir de atividades chamadas de *glideins*, consegue acoplar recursos remotos ao conjunto de recursos já existentes. Quando existem mais atividades prontas para serem executadas do que recursos disponíveis, recursos extras podem ser alocados de forma automatizada, aumentando a vazão da execução de atividades. Quando os recursos extras ficam livres e inativos, eles podem ser liberados. Essa solução foi feita originalmente para que grades computacionais conseguissem expandir seus recursos com recursos remotos e aumentar o *throughput* de tarefas executadas como acontece na grades *Science Grid* (*OSG*) e *European Grid Infrastructure* (*EGI*). Já existem adaptações para que recursos sejam provisionados de plataformas de computação em nuvem [MTH+14]. contudo essa solução ainda visa maximizar o número de tarefas em execução, sem se preocupar com questões como custo monetário do provisionamento, e a utilização de informações das atividades para prever e melhorar escalonamentos futuros.

3.1 Conclusão do Capítulo

Os trabalhos citados neste capítulo, têm relação com o provisionamento dinâmico de recursos em execuções de workflows científicos em nuvens. O glideinWMS é uma solução válida e próxima ao objetivo deste trabalho. Contudo, essa solução, apesar de dinâmica, não utiliza informações provenientes do workflow para gerar uma visão futura da execução para tomadas de decisão que podem influenciar o desempenho da execução. A solução do glideinWMS toma decisões reativas com base no estado atual da execução. De qualquer modo, os trabalho citados mostram o interesse emergente em execuções de workflows científicos em plataformas de computação em nuvem, e são fonte de inspiração para estratégias e métodos utilizados em nossa abordagem.

Capítulo 4

Contexto Experimental

Dois experimentos são descritos neste trabalho. Apesar de terem objetivos e resultados distintos, ambos experimentos utilizam de partes comuns em sua ambientação. Este capítulo descreve esse contexto que é presente nos experimentos.

4.1 Pegasus

O SGWC *Pegasus* [peg, DSS+05] foi utilizado como caso de estudo e ambiente de execução para ambos experimentos. O Pegasus faz a ponte de ligação entre o domínio científico e ambiente experimental ao mapear workflows abstratos, descritos em alto nível, em recursos distribuídos. Ele gerencia os dados pelo usuário, inferindo automaticamente transferências de arquivos necessários, além de capturar informações de desempenho.

Os workflows, geralmente representados como DAGs (Seção 2.1.2), são armazenados em arquivos *DAX* (*DAG in XML*). Os arquivos DAX são representações dos workflows em alto nível no formato *XML*. O DAX descreve todas as atividades: suas dependências, entradas necessárias, saídas esperadas e argumentos de chamada. Eles são os arquivos principais de entrada para execução de workflows no Pegasus.

No DAX não há informações sobre recursos computacionais e nem sobre os executáveis utilizados pelas atividades. Contudo essas informações ainda são necessárias para execução de um workflow. Essas informações são descritas em outros arquivos que também são entradas do Pegasus. O *Replica Catalog* mantém um mapa da localização dos arquivos utilizados pelo workflow. O *Site Catalog* descreve os recursos computacionais disponíveis para execução do workflow. O *Transformation Catalog* guarda a localização dos executáveis utilizados pelas atividades do workflow.

A Figura 4.1 mostra uma visão geral da arquitetura do Pegasus. O *Workflow Mapper* é responsável por planejar a execução do workflow. O Pegasus transforma o workflow abstrato em um workflow executável, determinando os executáveis, dados e recursos necessários para sua execução. A execução do workflow é gerenciada pelo *DAGMan* [Fre02], que gerencia os dados, monitora a execução e cuida do tratamento de falhas. A execução individual de tarefas é feita pelo gerenciador de recursos *HTCondor* [htc, CKR+07] que supervisiona a execução em recursos locais e remotos. O HTCondor funciona em diferentes ambientes de execução. Neste trabalho, utilizamos a plataforma de computação em nuvem contudo sem um sistema compartilhado de arquivos.

O Pegasus foi escolhido como plataforma de execuções dos experimentos deste trabalho por ser um SGWCs de código aberto, que possui uma comunidade ativa e por suas várias contribuições na

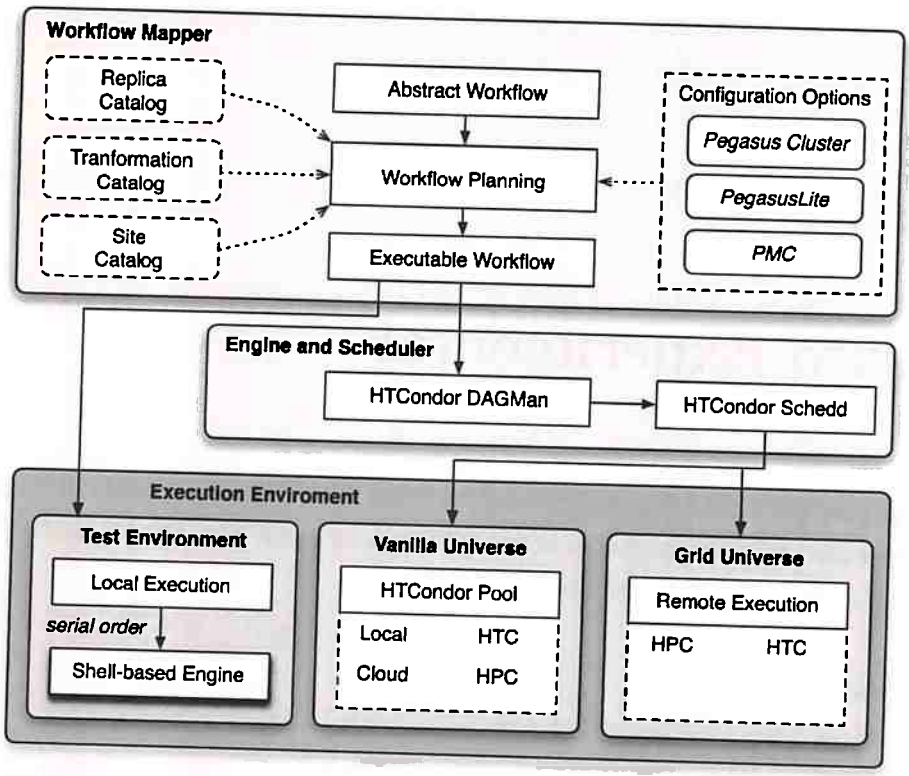

Figura 4.1: *Visão geral da arquitetura do Pegasus.*

área de workflows científicos.

4.2 Aplicações Científicas

Existe um conjunto de aplicações de workflows científicos [BCD+08, JCD+13] que é amplamente utilizado por pesquisadores como uma espécie de *benchmark* para a avaliação e comparação de técnicas e ferramentas criadas para esse domínio. Nesse conjunto, foram escolhidas o *Montage* [mon] e o *Epigenomics* [epi] como caso de estudo para execução dos experimentos.

As Figuras 4.2 e 4.3 são imagens que expõem a estrutura de exemplos sintéticos dessas aplicações de workflows. Esses exemplos foram criadas a partir de um gerador de workflows [gen] desenvolvido no *USC Information Sciences Institute*. O gerador utiliza informações obtidas de execuções reais bem como informações conhecidas para gerar workflows sintéticos que se assemelham aos utilizados em aplicações reais.

O Montage é uma ferramenta de código aberto desenvolvida pelo IRSA (*NASA/IPAC Infrared Science Archieve*) para agregar imagens em mosaicos personalizados. É aplicado na área de astronomia para criar composições do céu a partir de dados coletados por telescópios. Utiliza imagens no formato FITS (*Flexible Image Transport System*), o formato padrão usado em astronomia. As imagens de entrada são todas reprojetadas para uma mesma escala espacial e rotação, além de serem uniformizadas e corrigidas.

O Epigenomics é uma aplicação criada pelo *Epigenome Center* da *USC* University Southern California) e pela equipe de desenvolvimento do Pegasus. É utilizada para automatizar várias operações em processamento de sequências de genoma. O workflow processa vários conjuntos de sequências

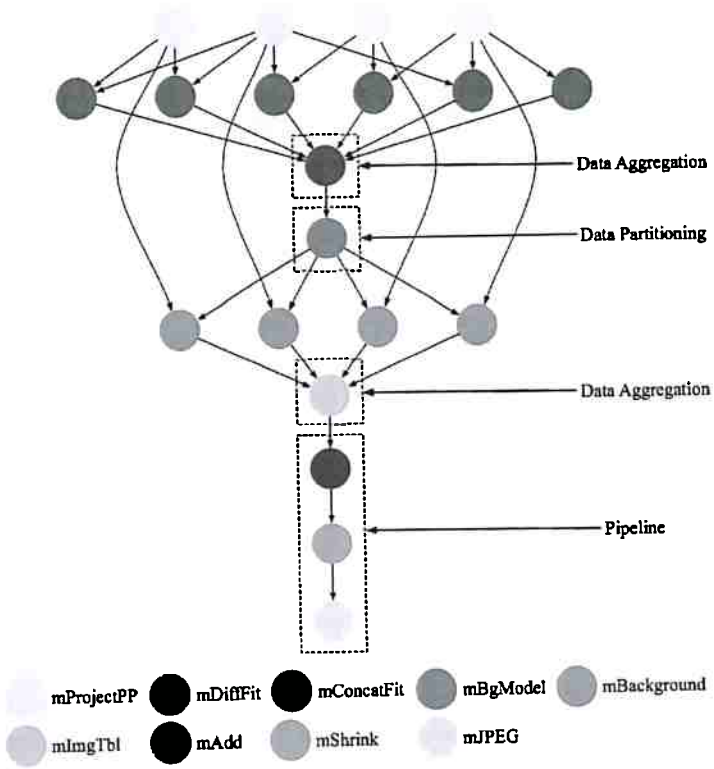

Figura 4.2: *Exemplo sintético do workflow Montage apresentado em Bharathi et al. [BCD+08]. Os vértices do grafo representam atividades, enquanto as arestas expressam os fluxos de dados entre as atividades e, consequentemente, a precedência entre elas. Retirado de https:// confluence.pegasus.isi.edu/ display/ pegasus/ WorkflowGenerator em 08/08/2016.*

Aplicação	Disco	Memória	CPU
Montage	alto	baixo	baixo
Epigenomics	baixo	médio	alto

Tabela 4.1: *Comparação da utilização de recursos de aplicações de workflows, apresentado por* Juve et al. *[BDJ+13, JDV+09].*

de genoma em paralelo. Essas sequências são subdivididas em subconjuntos, que posteriormente são filtradas para remoção de ruído, reformatadas e mapeadas para um genoma de referência. Os genomas mapeados são finalmente unidos e indexados para uma análise posterior.

É interessante notar que a utilização de recursos computacionais é muito diferente entre essas duas aplicações [BDJ+13, JDV+09]. A ferramenta Montage é considerada uma aplicação de uso intensivo de disco por passar 95% do tempo esperando por operações de entrada e saída. O desempenho do workflow Epigenomics é considerado limitado pela CPU, pois utiliza a CPU em 99% do seu tempo de execução; e o 1% restante em entrada e saída e outras operações. A Tabela 4.1 mostra uma comparação do uso de recursos de cada workflow.

Durante os experimentos, execuções de cada workflow foram executadas com diferentes conjuntos de entrada ou diferentes parâmetros de entrada. No caso do Montage, um único conjunto foi utilizado, mas diferentes parâmetros de gradação (0.1, 0.5, 1.0, 2.0, e 4.0 graus) foram utilizados. O parâmetro de gradação influi no tamanho do workflow gerado em termos de número de atividades. A tabela 4.2 mostra o número de atividades por tipo de atividade do workflow Montage. Dois conjuntos de entradas distintos foram utilizados para execução do workflow Epigenomics, os quais

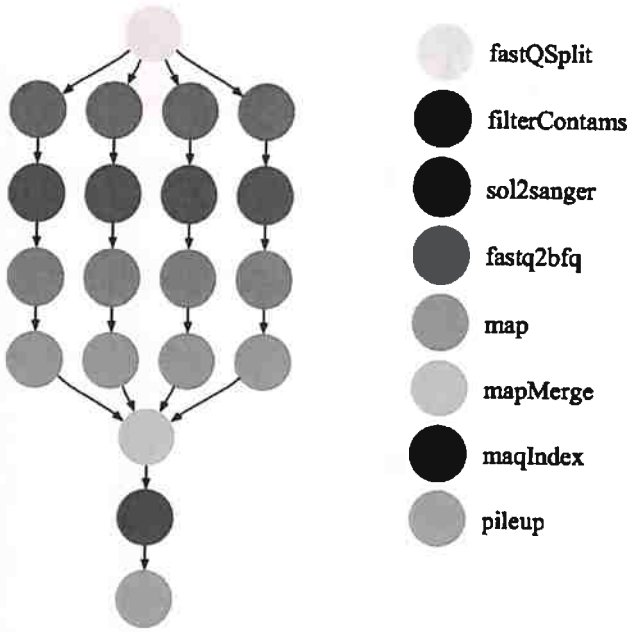

Figura 4.3: *Exemplo sintético do workflow Epigenomics apresentado em Bharathi et al. [BCD+08]. Os vértices do grafo representam atividades, enquanto as arestas expressam os fluxos de dados entre as atividades e, consequentemente, a precedência entre elas. Retirado de https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator em 08/08/2016.*

têm o número de atividades descritos na Tabela 4.3.

Atividade	0.1	0.5	1	2.0	4.0
mAdd	1	1	1	1	5
mBackground	8	32	84	300	603
mBgModel	1	1	1	1	1
mConcatFit	1	1	1	1	1
mDiffFit	13	73	213	836	2316
mImgtbl	1	1	1	1	5
mJPEG	1	1	1	1	1
mProjectPP	8	32	84	300	802
mShrink	1	1	1	1	4

Tabela 4.2: *Número de atividades por tipo de atividade no workflow Montage para diferentes valores do parâmetro de gradação.*

Atividade	TAQ	HEP
chr21	1	1
fast2bfq	729	1360
fastqSplit	2	7
filterContams	729	1360
map	729	1360
mapMerge	3	8
pileup	1	1
sol2sanger	729	1360

Tabela 4.3: *Número de atividades por tipo de atividades do workflow Epigenomics para conjuntos de dados TAQ e HEP.*

A Tabela 4.4 evidencia o tamanho dos arquivos de entrada de cada workflow. Essa caracterização do perfil de execução dos workflows científicos é possível utilizando a ferramenta *Kicks-*

tart [VMZ+06, JTFdS+15]. O Kickstart monitora e armazena informações sobre a execução individual das atividades do workflow. Captura dados sobre a entrada e saída, tempo de execução, uso memória, e uso de CPU. Essas informações possibilitam a análise dos resultados experimentais.

Workflow	Caso	Tamanho de Entrada
Montage	0.1	12.41 MB
Montage	0.5	49.79 MB
Montage	1.0	130.43 MB
Montage	2.0	489.75 MB
Montage	4.0	1.81 GB
Epigenomics	TAQ	1.01 GB
Epigenomics	HEP	1.79 GB

Tabela 4.4: *Tamanho total dos arquivos de entrada em cada workflow*

4.3 Ambientes de Execução

Ambos experimentos foram executados em plataformas reais de computação em nuvem. A plataforma GCloud da Google [goo] e a Azure da Microsoft [azuc] foram os ambientes de execução dos experimentos. Ambas foram utilizadas como infraestrutura (2.2), onde uma aglomerado virtual foi configurado para execução dos experimentos. A implantação do conjunto de máquinas virtuais e gerenciamento dos experimentos foi feito com a ferramenta *PRECIP* [prea]. Um gerenciador de experimentos que automatiza o provisionamento e implantação inicial das máquinas e controla os experimentos a partir roteiros de execução. Foi implementada uma extensão [preb] da ferramenta PRECIP para utilizar as plataformas de computação em nuvem GCloud e a Azure. Isso possibilitou a automação da execução dos experimentos deste trabalho nessas plataformas comerciais de computação em nuvem. As especificidades da configuração de cada ambiente está descrito nos Capítulos 5 e 6.

Capítulo 5

Compartilhamento de Recursos

A execução de workflows científicos pode ser custosa em termos de recursos computacionais. A otimização da utilização e eficiência dos recursos é algo desejável, mesmo em ambientes onde os recursos computacionais são abundantes como a plataforma de computação em nuvem. O experimento descrito neste capítulo estuda o compartilhamento do processamento dentro de uma mesma máquina virtual para execução das atividades de um workflow científico.

5.1 Motivação

Os SGWCs assumem que suas atividades executam isoladamente. Cada atividade possui um pedaço de recurso separado para sua execução. No caso de atividades que possuem uma única linha de execução (*single threaded*), cada atividade possui um núcleo de processamento isolado para sua execução. Mesmo em trabalhos relacionados à escalonamento de atividades [HMF+08, JDV+09, DSL+08] essa suposição é uma suposição comum.

Em plataformas de computação em nuvem, a quantidade de recursos disponíveis pode ser personalizada. Em uma situação ideal, o aumento do número de recursos deveria aumentar o desempenho da execução dos workflows científicos. Caso a execução de uma atividade esteja sobrecarregando um recurso, idealmente, a alocação de mais recursos poderia auxiliar sua execução. Contudo uma atividade não pode se beneficiar de mais processadores do que o número de linhas de execução que possui. Apesar dessa limitação, o compartilhamento de recursos permite um aumento na eficiência na utilização de recursos.

O experimento descrito neste capítulo estuda o compartilhamento do processamento entre atividades em ambientes de computação em nuvem para aumentar a eficiência da utilização de recursos durante a execução de um workflow científico.

5.2 Ambiente Experimental

Nesta seção, listamos características do ambiente onde foram executados os experimentos, como a infraestrutura, versões de software e detalhes específicos de configuração.

5.2.1 Condições Experimentais

Os workflows deste experimento foram executados na plataforma de computação em nuvem *GCloud* da Google, mais especificamente na *Google Compute Engine* (*GCE*). A GCE é o componente de IaaS fornecido pela GCloud. Para cada execução, implantamos um conjunto de máquinas de virtuais já configuradas com o HTCondor através do PRECIP (descrito na Seção 4.3).

Neste experimento, foram utilizadas máquinas virtuais do tipo *n1-standard-1*. Em uma máquina da série *n1*, uma CPU virtual é implementada como uma única *hardware hyper-thread* em um processador 2.6GHz Intel Xeon E5 (Sandy Bridge), 2.5GHZ Intel Xeon E5 v2 (Ivy Bridge) ou em um 2.3GHz Intel Xeon E5 v3 (Haswell). Uma máquina *n1-standard-1* é composta de um núcleo virtual de 2.75 *GCEUs* (*Google Compute Engine units*) e 3.75GB de memória.

A imagem utilizada para instanciar as máquinas virtuais possui o sistema operacional Ubuntu 15.04, utiliza o Pegasus 4.5.0 como SGWC e o HTCondor 8.3.5 como gerenciador de recursos. Para cada execução foram usadas 9 instâncias da máquina *n1-standard-1*, onde uma das máquinas era utilizada como nó principal (de submissão), e as demais como nós de execução.

5.2.2 Gerenciamento de Recursos

Dentro do Pegasus (4.1), o HTCondor gerencia os recursos disponíveis através de partições de execução, nas quais cada atividade pode ser escalonada para ser executada. Geralmente, o HTCondor cria uma única partição por núcleo de processamento. Por exemplo, em uma infraestrutura composta por oito nós de execução com um processador cada, o gerenciador de recursos anunciará oito partições de execução quando a configuração padrão é utilizada. Para este experimento, o HTCondor foi configurado para criar múltiplas partições por núcleo de processamento, *i.e.* atividades podem compartilhar a mesma CPU quando executadas.

As configurações descritas em 5.1 mostram as configurações que devem ser alteradas para que o particionamento funcione. `NUM_CPUS` delimita quantas partições serão criadas e `DETECTED_CPUS` são número de CPUs encontradas na máquina em questão, o N deve ser alterado para um inteiro indicando a multiplicidade do particionamento do experimento. Como utilizamos Ubuntu 15.04 foi necessário criar *CGroups* para garantir o isolamento dos recursos dentre as partições e compartilhamento dos processadores alocados para execução de atividades distintas. A comando *bash* 5.2 descreve como criar os CGroups com o nome de `htcondor` para o usuário `condor` no Ubuntu 15.04.

```
NUM_CPUS = N * $(DETECTED_CPUS)
BASE_CGROUP = htcondor
```

Código 5.1: *Configuração do HTCondor, no caso N deve ser alterado para o número de partições desejáveis.*

```
sudo cgcreate -a condor:condor -t condor:condor -g cpu,cpuacct,memory,freezer,
    blkio:htcondor
```

Código 5.2: *Configuração do CGroups*

A divisão de memória e disco disponível entre cada partição também podem ser configuradas, contudo a análise do impacto da necessidade de disco e memória está fora do escopo deste experi-

mento. A configuração padrão de divisão ambos recursos igualmente foi utilizada, sendo que ambos recursos, mesmo particionados, são suficientes para execução das tarefas.

5.2.3 Execução

Neste experimento foram feitas 18 repetições de execução para cada aplicação. Na execução do workflow Epigenomics ambos conjuntos de entrada, HEP e TAQ, foram utilizados. Durante a execução do Montage os parâmetros de invocação: 0.1, 0.5, 1.0, 2.0 e 4.0 graus foram utilizados. O particionamento variou nas seguintes quantidades: 1, 2, 4 e 8 partições. As execuções de 1 partição foram utilizadas como base de comparação, pois representam a situação onde não compartilhamento de recursos.

5.3 Resultados

Nesta seção, é avaliado o impacto o particionamento de núcleos de processamento sobre o desempenho geral da execução de workflows. Também são investigados os ganhos de desempenho individual das atividades. Os resultados do experimento para partição 1 (*i.e.*, sem compartilhamento de recursos) foram utilizados como base de comparação.

A Figura 5.1 mostra o *makespan* das execuções. O *makespan* representa o tempo total gasto durante a execução de todas atividades do workflow, incluindo as atividades de computação e as atividades extras de gerenciamento de dados inferidas pelo Pegasus (preparação dos arquivos, limpeza das atividades, gerenciamento de diretório). Para o workflow Montage, o ganho/perda é desprezível para as pequenas instâncias (0.1, 0.5 e 1.0 graus) devido ao pequeno número de atividades, e consequentemente baixo grau de paralelismo (veja Tabela 4.2 e Figura 4.2), à duração curta das atividades (veja Figuras 5.2 e 5.3), e ao *overhead* inerente do SGWCs, *e.g.* o tempo para liberar o próxima atividade [CD11]. Para as instâncias de grau maior (2.0 e 4.0), o desempenho melhorou até 9% como é visto na Tabela 5.1. As Figuras 5.2, 5.3 e 5.4 mostram o tempo médio da execução das atividades por tipo de atividade e grau da instância do workflow Montage. Apesar do particionamento causar pequenos impactos no tempo individual de execução das atividades, o paralelismo supera essas perdas.

Para o workflow Epigenomics, um ganho geral de até 6% é observado para partições de tamanho 2. A atividade `map` é predominante durante a execução do workflow Epigenomics, como é visível na Figura 5.5. Essa atividade processa uma quantidade significativa de dados e utiliza intensivamente a CPU da máquina que a hospeda [JCD+13, FdSJD+13]. Tamanhos grandes de particionamento têm um impacto significativo sobre o tempo de execução da atividade `map` e consequentemente diminui o tempo de execução do workflow.

A demora maior nas execuções individuais de cada atividade é causada principalmente pela menor utilização de CPU. Para cada caso do experimento, foi medido a utilização média de CPU $\bar{F}$ para cada tipo de atividade T, definida a seguir:

$$\bar{F}(T) = \frac{1}{|\mathcal{T}|} \times \sum_{t \in \mathcal{T}} \left(\frac{\text{CPU time}_t}{\text{runtime}_t} \right) \tag{5.1}$$

Onde $\mathcal{T}$ é o conjunto de todas as atividades do tipo T dentro de uma instância de workflow. Quanto maior o valor de $\bar{f}$, maior será o valor da utilização de CPU (no máximo 1, que representa

100% de utilização). As Figuras 5.6 e 5.7 mostram Os valores médios da utilização de CPU nos workflows Montage e Epigenomics. Note que para o workflow do Montage, são somente exibidos a utilização para 2.0 e 4.0 graus pois são os casos onde há um ganho notável de desempenho. No geral, o workflow Epigenomics é intensivo em CPU enquanto o workflow Montage é intensivo em disco [JCD+13].

Os resultados obtidos sugerem que a otimização pelo compartilhamento de recursos se limitam à um pequeno número de partições como mostra a Figura 5.5. Atividades intensivas em processamento como a atividade `map` apresentam uma perda exponencial na utilização de CPU conforme o número de partições aumenta, que afeta significante o tempo de execução, e portanto aumenta o *makespan* de seus workflows. Por outro lado, perdas lineares na utilização de CPU, como ocorrem nas atividades `mBackground`, `mDiffFit`, e `mProjectPP` presentes no workflow Montage; ao custo da perda individual no tempo de execução, promovem ganhos no desempenho geral.

A Tabela 5.1 resume os resultados do experimento para avaliar o impacto do compartilhamento do processamento, através de partições, ao rodar workflows científicos em uma plataforma comercial de computação em nuvem. A tabela destaca a duração média dos workflows, e dos ganhos (+) ou perdas (−) de desempenho para diferentes tamanhos de partições.

Workflow	Caso	Base		2 Partições		4 Partições		8 Partições	
		Makespan	Ganho	Makespan	Ganho	Makespan	Ganho	Makespan	Ganho
Montage	0.1	334.3	0%	336.5	−1%	336.9	−1%	343.1	−3%
	0.5	511.6	0%	550.8	−7%	573.4	−11%	526.2	−3%
	1.0	892.4	0%	893.6	0%	870.6	+3%	877.5	+2%
	2.0	2825.3	0%	2627.9	+8%	2589.4	+9%	2627.8	+8%
	4.0	6730.0	0%	6420.0	+5%	6443.3	+4%	6390.0	+5%
Epigenomics	TAQ	6310.0	0%	6010.0	+5%	6416.0	−2%	7356.7	−14%
	HEP	11493.3	0%	10840.0	+6%	11683.3	−2%	13020.0	−12%

Tabela 5.1: *Resumo dos ganhos e perdas de desempenho para diferentes tamanhos de partição. A base de comparação é definida pelas execução em uma única partição, i.e. sem compartilhamento de recursos.*

5.4 Conclusão do Capítulo

Este experimento estudou e avaliou o impacto do compartilhamento de recursos durante a execução de workflows científicos em plataformas de computação em nuvem. No geral, o compartilhamento de núcleos de processamento dentro de uma máquina virtual pode melhorar a execução de workflows científicos, contudo devem ser usados com moderação. Os resultados indicam que workflows maiores se beneficiam mais do compartilhamento de recursos.

Como esperado, atividades de processamento intensivo tendem a sofrer degradações de desempenho. Logo, mesmo que o tempo de execução de workflows possam ser melhorados com esse método, idealmente, o uso do compartilhamento de recursos deve ser estudado caso a caso para cada atividade.

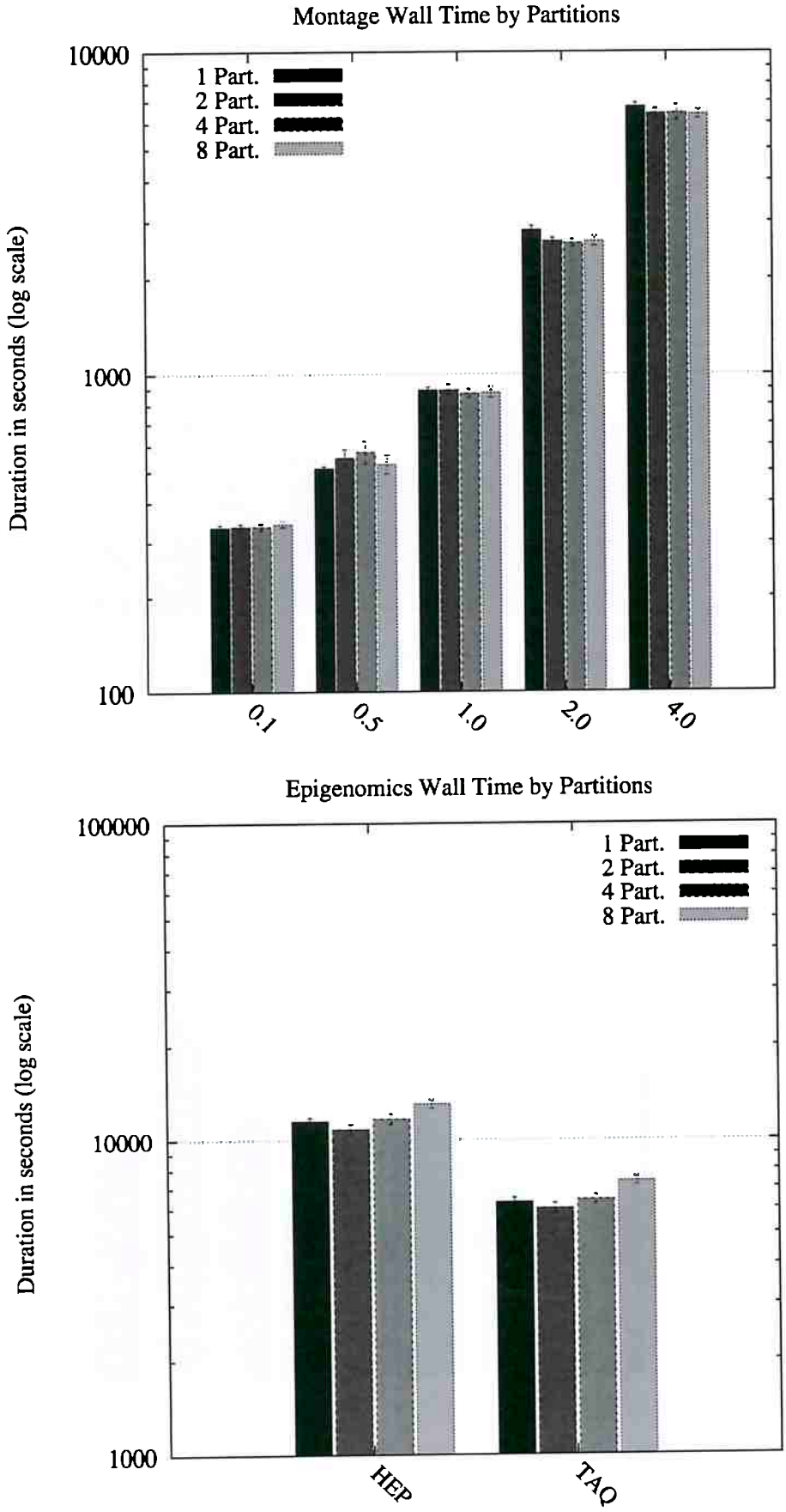

Figura 5.1: *Avaliação do makespan em diferentes particionamentos.*

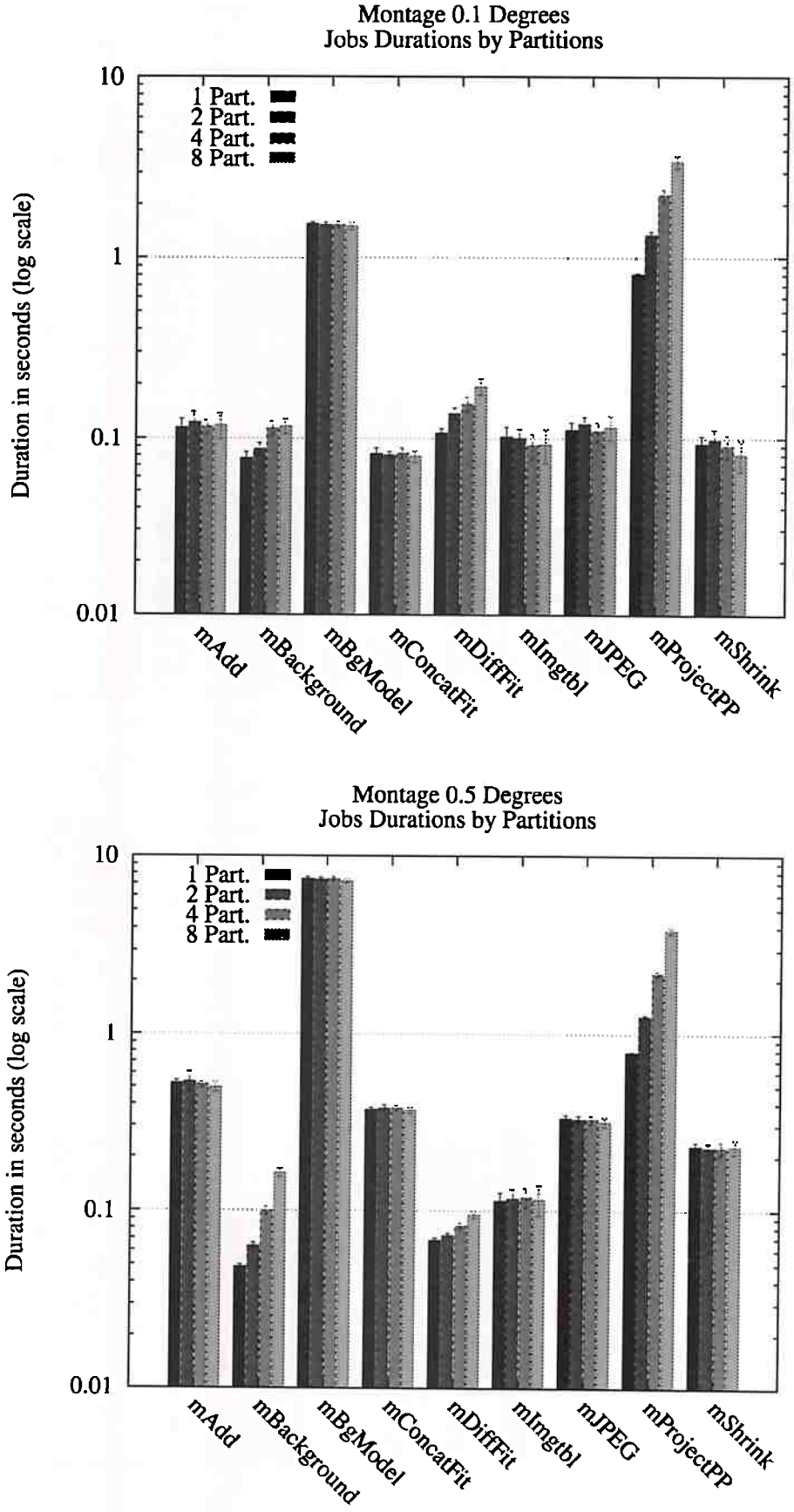

Figura 5.2: *Tempo médio de execução por tipo atividade do workflow Montage de graus 0.1 e 0.5.*

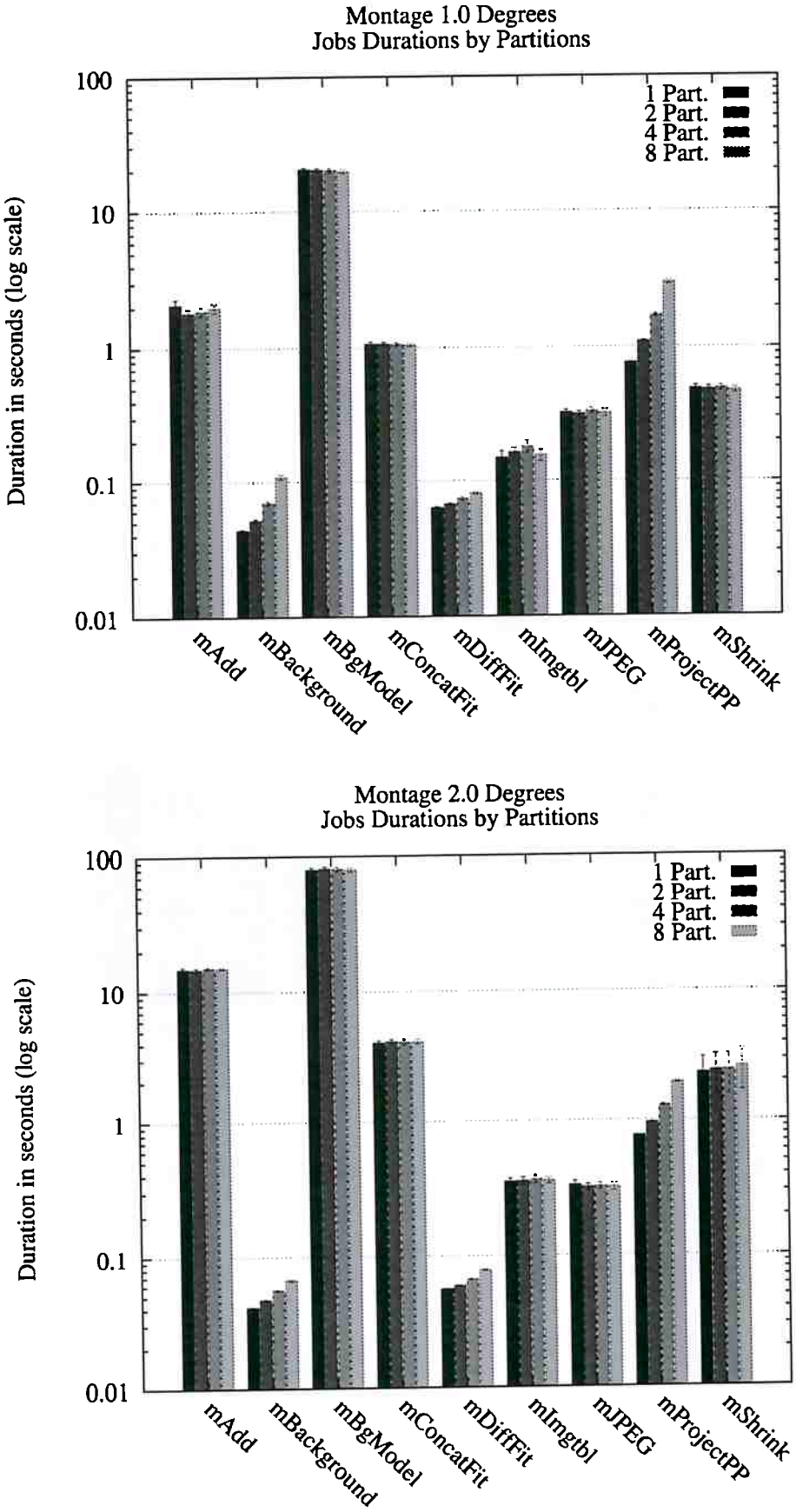

Figura 5.3: *Tempo médio de execução por tipo atividade do workflow Montage de graus 1.0 e 2.0.*

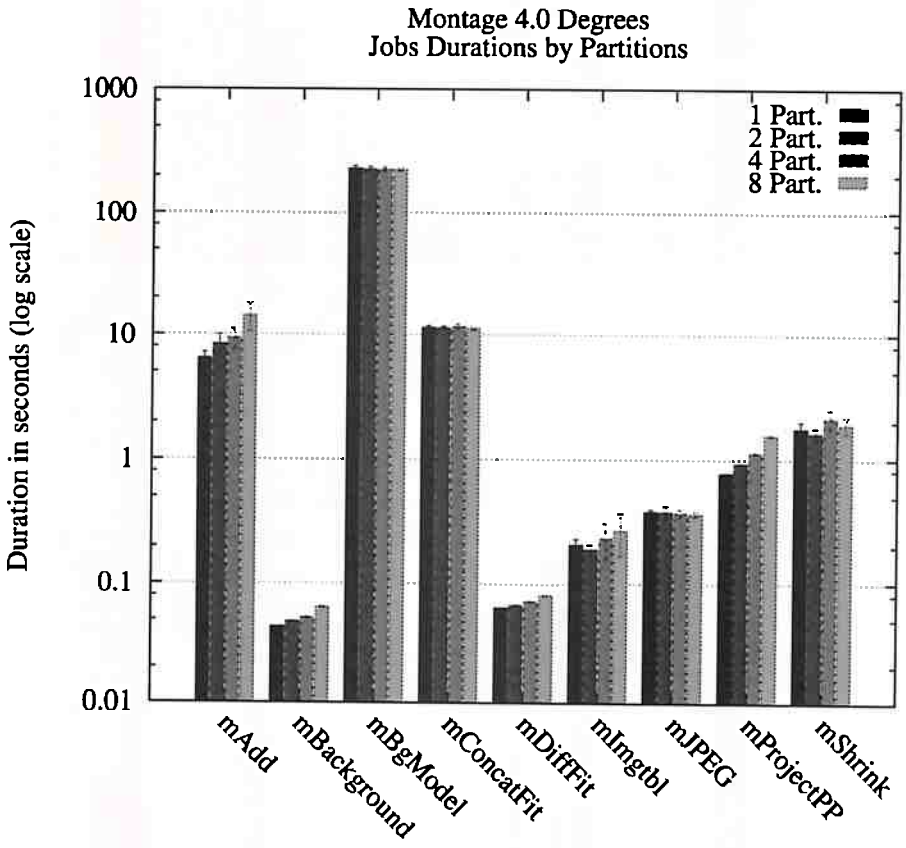

Figura 5.4: *Tempo médio de execução por tipo atividade do workflow Montage de graus 4.0.*

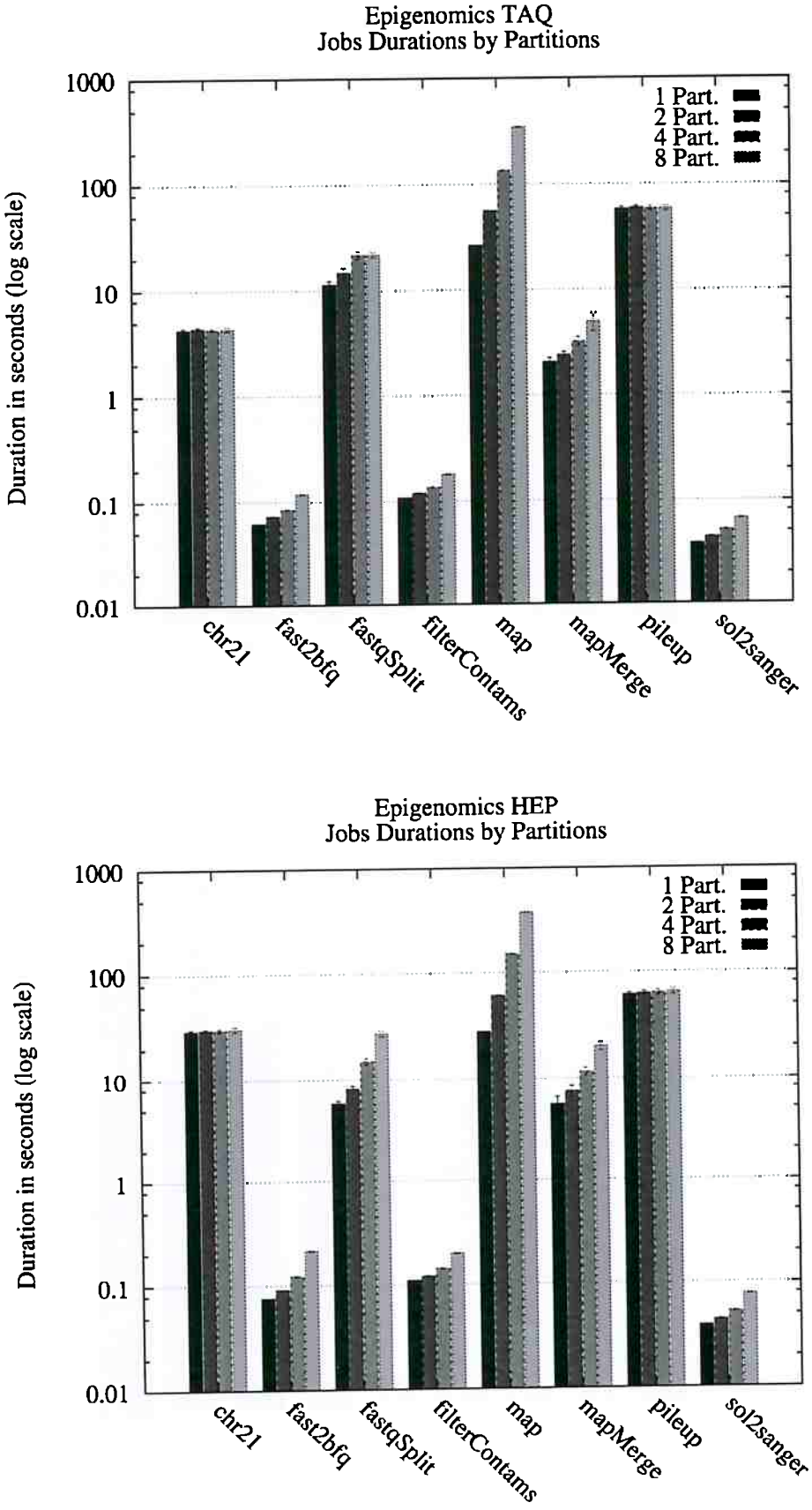

Figura 5.5: *Tempo médio de execução por tipo de atividade e diferentes conjuntos de entrada do workflow Epigenomics.*

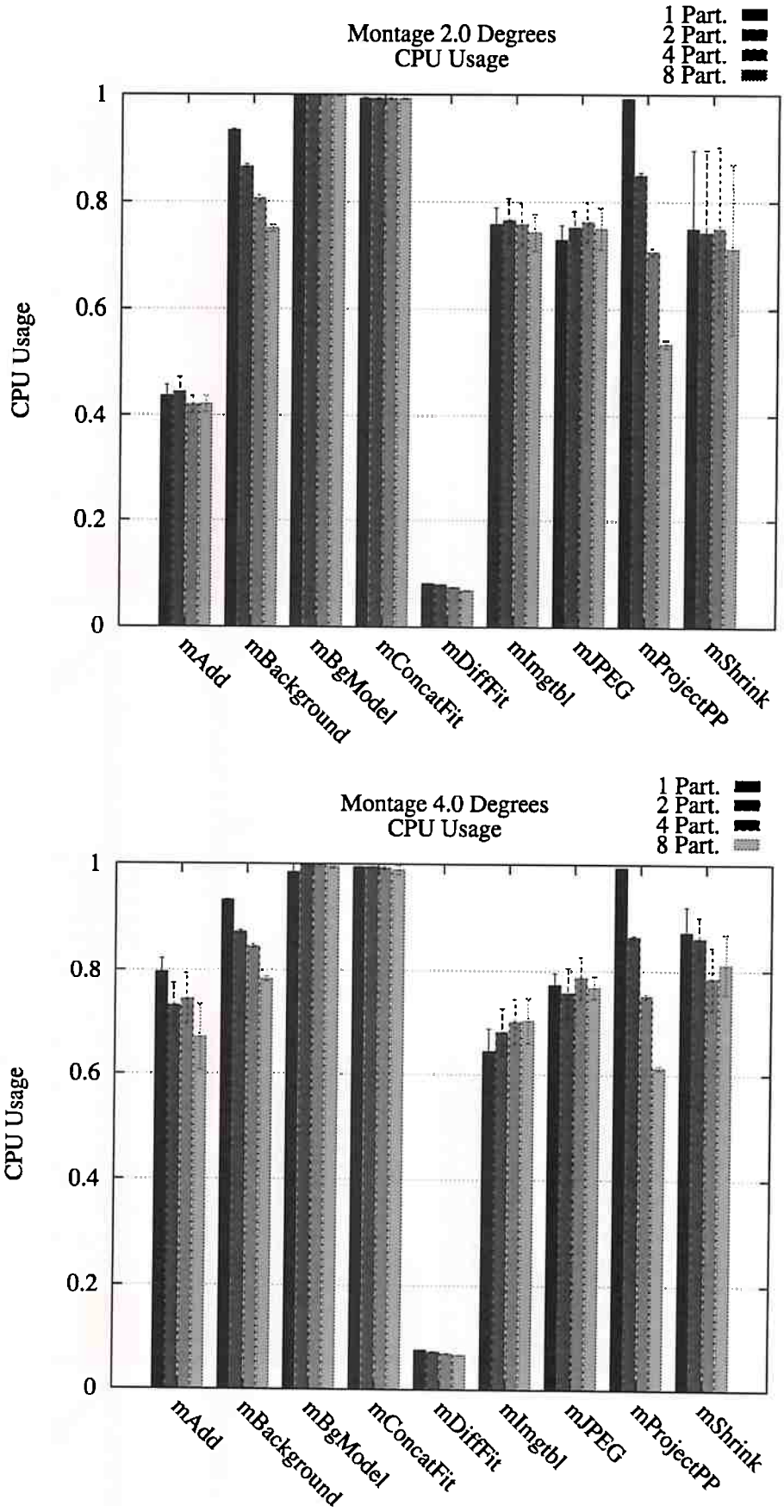

Figura 5.6: *Utilização média de CPU do workflow Montage por tipo de atividade.*

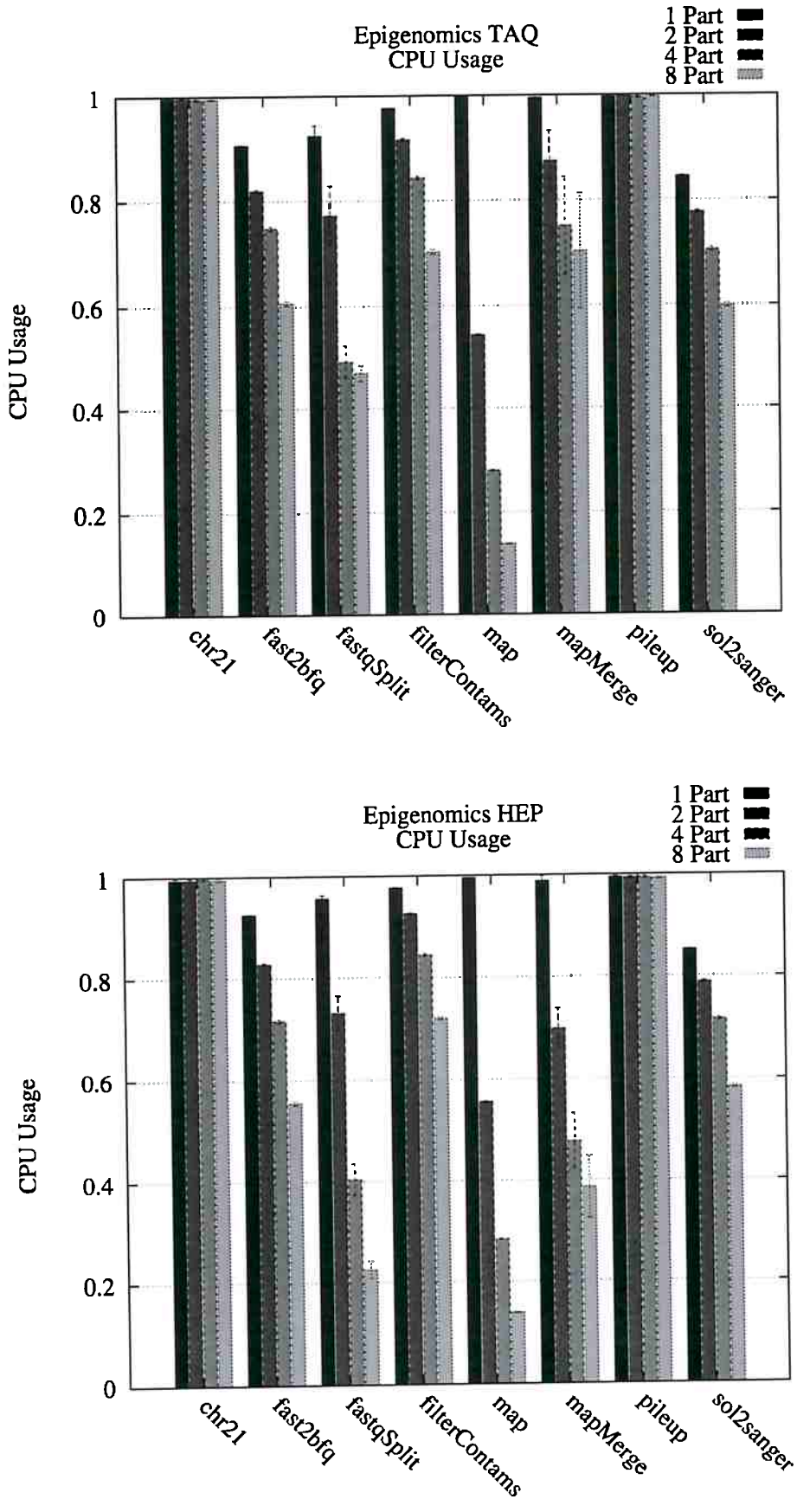

Figura 5.7: *Utilização média de CPU do workflow Epigenomics por tipo de atividade.*

Capítulo 6

Provisionamento Dinâmico de Recursos

Neste capítulo, é proposto um novo mecanismo de provisionamento dinâmico e escalonamento de atividades durante a execução de workflows científicos. Esse mecanismo utiliza informações provenientes do workflows científicos para tomadas de decisão. Tais decisões se beneficiam das características únicas da plataforma de computação em nuvem para provisionar recursos em tempo de execução.

6.1 Motivação e Objetivo

A elasticidade, característica singular das plataformas de computação em nuvem, permite que o poder computacional varie conforme uma demanda. Os recursos, em forma de máquinas virtuais, podem ser provisionados ou liberados em qualquer momento. A utilização, dessa liberdade de controle sobre os recursos, pode otimizar o desempenho de aplicações científicas. Contudo os SGWCs ainda não se beneficiam plenamente da elasticidade proveniente das plataformas de computação em nuvem.

A solução proposta pelo *glideinWMS*, descrita no final do Capítulo 3, utiliza da elasticidade para maximizar o *throughput* (vazão) das atividades em execução. É uma solução válida de provisionamento dinâmico contudo essa solução visa maximizar o número de tarefas em execução, sem se preocupar com restrições existentes como custo monetário do provisionamento ou término da execução de workflows. Também não utiliza informações das atividades para prever e melhorar escalonamentos futuros.

O experimento descrito neste capítulo utiliza-se de informações do workflow para tomadas de decisão sobre o provisionamento de recursos. Assim, a elasticidade da nuvem pode ser utilizada de forma a otimizar o desempenho da execução. O mecanismo proposto consegue, em um dado momento, decidir a quantidade exata de máquinas necessárias para execução do workflow em cada ponto de sua execução. Ela provisiona e gerencia esses recursos, além de distribuir as atividades entre os recursos através do escalonamento que efetua. A partir das informações do workflow, ele prevê eventos como início e término de tarefas que são utilizados para tomada de decisão do provisionamento. O programa também monitora a execução do workflow para ajustar-se às pequenas diferenças entre as previsões e os eventos reais, e se necessário reescalonar o workflow.

6.2 Ambiente Experimental

Nesta seção, descrevemos o ambiente de execução dos experimentos: infraestrutura, versões de software utilizados e especificações de configuração. Além disso descrevemos, em detalhes, o mecanismo proposto para o provisionamento dinâmico de recursos.

6.2.1 Condições Experimentais

As execuções deste experimento utilizaram a plataforma de computação em nuvem *Azure* da Microsoft. A novo modelo *ARM* (*Azure Resource Manager*) [azua] de criações de recursos da Azure foi utilizado através do conjunto de ferramentas *Microsoft Azure SDK for Python* [azud]. O PRECIP (descrito na Seção 4.3) foi utilizado para automatizar os experimentos.

Neste experimento, foram utilizadas máquinas virtuais *F-Series* do tipo *F1*. Uma máquina do tipo *F1*, possui 2GB de memória RAM, 16GB de SSD (unidade de disco sólido), e um núcleo virtual de CPU baseado no processador 2.4GHz Intel Xeon E5-2673 v3 (Haswell).

A imagem base das instâncias do experimento possui o sistema operacional Ubuntu 16.04, utiliza o Pegasus 4.6.1 como SGWC e o HTCondor 8.4.6 como gerenciador de recursos. Execuções com e sem o mecanismo proposto foram efetuadas e comparadas.

6.2.2 Escalonamento

Como padrão o SGWC Pegasus delega o escalonamento de atividades ao seu gerenciador de recursos HTCondor, caso contrário as atividades devem ser explicitamente mapeadas à recursos existentes. Mapear explicitamente atividades em recursos, em tempo de planejamento, geralmente é um método engessado e, por isso, impraticável.

O HTCondor escalona as atividades utilizando o algoritmo *myoptic* [YBR08]. Myopic é um método guloso que mapeia uma atividade pronta para ser executada no recurso que a terminará primeiro, mas como o Pegasus nem o HTCondor utilizam informações de previsões sobre a execução, a atividade é simplesmente casada com um recurso compatível. A compatibilidade entre recursos e atividades são descrita através de meta informações chamadas *ClassAds*. Cada atividade e recurso possuem informações que são publicados e utilizados para o mapeamento. É utilizando os ClassAds que o Pegasus pode mapear explicitamente uma atividade em um recurso, mas como dito antes, isso ocorre em tempo de planejamento, antes da execução do workflow.

O HTCondor tem uma interface na linha de comando que pode conversar e alterar os ClassAds. Um modo de controlar o escalonamento do HTCondor é utilizar-se dos ClassAds. Em tempo de planejamento, as atividades são associadas à um recurso inexistente que serve como fila de espera para o escalonamento. Em tempo de execução os ClassAds são atualizados para que as atividades executem nos recursos desejados.

Nosso mecanismo infere o escalonamento no HTCondor através do método descrito acima. A linha de comando descrita no Código 6.1 mostra como uma atividade pode ter seu ClassAd alterado através do comando `condor_qedit` [con].

```
condor_qedit -constraint 'GlobalJobId == "ATIVIDADE" ' Requirements '(( Target.
    Machine == "MAQUINA" ))'
```

Código 6.1: *Alterando o ClassAd da ATIVIDADE para ser executada no recurso MAQUINA*

6.2.3 Provisionamento

Nem o Pegasus e nem o HTCondor cuidam do provisionamento de recursos. O nó central do HTCondor sempre deve ser provisionado a priori, e o endereço desse nó deve estar descrito na configuração do workflow a ser executado pelo Pegasus. É através do nó central que o Pegasus consegue conversar com o conjunto gerenciado pelo HTCondor. Os nós de trabalho podem ser provisionados posteriormente. Desde que os nós de trabalho estejam configurados de forma correta e apontando o nó central, o HTCondor gerencia conversa e os integra ao conjunto de recursos disponíveis automaticamente.

O provisionamento dos nós de execução, geralmente, são feitos antes da execução dos workflows e de forma manual. O mecanismo proposto é executado no nó principal e gerencia o provisionamento de recursos. Ele conversa com a plataforma de computação em nuvem através de sua API para criar e destruir as máquinas virtuais. Uma camada sobre o API da Azure foi implementada utilizando o *Microsoft Azure SDK for Python* [azud] e encontra-se disponível no repositório [azub].

6.2.4 Mecanismo

O mecanismo proposto visa atacar o provisionamento dinâmico de recursos em execuções de workflows científicos, ficando também responsável pelo escalonamento das atividades nos recursos disponíveis. Em um dado momento, ele deve saber qual o número de recursos a serem alocados ou liberados. Deve gerenciar o provisionamento desses recursos, bem como mapear as atividades nos recursos disponíveis, e armazenar as informações contidas no escalonamento sobre o início e término das atividades. Essas informações são utilizadas para decidir se um novo escalonamento é necessário quando eventos reais diferem dos eventos previstos.

A Figura 6.1 mostra um diagrama de fluxo do funcionamento do mecanismo proposto. Como entrada do mecanismo temos: uma descrição do workflow, o orçamento disponível para execução do workflow de entrada e uma previsão do tempo de execução de cada atividade do workflow nos tipos de recursos disponíveis. Caso existam outros workflows em execução, existe um passo que combina os workflows formando um workflow único. Depois o workflow inteiro é escalonado e, se necessário, o número de VMs disponíveis é ajustado. VMs que serão necessárias somente posteriormente podem ser agendadas para serem provisionadas, e VMs que não serão mais utilizadas podem ser liberadas. No ciclo, mecanismo espera por novos workflows ou eventos, como o fim de uma atividade, o momento agendado para alocação de uma VM, ou, até mesmo, erros de execução. Quando uma atividade termina é necessário verificar se houve uma diferença muito grande da previsão proveniente do escalonamento e dos acontecimentos reais. Caso a diferença seja notável, um reescalonamento é necessário. Caso contrário, o mecanismo espera pelo próximo evento.

O Algoritmo 1 apresenta a função `NumeroVMs` que encontra o número de VMs necessário para um dado momento da execução a partir do orçamento existente. A entrada $nmax$ representa um limiar no número de máquinas do conjunto de recursos. O algoritmo faz uma busca entre 1 e $nmax$ para encontrar o número ideal de máquinas. A busca é feita de forma logarítmica sempre reduzindo o espaço entre o limite superior e o limite inferior pela metade, como mostra a Linha 10. A decisão é feita pelo função `Satisfaz`.

A função `Satisfaz` verifica se é possível escalonar o workflow com um número n de VMs utilizando o orçamento existente. A função de escalonamento é uma função simples de ordenação

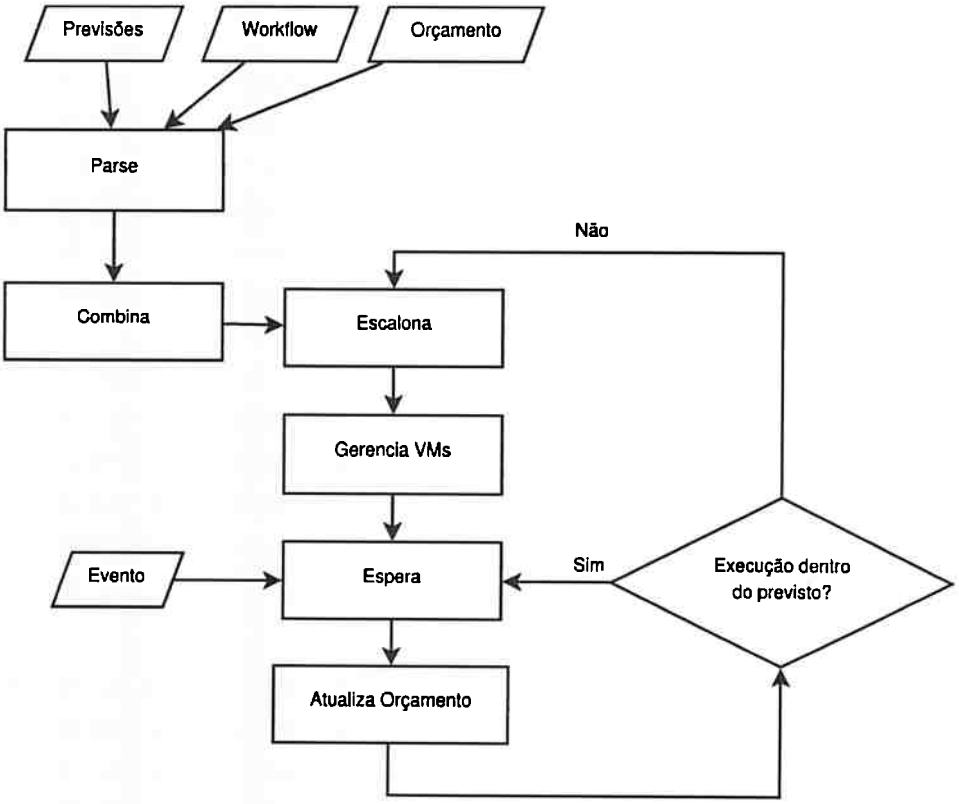

Figura 6.1: *Fluxograma do mecanismo de provisionamento.*

por *ranking*.

Note que só é possível fazer a busca pelo número de máquinas ideal, pois fixamos um único tipo de recurso. No caso de vários tipos de recursos devemos saber o número de cada recurso. Isso torna a busca complexa e impraticável.

6.2.5 Execução

Foram feitas execuções base com conjuntos de 17 máquinas do tipo *F1* sendo uma o nó principal, e as demais de execução. As execuções base não utilizaram o mecanismo proposto e serviram para calcular o tempo médio individual da execução das atividades de cada workflow. Esses valores foram utilizados como entrada de previsão para o mecanismo. Os custos monetários médios da execução de cada workflow também foi medido para servir de entrada, como orçamento, no mecanismo proposto. Dessa forma, foi possível comparar o mecanismo proposto com uma configuração padrão.

6.3 Conclusão do Capítulo

Neste capítulo descrevemos o mecanismo proposto para otimizar a utilização de recursos durante a execução de workflows científicos. Foi apresentado um fluxograma, retratando o funcionamento do mecanismo e cada passo do fluxo foi explicado. Os algoritmos utilizados pelo mecanismo foram apresentados e analisados.

Algoritmo 1 Algoritmo para encontrar o número de VMs necessárias em um dado momento

```
1: função NumeroVMs(nmax, oramento)
2:     satisfeito, custo[nmax] ← Satisfaz(nmax, oramento)
3:     se satisfeito == true então
4:         devolve nmax, custo[nmax]
5:     fim se
6:     limiteinf ← 0
7:     limitesup ← nmax − 1
8:     encontrado ← false
9:     enquanto encontrado ≠ true faça
10:        i ← ⌈(limiteinf + limitesup)/2⌉
11:        satisfeito, custo[i] ← Satisfaz(i, oramento)
12:        se satisfeito == true então
13:            limiteinf ← i
14:        senão
15:            limitesup ← i − 1
16:        fim se
17:        se limiteinf == limitesup então
18:            encontrado ← true
19:        fim se
20:    fim enquanto
21:    se limiteinf == 0 então
22:        Exceção                          ▷ O orçamento não foi suficiente
23:    fim se
24:    devolve limiteinf, custo[i]          ▷ O limite inferior representa o número de VMs
25: fim função
```

Capítulo 7

Considerações Finais

Este trabalho aborda o problema do provisionamento dinâmico de recursos em nuvens com foco workflows científicos. Delineia o estado atual da relação entre provisionamento de recursos e execução de workflows científicos. E propõe métodos de utilização das plataformas de computação em nuvem de forma a melhorar a utilização dos recursos, que influência diretamente no tempo e/ou no custo monetário de execuções de aplicações científicas.

As principais contribuições do trabalho foram: a proposta de um mecanismo que busca manter um número ideal de VMs alocados através do gerenciamento dinâmico do provisionamento de recursos e escalonamento de atividades; a implementação do mecanismo proposto e sua validação em nuvens reais; e o estudo de caso do compartilhamento de processadores entre atividades a fim de melhorar a utilização dos recursos.

O mecanismo proposto é, principalmente, responsável por alocar e liberar recursos em tempo de execução. O próprio mecanismo infere o número de recursos ideal para um dado momento. Essa inferência ocorre com base nas informações de execuções prévias, no orçamento disponível para execução, e no algoritmo de escalonamento utilizado. Diferente do que acontece em outros trabalhos, esse método beneficia-se da elasticidade da nuvem em tempo de execução otimizando a utilização dos recursos.

Sobre o estudo de caso do compartilhamento de processadores entre atividades, um impacto negativo ocorre sobre o tempo de execução individual das atividades, contudo esse compartilhamento promove um aumento no paralelismo da execução e pode, portanto, melhorar a execução do workflow como um todo.

Por fim, percebemos que as nuvens são ambientes atraentes para execução de workflows científico, pela sua característica de elasticidade que se bem aproveitada pode melhorar a utilização dos recursos. Mas, ainda existem várias questões não resolvidas que podem ser analisadas em trabalhos futuros.

7.1 Trabalhos Futuros

Para trabalhos futuros, existem várias possibilidades que podem ser expandidas. Como, por exemplo, o compartilhamento de recursos que pode ser feito de forma aplicada a atividades específicas com base em caracterizações dos workflows.

Várias questões são geradas quando consideramos diferentes tipos de máquina virtual. Quando existem vários tipos de recursos, a previsão utilizada sobre a execução deve ser revista, já que

uma atividade terá tempos distintos de execução em diferentes tipos de recursos. O mecanismo proposto para provisionamento não se aplica mais, pois não é mais possível saber o número ideal de recursos necessários em um dado momento. É preciso saber quantos recursos de cada tipo devem ser alocados, o que era, antes, um problema linear, torna-se um problema multidimensional.

Referências Bibliográficas

[AFG+09] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica e Matei Zaharia. Above the clouds: A berkeley view of cloud computing. Relatório Técnico UCB/EECS-2009-28, EECS Department, University of California, Berkeley, Feb 2009.

[app] Google app engine. https://appengine.google.com. Visitado: 08/08/2016.

[ARJ+13] Sepideh Azarnoosh, Mats Rynge, Gideon Juve, Ewa Deelman, Michal Niec, Maciej Malawski e Rafael Ferreira da Silva. Introducing PRECIP: An API for Managing Repeatable Experiments in the Cloud. Em *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, volume 2, páginas 19–26. IEEE, 2013.

[azua] Azure Resource Manager overview. https://azure.microsoft.com/en-us/documentation/articles/resource-group-overview/. [Visitado: 08/08/2016].

[azub] Gerenciamento de VMs para Azure (ARM). https://github.com/rika/azure_arm_vm_manager.

[azuc] Microsoft Azure. http://azure.microsoft.com/. [Visitado: 08/08/2016].

[azud] Microsoft Azure SDK for Python. https://github.com/Azure/azure-sdk-for-python. [Visitado: 08/08/2016].

[BCD+08] Shishir Bharathi, Ann Chervenak, Ewa Deelman, Gaurang Mehta, Mei-Hui Su e Karan Vahi. Characterization of scientific workflows. Em *Workflows in Support of Large-Scale Science, 2008. WORKS 2008. Third Workshop on*, páginas 1–10. IEEE, 2008.

[BDJ+13] G Bruce Berriman, Ewa Deelman, Gideon Juve, Mats Rynge e Jens-S Vöckler. The application of cloud computing to scientific workflows: a study of cost and performance. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1983):20120066, 2013.

[BHS09] Gordon Bell, Anthony Hey e Alex Szalay. Beyond the data deluge. *Science*, 323(5919):1297–1298, 2009.

[bot] boto. http://docs.pythonboto.org/. Visitado: 08/08/2016.

[CBGK13] Daniel Cordeiro, Kelly Braghetto, Alfredo Goldman e Fabio Kon. Da ciência à e-ciência: paradigmas da descoberta do conhecimento. *Revista USP*, 0(97), 2013.

[CD11] Weiwei Chen e Ewa Deelman. Workflow overhead analysis and optimizations. Em *Proceedings of the 6th workshop on Workflows in support of large-scale science*, páginas 11–20. ACM, 2011.

[CKR+07] Peter Couvares, Tevfik Kosar, Alain Roy, Jeff Weber e Kent Wenger. Workflow management in Condor. Em *Workflows for e-Science*, páginas 357–375. Springer, 2007.

[con] condor_qedit. http://research.cs.wisc.edu/htcondor/manual/current/condor_qedit.html. Visitado: 08/08/2016.

[CRB+11] Rodrigo N Calheiros, Rajiv Ranjan, Anton Beloglazov, César AF De Rose e Rajkumar Buyya. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1):23–50, 2011.

[DES98] Weimin Du, Graham Eddy e Ming-Chien Shan. Distributed workflow resource management system and method, Outubro 20 1998. US Patent 5,826,239.

[DGST09] Ewa Deelman, Dennis Gannon, Matthew Shields e Ian Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.

[DJMN13] Ewa Deelman, Gideon Juve, Maciej Malawski e Jarek Nabrzyski. Hosted science: Managing computational workflows in the cloud. *Parallel Processing Letters*, 23(02), 2013.

[DSL+08] Ewa Deelman, Gurmeet Singh, Miron Livny, Bruce Berriman e John Good. The cost of doing science on the cloud: the Montage example. Em *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, página 50. IEEE Press, 2008.

[DSS+05] Ewa Deelman, Gurmeet Singh, Mei-Hui Su, James Blythe, Yolanda Gil, Carl Kesselman, Gaurang Mehta, Karan Vahi, G Bruce Berriman, John Good et al. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.

[ec2] Amazon EC2. http://aws.amazon.com/ec2/. Visitado: 08/08/2016.

[epi] Epigenomics. http://epigenome.usc.edu/. Visitado: 08/08/2016.

[FdSJD+13] Rafael Ferreira da Silva, Gideon Juve, Ewa Deelman, Tristan Glatard, Frédéric Desprez, Douglas Thain, Benjamin Tovar e Miron Livny. Toward fine-grained online task characteristics estimation in scientific workflows. Em *8th Workshop on Workflows in Support of Large-Scale Science*, WORKS '13, páginas 58–67, 2013.

[FFK+06] Ian Foster, Timothy Freeman, Kate Keahy, Doug Scheftner, Borja Sotomayer e Xuehai Zhang. Virtual clusters for grid communities. Em *Cluster Computing and the Grid, 2006. CCGRID 06. Sixth IEEE International Symposium on*, volume 1, páginas 513–520. IEEE, 2006.

[Fre02] James Frey. Condor dagman: Handling inter-job dependencies. *University of Wisconsin, Dept. of Computer Science, Tech. Rep*, 2002.

[gen] WorkflowGenerator. https://confluence.pegasus.isi.edu/display/pegasus/WorkflowGenerator. Visitado: 08/08/2016.

[gli] GlideinWMS. http://glideinwms.fnal.gov/. Visitado: 08/08/2016.

[GLLK79] Ronald L Graham, Eugene L Lawler, Jan Karel Lenstra e AHG Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete Mathematics*, 5:287–326, 1979.

[glu] Gluster. http://www.gluster.org/. Visitado: 08/08/2016.

[gma] Gmail. https://mail.google.com/. Visitado: 08/08/2016.

[gog] GoGrid. http://www.gogrid.com/. Visitado: 08/08/2016.

[goo] Google Cloud. https://cloud.google.com/. [Visitado: 08/08/2016].

[HKR13] Nikolas Roman Herbst, Samuel Kounev e Ralf Reussner. Elasticity in cloud computing: What it is, and what it is not. Em *ICAC*, páginas 23–27, 2013.

[HMF+08] Christina Hoffa, Gaurang Mehta, Timothy Freeman, Ewa Deelman, Kate Keahey, Bruce Berriman e John Good. On the use of cloud computing for scientific workflows. Em *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, páginas 640–645. IEEE, 2008.

[HT03] Anthony Hey e Anne Trefethen. The data deluge: An e-science perspective. Em *Grid Computing*, páginas 809–824. Wiley and Sons, Chichester, UK, 2003.

[htc] HTCondor. http://research.cs.wisc.edu/htcondor/. Visitado: 08/08/2016.

[HTT09] Anthony Hey, Stewart Tansley e Kristin M. Tolle. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

[JCD+13] Gideon Juve, Ann Chervenak, Ewa Deelman, Shishir Bharathi, Gaurang Mehta e Karan Vahi. Characterizing and profiling scientific workflows. *Future Generation Computer Systems*, 29(3):682 – 692, 2013. Special Section: Recent Developments in High Performance Computing and Security.

[JD10] Gideon Juve e Ewa Deelman. Scientific workflows and clouds. *Crossroads*, 16(3):14–18, 2010.

[JD11a] Gideon Juve e Ewa Deelman. Automating application deployment in infrastructure clouds. Em *Proceedings of the 2011 IEEE Third International Conference on Cloud Computing Technology and Science*, CLOUDCOM '11, páginas 658–665, Washington, DC, USA, 2011. IEEE Computer Society.

[JD11b] Gideon Juve e Ewa Deelman. Scientific workflows in the cloud. Em *Grids, Clouds and Virtualization*, páginas 71–91. Springer, 2011.

[JD11c] Gideon Juve e Ewa Deelman. Wrangler: Virtual cluster provisioning for the cloud. Em *Proceedings of the 20th International Symposium on High Performance Distributed Computing*, HPDC '11, páginas 277–278, New York, NY, USA, 2011. ACM.

[JDV+09] Gideon Juve, Ewa Deelman, Karan Vahi, Gaurang Mehta, Bruce Berriman, Benjamin P Berman e Phil Maechling. Scientific workflow applications on Amazon EC2. Em *E-Science Workshops, 2009 5th IEEE International Conference on*, páginas 59–66. IEEE, 2009.

[JTFdS+15] Gideon Juve, Benjamin Tovar, Rafael Ferreira da Silva, Dariusz Król, Douglas Thain, Ewa Deelman, William Allcock e Miron Livny. Practical Resource Monitoring for Robust High Throughput Computing. Em *Workshop on Monitoring and Analysis for High Performance Computing Systems Plus Applications*, HPCMASPA'15, página to appear, 2015.

[LAB+06] Bertram Ludäscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A Lee, Jing Tao e Yang Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065, 2006.

[LLF+09] Cui Lin, Shiyong Lu, Xubo Fei, Artem Chebotko, Darshan Pai, Zhaoqiang Lai, Farshad Fotouhi e Jing Hua. A reference architecture for scientific workflow management systems and the VIEW SOA solution. *Services Computing, IEEE Transactions on*, 2(1):79–92, 2009.

[LWMB09] Bertram Ludäscher, Mathias Weske, Timothy McPhillips e Shawn Bowers. Scientific workflows: business as usual? Em *Business Process Management*, páginas 31–47. Springer, 2009.

[MG09] Peter Mell e Tim Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology*, 53(6):50, 2009.

[MJDN12] Maciej Malawski, Gideon Juve, Ewa Deelman e Jarek Nabrzyski. Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. Em *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, página 22. IEEE Computer Society Press, 2012.

[mon] Montage. http://montage.ipac.caltech.edu. Visitado: 08/08/2016.

[MTH+14] Parag Mhashilkar, Anthony Tiradani, Burt Holzman, Krista Larson, Igor Sfiligoi e Mats Rynge. Cloud Bursting with GlideinWMS: Means to satisfy ever increasing computing needs for Scientific Workflows. *Journal of Physics: Conference Series*, 513(3):032069, 2014.

[nim] Nimbus Context Broker. http://www.nimbusproject.org/doc/ctxbroker/latest/. Visitado: 08/08/2016.

[par] Paramiko. http://www.paramiko.org/. Visitado: 08/08/2016.

[pbs] PBS. http://www.pbsworks.com/. Visitado: 08/08/2016.

[peg] Pegasus. https://pegasus.isi.edu. Visitado: 08/08/2016.

[prea] PRECIP. https://github.com/pegasus-isi/precip. Visitado: 08/08/2016.

[preb] PRECIP para GCloud e Azure. https://github.com/rika/precip.

[pro] Provisionamento dinâmico de recursos. https://github.com/rika/dynamic-provisioning.

[pvf] PVFS. http://www.parl.clemson.edu/pvfs/. Visitado: 08/08/2016.

[RHRB13] Mustafizur Rahman, Rafiul Hassan, Rajiv Ranjan e Rajkumar Buyya. Adaptive workflow scheduling for dynamic grid and cloud computing environment. *Concurrency and Computation: Practice and Experience*, 25(13):1816–1842, 2013.

[s3] Amazon S3. http://aws.amazon.com/s3/. Visitado: 08/08/2016.

[Sfi08] Igor Sfiligoi. glideinWMS—a generic pilot-based workload management system. *Journal of Physics: Conference Series*, 119(6):062044, 2008.

[TDGS07] I.J. Taylor, E. Deelman, D.B. Gannon e M. Shields. *Workflows for e-Science*. Springer, 2007.

[tor] TORQUE. http://www.adaptivecomputing.com/products/open-source/torque/. Visitado: 08/08/2016.

[twi] Twitter. https://twitter.com. Visitado: 08/08/2016.

[vDATHKB03] Wil MP van Der Aalst, Arthur HM Ter Hofstede, Bartek Kiepuszewski e Alistair P Barros. Workflow patterns. *Distributed and parallel databases*, 14(1):5–51, 2003.

[VMZ+06] Jens S. Vockler, Gaurang Mehta, Yong Zhao, Ewa Deelman e Mike Wilde. Kickstarting remote applications. Em *International Workshop on Grid Computing Environments*, GCE '06, 2006.

[VRMB11] Luis M. Vaquero, Luis Rodero-Merino e Rajkumar Buyya. Dynamically scaling applications in the cloud. *ACM SIGCOMM Computer Communication Review*, 41(1):45–52, 2011.

[WDL+13] Long Wang, Rubing Duan, Xiaorong Li, Sifei Lu, T. Hung, R.N. Calheiros e R. Buyya. An iterative optimization framework for adaptive workflow management in computational clouds. Em *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, páginas 1049–1056, July 2013.

[WfM99] WfMC. Terminology and glossary. Relatório técnico, Document No WFMC-TC-1011. Workflow Management Coalition. Winchester, 1999.

[YB05] Jia Yu e Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *ACM Sigmod Record*, 34(3):44–49, 2005.

[YBR08] Jia Yu, Rajkumar Buyya e Kotagiri Ramamohanarao. *Workflow Scheduling Algorithms for Grid Computing*, páginas 173–214. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[YMF+10] Y.O. Yazir, C. Matthews, R. Farahbod, S. Neville, A Guitouni, S. Ganti e Y. Coady. Dynamic resource allocation in computing clouds using distributed multiple criteria decision analysis. Em *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, páginas 91–98, July 2010.