UMA ESTRATÉGIA DE EXECUÇÃO PARALELA ADAPTÁVEL DE *WORKFLOWS* CIENTÍFICOS

Vítor Silva Sousa

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia de Sistemas e Computação.

Orientadores: Marta Lima de Queirós Mattoso
Daniel Cardoso Moraes de Oliveira

Rio de Janeiro
Agosto de 2014

UMA ESTRATÉGIA DE EXECUÇÃO PARALELA ADAPTÁVEL DE *WORKFLOWS* CIENTÍFICOS

Vítor Silva Sousa

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

_____

Prof[a]. Marta Lima de Queirós Mattoso, D.Sc.

_____

Prof. Daniel Cardoso Moraes de Oliveira, D.Sc.

_____

Prof. Alexandre de Assis Bento Lima, D.Sc.

_____

Dr. Kary Ann del Carmen Soriano Ocaña, D.Sc.

_____

Prof. Antônio Tadeu Azevedo Gomes, D.Sc.

RIO DE JANEIRO, RJ - BRASIL

AGOSTO DE 2014

Sousa, Vítor Silva

Uma Estratégia de Execução Paralela Adaptável de *Workflows* Científicos / Vítor Silva Sousa. – Rio de Janeiro: UFRJ/COPPE, 2014.

X, 96 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso

Daniel Cardoso Moraes de Oliveira

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2014.

Referências Bibliográficas: p. 89-96.

1. Paralelismo de dados. 2. *Workflows* Científicos. 3. Ponto-a-ponto. I. Mattoso, Marta Lima de Queirós *et al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III Título.

À minha família.

AGRADECIMENTOS

À Prof[a]. Marta Mattoso, minha orientadora, por ter acreditado no meu potencial desde o segundo período do curso de graduação, pelas críticas muito valiosas no amadurecimento dos trabalhos, pelas sugestões do meu encaminhamento acadêmico e pelas excelentes oportunidades oferecidas;

Ao Daniel de Oliveira, Jonas Dias e Eduardo Ogasawara, que sempre me proporcionaram um ambiente incomparável para aquisição de conhecimento, aplicação dos conceitos aprendidos em aulas, experiências de vida e mesmo a difícil habilidade de separar o ambiente do trabalho com o da amizade;

Aos membros da banca, Prof. Antônio Tadeu, Prof. Alexandre e Prof[a]. Kary, por aceitarem em participar da apresentação da minha dissertação de mestrado;

À Mara Prata e Patrícia Leal por sempre me ajudarem com as questões administrativas;

A todos os amigos, que me apoiaram ao longo do curso e acreditaram no meu potencial para o desenvolvimento desse projeto. Amigos que manifestaram em pequenos atos esse sentimento tão difícil de ser mensurado, a amizade. Alguns exemplos podem ser citados, como: as caronas para Niterói da Carolina e as companhias da Thayana Bruno, Igor Campbell, Lívia Pimentel, Luiz Alves, Felipe Horta, Talita Lopes e Thaiana Pinheiro;

Agradeço.

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

UMA ESTRATÉGIA DE EXECUÇÃO PARALELA ADAPTÁVEL DE *WORKFLOWS* CIENTÍFICOS

Vítor Silva Sousa

Agosto/2014

Orientadores: Marta Lima de Queirós Mattoso
Daniel Cardoso Moraes de Oliveira
Programa: Engenharia de Sistemas e Computação

Muitos experimentos científicos de larga escala são modelados como *workflows* científicos e executados por meio de Sistemas de Gerência de *Workflows* Científicos (SGWfC). Em virtude do volume de dados a serem processados, alguns SGWfC têm utilizado técnicas de paralelismo em ambientes de processamento de alto desempenho para produzir seus resultados. A ocorrência de falhas também tornou-se uma certeza com o significativo crescimento no número de núcleos computacionais e o volume de dados processados nesses ambientes. Nesse sentido, alguns SGWfC apresentam estratégias adaptativas dos nós computacionais em tempo de execução de acordo com a disponibilidade dos mesmos ou a ocorrência de falhas. Contudo, tais soluções são normalmente baseadas em um escalonamento estático dos *workflow*s e não apoiam a gerência das transformações de dados envolvidas na execução desses *workflows*. Somando-se a isso, tais adaptações precisam garantir a eficiência desses nós computacionais, evitando altos custos para eventuais atualizações na alocação de ativações. Logo, esta dissertação propõe uma estratégia de execução paralela adaptável de *workflows* científicos, garantindo a eficiência dos nós computacionais utilizados.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

A STRATEGY FOR ADAPTIVE PARALLEL EXECUTION OF SCIENTIFIC WORKFLOWS

Vítor Silva Sousa

August/2014

Advisors: Marta Lima de Queirós Mattoso

Daniel Cardoso Moraes de Oliveira

Department: Systems and Computer Engineering

Many large-scale scientific experiments are modeled as scientific workflows and executed through the Scientific Workflow Management Systems (SWfMS). Due to the volume of data to be processed, some SWfMS are using techniques of parallelism in high performance computing environments to produce theirs results. Failures also become a certainty with the significant increase in number of computational cores and the volume of data processed in these environments. This way, some SWfMS propose an adaptive strategy of computational nodes at runtime according to the their availability and failure occurrences. However, these solutions are normally based on a static scheduling and they do not support the management of the data transformations involved in execution of these workflows. Moreover, these adaptations have to guarantee the efficiency of these computational nodes, avoiding high costs for eventual tasks allocation. Thus, this dissertation proposes a strategy for adaptive parallel execution of scientific workflows, in order to guarantee efficiency of the used computational nodes.

ÍNDICE

LISTAGEM DE FIGURAS

LISTAGEM DE TABELAS

Capítulo 1 – Introdução

Os experimentos científicos em larga escala são conduzidos por cientistas ou grupos de pesquisa em diferentes domínios, como a bioinformática (STEVENS *et al.*, 2003) e a exploração de petróleo em águas profundas (SILVA e SENGER, 2009). Tais experimentos representam estudos que utilizam a combinação de diferentes programas e serviços, assim como ambientes de computação distribuída e heterogênea. O ciclo de vida dos experimentos científicos (MATTOSO *et al.*, 2010) passa pelas etapas de composição, execução e análise desses experimentos. Uma das dificuldades em prover uma infraestrutura computacional para apoiar esse ciclo de vida está na gerência do encadeamento dos programas e de um volume de dados cada vez maior para ser processado e transformado ao longo dos programas (PENNISI, 2011).

Para apoiar essa questão, os experimentos científicos podem ser modelados como *workflows* científicos. Um *workflow* científico pode ser definido por um conjunto de atividades e um fluxo de dados entre elas. As atividades representam programas de simulação computacional que consomem e produzem conjuntos de dados, gerando o fluxo de dados. O *workflow* científico especifica a dependência dos dados, na qual o conjunto de dados de saída de uma atividade serve de conjunto de entrada de outra atividade, compondo um encadeamento de atividades (BROWN *et al.*, 2007). Nesse sentido, os Sistemas de Gerência de *Workflows* Científicos (SGWfC) surgiram para apoiar a modelagem, a execução e o monitoramento dos *workflows* científicos (BERRIMAN *et al.*, 2007, DIAS *et al.*, 2013, OCAÑA *et al.*, 2011). Como exemplos de SGWfC, pode-se citar o VisTrails (CALLAHAN *et al.*, 2006), o Taverna (MISSIER *et al.*, 2010), o Swift/T (WOZNIAK *et al.*, 2013), o Kepler (LUDÄSCHER *et al.*,

2006), o Pegasus (DEELMAN *et al.*, 2007), o Chiron (OGASAWARA *et al.*, 2013) e o Galaxy (GOECKS *et al.*, 2010).

Outra característica importante na gerência de experimentos científicos por meio de *workflows* é a captura de dados de proveniência (FREIRE *et al.*, 2008), que consiste no registro das informações relacionadas à execução do *workflow*. Tais registros podem permitir uma análise do fluxo de dados gerados ao longo das atividades, durante ou após a execução dos *workflows*. Portanto, além de serem essenciais para a análise dos resultados, os dados de proveniência contribuem para a confiabilidade e a reprodutibilidade desses resultados.

Devido ao grande volume de dados a serem processados nos *workflows* científicos, a execução sequencial desses *workflows* pode não ser suficiente para atender aos prazos estabelecidos pelos cientistas. Por esse motivo, técnicas de paralelismo em ambientes de Processamento de Alto de Desempenho (PAD) têm sido empregadas para reduzir o tempo total de execução dos mesmos. Como exemplos de ambientes de PAD têm-se os *clusters*, as grades computacionais e as nuvens computacionais. Mais especificamente, esta dissertação está pautada na análise de *workflows* científicos com acesso intensivo a dados.

Para permitir a execução paralela em ambientes de PAD, o SGWfC precisa ter capacidade de gerenciar o paralelismo ao longo da execução do *workflow*, sendo esses sistemas conhecidos como SGWfC paralelos. Considerando o grande volume de dados, SGWfC adotam com frequência o paralelismo de dados. Nesse sentido, instâncias de uma atividade do *workflow* são replicadas pelos nós computacionais (representados pela sigla NC) e executadas em paralelo, sendo que cada instância da atividade consome um elemento do conjunto de dados de entrada da atividade no *workflow*. Cada

instância da atividade criada para o paralelismo de dados é também conhecida como tarefa. O processo de gerência de execução paralela de instâncias de atividades é denominado de ativação de atividade, conforme definido na abordagem algébrica de workflows proposta por OGASAWARA (2011). Além disso, uma ativação de atividades consiste na menor unidade de dados necessária para executar a atividade em qualquer núcleo dos nós computacionais disponíveis, determinando quais programas devem ser invocados e quais dados precisam ser acessados.

Uma ativação de atividade também consiste de três etapas: instrumentação dos dados de entrada, invocação do programa e extração dos dados de saída. A instrumentação dos dados é caracterizada pela extração dos valores dos dados de entrada e os prepara para a execução do programa especificado. A invocação do programa, por conseguinte, permite o disparo e o monitoramento da execução programa após a instrumentação. Por último, a extração dos dados de saída captura os valores de dados de saída produzidos após a invocação do programa, a fim de estabelecer os dados de saída para a ativação.

A execução paralela de *workflows* científicos em ambientes de PAD por meio de técnicas de paralelismo aumentou a quantidade de dados processados em um mesmo intervalo de tempo. Com isso, assim como em qualquer execução paralela em larga escala, a probabilidade de falhas em tempo de execução também aumenta. Da mesma forma, o crescimento do número de nós computacionais utilizados em paralelo também aumentam a probabilidade de indisponibilidade de um NC. Sendo assim, os SGWfC precisam apoiar o uso dos NC durante a execução paralela, considerando eventuais mudanças (falhas ou indisponibilidade dos NC) em tempo de execução. Três características desejáveis em sistemas paralelos, que iremos abordar no contexto de

SGWfC paralelos são as seguintes: escalonamento adaptativo, tolerância a falhas e balanceamento de carga.

O escalonamento de ativações em *workflows* é tradicionalmente definido pelo mapeamento das ativações das atividades para os diferentes NC. Além disso, o escalonamento pode ser classificado como estático ou adaptativo (BUX e LESER, 2013). O escalonamento estático, por definição, determina como as ativações serão distribuídas para os diferentes NC antes de iniciar a execução do *workflow*. Logo, o escalonamento estático define um mapeamento para a distribuição das ativações de acordo com os NC disponíveis previamente, não permitindo adaptações nesse mapeamento em tempo de execução.

Ao contrário da proposta estática, o escalonamento adaptativo (também conhecido como dinâmico) permite alterações nesse mapeamento em tempo de execução, realizando mudanças específicas no mesmo para a distribuição das ativações. Nesse caso, para permitir a adaptabilidade dos NC em relação às falhas no ambiente de PAD ou ao desbalanceamento de carga, devemos utilizar um escalonamento adaptativo. Do mesmo modo, o escalonamento adaptativo pode considerar também a disponibilidade (PRICE e TINO, 2009), a confiabilidade (OLIVEIRA *et al.*, 2012, RASOOLI e DOWN, 2011), e o custo financeiro dos NC disponíveis no ambiente de PAD utilizado (OLIVEIRA *et al.*, 2010).

Algumas tecnologias têm sido propostas para apoiar o desenvolvimento de aplicações paralelas que consideram a adaptabilidade de NC, como o *Adaptive* MPI (HUANG *et al.*, 2006). Além disso, uma das formas possíveis de implementar soluções adaptativas em relação aos NC para a execução paralela de aplicações científicas consiste no uso de técnicas P2P (do inglês *peer-to-peer*, ou simplesmente ponto-a-

ponto) (PAPUZZO e SPEZZANO, 2011). As técnicas P2P são voltadas para a adaptabilidade inerente ao tratamento de ingressos e de saídas frequentes de pontos da rede (conhecidos como *churns*). Chamaremos de volatilidade de NC aos *churns* das redes P2P (SHEN *et al.*, 2010). Assim, técnicas P2P favorecem a tolerância a falhas, pois a partir da identificação de falhas nos NC em tempo de execução, os algoritmos P2P adaptam a rede removendo os NC com falha, eventualmente redirecionando a execução para outros pontos ou incluindo outros pontos de execução. O cenário de adaptação relacionado à tolerância a falhas implica a alteração do mapeamento das ativações (escalonamento de ativações do *workflow*).

A ocorrência de falhas também tornou-se uma certeza com o significativo crescimento no número de NC e no volume de dados processados nos ambientes de PAD. Portanto, observa-se a importância do desenvolvimento de SGWfC capazes de continuar a execução dos *workflows* mesmo diante de eventuais falhas nos NC. De acordo com o nível de tolerância a falhas (por exemplo, falhas relacionadas a software ou a hardware), esses SGWfC podem ser definidos como tolerantes a falhas (HANMER, 2007, ISERMANN, 2005).

Como exemplo temos a ocorrência de falhas irrecuperáveis (EDELWEISS e NICOLAO, 1998) em um NC, conhecido como NC_i, durante a execução de um *workflow*. Nesse cenário, a redistribuição de ativações de NC_i para os outros NC disponíveis não é possível, pois NC_i não pode ser acessado (e, consequentemente, também não pode redistribuir suas ativações que estavam pendentes ou executando). Entretanto, pode-se identificar tal evento pelo bloqueio dessas ativações pelo NC_i por um tempo superior ao constatado em execuções anteriores dessas ativações. Além da sua identificação, esse evento pode ser evitado caso um ou mais NC diferentes de NC_i também possuam as ativações que estavam executando em NC_i (conjunto conhecido

como SNC_j). Portanto, ao ocorrer mudanças no mapeamento das ativações em tempo de execução, as ativações que estavam executando em NC_i podem ser redistribuídas pelos NC a partir das réplicas dessas ativações em SNC_j.. Esse cenário de adaptação relacionado à tolerância a falhas implica a alteração do mapeamento das ativações (escalonamento do *workflow*).

O balanceamento de carga é outra característica essencial em execuções paralelas. No caso dos SGWfC paralelos, o balanceamento é ainda mais crítico, uma vez que as aplicações científicas são "caixas-pretas" e tendem a processar muitas ativações. Cada ativações podem apresentar um tempo de processamento diferente (também chamado de carga). Tais ativações, ao serem processadas em paralelo, podem gerar um mapeamento irregular ou desbalanceado das ativações para os NC disponíveis. Logo, observa-se a importância dos SGWfC paralelos de identificarem e tratarem o desbalanceamento de carga, a fim de reduzir o tempo total de execução dos *workflows* científicos. Vale ressaltar também que as adaptações no mapeamento das ativações para os NC são necessárias para realizar o balanceamento de carga durante a execução.

Considerando as três características apresentadas para apoiar a execução paralela em *workflows* científicos, pode-se constatar que alguns SGWfC paralelos apresentam estratégias de escalonamento adaptativo (COSTA *et al.*, 2012, LEE *et al.*, 2008, 2011, OLIVEIRA *et al.*, 2012, PAPUZZO e SPEZZANO, 2011, RASOOLI e DOWN, 2011). Portanto, tais SGWfC paralelos modificam o mapeamento das ativações para os NC em tempo de execução. Em relação às outras características, COSTA *et al.*, (2012) propuseram uma estratégia de escalonamento adaptativo aliada à tolerância a falhas, a partir da captura e consulta dos dados de proveniência em tempo de execução. Já PAPUZZO e SPEZZANO (2011); e RASOOLI e DOWN (2011)

propuseram estratégias adaptativas associadas ao balanceamento de carga. Mais especificamente, PAPUZZO e SPEZZANO (2011) desenvolveram métricas de qualidade de serviço (do inglês, *Quality of Service* ou QoS) para determinar mudanças no mapeamento das ativações para os NC, enquanto RASOOLI e DOWN (2011) apresentaram métricas baseadas no tempo de execução dos *workflows* e na localidade dos dados. Além desses, o SciCumulus (OLIVEIRA *et al.*, 2012) e o SciMultaneous (COSTA *et al.*, 2012) são exemplos de dois sistemas que, em conjunto, apoiam o escalonamento adaptativo, o balanceamento de carga (SciCumulus) e a tolerância a falhas (SciMultaneous).

Apesar de a integração SciCumulus e SciMultaneous proporcionar as três características para o paralelismo em SGWfC, essa integração é diferente desta dissertação quanto ao próprio escalonamento adaptativo. Os trabalhos relacionados que possuem escalonamento adaptativo identificam a necessidade de mudanças em relação ao mapeamento das ativações para os NC a partir de uma análise global das atividades do *workflow* e dos NC. Ou seja, a realização de mudanças implica a definição de um novo mapeamento das ativações para todos os NC antes de começar ou continuar a execução do *workflow*. Dessa forma, um novo mapeamento gerado (após uma mudança) pode redistribuir ativações entre NC. Contudo, algumas dessas ativações poderiam permanecer em um mesmo NC, caso esse NC permanecesse disponível após a mudança. Enquanto isso, a proposta desta dissertação realiza o escalonamento adaptativo a partir de uma análise dedicada a cada NC. Essa análise considera que, em cada mudança, o mapeamento de ativações é realizado apenas nos conjuntos de NC que foram afetados pela mudança. Outra característica da proposta desta dissertação é a decisão da distribuição de ativações para NC em tempo de execução, que tem a

finalidade de evitar a redistribuição de muitas ativações em casos de eventuais mudanças (por exemplo, falha irrecuperável de um NC).

Outra diferença entre as soluções em (LEE *et al.*, 2008, PAPUZZO e SPEZZANO, 2011, RASOOLI e DOWN, 2011) e esta dissertação consiste no fato de as soluções existentes terem uma visão orientada a tarefas individualmente, considerando um *workflow* como um conjunto de ativações isoladas. Idealmente para *workflows*, a abordagem adaptativa deveria ser orientada ao fluxo de dados, sendo capaz de antecipar as transformações de dados entre as atividades do *workflow* para poder prever ações futuras para o escalonamento, quando considera-se a recuperação de falhas e o balanceamento de carga. Assim, as transformações de dados envolvidas entre as atividades devem estar monitoradas e passíveis de consulta, para que a abordagem possa ser orientada ao fluxo de dados.

Dessa forma, a proposta desta dissertação consiste no desenvolvimento de uma estratégia de execução paralela adaptável de *workflows* científicos, considerando o escalonamento adaptativo, a tolerância a falhas e o balanceamento de carga. O Demeter foi desenvolvido nesta dissertação para proporcionar essa estratégia. A adaptação dos NC do Demeter visa ao escalonamento adaptativo das ativações para os NC e ao monitoramento dos *workflows* quanto às transformações de dados entre atividades na execução desses *workflows*. A estratégia adaptativa descrita nesta dissertação também considera a análise de disponibilidade dos NC utilizando técnicas P2P, como propostas por HAYEK *et al.* (2008); PAPUZZO e SPEZZANO (2011); e PRICE e TINO (2009). Em relação à tolerância a falhas, considerou-se a identificação e tratamento de falhas nos níveis de software e de hardware, de modo complementar às recuperações do sistema operacional, como discutido por COSTA *et al.* (2012). Somando a isso, o

mapeamento das ativações para os NC foi desenvolvido com o intuito de favorecer o balanceamento de carga em tempo de execução.

O Demeter também tira proveito da abordagem algébrica para *workflows* científicos centrada em dados no modelo relacional proposta por OGASAWARA *et al.* (2011), para consultar os dados e operações algébricas previstas na execução do *workflow*. A análise da representação algébrica relacional dos *workflows* evidencia as dependências entre dados das atividades, as quais são representadas por relações de dados de entrada e saída. Essas transformações de dados são incluídas como tuplas em relações que são preenchidas à medida em que ativações de atividades do *workflow* são executadas pelo SGWfC.

A abordagem algébrica relacional para workflows permite consultar os dados envolvidos no consumo e na produção por cada ativação, mesmo diante de ocorrências de falhas. As ativações também são representadas em relações, indicando o que foi executado, o que está em execução, e o que está pronto para ser executado. Com essa estrutura de representação do fluxo de ativações e de dados, é possível realizar a redistribuição de ativações, durante a execução do *workflow*, de acordo com o escalonamento adaptativo. Do mesmo modo, é possível mudar, com uma operação simples da álgebra relacional, a ordem ou planejamento inicial das ativações. Esses aspectos foram considerados no desenvolvimento da estratégia de execução paralela adaptável do Demeter.

Dois estudos experimentais foram realizados nesta dissertação com o intuito de comparar o desempenho da estratégia adaptativa do Demeter com a estratégia não adaptativa de um SGWfC. Nesse caso, esta dissertação utilizou o Chiron (OGASAWARA *et al.*, 2013). Entretanto, os estudos apresentam diferenças quanto à

sua condução e a análise realizada. O primeiro estudo consistiu na análise de diferentes configurações de um *workflow* quanto à complexidade de processamento das ativações, a fim de analisar como o custo de processamento de cada ativação poderia interferir na comunicação entre os NC e, consequentemente, na eficiência. Já o segundo estudo foi caracterizado pela execução de um *workflow* do domínio de astronomia para comparar o desempenho da estratégia adaptativa com a estratégia não adaptativa em cenários reais de *workflows* científicos. Portanto, o *workflow* de astronomia utilizado foi o Montage (JACOB *et al.*, 2009), que vem sendo adotado como um benchmark *de fato* na literatura de sistemas de *workflows* com paralelismo. O Montage é um *workflow* com aspectos representativos de diversos experimentos científicos reais por apresentar transformações de dados complexas entre as atividades e diferentes comportamentos de desempenho para as mesmas. Os resultados do primeiro estudo evidenciaram um desempenho semelhante ou superior nas diferentes configurações. Enquanto isso, o segundo estudo evidenciou uma eficiência para a estratégia adaptativa superior a 27%, em média, em comparação com a estratégia não adaptativa para configurações típicas de execução paralela em média ou larga escala (utilizando mais de 100 núcleos computacionais ou *cores*).

O restante desta dissertação está organizado da seguinte maneira: No Capítulo 2, discutimos os principais conceitos sobre *workflows* científicos, a execução paralela de *workflows* e as técnicas P2P. Quanto ao primeiro tópico, apresentamos os sistemas de *workflows* científicos centrados em dados e os SGWfC paralelos. Já em relação às técnicas P2P, abordamos suas características, vantagens e desvantagens. Além disso, apresentamos a plataforma JXTA (GRADECKI, 2002), uma das tecnologias existentes para o desenvolvimento de aplicações P2P. No Capítulo 3 são descritos os trabalhos relacionados ao Demeter. Em seguida, no Capítulo 4, a estratégia de execução paralela

adaptável desta dissertação é discutida quanto aos seus objetivos, suas características e seu funcionamento. Além disso, esse capítulo apresenta a implementação do gerente de execução paralela adaptável de *workflows* científicos, proposto nesta dissertação, denominado por Demeter, incluindo também a especificação e o desenvolvimento das arquiteturas AWAPE (do inglês *Architecture for Workflow Adaptive Parallel Execution*) e APON (*Architecture for P2P Overlay Networks*) para o uso pelo Demeter. O Capítulo 5 apresenta os resultados experimentais obtidos com o planejamento de dois estudos analíticos, assim como suas fases de desenvolvimento. Por último, o Capítulo 6 apresenta a conclusão e os trabalhos futuros.

Capítulo 2 – *Workflows* Científicos

Este capítulo tem o objetivo de apresentar os principais conceitos envolvidos em sistemas para a execução paralela de *workflows* científicos, assim como as definições e características da tecnologia P2P. Como os sistemas de *workflows* científicos são centrados em dados, a Seção 2.1 trata especificamente desse conceito. Já a Seção 2.2 aborda os SGWfC, apresentando sua definição e a importância do seu uso em experimentos científicos. Em seguida, a Seção 2.3 descreve o que são SGWfC paralelos, a sua importância em ambientes de PAD e as características desejáveis para o seu funcionamento. Por último, a Seção 2.4 apresenta os principais conceitos da tecnologia P2P e um exemplo de plataforma, conhecida como JXTA, para o desenvolvimento de aplicações P2P.

2.1 Sistemas de *Workflows* Científicos Centrados em Dados

Os *workflows* podem ser modelados por meio de processos, que são definidos pelo encadeamento de atividades e suas dependências (AALST e HEE, 2002). De forma análoga, os *workflows* científicos são definidos a partir de um grafo, em que os nós representam as atividades e as arestas representam as dependências entre elas. Um *workflow* científico possibilita estruturar um experimento científico, em que as atividades realizam o processamento de dados por meio de programas e serviços, enquanto as arestas representam as dependências entre as atividades por meio de dados.

Nesta dissertação considera-se a definição de sistemas de *workflows* científicos centrados em dados, em que nós do grafo representam o processamento dos dados consumidos pelas atividades e as arestas consistem no fluxo de dados entre as atividades (TANNEN *et al.*, 2013). Nesse sentido, as atividades de um *workflow* estão

associadas ao consumo e à produção de dados (descritos como relações de entrada e saída, respectivamente). Diferentemente dos sistemas de *workflows* científicos centrados em dados, os sistemas de *workflows* centrados em tarefas se concentram mais na dependência entre as tarefas, considerando um pequeno volume de dados. Por isso são voltados à modelagem de processos de negócios.

Nesta dissertação utilizou-se da abordagem algébrica relacional para *workflows* científicos centrada em dados proposta por OGASAWARA *et al.* (2011). Chiron (OGASAWARA *et al.*, 2013), um SGWfC paralelo, implementa essa abordagem algébrica. Além dos conceitos de atividade e de dependência, os sistemas de *workflows* científicos centrados em dados, que utilizam essa álgebra relacional, contemplam outros cinco conceitos importantes, sendo os seguintes: relação, tupla, atributo, valor e ativação. Uma relação dessa álgebra é definida por um conjunto de atributos e valores para o consumo ou a produção de dados por uma determinada atividade, sendo que cada tupla representa um conjunto de valores que tais atributos assumem.

Essa álgebra é apresentada resumidamente a seguir, enquanto que mais detalhes podem ser encontrados em OGASAWARA *et al.* (2011). Os dados nessa abordagem algébrica relacional são operandos e as atividades são encapsuladas pelos operadores algébricos. Nesse sentido, os operadores algébricos determinam o comportamento do consumo e da produção de tuplas pelas atividades.

Alguns desses operadores são semelhantes aos existentes na álgebra relacional de banco de dados, enquanto que outros foram criados ou estendidos. As atividades são definidas por expressões algébricas para realizarem o processamento de dados. Logo, uma expressão algébrica é definida a partir de um operador, que recebe como argumentos a invocação do programa a ser executado, os esquemas das relações de

entrada e saída, eventuais operandos adicionais, assim como os dados a serem consumidos pelo operador algébrico especificado. A Figura 1 apresenta a representação de uma expressão algébrica.

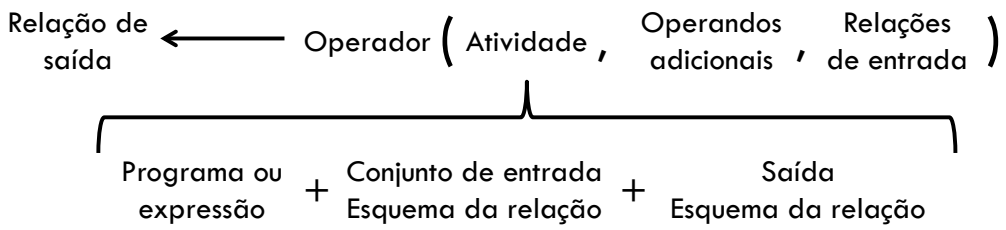

Figura 1. Representação de uma expressão algébrica (OGASAWARA *et al.*, 2011).

A Tabela 1 apresenta como devem ser definidos os operadores algébricos. Vale ressaltar também que a relação do número de tuplas consumidas e produzidas é a principal característica para a definição do tipo de operador algébrico a ser utilizado.

Tabela 1. Operações da álgebra de *workflows* (OGASAWARA *et al.*, 2011)

Operador	Tipo da atividade	Operandos adicionais	Resultado	Relação de tuplas consumidas / produzidas
Map	Programa	Relação	Relação	1:1
SplitMap	Programa	Referência a arquivo, relação	Relação	1:m
Reduce	Programa	Conjunto de atributos de agrupamento, relação	Relação	n:1
Filter	Programa	Relação	Relação	1:(0-1)
SRQuery	Expressão da álgebra relacional	Relação	Relação	n:m
MRQuery	Expressão da álgebra relacional	Conjunto de relações	Relação	n:m

A Figura 2 apresenta um exemplo de *workflow* composto por duas atividades (*A* e *B*) utilizando, respectivamente, as operações de *Map* e de *SR_Query*. Para a atividade *A*, a linha de comando para a execução dessa atividade é caracterizada pela chamada do programa *progA* seguido dos valores de atributos a serem consumidos por essa atividade. Nesse caso seriam os valores dos atributos *LIVRO*, *VENDAS* e *VALOR_UNITARIO*. Somando-se a isso, o programa *progA* é mapeado para a operação

da álgebra conhecida como *Map*, uma vez que para cada tupla processada (cada livro processado), uma tupla de saída é produzida. Portanto, a relação de tuplas consumidas e produzidas possui cardinalidade (1:1).

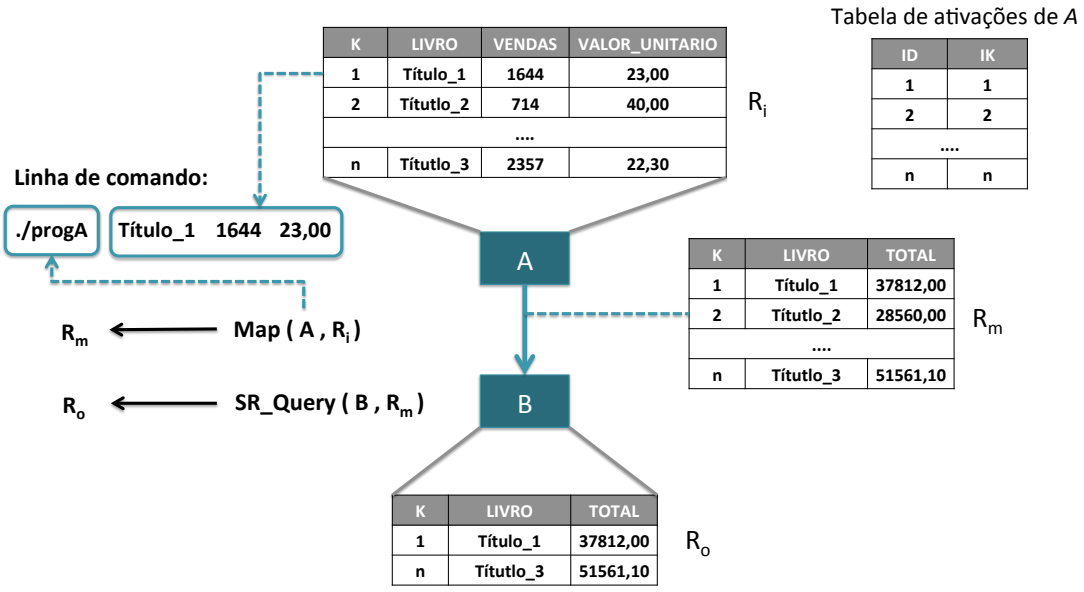

Figura 2. Exemplo de *workflow* como uma expressão algébrica.

Para a especificação da linha de comando a ser processada, os dados de entrada precisam ser instrumentados para uma relação de acordo com o esquema definido pelo *workflow* modelado. Nesse caso, esse esquema é representado por R_i. Do mesmo modo, o término da execução da atividade consiste na produção da relação de saída R_m, que também necessita da instrumentação de seus dados para o processamento da atividade subsequente (atividade *B*).

Além disso, o SGWfC paralelo deve contar também com a tabela de ativações para cada atividade, a fim de direcionar a execução das ativações quanto às tuplas consumidas pela relação R_i, à ativação do programa *progA* e à inserção de tuplas na relação de saída R_m. Desse modo, a tabela de ativações também permite o controle de término da execução de uma atividade, pois ao processar todas as ativações, todas as *n* tuplas de R_i já foram consumidas por alguma ativação.

Por conseguinte, o conceito de ativação foi inspirado pela definição de ativação de tuplas de banco de dados relacional (BOUGANIM *et al.*, 1996). A ativação é a menor unidade de dados autocontida para a execução de uma atividade, sendo composta de um conjunto de valores (dos atributos de uma relação) para realizar o processamento necessário da atividade. Para isso, uma ativação é definida em função de três procedimentos: instrumentação dos dados de entrada (substituições de identificadores de atributos por valores específicos), invocação do programa associado à atividade e a extração dos dados para a relação de saída.

Vale destacar que o mesmo procedimento ocorre para o processamento da atividade *B*, contudo por se tratar de um operador algébrico de *SR_Query*, a linha de comando consiste em uma expressão da álgebra relacional (no caso do Chiron, a invocação desse operador consiste em uma consulta SQL), mais detalhes em (OGASAWARA *et al.*, 2013). Dessa forma, apenas uma ativação é criada, a fim de consumir todas as tuplas da relação R_m. Além disso, dependendo do modelo de escalonamento e do modelo de execução, algumas ativações da atividade *B* podem ser ativadas em paralelo à atividade *A* em *pipeline*.

2.2 Sistema de Gerência de *Workflows* Científicos (SGWfC)

Sistemas de Gerência de *Workflows* Científicos (SGWfC) gerenciam a composição, execução, monitoramento e análise de resultados de *workflows* científicos. O uso dos SGWfC permite a especificação dos programas e serviços em relação ao experimento científico a ser validado. O Kepler (ALTINTAS *et al.*, 2004) e o VisTrails (CALLAHAN *et al.*, 2006) são exemplos de SGWfC.

O Kepler (ALTINTAS *et al.*, 2004) é um SGWfC de código aberto, com o objetivo de permitir o desenvolvimento de *workflows* nas mais diversas áreas da ciência. Suas características estão relacionadas ao fácil uso da interface, por meio de ações de arraste de ícones para permitir a especificação de *workflows*. No SGWfC Kepler, as atividades são definidas como atores, sendo que já existem diferentes tipos de atividades primitivas para auxiliar a modelagem. Quanto aos relacionamentos ou dependência entre as atividades, os mesmos são representados pelo elemento diretor, que define o modelo de execução do *workflow*, respeitando as dependências das relações e controla o momento da execução de uma atividade.

O VisTrails é outra proposta com o principal objetivo de favorecer a análise visual do resultado de *workflows*. Para tanto, o VisTrails captura dados de proveniência dos *workflows*, além de possuir algumas funcionalidades para o controle de versões durante a composição dos *workflows* e um mecanismo para controle de execuções anteriores. Somando-se a isso, uma linguagem, conhecida como *VisTrails Query Language* (ANDERSON *et al.*, 2007) foi criada para proporcionar a análise dos resultados da execução dos *workflows* por meio de consultas. Outro diferencial do Vistrails (CALLAHAN *et al.*, 2006) é a presença de uma área de visualização (*spreadsheet*), que possibilita verificar e comparar os resultados das execuções de diferentes versões de um mesmo *workflow*.

Apesar das diferentes funcionalidades desses sistemas, os *workflows* científicos em larga escala tem necessitado do apoio ao paralelismo, a fim de processar o volume grande e crescente de dados envolvidos na execução dos *workflows*. Nesse sentido, outros SGWfC foram propostos para apoiar a execução paralela de *workflows* científicos, como o Pegasus (LEE *et al.*, 2008), o Swift/T (WOZNIAK *et al.*, 2013) e o Chiron (OGASAWARA *et al.*, 2013).

O Pegasus é um SGWfC para a execução em ambientes de PAD (*clusters*, grades computacionais e nuvens), sem a necessidade de o usuário (nesse caso, o cientista) se preocupar com o tipo de ambiente que deseja executar. O Pegasus realiza o mapeamento do *workflow* com seus dados de entrada em *workflows* ditos executáveis. Os *workflows* executáveis são definidos como *workflows* que já apresentam as configurações de ambiente necessárias para a sua execução, sendo que a escolha do ambiente é feita pelo cientista. Além disso, esse mapeamento automático implica algumas estruturações no *workflow*, o que favorece melhorias no desempenho de execução (DEELMAN *et al.*, 2007). Outras características do Pegasus são o monitoramento por meio do Stampede (GUNTER *et al.*, 2011) e a recuperação de falhas, realizando a redistribuição de ativações. No caso da recuperação de falhas, uma análise completa do *workflow* é realizada em função, por exemplo, de um erro na modelagem dos dados de entrada do *workflow*. Entretanto, a alocação de NC não é adaptativa na versão original do Pegasus. Para contornar essa limitação, o Pegasus implementa sua máquina de execução tendo como base o Condor (COUVARES *et al.*, 2007) e o DAGMan (*Directed Acyclic Graph Manager*) (CONDOR TEAM, 2005). O Condor é um escalonador orientado a tarefas que possui uma interface para submissão e execução de tarefas em ambientes de PAD, enquanto que o DAGMan automatiza a submissão e a gerência de *workflows* científicos utilizando o Condor. Tais tecnologias permitem a adaptação da execução dos *workflows*, contudo tais modificações implicam a modificação do plano de alocação das ativações definido antes do início do *workflow*, o que pode ser custoso do ponto de vista de identificação de ativações já executadas ou mesmo da redistribuição de ativações. Outra característica do Pegasus é a captura de proveniência em tempo de execução.

O Swift/T é outro SGWfC que permite a paralelização da execução de *workflows* científicos em ambientes de PAD. Para isso, o Swift/T apresenta dois componentes para favorecer a execução paralela de *workflows*, sendo esses conhecidos como compilador Swift-Turbine e o Turbine escalável em tempo de execução. Esses componentes também permitem otimizações durante a execução dos *workflows*. Além disso, a principal vantagem do Swift/T são os algoritmos otimizados para permitir a execução paralela. Já o modelo de execução do Swift/T caracteriza-se pela partição em ativações e a distribuição dessas ativações menores a partir da interpretação de um programa modelado por meio do Swift Script. Além dessa etapa de interpretação em uma linguagem de mais alto nível, o Turbine apresenta mais três componentes, sendo os seguintes: serviços de acesso aos dados e de balanceamento de carga; uma máquina lógica para análise da dependência de dados; e algumas funcionalidades para execução de programas externos e operações sobre dados. Swift/T possui resultados de escalabilidade acima de 30.000 cores. Contudo, essa abordagem não possui um apoio ao monitoramento de dados de execução ou de proveniência durante a execução, o que dificulta a depuração de experimentos científicos muitos complexos e em larga escala.

Além das propostas apresentadas, o Chiron é um SGWfC paralelo de *workflows* científicos baseada em uma álgebra relacional de *workflows* centrada em dados (a mesma apresentada na Seção 2.1). Por meio dessa álgebra relacional, o Chiron permite a execução de diferentes operadores algébricos relacionais, incluindo operadores estendidos por essa álgebra relacional (DIAS *et al.*, 2013). Outras características desse SGWfC paralelo são os modelos de execução e as operações algébricas relacionais que permitem a otimização dos *workflows* em tempo de execução. A principal característica do Chiron é integrar na mesma base de dados os dados de proveniência com os dados de execução do workflows por meio do modelo algébrico. Essa

característica proporciona flexibilidade no modelo de execução, adaptação na definição do workflow e monitoramento da execução com a semântica do domínio da aplicação.

No que diz respeito à sua arquitetura, o Chiron foi desenvolvido para apoiar a execução de *workflows* em ambientes de *clusters*, considerando arquiteturas com disco compartilhado. Tais arquiteturas representam o estado da arte em computação científica de PAD, por exemplo 90% das máquinas do Top 500 possuem arquitetura paralela de disco compartilhado. Entretanto, o Chiron apresenta uma abordagem em que a gerência de execução é centralizada (Figura 3), em que o *Nó 0* é o responsável pela distribuição das ativações para os outros nós existentes por meio do processador de *workflow* (conhecido como PW). O PW tem a função de analisar a estrutura do *workflow*, considerando, por exemplo, as dependências de dados entre as atividades, antes de realizar a distribuição das ativações. Além disso, cada nó apresenta apenas um escalonador de ativação (conhecido como EA) e um ou mais processadores de ativação (conhecido como PA). O EA é responsável por receber as ativações do PW e distribuir para os PA existentes nesse nó. Por último, cada PA realiza a execução da ativação recebida e envia os dados de proveniência produzidos para o EA, que enviará em seguida para o PW realizar as operações na base de proveniência do Chiron.

Portanto, o Chiron coleta a proveniência distribuída pelos seus nós de execução, apesar de a sua gerência com a base de dados ser realizada apenas por meio do nó central, ou seja, o *Nó 0*. Assim, todos os dados de proveniência gerados pela execução das ativações nos outros nós precisam ser enviados para o nó central realizar a atualização na base de dados. O sistema de banco de dados utilizado pelo Chiron é o PostgreSQL (POSTGRESQL, 2014).

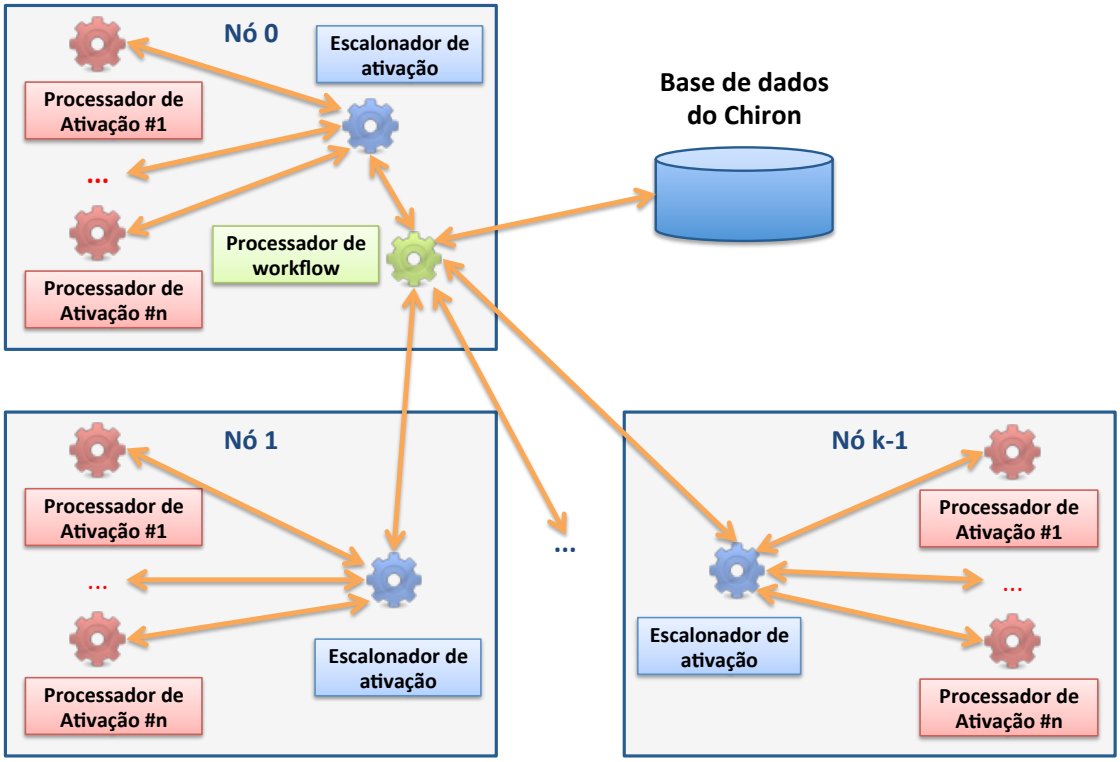

Figura 3. Arquitetura do Chiron. Adaptada de OGASAWARA *et al.* (2013).

Além disso, o Chiron realiza o escalonamento das ativações a partir da definição de duas estratégias ou algoritmos: estratégia do fluxo de dados (do inglês *dataflow strategy*) e estratégia de despacho (do inglês *dispatching strategy*). A estratégia do fluxo de dados determina como as ativações são escalonadas pelos diferentes NC de acordo com as dependências de dados, podendo ser de dois tipos: FAF (do inglês *First Activity First*) e FTF (do inglês *First Tuple First*). A estratégia FAF realiza a distribuição das ativações por atividade, de modo que as atividades pendentes (que já tenham todas as ativações das suas atividades dependentes executadas) podem realizar o envio de suas ativações para os NC disponíveis.

Já a estratégia FTF apresenta uma abordagem baseada no conceito de tupla (referente à relação de entrada da atividade). Como exemplo, consideraremos um *workflow* com duas atividades (*Act1* e *Act2*), sendo a atividade *Act2* dependente da atividade *Act1*. Na estratégia FTF, o escalonamento não precisa esperar que todas as

ativações da atividade *Act1* sejam executadas para executar uma ativação da atividade *Act2*, desde que a tupla utilizada nessa ativação já tenha sua respectiva ativação dependente processada na atividade *Act1*. Semelhante à estratégia FTF, o Swift/T permite o escalonamento de *workflows* utilizando uma abordagem conhecida como *pipelining*.

Já a estratégia de despacho pode apresentar também dois tipos de comportamento: dinâmico e estático. O comportamento dinâmico é baseado no envio de uma ativação para os NC de acordo com os pedidos de novas ativações pelos nós ociosos. Sendo assim, o número total de pedidos de ativação é igual ao número de ativações no término do *workflow*. Por outro lado, a estratégia estática é responsável pelo envio de um conjunto pré-definido de ativações para os NC a cada envio. Dessa forma, o número de pedidos está diretamente relacionado ao tamanho dos conjuntos de ativações.

2.3 Escalonamento em Sistemas de Gerência de *Workflows* Científicos

SGWfC paralelos precisam distribuir as atividades dos *workflows* por meio da divisão de cada atividade em partes menores, definidas como ativações. Nesse sentido, o escalonamento de *workflows* pode ser definido pelo mapeamento de todas as ativações constituintes do *workflow* para os NC disponíveis no ambiente de PAD. As decisões relacionadas às ativações a serem distribuídas e aos NC que receberão tais ativações são associadas à estratégia de escalonamento. Tal estratégia pode ser eficiente em *clusters* (BARGA *et al.*, 2008), baseada em dois objetivos para a distribuição em ambientes heterogêneos (BOERES *et al.*, 2011) ou mesmo adaptativo, que permita a adição e a remoção de NC em tempo de execução (OLIVEIRA *et al.*,

2012). Além disso, os SGWfC paralelos precisam considerar as dependências de dados e de controle definidas na modelagem do *workflow*.

As estratégias de escalonamento também podem ser classificadas das seguintes formas: estática e adaptativa. Primeiramente, o escalonamento estático é caracterizado por determinar o mapeamento das ativações do *workflow* para os NC antes do início de sua execução. Sendo assim, essa estratégia não permite alterações no mapeamento das ativações do *workflow* para os NC durante a sua execução (BUX e LESER, 2013).

Diferentemente dessa estratégia, o escalonamento adaptativo consiste no mapeamento das ativações para os NC em tempo de execução (BUX e LESER, 2013). Desse modo, o mapeamento de ativações não precisa ser definido antes de iniciar a execução do *workflows*, pois as decisões quanto à distribuição destas ativações são realizadas de acordo com o progresso da execução do *workflow*. Desse modo, mudanças em tempo de execução quanto aos NC e a eventuais falhas podem ser tratados. Mais especificamente, esse escalonamento não precisa estabelecer um novo mapeamento das ativações para os NC antes de continuar com a execução do *workflow*, pois apenas os NC indisponíveis após a mudança deverão ter suas ativações redistribuídas. Consequentemente, tal solução é considerada a mais apropriada para a execução de *workflows* científicos em larga escala, em função de sua capacidade em lidar com as mudanças em tempo de execução e o desbalanceamento de carga.

Além disso, o objetivo de um SGWfC paralelo não pode estar restrito ao apoio do escalonamento. Essa proposta tem que promover uma forma eficiente e escalável de realizar o escalonamento. Para isso, o SGWfC paralelo precisa distribuir as ativações da melhor forma possível, a fim de evitar que NC fiquem ociosos ou mesmo que sejam utilizados de uma forma menos eficiente. Já em relação à escalabilidade, o SGWfC

paralelo deve manter o desempenho em relação ao tempo de execução, mesmo adicionando-se mais NC. Contudo, o número de NC máximos alocados deve ser analisado de acordo com o problema em questão. Ou seja, existem aplicações em que não é possível paralelizar com mais de uma quantidade de processadores, uma vez que não há mais ativações a serem distribuídas, ocasionando NC ociosos (FOSTER *et al.*, 2008).

Outras características desejáveis para os SGWfC paralelos são a adaptabilidade, a tolerância a falhas e o balanceamento de carga. Em relação à adaptabilidade, essa característica permite a adição e a remoção dos NC em tempo de execução, favorecendo, por exemplo, o uso de um algoritmo adaptativo por uma dada máquina de execução paralela. A adaptabilidade também está associada à continuação ou à interrupção no uso de um NC, em virtude das ocorrências de falhas no mesmo. Ao mesmo tempo, a adaptabilidade considera as mudanças necessárias no mapeamento das ativações para os NC em tempo de execução.

Já a ocorrência de falhas tornou-se uma certeza com o significativo aumento no número de NC e no volume de dados processados nos ambientes de PAD. Portanto, observa-se a importância do desenvolvimento de SGWfC paralelos capazes de continuar a execução dos *workflows* mesmo diante de eventuais falhas nos NC. De acordo com o nível de tolerância a falhas (por exemplo, falhas relacionadas a software ou a hardware), esses SGWfC podem ser definidos como tolerantes a falhas (HANMER, 2007, ISERMANN, 2005). As ocorrências de falhas também podem implicar a redistribuição de ativações, a partir de mudanças no mapeamento das ativações para os NC. Como exemplo, pode-se citar as ocorrências de falhas irrecuperáveis (EDELWEISS e NICOLAO, 1998) em um NC.

O balanceamento de carga é outra característica desejável para os SGWfC paralelos, uma vez que as aplicações científicas tendem a processar muitas ativações com diferentes custos de processamento (também chamados de cargas). No cenário de ambientes de PAD, tais custos ao serem processados em paralelo podem favorecer um mapeamento irregular ou desbalanceado das ativações para os NC disponíveis. Logo, observa-se a importância dos SGWfC paralelos de identificarem e tratarem o desbalanceamento de carga, a fim de reduzir o tempo total de execução dos *workflows* científicos. Vale ressaltar também que as mudanças no mapeamento das ativações para os NC são necessárias para prover o balanceamento de carga em tempo de execução.

Vale ressaltar também a importância da coleta de proveniência, uma vez que permite o registro sólido da execução do *workflow* nos diferentes NC, colaborando com a reprodutibilidade do experimento, a análise de resultados pelos usuários, ou mesmo para a análise de desempenho e de erros de um SGWfC paralelo presente em um ambiente de PAD. Dessa forma, os cientistas também podem consultar a base de proveniência com o objetivo de analisar as transformações de dados envolvidas entre as atividades, evidenciando o poder analítico desses SGWfC paralelos ao suportarem tais propriedades.

Dentre as características desejáveis, o Chiron não apresenta o apoio a nenhuma dessas características (adaptabilidade, tolerância a falhas e balanceamento de carga). Nesse sentido, esta dissertação tem o objetivo de propor o Demeter, um gerente de execução paralela adaptável para uma álgebra relacional de *workflows* científicos centrada em dados que estende a versão atual do Chiron. Além disso, duas arquiteturas (AWAPE e APON) foram desenvolvidas e utilizadas pelo Demeter para apoiar a adaptabilidade, a tolerância a falhas e o balanceamento de carga. Outra característica dessa máquina é o uso de técnicas P2P para a sobreposição de uma rede física

(implementadas na arquitetura APON), por meio da abordagem centralizada. As características, o funcionamento e as vantagens dessa tecnologia são abordados na próxima seção.

2.4 Tecnologia P2P

2.4.1 Definição e características

A tecnologia P2P permite o estabelecimento de arquiteturas distribuídas e descentralizadas. Para isso, tal tecnologia permite o estabelecimento de redes para mapear os recursos físicos disponíveis em NC, sendo que cada nó de uma rede P2P é conhecido como *peer* e pode apresentar o comportamento tanto de consumidor, como de fornecedor de recursos simultaneamente (HAYEK *et al.*, 2008). A Figura 4 apresenta as duas abordagens descritas.

As redes P2P normalmente implementam uma rede virtual sobreposta sobre os recursos físicos, com o propósito de favorecer o mapeamento desses recursos em *peers*, assim como permitir a comunicação entre esses *peers*. Dessa forma, os *peers* são tratados como nós virtuais a partir dessa sobreposição, tornando-os independentes da topologia da rede física para os sistemas P2P. Ao mesmo tempo, a sobreposição permite a comunicação entre os *peers* da rede por meio de links lógicos com a camada de rede física, utilizando o protocolo TCP/IP. As sobreposições também permitem a indexação dos NC mapeados e o uso de um mecanismo para a descoberta de *peers* existentes nessa rede P2P sobreposta.

Outra característica importante está relacionada às abordagens para estabelecer uma rede P2P sobreposta. Como mencionado anteriormente, existe uma abordagem em que os *peers* são mapeados para apenas um NC, da mesma forma que existe uma outra

abordagem baseada no mapeamento de um *peer* da rede P2P como um conjunto de NC. O trabalho proposto por (RAHMAN *et al.*, 2010) apresenta uma rede P2P sobreposta baseada em grades geograficamente dispersas. Nesse trabalho, cada ambiente de grades (com vários NC) é tratado como apenas um nó na rede P2P.

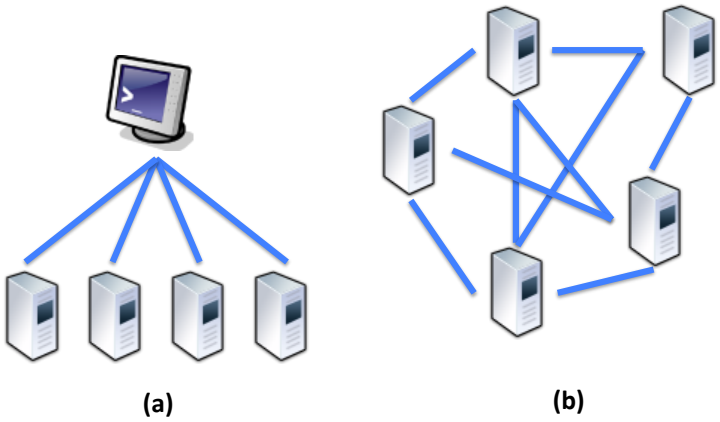

Figura 4. Abordagem (a) cliente-servidor e (b) descentralizada.

A comunicação entre nós em uma rede P2P pode ser definida a partir de três mecanismos: publicação de anúncio, descoberta de anúncio e troca de mensagens por *pipes*. O termo anúncio é dedicado a um documento, normalmente no formato XML, que é compartilhado entre os nós da rede para proporcionar a identificação dos nós presentes nessa rede (anúncio de nós), de grupos de nós definidos na rede (anúncio de grupos) e de canais de comunicação para troca de mensagens entre nós (anúncio de *pipes*). Já o termo *pipe* é dado a um túnel ou canal de comunicação que é estabelecido entre dois nós da rede para a troca de mensagens.

A publicação de anúncios é baseada no compartilhamento desse documento pelos nós da rede, por meio de mensagens *multicast*. Já o mecanismo de descoberta de anúncios baseia-se na captura dos anúncios pelos nós e o seu processamento de acordo com o anúncio obtido e o controle desenvolvido para o sistema. Por último, a troca de mensagens por *pipes* é estabelecida pelo recebimento do identificador do *pipe* pelo

anúncio de mesmo tipo e o uso desse identificador para manter um canal de comunicação com outro nó. A partir do momento que esse canal é definido, a mensagem pode ser enviada de um nó para outro.

As redes P2P também podem ser classificadas quanto a sua topologia em três tipos: desestruturadas, estruturadas e híbridas. As redes desestruturadas são conhecidas por não apresentarem uma topologia definida para a sobreposição da rede P2P, permitindo a inserção de nós sem a determinação de dependências hierárquicas com outros nós. O Gnutella (RIPEANU, 2001) e o Kazaa (KAZAA, 2011) são exemplos de redes desestruturadas. Além disso, pelo fato da rede desestruturada não respeitar nenhuma topologia, a mesma apresenta como vantagem o custo de construção da rede e uma robustez em taxas altas de ocorrência de *churns*. O evento de *churns* é caracterizado pela entrada e saída frequente de nós na rede. Nesta dissertação, chamamos de volatilidade de nós os eventos de *churns* das redes P2P.

Contudo, as redes desestruturadas podem apresentar dificuldade em consultas de dados na rede em função da falta de uma estrutura global dos nós, uma vez que a consulta precisa ser realizada em cada nó da rede até encontrar o dado desejado. Somando-se a isso, a busca implica em um custo de tráfego, de processamento e de uso de memória maior, ao mesmo tempo em que não garante que o dado seja encontrado.

Por outro lado, as redes estruturadas permitem um controle dos dados a partir de nós com nível hierárquico maior, facilitando a busca (SHEN *et al.*, 2010). As redes estruturadas são organizadas de acordo com uma topologia específica e garantem buscas eficientes na rede por arquivos e recursos a partir do uso de protocolos de comunicação. Apesar do melhor desempenho para obtenção de arquivos e recursos, a rede estruturada apresenta menor robustez para ocorrências de volatilidade, uma vez

que ingressos e saídas frequentes de um nó implicam na reorganização hierárquica dessa topologia.

Já a topologia híbrida é caracterizada por aliar tanto características da topologia desestruturada como da estruturada. Para isso, uma rede híbrida define um nó central responsável por fornecer o roteamento para a comunicação entre dois nós da rede. Assim, o nó central apresenta o comportamento de um mediador na comunicação entre os nós da rede. Um exemplo de rede híbrida é o Spotify (KREITZ e NIEMELA, 2010), um sistema de reprodução de música por *streaming*. Como resultado desta dissertação, um estudo sobre diferentes topologias foi realizado para a execução de *workflows* científicos em ambientes de PAD (SILVA *et al.*, 2013).

Além disso, as técnicas P2P permitem a tolerância a falhas e a adaptabilidade da rede. A tolerância a falhas pode ser gerenciada pelo mecanismo de anúncio, que confirmam em cada intervalo de tempo, a permanência do nó na rede P2P. Dependendo da falha ocorrida em tempo de execução, o envio desse anúncio pode ser comprometido, evidenciando um comportamento anormal desse nó. Ao mesmo tempo, a ocorrência de outros tipos de falhas nesse nó que não interferem no funcionamento da abordagem P2P, mas comprometem a execução de experimentos, podem ser notificados por meio dos próprios anúncios. Dessa forma, apesar de estar presente na rede P2P, tal nó não seria utilizado para processamento até a resolução das dificuldades notificadas.

Do mesmo modo, a adaptabilidade pode ser garantida pelos anúncios, uma vez que o ingresso de um nó pode ser identificado pela divulgação de um anúncio desse novo nó na rede. No caso da saída de um nó da rede, o mesmo é identificado pela falta do seu anúncio durante um determinado período de tempo, semelhante ao

comportamento de tolerância a falhas. Esse mecanismo de verificação de anúncios é baseado no algoritmo *keep-alive* (PRICE e TINO, 2009).

Outro caso possível consiste na falha de um nó de controle (ou de coordenação) em uma determinada topologia, como o nó central na topologia híbrida. A queda desse nó, por exemplo, desestrutura a topologia, exigindo um mecanismo para definir outro nó de controle e, consequentemente, para retomar a execução. Logo, os algoritmos de eleição de líder (HAYEK *et al.*, 2008) são conhecidos por permitir a reestruturação das redes P2P frente a ocorrência de falhas nesses nós de controle, que também são conhecidos como nós centrais (KIM *et al.*, 1995). A Figura 5 ilustra um exemplo de falhas que podem motivar a eleição de um novo líder.

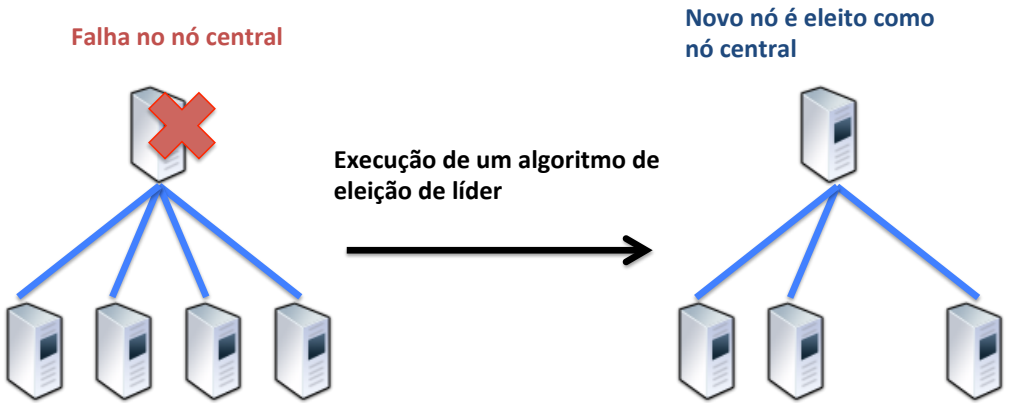

Figura 5. Algoritmo de eleição de líder.

2.4.2 Plataforma JXTA

A plataforma JXTA foi desenvolvida pela Sun Microsystems em 2001, com o intuito de permitir a implementação de aplicações P2P (GRADECKI, 2002). O seu funcionamento é baseado na sobreposição da topologia física da rede, por meio de uma rede virtual JXTA, independentemente da linguagem de programação, de plataforma ou da própria tecnologia de rede (Figura 6). Por meio dessa sobreposição, os nós podem efetuar a troca de mensagens a partir de arquivos no formato XML, desde que

estejam conectados nessa rede virtual. Além disso, a JXTA garante a ubiquidade, podendo qualquer dispositivo se conectar na rede P2P sobreposta. As implementações disponíveis atualmente são destinadas às linguagens de programação Java SE, Java ME, C/C++ e C#. Vale ressaltar também que algumas dessas versões não estão completamente implementadas.

Somando-se a isso, a JXTA pode ser representada pelos componentes da Figura 7. No nível mais baixo, tem-se os NC disponíveis em uma ou mais redes físicas para participarem dessa rede P2P sobreposta. No núcleo do JXTA, define-se as propriedades de segurança, assim como características específicas a serem utilizadas pelos nós, como o seu identificador, os anúncios a serem publicados e descobertos, os *pipes* para proporcionarem a troca de mensagens entre nós e um mecanismo próprio para controle de nós presentes na rede P2P. Já no nível de serviços, temos a sobreposição imposta pela JXTA e os protocolos para o controle da rede. Por último, apresenta-se a camada de aplicação, em que os usuários dessa plataforma utilizam os componentes disponibilizados por meio dos serviços da JXTA para criar e gerenciar as suas redes P2P (BAROLLI e XHAFA, 2011).

O controle da rede P2P é inicializado pelas entidades definidas no núcleo do JXTA. Primeiramente, os nós ou *peers* são mapeados na rede sobreposta de acordo com a rede física, definindo-se um identificador único de 128 bits, sendo que esse não tem garantia de ser globalmente único. Tais nós são entidades capazes de compreender todos os protocolos existentes. Já os anúncios são responsáveis por descrever em documentos XML a existência de um nó, *pipe*, grupo de nós ou serviço. A entidade grupo de nós (do inglês *Peer Group*) define um conjunto de nós da rede P2P, em que o *World Peer Group* corresponde ao grupo com todos os nós da rede global (GRADECKI, 2002).

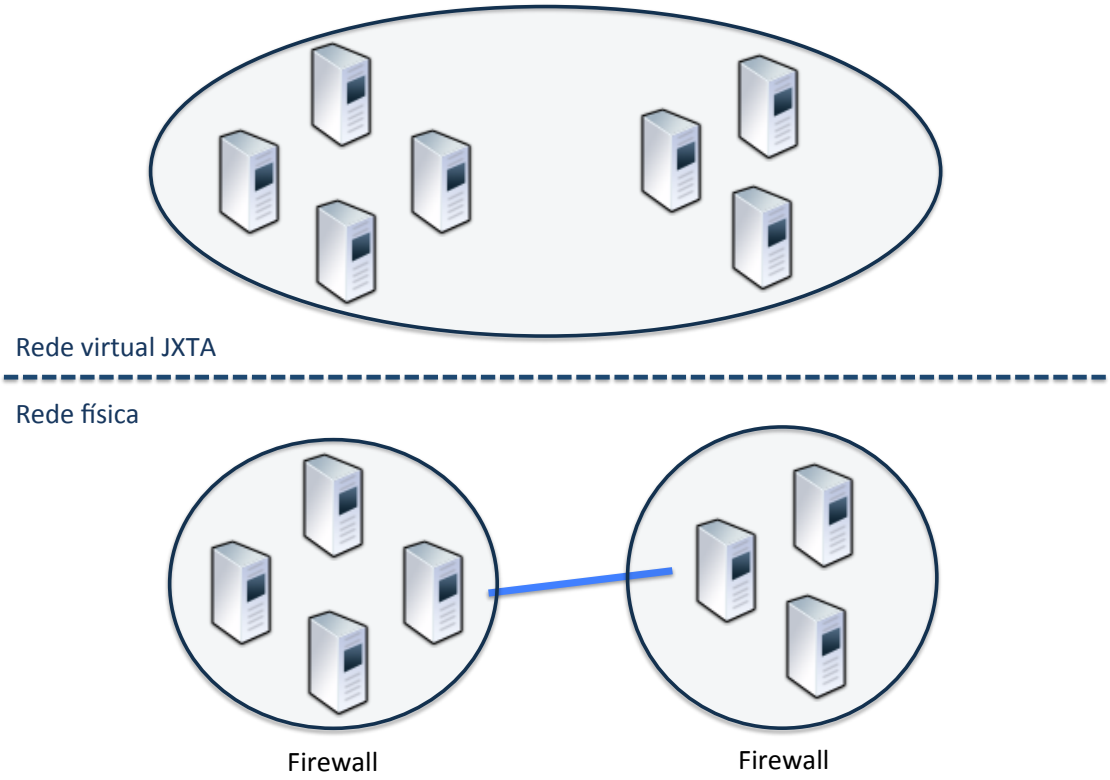

Figura 6. Sobreposição de rede pela plataforma JXTA.

Já os *pipes* são canais de comunicação que permitem o envio e o recebimento de mensagens. Os *pipes* também podem ser de dois tipos: *pipe* unidirecional ou *pipe* de propagação. O *pipe* unidirecional conecta apenas dois nós, enquanto que o *pipe* de propagação é capaz de conectar múltiplos nós. A unidade básica de comunicação em um *pipe* é a mensagem, sendo composta de um envelope e um corpo. As mensagens podem apresentar diferentes comportamentos, como por exemplo um comportamento assíncrono e sem confirmação (não confiável). A Figura 8 apresenta o funcionamento da troca de mensagens utilizando *pipe* pelo JXTA. Essa figura destaca a necessidade de definir uma porta de saída (*pipe* de saída) e outra de entrada (*pipe* de entrada) para que a mensagem possa ser transmitida entre dois nós.

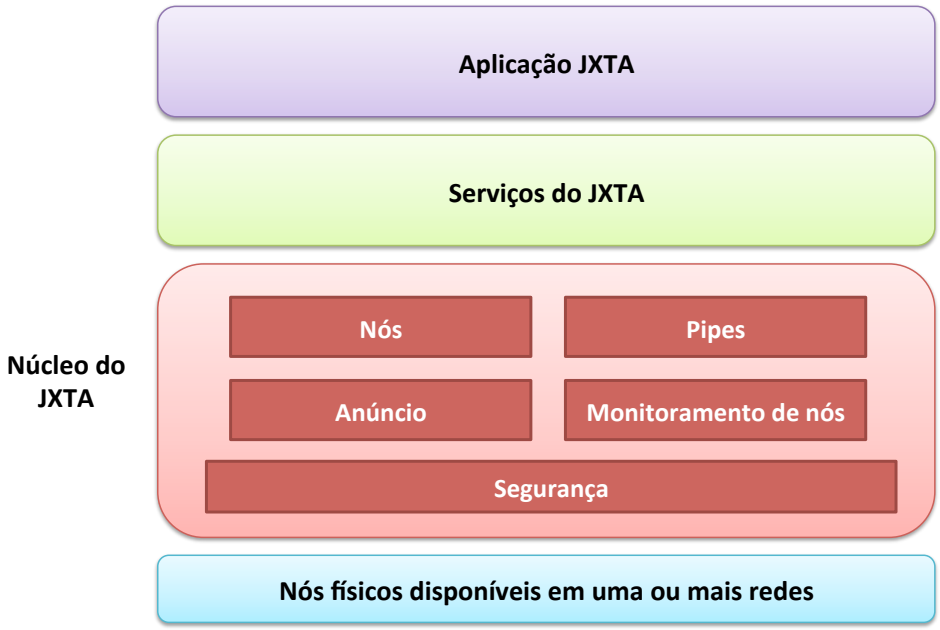

Figura 7. Representação em camadas da plataforma JXTA.

Os serviços da JXTA são realizados a partir da sobreposição da rede e dos diferentes protocolos para a gerência dos nós e da comunicação. Os protocolos utilizados pela JXTA são os seguintes: *Peer Discovery*, *Peer Resolver*, *Peer Information*, *Peer Membership*, *Pipe Binding* e *Endpoint Routing*. O *Peer Discovery* é responsável pela descoberta de anúncio de outros nós, independentemente dos tipos (anúncios de nós, grupo de nós e *pipes*). Outro ponto é que um anúncio pode apresentar uma solicitação de *pipe* por um nó da rede. Já o *Peer Resolver* permite a requisição e o recebimento de consultas genéricas para um nó específico da rede ou para um conjunto de nós dentro do mesmo grupo.

Já o protocolo *Peer Information* lida com um mecanismo que obtém dados sobre o estado de um nó na rede P2P. Vale ressaltar que esse protocolo não garante que as suas mensagens sejam recebidas pelo(s) outro(s) nó(s), assim como utiliza um canal de comunicação sem segurança. O *Peer Membership* proporciona a atualização de nós na rede P2P a partir de pedidos de adesão, de atualização e de cancelamento. Já o mecanismo de *Pipe Binding* é o responsável por utilizar um anúncio de *pipe* (*Pipe*

Advertisement) para permitir a troca de mensagens pelos nós desse canal de comunicação, por meio das portas de entrada e de saída. O *Pipe Binding* também é responsável pela determinação do roteamento da mensagem. Por último, o *Endpoint Routing* é um mecanismo complementar, que permite consultas a um nó roteador sobre possíveis roteamentos para o envio de uma mensagem.

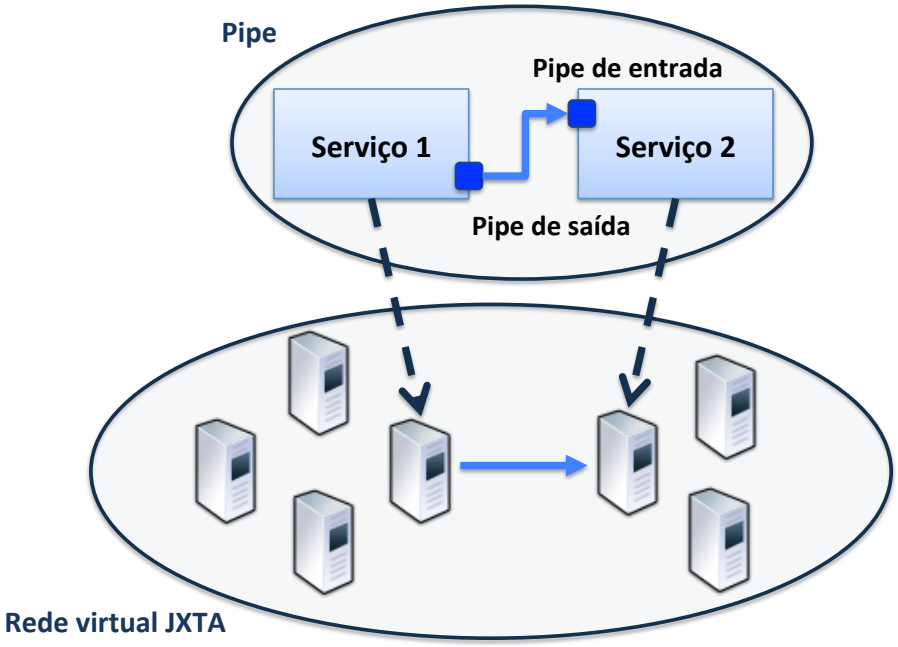

Figura 8. Funcionamento da entidade de *pipe* unidirecional.

A última camada da Figura 7 apresenta a aplicação JXTA, que corresponde ao desenvolvimento de um sistema pelo usuário desse arcabouço. No caso desta dissertação, a arquitetura APON utilizou o JXTA para o desenvolvimento das técnicas P2P discutidas nessa seção. Portanto, tal arquitetura é representada como uma aplicação JXTA.

Capítulo 3 – Trabalhos Relacionados

Semelhante à proposta desta dissertação, que contempla o desenvolvimento de uma estratégia para a execução paralela adaptável de *workflows* científicos através do Demeter (e suas arquiteturas AWAPE e APON), outros trabalhos também utilizam os conceitos de adaptabilidade, de recuperação de falhas e de balanceamento de carga para o escalonamento de *workflows* científicos. Nesse sentido, o objetivo desse capítulo é descrever alguns desses trabalhos, destacando a contribuição desta dissertação em relação às abordagens existentes.

Primeiramente, o trabalho proposto por LEE *et al.* (2011) apresenta uma extensão para o Pegasus, a fim de proporcionar a execução adaptativa de *workflows* científicos. A solução desenvolvida nessa extensão do Pegasus baseia-se na revisão das tomadas de decisão para distribuição de ativações durante a execução do *workflow*, considerando adaptações nos NC disponíveis. Dessa forma, uma vez que seja identificada a necessidade de mudança no escalonamento do *workflow*, a alocação inicial das ativações definida pelo compilador de *workflow* abstrato precisa ser alterada. Para isso, funções de utilidade baseadas no lucro financeiro e no tempo de resposta são empregadas para avaliar as adaptações nos NC disponíveis. A política utilizada para realizar tais adaptações é conhecida como política de função de utilidade, do inglês *utility function policy* (KEPHART e DAS, 2007).

Contudo, essa extensão para o Pegasus não considera a recuperação de falhas e o escalonamento adaptativo realiza uma análise global quanto ao mapeamento das ativações para os NC, o que pode implicar em um alto custo de adaptação da alocação de ativações (definida antes do início da execução do *workflow*) nos NC, pois o escalonador precisaria ser atualizado com essas novas configurações (de alocação de

ativações ou de mudanças nos NC). Diferentemente da abordagem de LEE *et al.* (2011), a estratégia adaptativa implementada no Demeter é baseada em um escalonamento adaptativo que analisa eventuais mudanças isoladamente para cada NC. Tal comportamento adaptativo está relacionado ao fato de que uma ativação ou conjunto de ativações é alocado aos NC durante a execução do *workflow*. Logo, as principais vantagens estão relacionadas ao balanceamento de cargas das ativações, principalmente em cenários de larga escala, e à capacidade de adaptações em tempo de execução, que requer apenas redistribuição de ativações alocadas em NC não disponíveis após as mudanças.

Vale lembrar que, para *workflows* baseados em álgebra relacional, o termo ativação contém a menor unidade de dado necessária para executar uma atividade e apresenta o mesmo funcionamento que o termo ativação em outros sistemas. Somando-se a isso, essa distribuição de ativações do Demeter em tempo de execução é conhecida pela política de ação, do inglês *action policy* (KEPHART e DAS, 2007). Ao mesmo tempo, os desenvolvedores dessa extensão do Pegasus apresentam outros trabalhos relacionados (LEE *et al.*, 2008, PATON *et al.*, 2009) na mesma linha de pesquisa que a descrita nesses parágrafos.

Um algoritmo adaptativo para o Hadoop (RASOOLI e DOWN, 2011) é outro trabalho relacionado que permite a identificação de diferenças no desempenho de NC heterogêneos em tempo de execução e a redução do custo de comunicação pela avaliação dos dados. Apesar desse algoritmo do Hadoop apresentar um escalonamento adaptativo semelhante ao Pegasus, o mesmo permite o balanceamento de carga, analisando inclusive a localidade do dado. Para realizar a distribuição das ativações em tempo de execução, a taxa estimada de chegada de ativações e o tempo médio para a execução das ativações são também considerados para determinar se há a necessidade

de redistribuição de certas ativações. A tomada de decisão pelo algoritmo adaptativo é baseada em um problema de otimização que utiliza o tempo de execução e o custo de transferência de dados através de métricas de avaliação, como a localidade das ativações, a equidade (do inglês *fairness*) e a menor insatisfação de compartilhamento (do inglês, *minimum share dissatisfaction*). Outro ponto é que o algoritmo de RASOOLI e DOWN (2011) considera apenas o problema da adaptabilidade e do balanceamento de carga, não abordando a recuperação de falhas, como proposto pelo Demeter.

Já o Swift/T (WOZNIAK *et al.*, 2013) é uma máquina de execução paralela de *workflows* científicos que permite desempenho em ambientes de larga escala (mais de 30.000 *cores*). Além disso, essa máquina conta com uma parte de sua gerência e da distribuição de ativações em uma arquitetura descentralizada, o que permite uma menor sobrecarga em comparação com sistemas que apresentam uma arquitetura centralizada. Apesar de excelentes resultados experimentais no quesito escalabilidade, o Swift/T não apresenta um mecanismo de recuperação de falhas ou de associação do fluxo de dados no nível dos elementos de uma coleção em um arquivo. Portanto, o mesmo se diferencia da proposta desta dissertação quanto à orientação dos *workflows* científicos apoiados, pois enquanto o Swift/T é um sistema de *workflows* científicos voltado a alta escalabilidade, ele desconsidera o monitoramento da execução com a participação do usuário. Já o Demeter apoia a execução paralela de *workflows* científicos orientada ao fluxo de dados, provendo ao cientista a possibilidade de acompanhar a geração do fluxo de dados e eventualmente até interferir na execução. Esse tipo de interferência possui um grande potencial de aumento da produtividade e mesmo do tempo de execução, conforme apresentado em DIAS (2013).

O SciCumulus (OLIVEIRA *et al.*, 2010) é um SGWfC paralelo para ambiente de nuvens computacionais, que utiliza o Chiron como sua máquina de workflows. SciCumulus possui diversas camadas sobre o Chiron para tirar proveito de características do ambiente de nuvens computacionais, por exemplo, elasticidade. SciCumulus apresenta assim, a mesma abordagem algébrica relacional utilizada pelo Demeter. O seu funcionamento baseia-se na instanciação de máquinas virtuais em um provedor de nuvem computacional, por exemplo, o provedor Amazon EC2 (AMAZON EC2, 2010). A partir da instanciação do ambiente para o processamento paralelo, o SciCumulus inicializa a execução de um *workflow* por meio de uma arquitetura de controle centralizado. A comunicação entre as máquinas virtuais é realizada pela tecnologia MPJ (extensão do MPI para a linguagem de programação Java) (CARPENTER *et al.*, 2000). Além disso, o SciCumulus conta com algumas propostas que visam a adaptabilidade dos NC (OLIVEIRA *et al.*, 2012, VIANA *et al.*, 2011) e a recuperação de falhas (COSTA *et al.*, 2012), explorando a elasticidade na alocação de máquinas virtuais em ambientes de nuvens computacionais. O SciCumulus também apresenta o escalonamento adaptativo semelhante ao proposto ao Demeter, favorecendo o balanceamento de carga.

As propostas de adaptabilidade dos NC no SciCumulus são baseadas em algoritmos que armazenam dados sobre o perfil de execução das atividades em bases de dados de proveniência e consultam esses dados em tempo de execução. A partir desse histórico no perfil de execução, decisões quanto à remoção ou à adição de NC no ambiente são avaliadas por modelos de custos com funções multi-objetivo, que consideram o custo financeiro, o tempo total de execução do *workflow* e a confiabilidade das máquinas virtuais alocadas na minimização da função de custos.

Mais especificamente, uma dessas avaliações do modelo de custos é realizada por algoritmos genéticos (VIANA *et al.*, 2011).

Por outro lado, a proposta desses algoritmos adaptativos é realizada por um componente separadamente do escalonador do *workflow*. Nessa situação, se houver uma indisponibilidade do escalonador, esse componente não fará as adaptações nos NC. Dessa forma, esse componente apresenta-se como um ponto único de falhas, pois necessita estar sempre disponível, não provendo a tolerância a falhas ou eleição de outro NC para assumir a sua funcionalidade.

Diferentemente disso, o Demeter considera a adaptação dos NC por ocorrência de falhas no nó central da topologia e a eleição de um novo líder (ou nó central) para continuar a análise do ambiente de execução paralela. Pode-se exemplificar a falha pelo uso de um conjunto de parâmetros incompatíveis com o programa a ser executado por uma atividade.

Outra questão é a detecção da necessidade de adaptação desses NC durante o processamento do *workflow*, em função, por exemplo, da métrica de confiabilidade. Diante de eventos como falhas nas máquinas virtuais ou nos programas envolvidos na execução do *workflow*, o SciCumulus pausa a execução do mesmo por completo para permitir a remoção do NC problemático ou a adição de novos NC (para atender aos prazos definidos pelo cientista), em função de uma limitação da versão da tecnologia MPI utilizada. Enquanto isso, por utilizar técnicas P2P do JXTA, o Demeter permite que o *workflow* continue o seu processamento, mesmo que haja alguma alteração na configuração dos NC em tempo de execução. Essa pausa mencionada no SciCumulus pode impactar gerando perda de desempenho, dependendo da quantidade do número de máquinas envolvidas na execução do *workflow*, do tempo de reinicialização do

SciCumulus com uma nova configuração e da frequência de ocorrência de alterações na configuração dos NC (por exemplo, os eventos de elasticidade).

Outro trabalho relacionado é o SciMultaneous (COSTA *et al.*, 2012), uma arquitetura desenvolvida com o intuito de detectar falhas durante a execução de *workflows* em nuvens computacionais e de recuperar o sistema após a sua ocorrência. Tal arquitetura foi acoplada ao SciCumulus. Algumas falhas são analisadas quanto ao tipo de erro ocorrido. Dependendo do erro, algumas ativações podem ser ignoradas na execução do *workflow*, assim como existem casos em que uma ativação fica executando indefinidamente em um NC, impedindo a recepção de novas ativações. Sendo assim, nessa última situação, o NC pode apresentar uma perda considerável no desempenho e precisa interromper essa ativação para proceder com a execução do *workflow*.

Outro ponto importante é que o SciMultaneous comporta-se como um módulo externo ao funcionamento do SciCumulus. Em função dessa característica, o SciMultaneous não permite executar novamente parte do *workflow*, pois o seu controle não é capaz de interferir diretamente no SciCumulus. Por exemplo, vamos assumir um *workflow* que tenha executado uma atividade com o operador *Map* e outra atividade com o operador *Reduce* e dependente desse *Map*. Para uma ativação que apresente um erro e precise de uma nova execução, caso a atividade com operador *Reduce* já tenha começado sua execução, mesmo que haja a interferência do SciMultaneous solicitando uma nova execução dessa ativação com erro (atividade com o operador *Map*), a atividade com o operador *Reduce* não será reiniciada após essa solicitação, pois o SciMultaneous não é capaz de influenciar no escalonamento e na análise de dependências entre atividades do SciCumulus.

Além dessas propostas, o Sunflower (PAPUZZO e SPEZZANO, 2011) é um arcabouço baseado em um agente P2P para configurar, gerenciar e adaptar *workflows* em ambientes com infraestrutura de grades computacionais e nuvens. As adaptações em tempo de execução são capturadas por mecanismos que utilizam técnicas P2P, tendo como objetivo garantir um bom desempenho e a recuperação da execução dos *workflows*, dada a ocorrência de eventuais anormalidades. Outra característica dessa proposta autonômica é uma métrica de QoS (do inglês *Quality of Service*) para avaliar a quantidade de NC a ser alocada para a execução do *workflow*, semelhante ao cálculo de confiabilidade do Demeter (apresentado no capítulo 4).

Por outro lado, o balanceamento de carga oferecido pelo Sunflower requer uma distribuição inicial do conjunto de ativações pendentes (e que atendem às dependências) para os nós da rede. Dessa forma, adaptações durante a execução podem obrigar eventuais balanceamentos de carga, implicando a transferência do conjunto de ativações que não estavam sendo executadas e, consequentemente, não precisariam ser redistribuídas. Diferentemente, a arquitetura AWAPE realiza uma avaliação dos NC antes de escalonar novas ativações, necessitando redistribuir apenas ativações que apresentaram alguma falha durante a sua execução.

Ao mesmo tempo, a proposta desta dissertação é compatível com as vantagens do Chiron relacionadas à álgebra relacional de *workflows* científicos centrada em dados utilizada e aos dados de proveniência relacionados à execução, como a otimização do plano de execução dos *workflows* e a reprodutibilidade dos experimentos científicos. Por exemplo, a otimização seria favorecida por operações algébricas para reduzir o processamento de dados (OGASAWARA, 2011). Já a reprodutibilidade estaria associada à capacidade de o cientista obter resultados de execuções anteriores com

determinados parâmetros, ou mesmo uma estimativa do tempo médio de execução de uma atividade.

Capítulo 4 – Uma Estratégia de Execução Paralela Adaptável de *Workflows* Científicos

Esse capítulo tem o objetivo de apresentar a estratégia proposta para a execução paralela adaptável de *workflows* científicos, que visa o uso eficiente dos nós computacionais (NC) e a adaptabilidade. A execução paralela está presente na maioria das máquinas de *workflows* científicos, enquanto que a execução paralela adaptativa é a principal contribuição desta dissertação, considerando também a tolerância a falhas e o balanceamento de carga. Vale ressaltar que um dos diferenciais dessa solução é a abordagem da execução orientada ao fluxo de dados, enquanto que outras soluções existentes em relação à adaptabilidade são orientadas a tarefas.

Com o intuito de atender esse objetivo, esta dissertação consistiu no desenvolvimento do Demeter, um gerente de execução paralela adaptável de *workflows* científicos. Além disso, técnicas P2P foram utilizadas para o desenvolvimento da estratégia adaptável, em função de suas características topológicas, da capacidade de adaptação às mudanças dos NC em tempo de execução, assim como para a recuperação de falhas e o balanceamento de carga. Mais especificamente, a rede P2P adotada na estratégia Demeter apresenta uma topologia estruturada por meio da definição de uma árvore de apenas dois níveis, em que o nó raiz coordena a troca de mensagens com todos os outros nós da rede. Apesar da troca de mensagens ser exclusiva com esse nó raiz (também conhecido como nó central), os outros nós podem se comunicar pelo mecanismo de anúncio, em função de eventuais anomalias na rede. Para implementar as técnicas P2P, utilizou-se a plataforma JXTA.

A implementação dessa estratégia adaptável foi realizada por meio do gerente de execução paralela adaptável para a álgebra de *workflows* científicos centrada em dados conhecido como Demeter, que utiliza as arquiteturas APON (do inglês *Architecture for P2P Overlay Networks*) e AWAPE (do inglês *Architecture for Workflow Adaptive Parallel Execution*) também desenvolvidas nesta dissertação. Tais arquiteturas possuem funcionalidades diferentes. A arquitetura APON foi definida para criar, gerenciar e monitorar os NC através de redes P2P em ambiente de PAD utilizando a plataforma JXTA.

Enquanto isso, a arquitetura AWAPE lida com aspectos da execução paralela adaptável de *workflows* científicos, a partir do uso das funcionalidades de adaptabilidade, de troca de mensagens e de tolerância a falhas em redes P2P presentes na arquitetura APON. Somando-se a isso, as arquiteturas APON e AWAPE foram utilizadas no Demeter, sendo que esse gerente tem como base a implementação do Chiron. Para verificar as contribuições propostas pela estratégia adaptável desta dissertação, alguns experimentos foram realizados para comparar o desempenho do Demeter (estratégia adaptativa) e do Chiron (estratégia não adaptativa), conforme apresentado no Capítulo 5. Enfatiza-se também que as arquiteturas descritas nesse capítulo (APON e AWAPE) foram desenvolvidas de forma que possam ser acopladas a outros SGWfC.

Esse capítulo é dividido em quatro seções. A Seção 4.1 descreve a topologia da rede P2P implementada pelo Demeter. A Seção 4.2 define a arquitetura AWAPE. A Seção 4.3 descreve a arquitetura APON. Por último, a Seção 4.4 apresenta as modificações realizadas na máquina Chiron para o desenvolvimento do Demeter e o uso das arquiteturas APON e AWAPE.

4.1 Topologia da rede P2P

Técnicas P2P foram utilizadas no projeto do Demeter para apoiar a execução paralela adaptável de *workflows* científicos. Para o desenvolvimento da rede P2P utilizou-se a plataforma JXTA, a partir de uma topologia estruturada. A topologia em questão é representada por uma árvore de apenas dois níveis, conforme apresentado na Figura 9. Os nós da rede P2P podem ser classificados como distribuição ou execução.

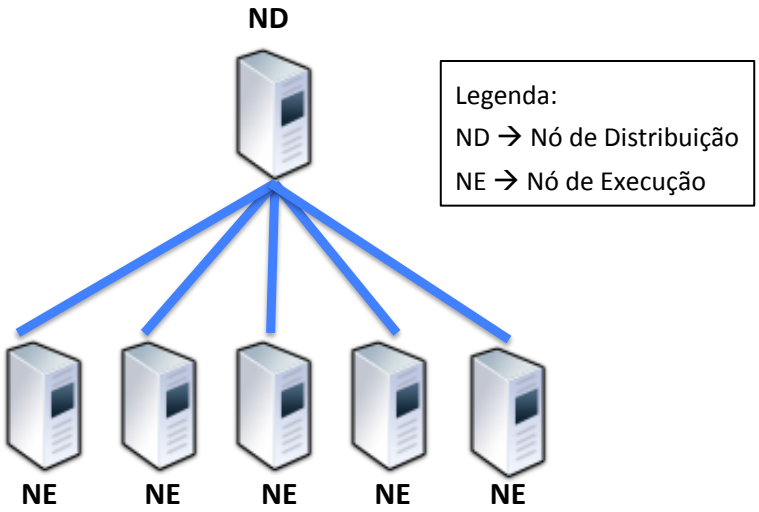

Figura 9. Topologia da rede P2P adotada.

O nó de distribuição é definido sempre como o nó raiz da topologia estruturada. Esse nó tem a função de gerenciar a distribuição do *workflow*, de acordo com a subdivisão das atividades do *workflow* em ativações, considerando as dependências existentes entre as atividades. O nó de distribuição também é o único habilitado para analisar os dados de proveniência e efetuar as transações na base de proveniência, garantindo assim a consistência da mesma. Além do escalonamento, o nó de distribuição apresenta as características dos nós de execução, de acordo com o número de processadores disponibilizados por esse nó mapeado na rede P2P.

Vale ressaltar que a solução proposta nesta dissertação considera uma base de está localizada em NC diferente do utilizado pelos nós de distribuição e execução. Somando-se a isso, esta dissertação assume que o NC que possui a base de proveniência é completamente confiável, portanto, não há a possibilidade de ocorrência de falhas nele em tempo de execução. Os NC dos sistemas de gerência de banco de dados também podem prover a replicação, a distribuição, o *rollback*, entre outras características para a execução de *workflows* científicos. O paralelismo é outro potencial associado aos sistemas de gerência de banco de dados (*e.g.* sistemas de banco de dados massivamente paralelos).

Por outro lado, o nó de execução é estabelecido sempre como um nó folha na topologia estruturada proposta. Esse tipo de nó é responsável pelo processamento propriamente dito das ativações enviadas pelo nó de distribuição. Portanto, o nó de execução recebe as ativações, analisa as configurações de ambiente necessárias para o seu processamento, configura tal ambiente e processa o conteúdo das mesmas. Tal execução produz dados de proveniência, que são relevantes para a base de proveniência. Ao final do processamento, o nó de execução notifica o seu término e armazena os dados de proveniência produzidos, por meio do envio desses dados ao nó de distribuição.

4.2 Arquitetura AWAPE

A arquitetura AWAPE (acrônimo do inglês *Architecture for Workflow Adaptive Parallel Execution*) foi desenvolvida para proporcionar a adaptabilidade durante a execução paralela de *workflows* científicos pelo Demeter, considerando também a tolerância a falhas e o balanceamento de carga. Para isso, a arquitetura AWAPE foi desenvolvida com o modelo de arquitetura baseada em componentes (Figura 10), tendo

os seguintes componentes: Analisador de Dependência do *Workflow*, Escalonador de Ativações, Gerente de Dados de Proveniência e o Gerente da Rede P2P. Essa arquitetura utiliza as técnicas P2P apresentadas na Seção 4.2. Em função da complexidade das técnicas P2P implementadas, uma nova arquitetura, conhecida como APON, foi desenvolvida para ser utilizada pelo Gerente da Rede P2P. A arquitetura APON é discutida em mais detalhes na Seção 4.3.

O Analisador de Dependência do *Workflow* é responsável pelo controle da estrutura propriamente dita do *workflow*, considerando as mudanças nos estados de execução das atividades (bloqueado, pendente, executando, finalizado ou finalizado com erro) e as análises de dependência entre as atividades. Tais mudanças nos estados de execução das atividades contam também com o monitoramento do *workflow*, a fim de verificar o término do seu processamento. Consequentemente, outra função desse componente é a identificação do momento adequado para a decomposição das atividades em ativações durante a execução do *workflow*.

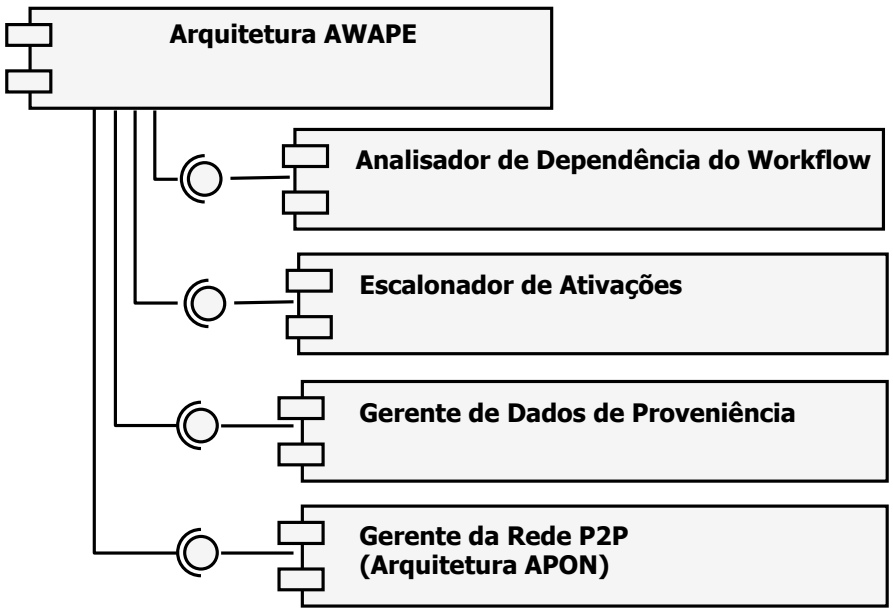

Figura 10. Arquitetura AWAPE.

O Escalonador de Ativações tem o propósito de decompor as atividades do *workflow* em ativações e de distribuir essas ativações para os NC disponíveis. A escolha de qual ativação deve ser enviada para cada nó cabe a esse componente. Por isso, esse componente utiliza propriedades do Analisador de Dependência do *Workflow* para determinar o início da decomposição em ativações. Do mesmo modo, o Gerente de Dados de Proveniência é utilizado pelo Escalonador de Ativações para verificar e atualizar o estado da execução das ativações, que podem estar prontas para serem executadas (estado conhecido como *READY*), executando em um determinado NC (estado conhecido como *RUNNING*), concluídas (estado conhecido como *FINISHED*) ou concluídas com erro (estado conhecido como *FINISHED_WITH_ERROR)*.

Portanto, o Gerente de Dados de Proveniência lida com a coleta, a análise e a consulta dos dados de proveniência. A coleta é realizada por meio das mensagens capturadas e tratadas pelo Escalonador de Ativações, que apresenta dados de proveniência de ativações que foram distribuídas em um dado momento (alteração do estado da ativação de *READY* para *RUNNING*) ou mesmo dados resultantes da execução de ativações pelos nós da rede (alteração do estado da ativação de *RUNNING* para *FINISHED*). Já a análise é utilizada para validar os dados antes do seu armazenamento na base de dados. Enquanto isso, a consulta dos dados de proveniência é utilizada (i) pelos cientistas, para obter resultados de suas execuções; e (ii) pela própria arquitetura, para monitorar a execução do *workflow* (por exemplo, a identificação do término do processamento do *workflow*). Outras características apoiadas por esse componente são a consistência dos dados e a recuperação da execução do *workflow*, caso haja uma falha no nó de distribuição.

No que diz respeito à recuperação de falhas, um novo estado foi estabelecido pelas ativações para identificar a sua ocorrência (estado *FINISHED_WITH_ERROR*).

Além desse novo estado, a base de proveniência também gerencia o número de ocorrências de erros para uma ativação e o NC que processou tal ativação. Ao mesmo tempo, um cálculo de confiabilidade é realizado na arquitetura APON para determinar quanto a continuação de um determinado NC. Mais especificamente, esta dissertação concentra-se principalmente nas falhas semânticas (relacionadas à software), que estariam associadas à configuração dos programas pelos cientistas (*e.g.* definição de valores de parâmetros a serem utilizados), ao invés de falhas de hardware.

De acordo com a adaptação em tempo de execução, o Demeter é baseado no escalonamento adaptativo. Portanto, eventuais mudanças nos NC durante o progresso do *workflow*, implica em uma redistribuição apenas de algumas ativações em nós que não estão mais presentes na rede de NC disponíveis. Enquanto isso, a maioria das outras soluções adaptativas existentes implementam um escalonamento adaptativo que analisa todos os NC em tempo de execução, como a extensão do Pegasus proposta por LEE *et al.* (2008). Nesse caso, a ocorrência de adaptações necessita que o escalonador de *jobs*, conhecido como DAGMan defina um novo mapeamento das ativações para todos os NC. Logo, as redistribuições de ativações nesse tipo de escalonamento adaptativo podem ser realizadas até mesmo com NC que não teriam sido influenciados pela adaptação.

Por último, o Gerente da Rede P2P tem o propósito de estabelecer a rede P2P com a topologia estruturada de apenas dois níveis, em que o nó raiz é o responsável pelo escalonamento do *workflow* e pelo controle dos dados de proveniência. Essa rede P2P utiliza a plataforma JXTA e seus NC para a sua manutenção. Em função da complexidade de definir todos os NC desse componente no que diz respeito à abordagem P2P, a arquitetura APON foi desenvolvida para representar os componentes

envolvidos nesse mecanismo de gerência da rede P2P. Logo, a arquitetura APON é descrita em mais detalhes na seção a seguir.

4.3Arquitetura APON

A arquitetura APON (do inglês *Architecture for P2P Overlay Networks*) foi desenvolvida para definir os componentes envolvidos na sobreposição da rede P2P e o seu mecanismo adaptativo, que é utilizado pelo Gerente da Rede P2P na arquitetura AWAPE. Portanto, diferentemente da arquitetura AWAPE, que apresenta um mecanismo adaptável para executar paralelamente *workflows* científicos, a arquitetura APON descreve as técnicas P2P usadas para garantir a adaptabilidade e a tolerância a falhas. A arquitetura APON é composta dos seguintes componentes apresentados na Figura 11: Inicializador de Nós Computacionais, Gerente Adaptável de Nós Computacionais, Controlador de Anúncios, Controlador de *Pipes* e Monitor de Falhas.

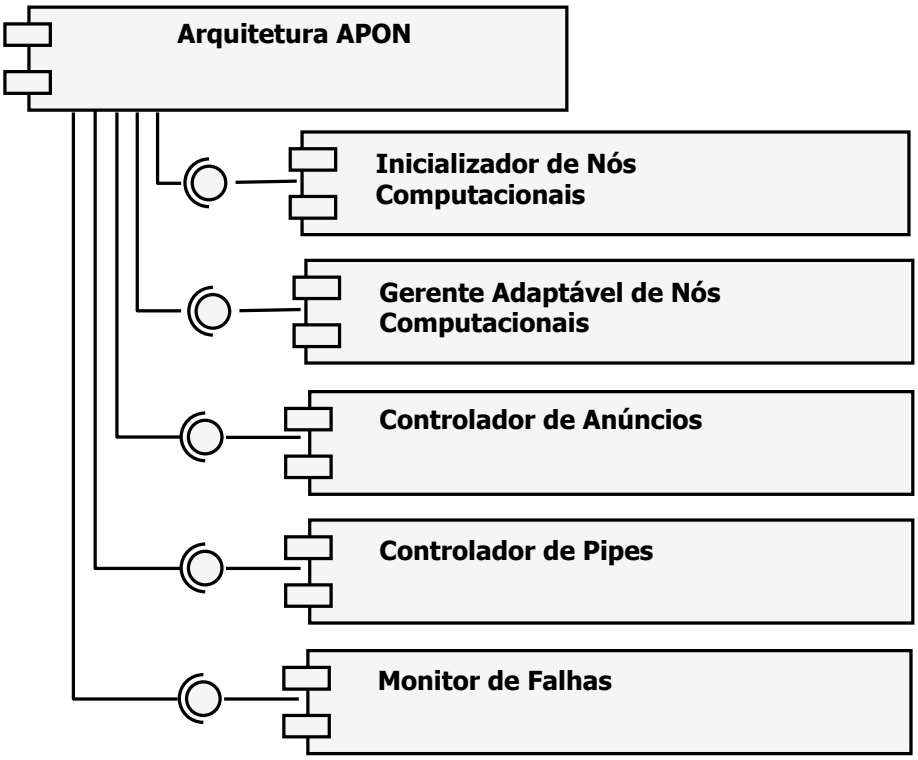

Figura 11. Arquitetura APON.

O Inicializador de Nós Computacionais é responsável pela definição de um novo nó na rede P2P. Para isso, esse componente apresenta a execução do algoritmo para a determinação do tipo de nó que será instanciado, podendo ser um nó de distribuição ou de execução, conforme apresentado na Seção 4.1. Do mesmo modo, no caso em que a rede ainda não apresenta um nó de distribuição (nó central da topologia proposta), um algoritmo de eleição de líder precisa ser processado para determinar qual dos pedidos de inserção ou de modificação do tipo de nó será escolhido para definir um novo nó de distribuição.

O algoritmo de eleição de líder desenvolvido começa o seu processamento identificando os primeiros nós de execução que perceberam a ausência do nó de distribuição (através da descoberta de anúncios), assim como todos os pedidos pendentes de novos nós na rede. Nesse algoritmo, a data de ingresso e o identificador do nó são as propriedades utilizadas na eleição. A partir dos nós capturados, o algoritmo define o nó mais antigo (o primeiro a ingressar na rede de acordo com o JXTA) como o nó de distribuição. Enquanto isso, os outros nós são definidos como nós de execução.

O algoritmo de eleição de líder também pode ser influenciado pela saída de nós da rede. Nesse sentido, há dois comportamentos possíveis. O primeiro é caracterizado pela saída de um nó de execução. Nesse caso, o nó de distribuição só remove o nó de execução do seu controle de nós presentes na rede e atualiza a base de proveniência para redistribuir as ativações que estavam sendo executadas por esse nó de execução. Enquanto isso, o segundo comportamento é caracterizado pela saída de um nó de distribuição. Nessa situação, a topologia da rede P2P é afetada substancialmente, uma vez que um dos nós de execução precisa ser definido como nó de distribuição para continuar a execução do *workflow*. Para que isso seja possível, os primeiros nós de

execução a identificar a saída do nó de distribuição são utilizados pelo algoritmo de eleição de líder.

Já o Gerente Adaptável de Nós Computacionais tem a função de garantir a adaptabilidade da execução do *workflow* por meio de técnicas P2P, sendo que uma adaptação é efetuada pela adição ou remoção de NC, ou mesmo pela ocorrência de falhas. Essa decisão de alteração dos NC é baseada em algumas métricas propostas pelo algoritmo adaptativo do SciCumulus (OLIVEIRA *et al.*, 2012), como a confiabilidade.

Nesse caso, a confiabilidade está diretamente associada ao percentual de ativações processadas com sucesso (sem falha) em relação ao número total de ativações processadas em um dado NC. Logo, o cálculo de confiabilidade pode ser expresso pela equação a seguir, sendo que as variáveis $\text{ativacoes}_{\text{processadas}}$ e $\text{numero}_{\text{falhas}}$ representam, respectivamente, o número de ativações processadas por um determinado NC e o número de ativações finalizadas com falhas, que foram constatadas nesse mesmo NC:

$$\text{confiabilidade} = \frac{\text{ativacoes}_{\text{processadas}} - \text{numero}_{\text{falhas}}}{\text{ativacoes}_{\text{processadas}}}$$

Ao mesmo tempo, a adição de NC em tempo de execução é permitida de forma manual pelo usuário, a partir da invocação da máquina de execução paralela. A mesma analisa os NC disponíveis para a execução do *workflow* e adequa o novo NC de acordo com a topologia centralizada proposta. Em virtude dos NC em *clusters* e grades computacionais normalmente utilizarem um gerente de recursos e de despacho, como o *Portable Batch System* ou PBS (BAYUCAN *et al.*, 2000), a adição de NC é efetivada apenas quando o usuário solicita tais NC extras e os mesmos são disponibilizados por

esses gerentes. Sendo assim, esse cenário não é apoiado automaticamente pelas heurísticas de adaptabilidade desse trabalho.

Além desses componentes, o Controlador de Anúncios lida com a publicação e a descoberta de anúncios pelos nós da rede. Esse componente permite a definição de todos os tipos de anúncios propostos pela plataforma JXTA (anúncios de nó, de grupo e de *pipe*). O Controlador de *Pipes* tem o propósito de estabelecer o tipo de *pipe* (unidirecional ou de propagação) e quais serão os nós com porta de entrada (receptor) e porta de saída (transmissor) para a troca da mensagem. Além de estabelecer esse canal de comunicação, esse componente efetua a troca propriamente dita da mensagem entre os nós.

Quanto a esse mecanismo de troca de mensagens entre os nós da rede P2P, o canal de comunicação é estabelecido de forma unidirecional através de três etapas: a divulgação de um anúncio de *pipe* pelo nó origem, descoberta de anúncio pelo nó destino e o envio ou recebimento da mensagem pelo *pipe*. O anúncio de *pipe* é criado e divulgado pelo nó origem que deseja enviar uma mensagem a um nó destino, assim como o nó origem estabelece um *pipe* de saída para que seja possível enviar a mensagem. Após a sua publicação, o nó destino é o responsável por capturar tal anúncio e estabelecer um *pipe* de entrada. Ao definir o *pipe* de entrada, os dois nós envolvidos na transmissão da mensagem estão aptos a realizar o envio ou o recebimento da mensagem, de acordo com a sua função nesse canal de comunicação.

As mensagens consideradas na comunicação entre os nós (sempre entre um nó de distribuição e outro de execução) no Demeter podem ser dos seguintes tipos: REQUEST, PROCESS, STORE, WAIT e FINISHED. A mensagem de REQUEST é utilizada para a solicitação de ativação pelos nós da rede. Enquanto isso, a mensagem

de PROCESS envia uma ativação pendente no *workflow* a ser executada no nó solicitante. Já a mensagem de STORE é responsável pelo envio dos dados de proveniência de uma ativação executada no nó, assim como o pedido de uma nova ativação para o nó de distribuição.

A mensagem de WAIT, por outro lado, informa ao nó solicitante que o *workflow* ainda não terminou a sua execução e que não há nenhuma ativação pendente no momento. O fato de não ter ativação pendente está relacionado ao escalonamento das últimas ativações de uma atividade para outros nós da rede, impedindo o escalonamento de novas ativações para o nó solicitante até o término dessas últimas ativações nos outros nós e, consequentemente, o prosseguimento da execução das outras atividades do *workflow*. Por último, a mensagem de FINISHED informa aos nós que o *workflow* foi completamente executado ou abortado, requisitando, dessa forma, a finalização dos nós.

Por último, o Monitor de Falhas baseia-se em dois mecanismos: algoritmo *keep-alive* e controle dos recursos físicos. O algoritmo *keep-alive* (KA) é conhecido pela verificação da conexão entre nós em uma rede. O seu funcionamento é inicializado pelo envio de uma mensagem para outro nó, a fim de confirmar a sua existência na rede. Para isso, esse algoritmo de KA na arquitetura APON confirma a existência dos nós a partir da captura dos seus anúncios na rede P2P.

Já o controle dos NC monitora as possíveis ocorrências de falhas de *hardware* ou de *software*, ajustando os NC disponíveis de acordo com a confirmação dessas falhas. A detecção de falhas ocorre tanto no nível de *software*, para falhas específicas da execução de ativações do *workflow*, como no nível de *hardware*, para falhas identificadas a partir das funcionalidades apoiadas pela plataforma JXTA. No caso de

falhas no nível de *software*, as ativações são distribuídas novamente para outros nós. Se houver a reincidência de falhas em uma ativação após um número limite de vezes estipulado (o valor padrão é de cinco tentativas), tal ativação é desconsiderada do *workflow*, respeitando as especificações dos operadores algébricos. Essa decisão foi baseada na possibilidade de fornecimento de parâmetros incompatíveis para a execução de uma determinada atividade.

Um ponto importante dessa abordagem é que não há redundância no envio das ativações. Ou seja, só ocorre o reenvio em caso de falhas. A detecção de falhas em tempo de execução também considera o bloqueio de um NC pelo fato de um programa não finalizar ou interromper a sua execução (por exemplo, o programa pode ter travado ou o parâmetro informado não é adequado para o programa em questão). Nesses casos de bloqueio, o tempo médio de processamento de ativações dessa atividade (referentes a execuções passadas) é computado e, quando o tempo de processamento da ativação em andamento for maior que 2,33 vezes o desvio padrão, então essa ativação é abortada e o Monitor de Falhas considera a mesma como concluída com erro. É importante enfatizar também que esse componente se responsabiliza apenas pela identificação das falhas, enquanto que o reenvio das ativações presentes no nó com falhas para outros NC (nós na rede P2P) é realizado pelo Escalonador de Ativações da arquitetura AWAPE.

4.4Adaptação da Arquitetura do Chiron

As arquiteturas AWAPE e APON incorporadas no Demeter apresentaram uma estratégia para execução paralela adaptável para a álgebra de *workflows* científicos centrada em dados. Além disso, o Demeter utilizou como base a arquitetura presente no Chiron. Nesse sentido, alguns componentes do Chiron foram adaptados, assim como

houve a inclusão de novos componentes, a fim de possibilitar o uso desse gerente de execução paralela adaptável. Uma das adaptações realizadas foi baseada na alteração do mecanismo de troca de mensagens, desenvolvido na versão original do Chiron por meio da tecnologia MPI. Nesse sentido, essa seção tem o objetivo de apresentar tais adaptações no Chiron, com o intuito de atender aos objetivos do Demeter.

A arquitetura do Chiron é representada pela Camada de Distribuição e pela Camada de Execução, conforme a Figura 12. A Camada de Distribuição é a responsável pela gerência do *workflow* e pela distribuição das ativações para os nós da rede. Para que isso seja possível, o monitor gerencia a execução do *workflow*, através de consultas e atualizações na base de proveniência. Por outro lado, o escalonador determina como as ativações devem ser distribuídas, de acordo com a análise de dependências entre as atividades realizada pelo monitor. Após a determinação das ativações pendentes e de seus respectivos destinos (nós disponíveis), o mecanismo de despacho envia tais ativações. Além desses mecanismos, há o receptor de mensagem da camada de distribuição, que recebe as ativações processadas por outros nós e encaminha os dados de proveniência pelo analisador de dados. Vale mencionar também que a arquitetura proposta pelo Chiron é centralizada, ou seja, só há um nó central responsável por controlar a execução do *workflow*. Logo, a Camada de Distribuição só é inicializada apenas uma vez (pelo nó central).

Já a Camada de Execução é inicializada por todos os nós da arquitetura, incluindo o nó central, pois o mesmo também utiliza seus nós computacionais para o processamento. A inicialização dessa camada é caracterizada pelo recebimento de uma mensagem do nó central a partir do receptor de mensagem. Após o recebimento dessa mensagem, as configurações do sistema operacional e a organização do sistema de arquivos são realizadas pelo inicializador do ambiente, que age de acordo com a

presença de ativações na mensagem recebida. Dado que o ambiente está devidamente configurado, então as ativações recebidas podem ser executadas pelo processador de ativação, que coleta em tempo de execução os dados de proveniência. Após o processamento das ativações, o mecanismo de despacho realiza a coleta dos dados de proveniência produzidos e os envia para o nó central, informando dados específicos da execução da ativação ou mesmo do domínio.

Diferentemente do modelo apresentado na Figura 12, a arquitetura adaptada do Chiron para o Demeter é representada por três camadas, sendo as seguintes: Camada de Distribuição, Camada de Execução e Camada APON. As duas primeiras camadas apresentam as mesmas características do Chiron, contudo possuem algumas adaptações e inclusões. Com exceção dos mecanismos de despacho e do receptor de mensagem, todas as outras modificações nessas duas camadas são referentes à arquitetura AWAPE. Já a Camada APON foi inserida para estabelecer a rede P2P proposta, monitorar a entrada e a saída de nós na mesma, assim como permitir a detecção e o tratamento de falhas em tempo de execução. Essa camada é dedicada apenas a implementar todos os componentes da arquitetura APON, sendo os componentes dessa camada iguais aos existentes na arquitetura APON. A Figura 13 apresenta a arquitetura adaptada do Chiron para o Demeter.

Com esse propósito, a Camada APON apresenta o inicializador de nós computacionais que realiza a verificação e a inserção de novos nós na rede, respeitando a topologia estruturada proposta. Além disso, nesse momento o nó pode ser classificado como nó de distribuição ou nó de execução, conforme apresentado na Seção 4.1. Considerando a arquitetura do Chiron, o nó central seria classificado como nó de distribuição na arquitetura do Demeter, enquanto que os outros nós seriam os nós folhas da topologia P2P, sendo definidos como nós de execução.

Após a inserção desse nó na rede, o gerente adaptável de NC, o controlador de anúncios, o controlador de *pipes* e o monitor de falhas são inicializados e funcionam continuamente. O gerente adaptável de NC é responsável pelas modificações dos nós da rede, seja de remoção ou de inserção de NC. Além disso, esse componente realiza a análise de vizinhança desses nós através da atualização da base de configuração. Enquanto isso, o controlador de anúncios é responsável pela identificação de anúncios na rede P2P e a sua análise para informar eventuais mudanças na topologia.

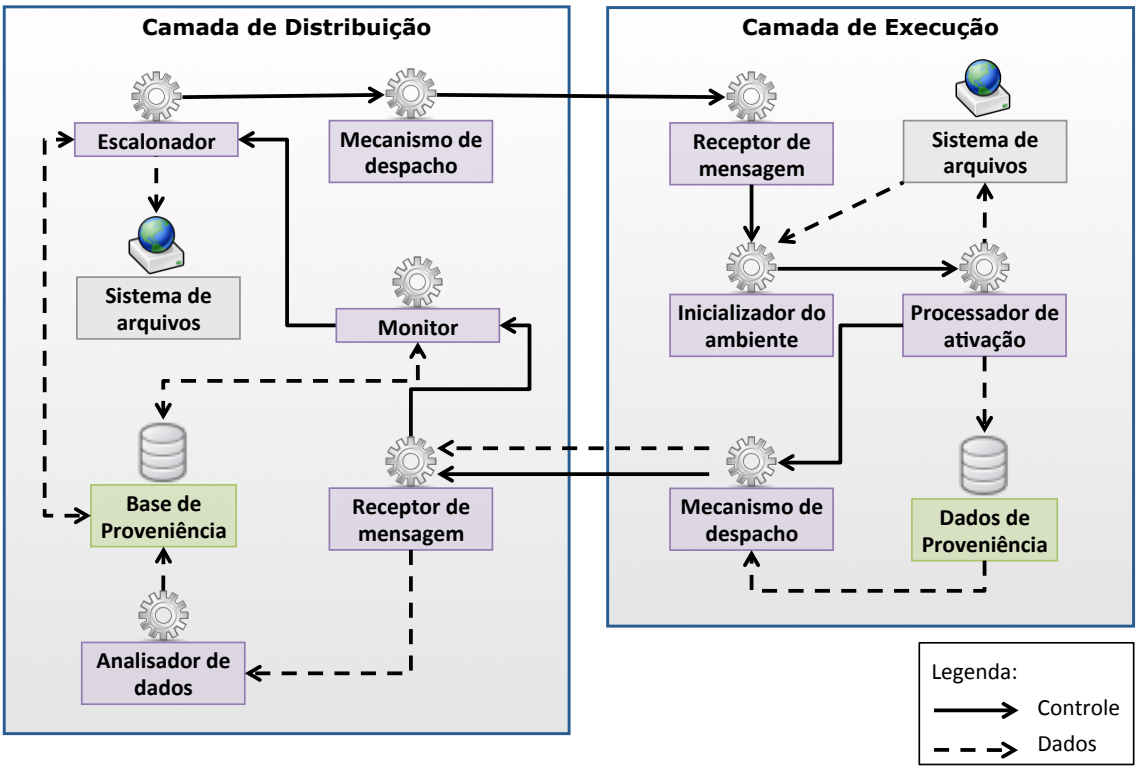

Figura 12. Adaptação da arquitetura do Chiron.

Já o controlador de *pipes* identifica tentativas de comunicação com o NC em questão ou tenta estabelecer um canal de comunicação com outro nó para o envio de mensagens. Por último, o monitor de falhas realiza uma verificação periódica da rede quanto à ocorrência de falhas. Em caso de eventuais falhas, as modificações na rede P2P são efetuadas e a base de configuração é atualizada. É importante enfatizar que

todos os dados de configuração apresentados na Figura 13 referem-se apenas a uma base, sendo separada graficamente por uma questão visual.

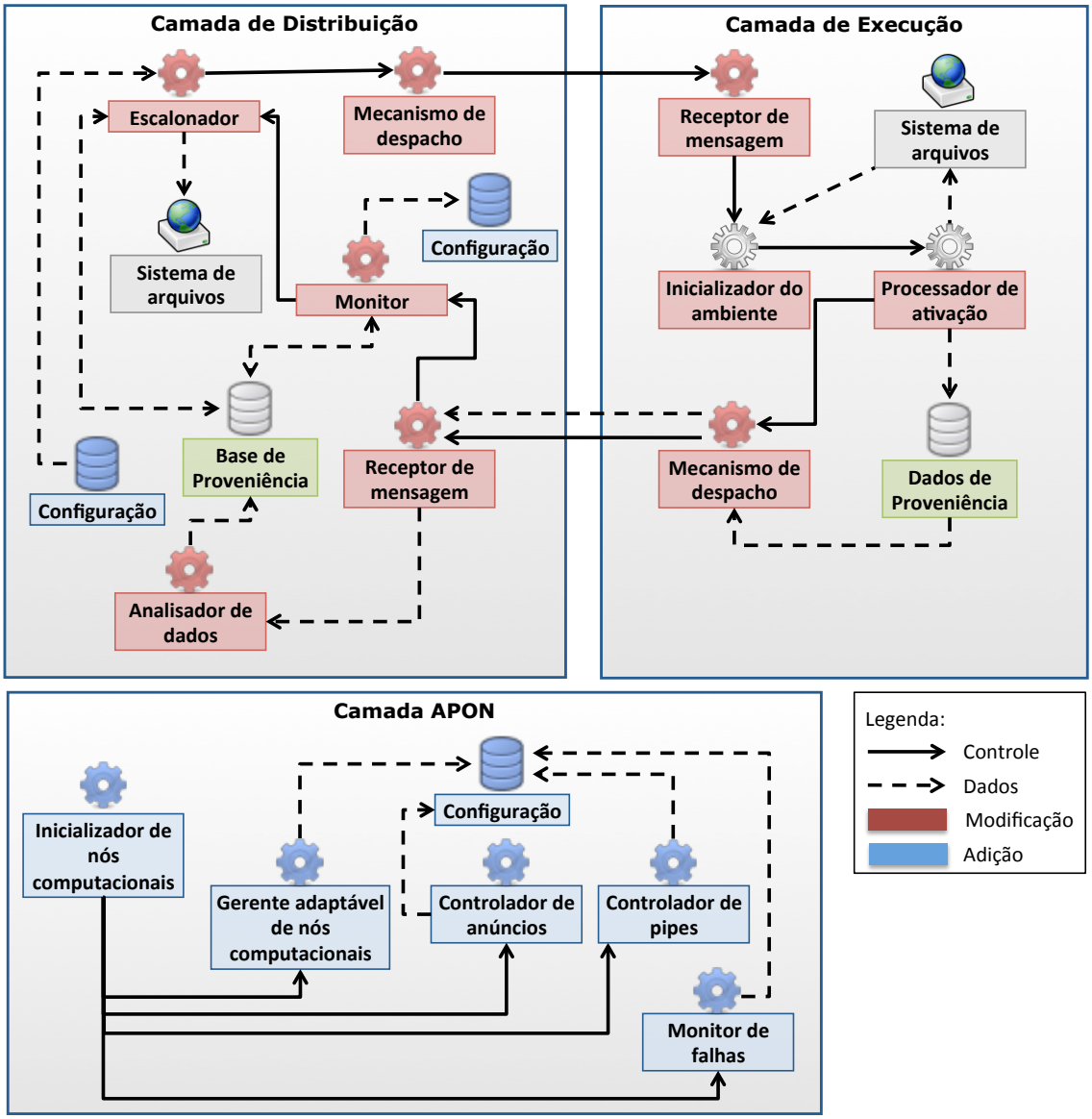

Figura 13. Arquitetura adaptada do Chiron para o Demeter.

Em relação à Camada de Distribuição e à Camada de Execução, os elementos com coloração vermelha foram adaptados de acordo com a arquitetura AWAPE. Primeiramente, o monitor (implementado pelo componente Analisador de Dependência do *Workflow* da arquitetura AWAPE) foi adaptado para permitir a gerência do *workflow*, considerando propriedades específicas das técnicas P2P. Por exemplo, os dados de um nó que enviou uma mensagem (ou mesmo uma ativação) são analisados e

armazenados com os dados de configuração da rede. Tais dados estão relacionados ao endereço IP do nó, ao último anúncio recebido, a quantas falhas de envio já ocorreram no seu log, entre outros. Os dados utilizados para a configuração da rede foram consultados na base de proveniência, refletindo em adaptações no modelo de proveniência do Chiron.

Do mesmo modo, o escalonador (implementado pelo componente Escalonador de Ativações da arquitetura AWAPE) foi adaptado para utilizar os dados de configuração da rede P2P na determinação de quais mensagens devem ser respondidas primeiramente. Nesse sentido, as mensagens recebidas são ordenadas quanto ao horário de seu recebimento. Contudo, em casos de mensagens recebidas em um horário muito próximo, os nós com menor frequência de falha (número de falhas em um intervalo de tempo) foram ordenados primeiramente na fila. Além disso, informações fornecidas pelo gerente adaptável de NC e pelo monitor de falhas foram aproveitadas antes do escalonamento, a fim de confirmar a conexão entre o nó de distribuição e o nó requerente de uma ativação.

Enquanto isso, os mecanismos de despacho e os receptores de mensagem foram modificados para apoiar a abordagem P2P do Demeter, removendo-se assim a implementação baseada em trocas de mensagens MPI do Chiron. As mudanças nesses elementos foram pautadas na introdução do mecanismo de publicação de anúncios de *pipe*, assim como na descoberta de tais anúncios e no estabelecimento de um *pipe* para a troca de mensagens entre os nós envolvidos. Pelo fato de utilizar conceitos inerentes das técnicas P2P, esses mecanismos de despacho e os receptores de mensagem implementaram o componente Gerente da Rede P2P da arquitetura AWAPE, que utiliza os componentes da arquitetura APON (presentes na camada APON do Demeter). Já o componente Gerente de Dados de Proveniência da arquitetura AWAPE

foi desenvolvido a partir da modificação do analisador de dados do Chiron, que é responsável pela captura e consulta de dados de proveniência do *workflow* em tempo de execução.

Dessa forma, a implementação da arquitetura AWAPE no Demeter proporcionou a execução paralela adaptável de *workflows* científicos. Tal adaptabilidade está relacionada à capacidade de recuperação da execução caso ocorra uma falha, à alocação dinâmica de NC e ao uso eficiente dos NC em tempo de execução. Em relação à validação dessas características, as duas primeiras características foram analisadas por meio de testes. Mais especificamente, no caso de falhas do nó de distribuição (caso mais complexo para a topologia desenvolvida), o tempo médio de restabelecimento da rede P2P por completo (todos os NC) foi de aproximadamente vinte e sete segundos, sendo esse teste realizado para uma amostra de dez ocorrências desse cenário.

Por outro lado, ao considerarmos a eficiência, experimentos foram conduzidos conforme apresentado no próximo capítulo. Nesse sentido, os experimentos foram baseados na comparação das métricas de *speedup* e eficiência entre a estratégia adaptativa do Demeter e a estratégia não adaptativa existente no Chiron.

Capítulo 5 – Avaliação Experimental do Uso Eficiente dos Nós Computacionais

Esse capítulo apresenta os experimentos realizados para a avaliação do Demeter. Os experimentos foram baseados na comparação do desempenho entre a estratégia adaptativa do Demeter e a estratégia não adaptativa existente no Chiron, sendo que as métricas utilizadas para a análise de desempenho são o *speedup* e a eficiência, definidas em mais detalhes a seguir. Assim, os experimentos também tem o propósito de constatar se a sobrecarga da estratégia de execução paralela adaptativa é vantajosa em relação ao tempo de processamento ao ser comparada com a estratégia de execução não adaptativa.

Para a descrição das diferentes etapas do desenvolvimento e execução dos experimentos, esse capítulo foi dividido em três seções. A seção 5.1 apresenta uma descrição dos experimentos realizados, das características do ambiente computacional utilizado e das métricas de desempenho empregadas. Já a seção 5.2 apresenta detalhes da configuração do ambiente para a execução dos experimentos, enquanto que a seção 5.3 discute os resultados obtidos.

5.1 Descrição dos Experimentos

Os experimentos desenvolvidos para a comparação da estratégia adaptativa do Demeter com a estratégia não adaptativa existente no Chiron foram baseados em dois *workflows*, sendo cada *workflow* foi analisado em um estudo de caso separado. O primeiro estudo de caso considera um *workflow* representado apenas por cinco atividades encadeadas utilizando o operador *Map* (Figura 14), de acordo com a álgebra de *workflows* científicos centrada em dados (OGASAWARA *et al.*, 2011) empregada no Demeter, que foi abordada na seção 2.1. Esse primeiro estudo de caso tem o

objetivo de comparar a sobrecarga entre as estratégias adaptativa e não adaptativa para diferentes configurações, variando-se a complexidade de tempo de processamento de cada atividade.

Já o segundo estudo de caso considera um *workflow* sintético do Montage (Figura 15), conhecido na área de astronomia por permitir a junção de imagens cósmicas para a criação de mosaicos, conforme apresentado por JACOB *et al.* (2009). Montage foi utilizado por diferentes SGWfC e vem se tornando um padrão na avaliação de execuções paralelas em *workflows*. Dessa forma, o segundo estudo de caso visa a análise de desempenho das estratégias em um *workflow* envolvendo uma modelagem semelhante a aplicações reais. Quanto ao modelo de execução, todos os experimentos foram executados utilizando o modelo de execução dinâmico e com a estratégia de fluxo de dados FAF (*First Activity First*). Houve apenas uma execução de configuração que executamos o experimento com a estratégia de execução estática, pois os resultados para a estratégia de execução dinâmica apresentou um valor de eficiência muito inferior aos demais experimentos.

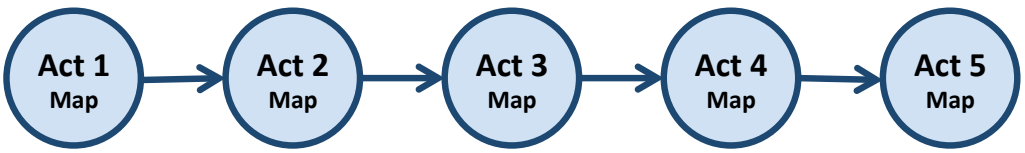

Figura 14. *Workflow* executado no primeiro estudo de caso.

Vale destacar também que, apesar dos dois estudos de caso apresentarem apenas dois operadores algébricos, os mesmos já contemplam os dois tipos possíveis de comportamentos esperados por todos os operadores algébricos da abordagem utilizada, conhecidos quanto ao fato de serem ou não bloqueantes. Nesse caso, os operadores *Map*, *Split Map* e *Filter* são conhecidos como não bloqueantes, ou seja, a sua execução não bloqueia a geração de novas ativações para as atividades que dependem dessa (para

isso, o modelo de execução deve ser baseado na estratégia FAF (*First Activity First*). Por outro lado, os operadores *SR Query*, *MR Query* e *Reduce* são ditos bloqueantes, quando podem gerar novas ativações apenas ao término do processamento desse operador algébrico.

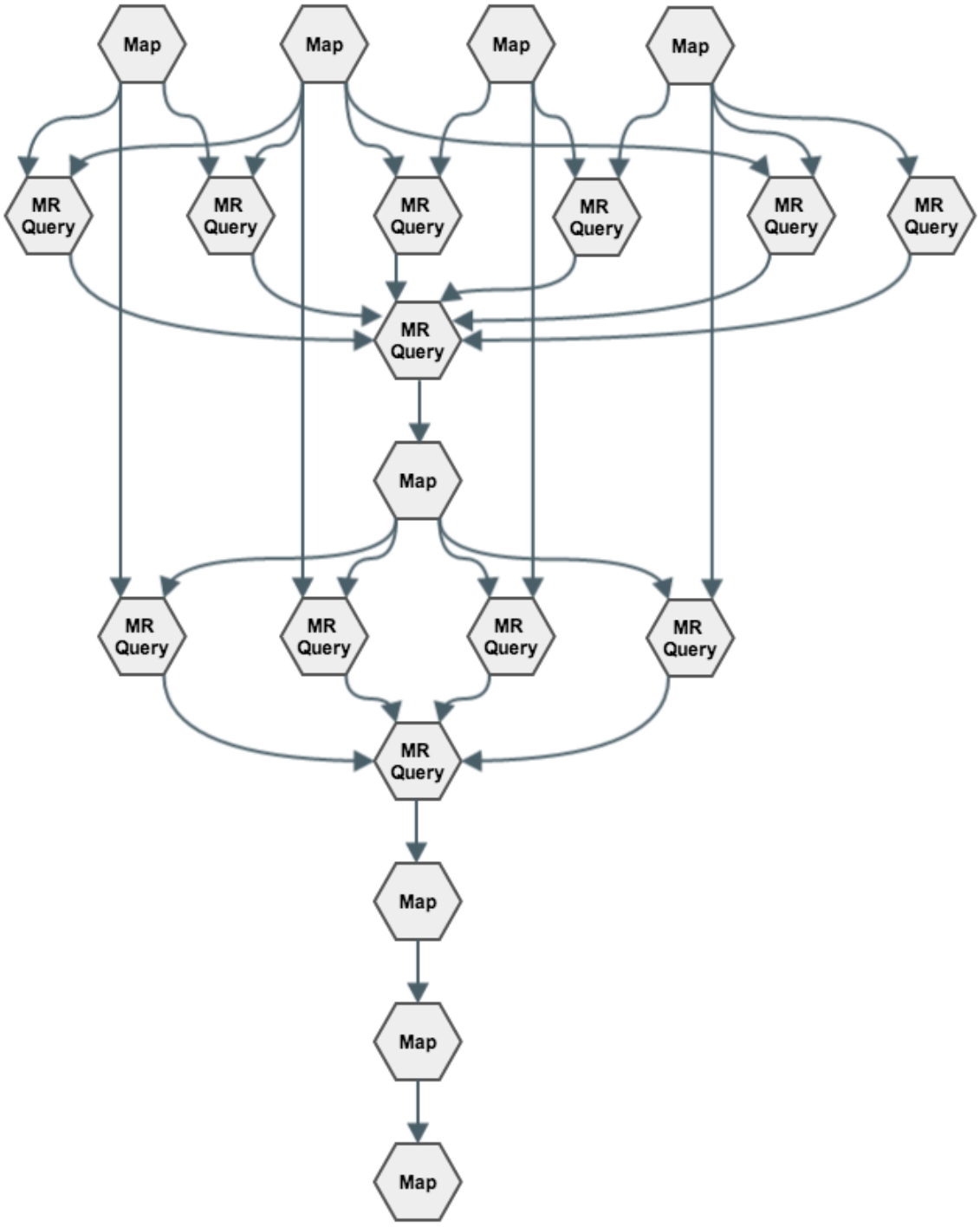

Figura 15. *Workflow* sintético do Montage com os operadores algébricos.

Em relação ao primeiro estudo de caso, a complexidade da execução das atividades do *workflow* foi definida a partir do custo de processamento e do número de ativações a serem executadas por cada atividade. Dessa forma, uma relação de entrada foi definida para a atividade *Act 1*, a fim de obter o custo de processamento de cada atividade. Mais especificamente, alguns fatores foram definidos e utilizados para o estabelecimento da complexidade das atividades desse estudo de caso.

As métricas para a definição do custo de processamento das atividades foram baseadas no benchmark de *workflows* científicos conhecido como SWB (*Scientific Workflow Benchmark*) (CHIRIGATI *et al.*, 2012). A definição do custo de processamento de uma atividade utilizou o fator de custo da atividade (da sigla *ACF* e do inglês *Activity Cost Factor*). A partir do fator *ACF*, o tempo de execução de uma ativação em milissegundos (da sigla *ACV* e do inglês *Activity Cost Value*) pode ser calculado pela fórmula $ACV = 2^{ACF+9}$, sendo $ACF > 0$. Enquanto isso, o número de ativações a serem executadas foi definido a partir do fator de tamanho das ativações (da sigla *ASF* e do inglês *Activation Size Factor*). O número de ativações a serem consumidas (da sigla *ASV* e do inglês *Activation Size Value*) é calculado pela fórmula $ASV = 2^{ASF+9}$, sendo $ASF > 0$. Dessa forma, o valor mínimo para *ACV* é de 1024 milissegundos, do mesmo modo que para *ASV* é de 1024 ativações.

Vale enfatizar que a definição das fórmulas levou em consideração o estudo a ser realizado. Por exemplo, pelo fato de os experimentos lidarem com uma análise de desempenho, aumentando o número de *cores*, o número de ativações precisa apresentar uma configuração mínima, a fim de que a paralelização seja vantajosa. No que diz respeito à vantagem de paralelização, temos que considerar a sobrecarga de inicialização dos NC e de comunicação entre os NC, assim como o número médio de

ativações a serem distribuídas para cada NC. Por esse motivo existem configurações mínimas nos experimentos em questão.

No segundo experimento, o *workflow* Montage apresenta a complexidade de execução de suas atividades inerente aos programas utilizados por essa aplicação da astronomia. Logo, uma análise variando a complexidade das atividades no segundo estudo de caso não foi considerada, e também por já ter sido realizada no primeiro estudo de caso. Somando-se a isso, no segundo estudo de caso desse capítulo, um *workflow* sintético foi modelado de acordo com as características do *workflow* Montage, respeitando as dependências de dados e a complexidade de execução das atividades (VÖCKLER *et al.*, 2011).

Os estudos de casos foram executados no *cluster* de nome Uranus, pertencente ao Núcleo Avançado de Computação de Alto Desempenho (NACAD) da COPPE/UFRJ (NACAD, 2014). A Uranus consiste de uma máquina SGI Altix ICE 8400 com 128 CPUs Intel Xeon (640 *cores* no total), sendo 64 CPUs Six Core Intel Xeon X5650 (Westmere) de 2,67 GHz (totalizando 384 *cores*) e 64 CPUs Quad Core Intel Xeon X5355 (Clovertown) de 2,66 GHz (totalizando 256 *cores*). Esse *cluster* conta com 1,28 TBytes de memória RAM, 72 TBytes de armazenamento em disco através da SGI InfiniteStorage NAS e com rede Infiniband QDR, DDR e Gigabit. O sistema operacional é o Suse Linux Enterprise Server (SLES) associado ao SGI Performance Suite. Apesar das 128 CPUs, apenas CPUs semelhantes foram consideradas nos experimentos, uma vez que os dois conjuntos de processadores (Westmere e Clovertown) são separados por duas filas de submissão, ocasionando dois escalonamentos de *jobs* distintos e independentes. Nesse caso, apenas as CPUs Six Core Intel Xeon X5650 (Westmere) foram utilizadas por apresentarem a configuração com o maior número total de processadores.

Antes da definição das configurações a serem utilizadas pelo *workflow* do primeiro estudo de caso, algumas análises foram realizadas para avaliar as limitações de *hardware* do ambiente computacional, utilizando as duas abordagens (estratégia adaptativa e não adaptativa), assim como aplicações externas intensivas em computação ou em processamento de dados. Por essas análises, constatou-se um custo muito alto desse *cluster* em lidar com o fator ASF maior que dois, em função do sistema de arquivos, afetando o desempenho de todas as aplicações executadas. Logo, um estudo variando o valor ASF não foi realizado. Ao mesmo tempo, não houve nenhuma limitação em relação ao fator ACF. Considerando as constatações mencionadas, as configurações para o primeiro estudo de caso foram definidas para a execução na Uranus, conforme a Tabela 2.

Tabela 2. Configurações para o primeiro estudo de caso.

# Configuração	ACF	ACV	ASF	ASV
1	1	1024	10	1024
2	2	2048	10	1024
3	3	4096	10	1024
4	4	8192	10	1024
5	5	16384	10	1024
6	6	32768	10	1024
7	7	65536	10	1024

Utilizando os casos da Tabela 2, uma análise de desempenho foi realizada a partir de cada configuração, variando-se o número de *cores* utilizados. As configurações para o número de *cores* utilizados foram as seguintes: 1, 12, 24, 48, 96, 108 e 120 *cores*. Cada nó utilizado nessas configurações é composto de 12 processadores e o tempo de execução com 1 *core* é um valor teórico, uma vez que os casos com valores de ACF altos impossibilitariam a execução dos *workflows* em tempo viável.

Os dois estudos têm o objetivo de avaliar se o custo de processamento da arquitetura para execução paralela adaptável é alto em relação à proposta original. Ou seja, caso os experimentos comprovem um desempenho semelhante, pode-se destacar que essa arquitetura apresentou vantagens ao garantir a adaptabilidade durante a execução paralela de *workflows* científicos. Apesar de não considerar a ocorrência de falhas ou mesmo a adaptação dos NC durante a execução dos experimentos, tais funcionalidades foram testadas de acordo com a descrição da arquitetura desta dissertação. Mesmo assim, o uso da estratégia adaptativa já implica um custo de gerência de suas funcionalidades. Outra análise em questão nesses experimentos é a eficiência dos NC utilizados, sendo que essa característica está diretamente relacionada à capacidade de distribuição das ativações para os nós disponíveis, evitando a ociosidade dos mesmos.

Dessa forma, utilizou-se as seguintes métricas para análise de desempenho: *speedup* e eficiência (HILL, 2006). O *speedup* consiste na relação entre o tempo gasto para a execução no caso sequencial e o tempo gasto para a execução no caso paralelo com *n* processadores. Sendo assim, a métrica de *speedup* tem o objetivo de determinar o ganho de tempo de uma aplicação executada com *n* processadores em comparação com a configuração sequencial (ou seja, com apenas um processador), conforme especificado na fórmula a seguir.

$$S_n = \frac{T_1}{T_n}$$

No caso ideal, o valor do *speedup* é igual ao número de processadores disponibilizados (*n*).

Já a eficiência é definida como a relação entre o *speedup* e o número de processadores alocados para a execução. Logo, essa métrica tem o objetivo de especificar o quanto é vantajoso adicionar processadores para a aplicação em paralelo (nesse caso, para a execução de *workflows* em ambiente de PAD). A fórmula a seguir apresenta a métrica de eficiência utilizada nos experimentos:

$$E_p = \frac{S_p}{p}$$

5.2 Configuração do Ambiente

A configuração do ambiente para as duas abordagens necessita da instalação da máquina virtual Java e do PostgreSQL, com o intuito de criar a base de proveniência para ambas estratégias (não adaptativa e adaptativa). Por outro lado, apenas a estratégia não adaptativa precisa instalar o MPJ (CARPENTER *et al.*, 2000), para que possa permitir a comunicação por mensagens entre nós. Além disso, a máquina virtual Java não precisou ser instalada, pois a mesma já faz parte de um dos módulos fornecidos pela Uranus. Já o PostgreSQL foi instalado em uma máquina conectada à rede local do NACAD, a fim de permitir o acesso à base de dados pelo *cluster* e apresentar um baixo custo de comunicação. Por último, o MPJ foi instalado e configurado na Uranus. Após a instalação dessas tecnologias mencionadas, dois arquivos XML foram configurados para a modelagem do *workflow* conceitual e de execução das duas abordagens.

As especificações das duas estratégias (não adaptativa e adaptativa) para *workflows* conceituais apresentam a estrutura propriamente dita do *workflow*, identificando as dependências entre atividades e como será realizada a invocação das atividades, sem especificar propriedades específicas da execução. Já o *workflow* de execução apresenta propriedades relacionadas ao processamento propriamente dito,

especificando o diretório da execução do *workflow*, o identificador da execução e os nomes dos arquivos que contêm as relações de entrada.

Além disso, diferentemente da estratégia não adaptativa, a especificação do *workflow* de execução da estratégia adaptativa possui alguns atributos extras no elemento *Workflow* (arquivo XML), sendo os seguintes: *max_failure*, *user_interaction*, *redundancy* e *reliability*. O atributo *max_failure* é responsável por permitir um número máximo de ocorrências de falhas em uma ativação durante a execução do *workflow*. No caso em que uma ativação falhou até o número máximo permitido, a estratégia adaptativa continua a execução do *workflow* sem considerar essa ativação e seus dados de saída na geração da relação de saída (da atividade a que essa ativação pertence). Contudo, o atributo *user_interaction* (do temo em português interação do usuário) permite a interrupção do *workflow* para o usuário realizar mudanças pontuais nos parâmetros da ativação com erro, quando esse atributo é definido como verdadeiro (valor *true*) no arquivo XML.

Já o atributo *redundancy* (do termo em português redundância) permite o escalonamento de ativações iguais para mais de um NC (aproximadamente 5 ativações redundantes), quando não há mais ativações diferentes para o escalonamento do *workflow*. Por último, o atributo *reliability* (do termo em português confiabilidade) permite uma análise dos NC em tempo de execução para a identificação do grau de confiabilidade dos mesmos, sendo que esse valor especificado pelo atributo corresponde ao valor mínimo necessário para que o NC permaneça na execução do *workflow*.

Para executar as duas estratégias (não adaptativa e adaptativa), o sistema de banco de dados PostgreSQL precisa ser configurado, a partir da criação de duas bases

de dados, uma para a estratégia não adaptativa (esquema original da base de dados) e outra para a estratégia adaptativa (esquema modificado de acordo com as adaptações das arquiteturas propostas). Outro ponto importante é que os arquivos XML do *workflow* conceitual diferem da estratégia não adaptativa para a estratégia adaptativa apenas no nome da base de dados. A Figura 16 apresenta a definição de um *workflow* conceitual para as duas estratégias. A Figura 17 define um *workflow* de execução usando esse *workflow* conceitual para a estratégia não adaptativa e a Figura 18 descreve um *workflow* de execução para a estratégia adaptativa (diferença apenas nos atributos adicionais explicados anteriormente). Vale enfatizar que esse *workflow* possui duas atividades, sendo a atividade *act2* dependente da atividade *act1*. Portanto, apenas a relação de entrada da atividade *act1* precisa ser informada pelo *workflow* de execução para começar o processamento desse *workflow*.

Somando-se a isso, após a instalação do MPJ, a execução dos *workflows* usando a estratégia não adaptativa requer a configuração de um arquivo para determinar as máquinas, as portas e os identificadores a serem usados por cada processo do MPI. Apenas para fim de exemplificação, a Figura 19 apresenta 4 máquinas a serem utilizadas para executar a estratégia não adaptativa na Uranus. Já a submissão do *job* que irá processar o *workflow* de execução necessita do desenvolvimento de um *script* PBS (BAYUCAN *et al*., 2000), como exemplificado pela Figura 20. Esse *script* apresenta os números de nós e de processadores utilizados (primeira linha do script) e a chamada da estratégia não adaptativa (linhas 15, 16 e 17). A Figura 20 apresenta o *script* PBS utilizado para 4 nós, sendo que cada nó possui 12 processadores (totalizando 48 *cores*).

```
<?xml version="1.0" standalone="no"?>
<Chiron>
  <database name="base" " server="localhost" port="5432"
username="postgres" password="pass"/>
  <Workflow tag="workflow_1" description="">
    <Activity tag="act1" description="" type="MAP" activation="java
-jar %=WFDIR%/bin/Program1.jar ID=%=ID% T2=%=T2%">
        <Relation reltype="Input" name="IAct1"/>
        <Relation reltype="Output" name="OAct1" />
        <Field name="ID" type="float" input="IAct1"
output="OAct1"/>
        <Field name="T1" type="float" input="IAct1"/>
        <Field name="T2" type="float" input="IAct1"
output="OAct1"/>
    </Activity>
    <Activity tag="act2" description="" type="MAP" activation="java
-jar %=WFDIR%/bin/Program2.jar ID=%=ID% T2=%=T2%">
        <Relation reltype="Input" name="IAct2" dependency="act1"/>
        <Relation reltype="Output" name="OAct2"/>
        <Field name="ID" type="float" input="IAct2"
output="OAct2"/>
        <Field name="T2" type="float" input="IAct2"
output="OAct2"/>
    </Activity>
  </Workflow>
</Chiron>
```

Figura 16. Definição de um *workflow* conceitual nas duas estratégias.

```
<?xml version="1.0" standalone="no"?>
<Chiron>
    <database name="chiron" server="localhost" port="5432"
username="postgres" password="pass"/>
    <Workflow tag="workflow_1" execmodel="DYN_FAF"
exectag="workflow_1-4-1"
wfdir="/scratch/silva/experiment/chiron/workflow_1/4"
expdir="%=WFDIR%/exp">
      <Relation name="IAct1" filename="Input.dataset"/>
    </Workflow>
</Chiron>
```

Figura 17. Definição de um *workflow* de execução na estratégia não adaptativa.

```
<?xml version="1.0" standalone="no"?>
<Chiron>
    <database name="demeter" server="localhost" port="5432"
username="postgres" password="pass"/>
    <Workflow tag="workflow_1" execmodel="DYN_FAF" maxfailure="5"
userinteraction="false" redundancy="true" reliability="0.9"
exectag="workflow_1-4-1"
wfdir="/scratch/silva/experiment/demeter/workflow_1/4"
expdir="%=WFDIR%/exp">
      <Relation name="IAct1" filename="Input.dataset"/>
    </Workflow>
</Chiron>
```

Figura 18. Definição de um *workflow* de execução na estratégia adaptativa .

```
# Number of Processes
4
# Protocol switch limit
131072
# Entry in the form of machinename@port@rank
r1i0n1@20000@0
r1i0n2@20002@1
r1i0n3@20004@2
r1i0n4@20006@3
```

Figura 19. Arquivo de configuração do MPJ para a estratégia não adaptativa.

```
#PBS -l nodes=4:ppn=12
#PBS -l walltime=06:00:00
#PBS -S /bin/bash
#PBS -N workflow_1-4
#PBS -j oe
#PBS -V
#PBS -m ae
#PBS -M silva@cos.ufrj.br
# change directory
cd ${PBS_O_WORKDIR}
# create list with names from data network
sort ${PBS_NODEFILE} | uniq > ~/tmp/${PBS_JOBID}
java -jar ~/bin/mpjEnv.jar ~/tmp/${PBS_JOBID}
cd $MPJ_HOME/bin
./mpjmanual.sh ~/tmp/${PBS_JOBID}.conf $EXP/bin/chiron.jar MPI 12
$EXP/chiron/workflow_1/4/EModel.xml
```

Figura 20. Arquivo de *job* no sistema PBS para a estratégia não adaptativa.

A estratégia adaptativa também precisa de um arquivo de configuração semelhante ao do MPJ, mas apenas com a identificação das máquinas, conforme a Figura 21. Ao contrário da estratégia não adaptativa, a estratégia adaptativa utiliza esse arquivo apenas para acessar as máquinas que executarão o *workflow*, sendo a definição da porta TCP utilizada definida automaticamente por essa abordagem e sem um identificador para a instância, como requerido pelo MPI. Além disso, o *script* PBS da estratégia adaptativa também apresenta uma modificação em relação à estratégia não adaptativa, baseada em uma linha de comando diferente para a invocação do Demeter, conforme apresentado pela Figura 22.

```
# Number of Processes
4
# Entry in the form of machine name
r1i0n1
r1i0n2
r1i0n3
r1i0n4
```

Figura 21. Arquivo de configuração do MPJ para a estratégia adaptativa .

```
#PBS -l nodes=4:ppn=12
#PBS -l walltime=06:00:00
#PBS -S /bin/bash
#PBS -N workflow_1-4
#PBS -j oe
#PBS -V
#PBS -m ae
#PBS -M silva@cos.ufrj.br
# change directory
cd ${PBS_O_WORKDIR}
# create list with names from data network
sort ${PBS_NODEFILE} | uniq > ~/tmp/${PBS_JOBID}
java -jar ~/bin/adaptiveEnv.jar ~/tmp/${PBS_JOBID}
./starter.sh ~/tmp/${PBS_JOBID}.conf $EXP/bin/demeter.jar 12
$CHIRON_EXP/demeter/workflow_1/4/EModel.xml
```

Figura 22. Arquivo de *job* no sistema PBS para a estratégia adaptativa.

A partir das configurações apresentadas, os estudos de caso puderam ser executados na Uranus, sendo os seus resultados apresentados na seção 5.3.

5.3 Resultados Experimentais

As figuras a seguir apresentam os resultados experimentais de *speedup* e eficiência para a execução das duas estratégias (não adaptativa e adaptativa) para os estudos de caso realizados. É importante destacar que os tempos de execução sequencial (1 *core*) são valores teóricos e que os resultados correspondem a três execuções dos *workflows* e suas configurações.

No primeiro estudo de caso, a nossa análise será dedicada, primeiramente, às configurações 1 e 2 da Tabela 2, que apresentam um baixo custo para o processamento da atividade. Essas configurações (Figuras 23 e 24, respectivamente) apresentaram desempenhos (valores de *speedup* e eficiência) ligeiramente diferentes, sendo que a estratégia adaptativa apresentou uma diferença média na eficiência para a estratégia

não adaptativa de aproximadamente 8% na configuração 1 e de 9% na configuração 2. Pelos resultados é possível constatar que a partir de 4 nós (totalizando 48 *cores*), a eficiência apresenta uma queda considerável para a estratégia não adaptativa, com um valor inferior a 60% de eficiência. Enquanto isso, a estratégia adaptativa apresenta essa mesma queda apenas com 8 nós (totalizando 96 *cores*). Esse comportamento é constatado pelo fato das ativações serem muito curtas, resultando em uma sobrecarga do nó central para processar diferentes pedidos para distribuir novas ativações pendentes e para armazenar os dados das ativações processadas na base de proveniência.

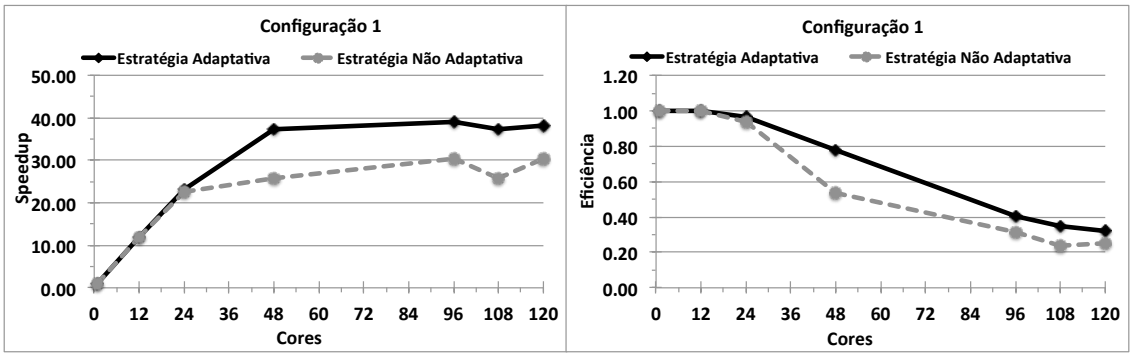

Figura 23. Resultados experimentais para a configuração 1.

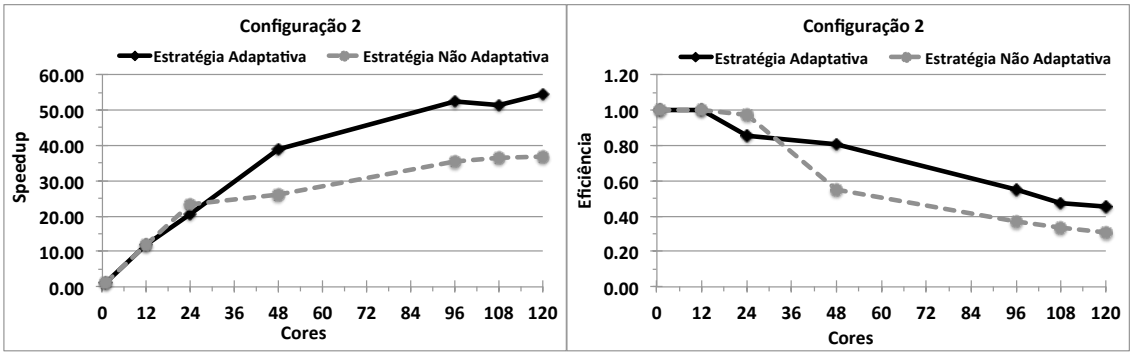

Figura 24. Resultados experimentais para a configuração 2.

Ao mesmo tempo, tal comportamento pode estar relacionado ao modelo de execução adotado para as atividades curtas. Nesse sentido, a configuração 1 foi modificada e reexecutada com o modelo de execução estático, a fim de distribuir um conjunto de ativações, ao invés de apenas uma ativação por pedido. Sendo assim, o

tempo de processamento de cada pedido é superior, reduzindo o custo de comunicação e evitando a sobrecarga do nó central. Os resultados apresentados na Figura 25 comprovam tal comportamento, uma vez que o desempenho para configurações com mais de 48 *cores* apresentou um desempenho superior ao constatado no modelo de execução dinâmico. Além disso, pode-se observar uma aproximação no desempenho das abordagens para configurações com até 4 nós, enquanto que as configurações a partir de 8 nós apresentaram melhores resultados para a estratégia adaptativa. Quantitativamente, o uso do modelo de execução estática apresentou uma eficiência superior a aproximadamente 20% em comparação com o modelo de execução dinâmica.

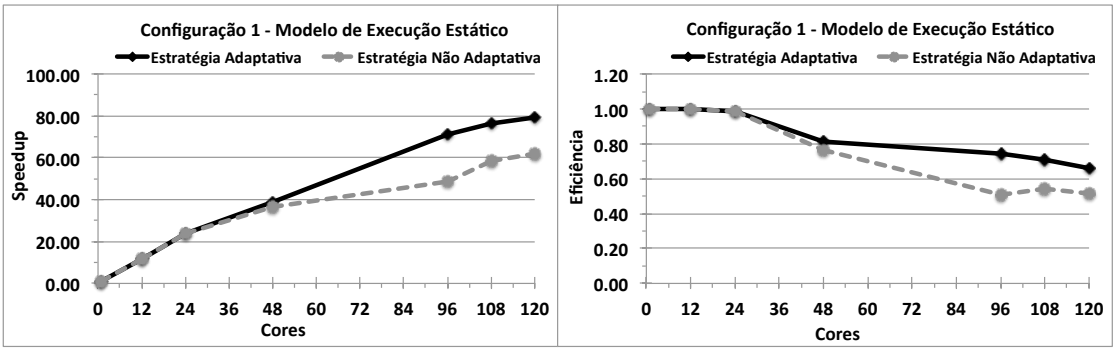

Figura 25. Resultados experimentais para a configuração 1 usando o modelo de execução estático.

As configurações 3, 4 e 5 apresentam um custo associado à sobrecarga para a execução das atividades (Figuras 26, 27 e 28). Nessas configurações, os resultados experimentais possuem desempenhos mais próximos em termos de *speedup* e eficiência entre as estratégias. Tais configurações possuem valores de eficiência melhores de acordo com o aumento do custo da atividade, quando comparados aos das configurações 1 e 2. Além disso, percebe-se o aumento da diferença entre as estratégias de acordo com o aumento da complexidade das atividades. Tal comportamento pode estar associado a capacidade de troca de mensagens assíncronas entre os diferentes

processadores na estratégia adaptativa, frente à troca de mensagens síncronas da estratégia não adaptativa.

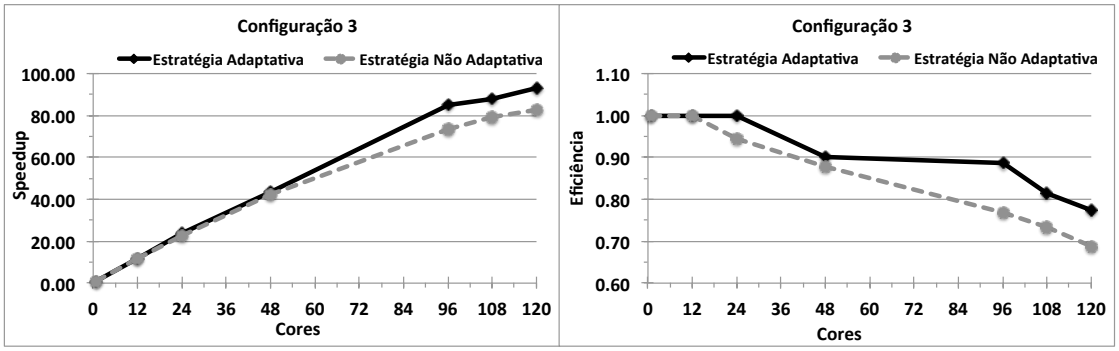

Figura 26. Resultados experimentais para a configuração 3.

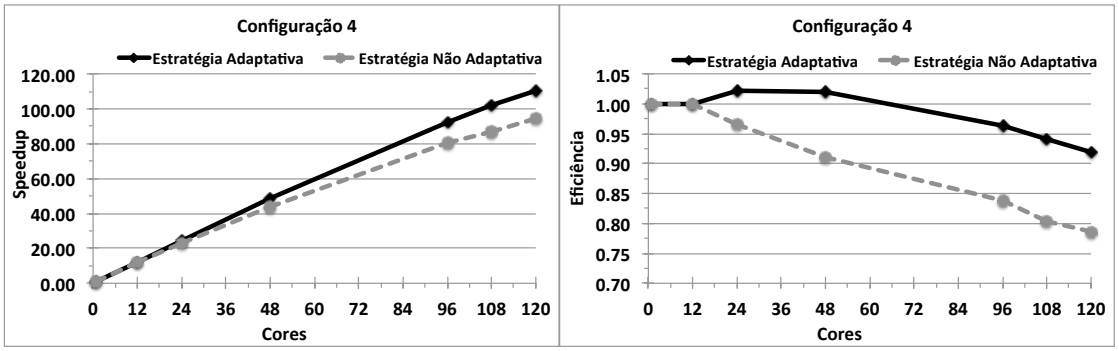

Figura 27. Resultados experimentais para a configuração 4.

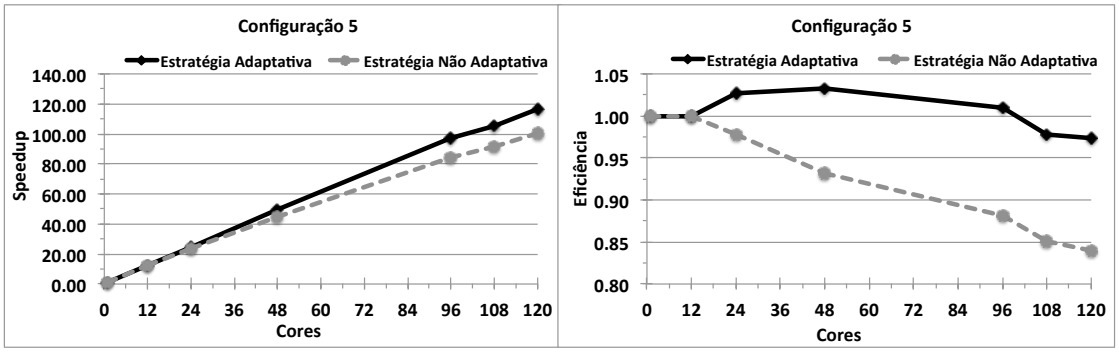

Figura 28. Resultados experimentais para a configuração 5.

Outra característica importante é a ocorrência de alguns resultados com *speedup* acima do número de processadores utilizados (ou seja, eficiência acima de 1,0), sendo esse comportamento conhecido como super linear. Esse comportamento pode ter ocorrido pelo efeito *cache* resultante da hierarquia de memória do *cluster* Uranus, uma vez que aumenta tanto com o número de processadores envolvidos na

execução de aplicações paralelas, como também com o tamanho acumulado de memória *cache* pelos processadores disponibilizados. Portanto, em caso de um acúmulo grande de memória *cache*, os dados envolvidos em um determinado processamento computacional podem utilizar apenas a memória cache disponível, reduzindo drasticamente o tempo de acesso a esses dados. Consequentemente, pode-se observar esse ganho extra no resultado do cálculo de eficiência.

Portanto, espera-se um desempenho melhor da estratégia adaptativa para as configurações 6 e 7. As Figuras 29 e 30 confirmam os resultados esperados para os casos com atividades longas, além de valores de *speedup* super linear para a estratégia adaptativa a partir de 24 *cores*.

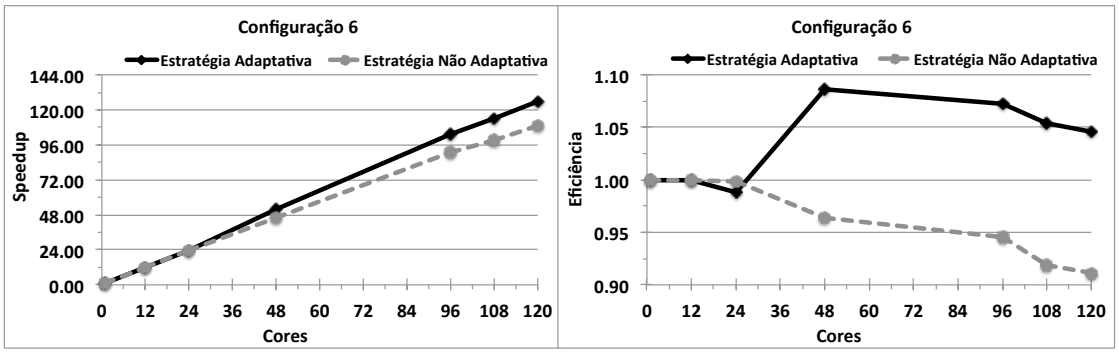

Figura 29. Resultados experimentais para a configuração 6.

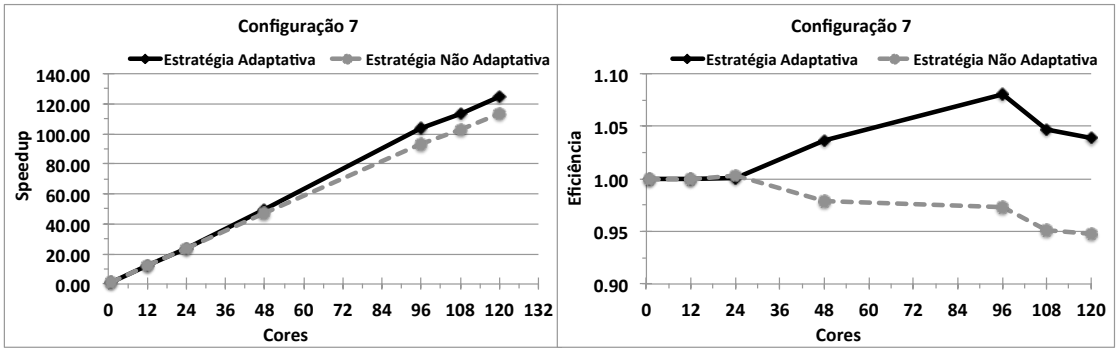

Figura 30. Resultados experimentais para a configuração 7.

Pelos resultados obtidos para cada configuração do primeiro estudo de caso, observa-se que a principal diferença nos valores de *speedup* e de eficiência entre a estratégia adaptativa e a estratégia não adaptativa está relacionada ao mecanismo de

troca de mensagens assíncronas. Sendo assim, uma análise foi realizada usando três estratégias: adaptativa (Demeter), não adaptativa (Chiron) e uma adaptativa, que consistiu na mudança do Demeter para o apoio à troca de mensagens síncronas. O objetivo dessa análise é determinar se os melhores desempenho do Demeter em relação ao Chiron estão restritos à troca de mensagens assíncronas e se há alguma vantagem associada aos algoritmos de adaptação desenvolvidos, como o algoritmo de *keepalive*.

A Figura 31 apresenta a comparação dessas estratégias utilizando o *workflow* do primeiro estudo de caso com a configuração 4, pois consiste da configuração média para todos os valores de ACF executados. Por essa figura, pode-se constatar que ao utilizar o Demeter com troca de mensagens síncronas, o seu desempenho sofre uma queda considerável. Entretanto, os valores de eficiência para o Demeter com mensagens síncronas são superiores à estratégia não adaptativa. Tal comportamento está associado ao fato de apesar de não ter a troca de mensagens assíncronas, essa versão do Demeter (com mensagens síncronas) ainda possui os algoritmos para análise de disponibilidade dos NC e de cálculo de confiabilidade. Ao considerar o erro associado às execuções para cada configuração quanto ao número de *cores*, percebe-se que para as configurações com poucos *cores* podem ser ditas com mesmo desempenho para a estratégia não adaptativa e o Demeter com mensagens síncronas.

Portanto, por essa análise baseada em estratégias quanto ao tipo de mecanismo de troca de mensagens (síncrono ou assíncrono), pode-se constatar que os experimentos realizados para as diferentes configurações no primeiro estudo de caso apresentaram o comportamento esperado pela análise da Figura 31. Além disso, a diferença de desempenho entre a estratégia adaptativa e a estratégia não adaptativa mantiveram as mesmas proporções, destacando que a diferença de desempenho tem grande contribuição quanto ao mecanismo de troca de mensagens. Em outras palavras,

caso a estratégia adaptativa fosse baseada na troca de mensagens síncronas, a diferença de desempenho em relação à estratégia não adaptativa seria desprezível. Contudo, isso seria vantajoso do ponto de vista desta dissertação devido às diferentes características e algoritmos desenvolvidos para apoiar a adaptabilidade na execução paralela de *workflows* científicos.

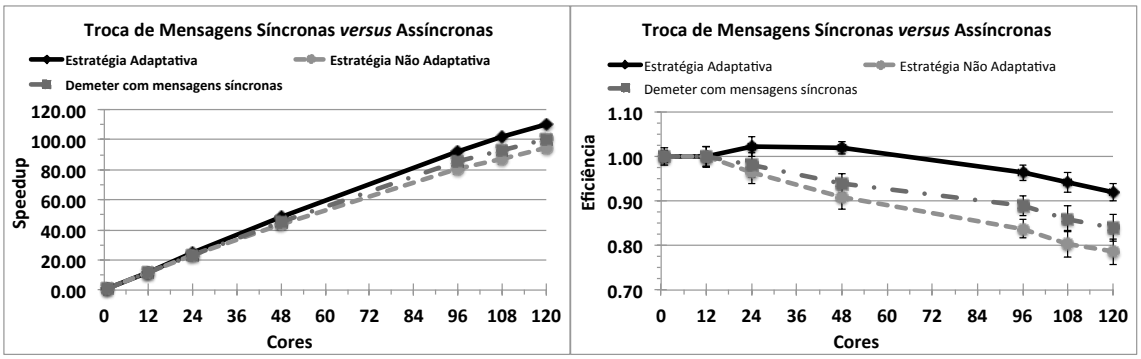

Figura 31. Comparação da troca de mensagens assíncronas e síncronas.

Já o segundo estudo de caso apresentou resultados diferentes dos discutidos no outro estudo. A Figura 32 apresenta os resultados experimentais do *workflow* Montage desse segundo estudo. Pelos resultados é possível observar um pior desempenho para a estratégia adaptativa nas primeiras configurações em número de *cores*, 24 e 48 *cores*, respectivamente. Esse comportamento inferior da estratégia adaptativa pode estar relacionado a duas características: modelagem do *workflow* com um operador bloqueante, nesse caso o MR Query; e o efeito cache dos NC.

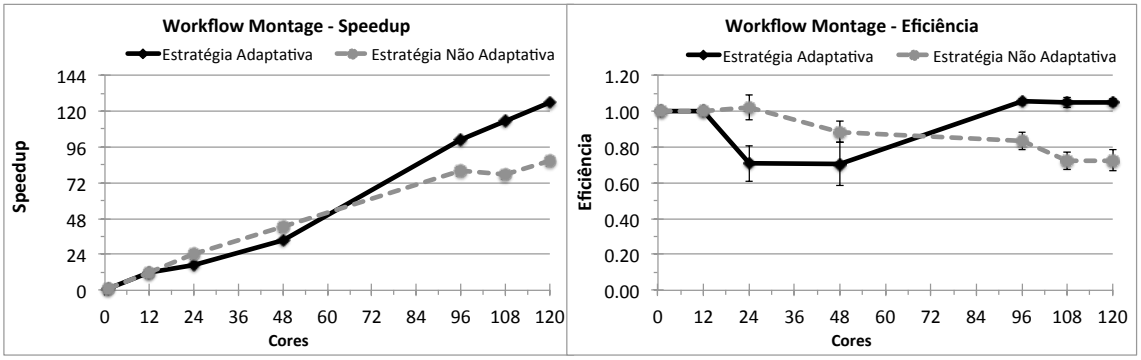

Figura 32. Resultados experimentais com o *workflow* Montage.

Em mais detalhes, a existência de uma atividade com o operador MR Query faz com que haja o bloqueio da execução nos outros NC, para que o nó central obtenha os dados produzidos pelas atividades dependentes e processe os dados por esse operador. Nessa situação, os outros NC podem ou não apresentar outras ativações sendo processadas durante a execução da atividade com o operador MR Query, interferindo consideravelmente no desempenho (em função da ociosidade dos outros NC). Como o mecanismo de troca de mensagens (síncrono ou assíncrono) é diferente para as estratégias adaptativas e não adaptativas, cada estratégia apresentará uma configuração específica ao começar a execução do operador MR Query. No caso do Demeter, identificou-se que o número de ativações pendentes era muito inferior, não favorecendo o paralelismo ao iniciar a execução de alguns operadores MR Query. Por outro lado, o Chiron possuía muitas ativações pendentes ao inicializar a maioria dos operadores MR Query, favorecendo o paralelismo.

Em relação ao efeito cache, no caso com configurações com poucos processadores, o acúmulo de memória cache não é suficiente para permitir um acesso rápido aos dados e favorecer a distribuição de ativações para os NC ociosos antes do início da atividade com o operador MR Query.

Apesar desse pior desempenho, em alguns casos, como para 12 e 48 *cores*, se considerarmos o erro associado das execuções, a eficiência das duas estratégias podem ser ditas iguais. Nesse caso, apenas a configuração para 24 *cores* apresentou um desempenho realmente inferior para a estratégia adaptativa.

Contudo, para configurações a partir de 96 *cores*, a estratégia adaptativa apresentou desempenho melhor do que a estratégia não adaptativa, sendo a diferença média de eficiência igual a 0,27. Já nesse caso, o efeito cache e a troca de mensagens

assíncronas favoreceram a estratégia adaptativa pela distribuição das ativações para os NC ociosos. Além disso, as execuções com configurações a partir 96 *cores* confirmaram esse comportamento por apresentarem o valor de eficiência superior a 1,0. Já em relação à existência do operador MR Query, o comportamento instável dos resultados evidencia essa distribuição não uniforme dos NC antes de inicializar esse operador algébrico.

A partir dos resultados experimentais e das análises realizadas, pode-se observar que o desenvolvimento da arquitetura para a execução paralela adaptável de *workflows* científicos não apresentou uma sobrecarga que afetasse consideravelmente o desempenho da estratégia adaptativa, quando comparada à estratégia não adaptativa. Nesse sentido, para o primeiro experimento, os casos de atividades curtas e médias apresentaram valores de *speedup* e de eficiência semelhantes, enquanto que nos casos de atividades longas constatou-se um desempenho superior da estratégia adaptativa..

Além disso, quando o *workflow* Montage foi analisado, a estratégia adaptativa apresentou dificuldades relacionadas a *workflows* modelados com operadores MR Query (operador bloqueante semelhante ao SR Query) em configurações de pequena escala (utilizando menos de 100 *cores*). Contudo, considerando a necessidade de execução em larga escala e a disponibilidade cada vez maior de *cores*, o efeito desse operador bloqueante foi minimizado pelo aproveitamento do cache dos processadores alocados. Cabe ressaltar também que o comportamento da curva de *speedup* não apresenta sinais de queda com mais de 100 *cores*, o que indica o potencial da estratégia, ao contrário da não adaptativa, confirmando resultados preliminares de simulação realizados com 256 *cores*.

Capítulo 6 – Conclusão

Esta dissertação se encaixa no cenário de experimentos científicos que demandam cada vez mais um volume de dados maior, motivando o uso de técnicas de paralelismo. As técnicas de paralelismo são empregadas com o intuito de reduzir o tempo total de execução dos experimentos. Além disso, devido ao encadeamento de programas e serviços nos experimentos científicos, os *workflows* científicos são utilizados para modelar tais experimentos por meio do encadeamento de atividades. Cada atividade seria, portanto, responsável pelo processamento de um programa ou serviço.

Os SGWfC são os sistemas que proporcionam a execução dos *workflows* científicos em ambientes de PAD, a partir do emprego de técnicas de computação paralela. A principal característica desejável desses sistemas é a eficiência no uso dos NC, a fim de garantir um escalonamento eficiente das ativações para os NC ao mesmo tempo em que evita a ociosidade desses nós, a interrupção da execução em função de falhas nos NC e o desbalanceamento de carga. Nesse sentido, a adaptabilidade vai ao encontro da execução paralela de *workflows* científicos, pois proporciona a capacidade de o sistema adicionar e remover NC em tempo de execução, de acordo com fatores como o prazo para a execução de um dado experimento (definido pelos cientistas), o custo financeiro (para ambientes de nuvem computacional) ou mesmo a ocorrência de falhas.

Destaca-se também a importância do apoio à captura e consulta do fluxo de dados envolvidos na execução dos *workflows* científicos, uma vez que permite que os dados de proveniência evolvidos na condução de experimentos científicos favoreçam a confiabilidade e a reprodutibilidade dos mesmos. Vale ressaltar ainda o suporte de

consultas analíticas por essa propriedade, na medida em que dados de domínio podem ser associados ao fluxo de dados envolvidos no processamento dos *workflows*.

Diante desse contexto, esta dissertação apresentou uma estratégia de execução paralela adaptável para *workflows* científicos implementada com o sistema Demeter, tendo como meta o escalonamento adaptativo eficiente de *workflows* científicos, a tolerância a falhas e o balanceamento de carga. Para isso, o Demeter modificou um SGWfC paralelo, nesse caso o Chiron, ao incorporar as arquiteturas dos componentes AWAPE e APON. Além disso, esta dissertação contribui com a proposta de uma estratégia adaptativa bem sucedida que pode ser aplicada a outros SGWfC através das arquiteturas AWAPE e APON. Alguns trabalhos realizados ao longo da elaboração desta dissertação geraram publicações direta ou indiretamente ligadas ao Demeter (LIU *et al.*, 2014, MATTOSO *et al.*, 2013, OLIVEIRA *et al.*, 2014, SILVA *et al.*, 2013, 2014).

Diferentemente das principais soluções adaptativas existentes para *workflows* científicos (LEE *et al.*, 2008, PAPUZZO e SPEZZANO, 2011, RASOOLI e DOWN, 2011), que utilizam um escalonamento adaptativo através de um mapeamento das ativações para todos os NC após eventuais adaptações, o Demeter realiza o escalonamento adaptativo do *workflow* por meio de uma análise dedicada a cada NC. Além disso, o diferencial associado ao uso da abordagem algébrica relacional proposta por OGASAWARA *et al.* (2011) também pode ser destacado pela captura e consulta dos dados de proveniência (e o fluxo de dados) em tempo de execução, favorecendo consultas analíticas.

No que diz respeito à eficiência do Demeter, a mesma foi avaliada nos experimentos realizados por meio da variação da complexidade das atividades de

workflows científicos representativos e do número de *cores* alocados para execuções reais. Os experimentos avaliaram a escalabilidade da solução ao mostrar que o ganho em desempenho foi preservado com diferentes configurações dessas variáveis. Já a adaptabilidade foi analisada do ponto de vista da adição e remoção de NC e pela identificação de falhas ou eventos de volatilidade durante o processamento do *workflow*, sendo esse primeiro aspecto adaptado da proposta do SciMultaneous (COSTA *et al.*, 2012). Dessa forma, a recuperação de falhas foi realizada pela identificação e remoção do NC com falha, caso esse erro fosse relacionado ao recurso físico (de *hardware*). Em caso de outro tipo de falha, o NC é reinicializado. Além disso, os dados de proveniência são analisados durante a recuperação de falhas com o intuito de preservar as propriedades da álgebra relacional de *workflows* centrada em dados.

Em relação à eficiência, o mecanismo de troca de mensagens, antes desenvolvido de forma síncrona pelo Chiron, foi modificado para ser executado de forma assíncrona, evitando condições de bloqueio no envio e recebimento de mensagens. Além disso, o mecanismo para troca de mensagens foi modificado para apoiar as técnicas P2P empregadas pela plataforma JXTA. Da mesma forma, técnicas P2P foram utilizadas para garantir a adaptabilidade, a partir da identificação de NC com falhas pelo algoritmo de *keepalive* e por falhas de *software* e *hardware* em tempo de execução, pela publicação de anúncios na rede P2P e pelo monitoramento da execução das ativações. Já no que diz respeito à adaptação pela recuperação de falhas, a mesma foi realizada por meio da análise da estrutura do *workflow*. No caso de falhas no nível de *hardware*, as mudanças foram realizadas pela plataforma JXTA, como a remoção de um determinado NC.

Os experimentos realizados para comparar o desempenho da estratégia adaptativa do Demeter com a proposta não adaptativa do Chiron foram executados em dois estudos de caso. No primeiro estudo, sete configurações de um *workflow* mapeado apenas com atividades do operador *Map* foram executadas, sendo que cada configuração apresentou o mesmo número de ativações por atividade, nesse caso 1024 ativações, e variou-se o custo de processamento das atividades. Para cada configuração, o mesmo *workflow* foi executado diferentes vezes, variando-se o número de *cores* utilizados, sendo iguais a 12, 24, 48, 96, 108 e 120 *cores*. Os tempos de execução sequencial (1 *core*) são valores teóricos.

Pela análise dos resultados experimentais, constatou-se um desempenho ligeiramente superior para a estratégia adaptativa em relação à estratégia não adaptativa para atividades curtas e médias. Por outro lado, percebeu-se uma dificuldade de escalar os *workflows* com atividades curtas, uma vez que havia uma sobrecarga do nó central com muitos pedidos para distribuir ativações pendentes ao utilizar o modelo de execução dinâmico. Para comprovar tal comportamento, alterou-se o modelo de execução para o modo estático, que cria um conjunto de ativações curtas para depois distribuir tal conjunto para um mesmo NC. Assim evita-se a sobrecarga do nó central, como evidenciado pelos resultados experimentais. Já as configurações com atividades longas apresentaram um desempenho bem superior para a abordagem do Demeter.

Por último, o segundo estudo consistiu na análise de *speedup* e eficiência para o *workflow* de astronomia Montage, representando aspectos de um experimento científico real e usado como um *benchmark* de fato em *workflows* científicos. Nos experimentos com *workflow* Montage, constatou-se que ao aumentar o número de processadores envolvidos, a eficiência para a estratégia adaptativa apresentou um ganho de em média 27% em relação à estratégia não adaptativa.

Portanto, a partir da proposta desta dissertação de proporcionar uma estratégia de execução paralela adaptável de *workflows* científicos centrada em dados e dos resultados experimentais, a estratégia do Demeter apresentou um desempenho melhor na maioria dos *workflows* executados nos dois estudos de caso existentes, ao ser comparado com a estratégia não adaptativa do Chiron. Além disso, a abordagem P2P do Demeter poderia ser aplicada em outras funções de um *workflow* científico, como a gerência de fragmentação e replicação dos dados da aplicação, o controle de notificações ou mesmo a consulta de dados de proveniência de modo distribuído.

Referências Bibliográficas

AALST, W., HEE, K., 2002, *Workflow Management: Models, Methods, and Systems*. The MIT Press.

ALMEIDA, E., SUNYÉ, G., TRAON, Y., *et al.*, 2008, "A Framework for Testing Peer-to-Peer Systems". In: *Proceedings of the 19th International Symposium on Software Reliability Engineering*, pp. 167–176, Los Alamitos, CA, USA.

ALTINTAS, I., BERKLEY, C., JAEGER, E., *et al.*, 2004, "Kepler: an extensible system for design and execution of scientific workflows". In: *Scientific and Statistical Database Management*, pp. 423–424, Greece.

AMAZON EC2, 2010, *Amazon Elastic Compute Cloud (Amazon EC2)*, http://aws.amazon.com/ec2/.

ANDERSON, E., FREIRE, J., KOOP, D., *et al.*, 2007, *Provenance Challenge - Vistrails*. Disponível em: http://twiki.ipaw.info/bin/view/Challenge/VisTrails,

ASSAYAD, I., GIRAULT, A., KALLA, H., 2004, "A bi-criteria scheduling heuristic for distributed embedded systems under reliability and real-time constraints". *2004 International Conference on Dependable Systems and Networks*, pp. 347–356

BARGA, R. S., FAY, D., GUO, D., *et al.*, 2008, "Efficient scheduling of scientific workflows in a high performance computing cluster". In: *Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments*, pp. 63–68, Boston, MA, USA.

BAROLLI, L., XHAFA, F., 2011, "JXTA-Overlay: A P2P Platform for Distributed, Collaborative, and Ubiquitous Computing", *IEEE Transactions on Industrial Electronics*, v. 58, n. 6, pp. 2163–2172.

BAYUCAN, A., HENDERSON, R. L., JONES, J. P., 2000, *Portable Batch System Administration Guide*. Mountain View, CA, USA, Veridian Systems.

BERRIMAN, G. B., DEELMAN, E., GOOD, J., *et al.*, 2007, "Generating Complex Astronomy Workflows", *Workflows for e-Science*, Springer, pp. 19–38.

BOERES, C., SARDIÑA, I., DRUMMOND, L., 2011, "An efficient weighted bi-objective scheduling algorithm for heterogeneous systems", *Parallel Computing*, v. 37, n. 8 (Agosto.), pp. 349–364.

BOUGANIM, L., FLORESCU, D., VALDURIEZ, P., 1996, "Dynamic load balancing in hierarchical parallel database systems". In: *Proceedings of the 22nd International Conference on Very Large Databases (VLDB)*, pp. 436–447

BROWN, D. A., BRADY, P. R., DIETZ, A., *et al.*, 2007, "A Case Study on the Use of Workflow Technologies for Scientific Analysis: Gravitational Wave Data Analysis", *Workflows for e-Science*, Springer, pp. 39–59.

BUX, M., LESER, U., 2013, *Parallelization in Scientific Workflow Management Systems*, CoRR/arXiv:1303.7195.

CALLAHAN, S. P., FREIRE, J., SANTOS, E., *et al.*, 2006, "VisTrails: visualization meets data management". In: *SIGMOD International Conference on Management of Data*, pp. 745–747, Chicago, Illinois, USA.

CARPENTER, B., GETOV, V., JUDD, G., *et al.*, 2000, "MPJ: MPI-like message passing for Java", *Concurrency: Practice and Experience*, v. 12, n. 11, pp. 1019–1038.

CHIRIGATI, F., SILVA, V., OGASAWARA, E., *et al.*, 2012, "Evaluating Parameter Sweep Workflows in High Performance Computing". In: *Proceeding of the 1st International Workshop on Scalable Workflow Enactment Engines and Technologies (SWEET'12)SIGMOD/PODS 2012*, Scottsdale, AZ, EUA.

CONDOR TEAM, 2005, *DAGMan: A Directed Acyclic Graph Manager, July 2005*.

COSTA, F., OLIVEIRA, D. DE, OCAÑA, K., *et al.*, 2012, "Handling Failures in Parallel Scientific Workflows Using Clouds". In: *High Performance Computing, Networking Storage and Analysis, SC Companion*, pp. 129–139, Los Alamitos, CA, USA.

COUVARES, P., KOSAR, T., ROY, A., *et al.*, 2007, "Workflow Management in Condor", *Workflows for e-Science*, Springer, pp. 357–375.

DANTAS, M., 2005, *Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais*. 1 ed. Rio de Janeiro, Axcel Books.

DEELMAN, E., MEHTA, G., SINGH, G., *et al.*, 2007, "Pegasus: Mapping Large-Scale Workflows to Distributed Resources", *Workflows for e-Science*, Springer, pp. 376–394.

DENG, K., KONG, L., SONG, J., *et al.*, 2011, "A Weighted K-Means Clustering Based Co-scheduling Strategy towards Efficient Execution of Scientific Workflows in Collaborative Cloud Environments". In: *Proceedings of the 9th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pp. 547–554

DIAS, J., OGASAWARA, E., DE OLIVEIRA, D., *et al.*, 2013, "Algebraic dataflows for big data analysis". In: *2013 IEEE International Conference on Big Data*, pp. 150–155

FOSTER, I., ZHAO, Y., RAICU, I., *et al.*, 2008, "Cloud Computing and Grid Computing 360-Degree Compared". In: *Grid Computing Environments Workshop, 2008. GCE '08*, pp. 10, 1

FREIRE, J., KOOP, D., SANTOS, E., *et al.*, 2008, "Provenance for Computational Tasks: A Survey", *Computing in Science and Engineering, v.10*, n. 3, pp. 11–21.

GOECKS, J., NEKRUTENKO, A., TAYLOR, J., 2010, "Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences", *Genome Biology*, v. 11, n. 8, pp. 86.

GRADECKI, J., 2002, *Mastering JXTA: building Java peer-to-peer applications*. Indianapolis, Ind, Wiley Pub.

GUNTER, D., DEELMAN, E., SAMAK, T., *et al.*, 2011, "Online workflow management and performance analysis with Stampede". In: *Proceedings of the 7th International Conference on Network and Service Management (CNSM)*, pp. 1 –10

HAYEK, R., RASCHIA, G., VALDURIEZ, P., *et al.*, 2008, "Summary management in P2P systems". In: *Proceedings of the 11th international conference on Extending database technology: Advances in database technology*, pp. 16

HILL, M., 2006, "What is scalability?", *Proceedings of the 28th International Conference on Software Engineering*, v. 18, n. 4, pp. 18 – 21.

HOFFA, C., MEHTA, G., FREEMAN, T., *et al.*, 2008, "On the use of cloud computing for scientific workflows". In: *Proceedings of the 4th IEEE International Conference on eScience (eScience 2008)*, pp. 7–12, Indianapolis, USA.

HUANG, C., ZHENG, G., KALÉ, L., *et al.*, 2006, "Performance evaluation of adaptive MPI". In: *Proceedings of the 11th ACM SIGPLAN symposium on Principles and practice of parallel programming*, pp. 12

JACOB, J. C., KATZ, D. S., BERRIMAN, G. B., *et al.*, 2009, "Montage: a grid portal and software toolkit for science-grade astronomical image mosaicking", *International Journal of Computational Science and Engineering (IJCSE)*, v. 4, n. 2, pp. 73–87.

KAZAA, 2011, *Kazaa Media Desktop*, http://www.kazaa.com/.

KEPHART, J., DAS, R., 2007, "Achieving Self-Management via Utility Functions", *IEEE Internet Computing*, v. 11, n. 1, pp. 40–48.

KIM, T., KIM, E., KIM, J., *et al.*, 1995, "A leader election algorithm in a distributed computing system". In: *Proceedings of the 5th IEEE Computer Society Workshop on Future Trends of Distributed Computing Systems*, pp. 481–485

KREITZ, G., NIEMELA, F., 2010, "Spotify -- Large Scale, Low Latency, P2P Music-on-Demand Streaming". In: *Proceedings of the 10th IEEE International Conference on Peer-to-Peer Computing (P2P)*, pp. 1–10

LEE, C., SUZUKI, J., VASILAKOS, A., *et al.*, 2010, "An evolutionary game theoretic approach to adaptive and stable application deployment in clouds". In: *Proceeding of the 2nd Workshop on Bio-inspired algorithms for distributed systems*, pp. 29–38, Washington, DC, USA.

LEE, K., PATON, N. W., SAKELLARIOU, R., *et al.*, 2008, "Adaptive Workflow Processing and Execution in Pegasus". In: *Proceedings of the 3rd International Conference on Grid and Pervasive Computing*, pp. 99–106, Kunming, China.

LEE, K., PATON, N. W., SAKELLARIOU, R., *et al.*, 2011, "Utility functions for adaptively executing concurrent workflows", *Concurrency and Computation: Practice and Experience*, v. 23, n. 6 (Apr.), pp. 646–666.

LIU, J., SILVA, V., PACITTI, E., *et al.*, 2014, "Scientific Workflow Partitioning in Multisite Cloud". In: *Proceeding of the 3rd Workshop on Big Data Management in CloudsEuro-Par 2014 Parallel Processing*, Porto, Portugal.

LUDÄSCHER, B., ALTINTAS, I., BERKLEY, C., *et al.*, 2006, "Scientific workflow management and the Kepler system", *Concurrency and Computation: Practice and Experience*, v. 18, n. 10, pp. 1039–1065.

MATTOSO, M., OCAÑA, K., HORTA, F., *et al.*, 2013, "User-steering of HPC workflows: state-of-the-art and future directions". In: *Proceedings of the 2nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies (SWEET)*, pp. 1–6, New York, NY, USA.

MATTOSO, M., WERNER, C., TRAVASSOS, G. H., *et al.*, 2010, "Towards Supporting the Life Cycle of Large-scale Scientific Experiments", *International Journal of Business Process Integration and Management*, v. 5, n. 1, pp. 79–92.

MISSIER, P., SOILAND-REYES, S., OWEN, S., *et al.*, 2010, "Taverna, reloaded". In: *Proceedings of the 22nd International Conference on Scientific and Statistical Database Management*, pp. 471–481, Berlin, Heidelberg.

NACAD. Núcleo Avançado de Computação de Alto Desempenho., 2014 Disponível em: http://www.nacad.ufrj.br/. Acesso em: 9 Apr 2014.

OCAÑA, K. A. C. S., OLIVEIRA, D., OGASAWARA, E., *et al.*, 2011, "SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes". In: *Advances in Bioinformatics and Computational Biology (BSB)*, pp. 66–70, Berlin, Heidelberg.

OGASAWARA, E., 2011, *Uma Abordagem Algébrica para Workflows Científicos com Dados em Larga Escala*, Universidade Federal do Rio de Janeiro

OGASAWARA, E., DIAS, J., OLIVEIRA, D., *et al.*, 2011, "An Algebraic Approach for Data-Centric Scientific Workflows", *Proceedings of the 37th International Conference on Very Large Data Bases (PVLDB)*, v. 4, n. 12, pp. 1328–1339.

OGASAWARA, E., DIAS, J., SILVA, V., *et al.*, 2013, "Chiron: A Parallel Engine for Algebraic Scientific Workflows", *Concurrency and Computation*, v. 25, n. 16, pp. 2327–2341.

OLIVEIRA, D., COSTA, F., SILVA, V., *et al.*, 2014, "Debugging Scientific Workflows with Provenance: Achievements and Lessons Learned". In: *Anais do XXIX Simpósio Brasileiro de Banco de Dados*, Curitiba, Paraná.

OLIVEIRA, D., OCAÑA, K., BAIÃO, F., *et al.*, 2012, "A Provenance-based Adaptive Scheduling Heuristic for Parallel Scientific Workflows in Clouds", *Journal of Grid Computing*, v. 10, n. 3, pp. 521–552.

OLIVEIRA, D., OGASAWARA, E., BAIÃO, F., *et al.*, 2010, "SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows". In: *Proceedings of the 3rd International Conference on Cloud Computing*, pp. 378–385, Washington, DC, USA.

PAPUZZO, G., SPEZZANO, G., 2011, "Autonomic management of workflows on hybrid grid-cloud infrastructure". In: *Proceedings of the 7th International Conference on Network and Services Management (CNSM'11)*

PATON, N., DE ARAGÃO, M. A. T., LEE, K., *et al.*, 2009, "Optimizing utility in cloud computing through autonomic workload execution", *Bulletin of the Technical Committee on Data Engineering*, v. 32, n. 1, pp. 51–58.

PENNISI, E., 2011, "Will Computers Crash Genomics?", *Science*, v. 331, n. 6018 (Feb.), pp. 666–668.

POSTGRESQL. PostgreSQL., 2014 Disponível em: http://www.postgresql.org/. Acesso em: 22 Mar 2014.

PRICE, R., TINO, P., 2009, "Still alive: Extending keep-alive intervals in P2P overlay networks".

RAHMAN, M., RANJAN, R., BUYYA, R., 2010, "Cooperative and decentralized workflow scheduling in global grids", *Future Generation Computer Systems*, v. 26, n. 5 (May.), pp. 753–768.

RASOOLI, A., DOWN, D. G., 2011, "An Adaptive Scheduling Algorithm for Dynamic Heterogeneous Hadoop Systems". In: *Proceedings of the 2011 Conference of the Center for Advanced Studies on Collaborative Research*, pp. 30–44, Riverton, NJ, USA.

RIPEANU, M., 2001, "Peer-to-peer architecture case study: Gnutella network". In: *Proceedings of the 1st International Conference on Peer-to-Peer Computing*, pp. 99–100

SHEN, X., YU, H., BUFORD, J., *et al.*, 2010, *Handbook of peer-to-peer networking*. New York ; London, Springer.

SILVA, F. A. B. DA, SENGER, H., 2009, "Improving scalability of Bag-of-Tasks applications running on master-slave platforms", *Parallel Computing*, v. 35, n. 2, pp. 57–71.

SILVA, V., DIAS, J., OLIVEIRA, D., *et al.*, 2013, "Uma Arquitetura P2P de Distribuição de Atividades para Execução Paralela de Workflows Científicos". In: *VII e-Science*, pp. 1–8, Maceió, Alagoas, Brazil.

SILVA, V., OLIVEIRA, D., MATTOSO, M., 2014, "SciCumulus 2.0: Um Sistema de Gerência de Workflows Científicos para Nuvens Orientado a Fluxo de Dados". In: *Sessão de Demos do XXIX Simpósio Brasileiro de Banco de Dados*, Curitiba, Paraná.

SMANCHAT, S., INDRAWAN, M., LING, S., *et al.*, 2009, "Scheduling multiple parameter sweep workflow instances on the grid". In: *Proceedings of the 5th IEEE International Conference on e-Science*, pp. 300–306

STEVENS, R., GLOVER, K., GREENHALGH, C., *et al.*, 2003, "Performing in silico experiments on the Grid: a users perspective". In: *Proceedings of the UK e-Science programme All Hands Conference*, pp. 2–4, Nottingham, UK.

TANNEN, V., WONG, L., LIBKIN, L., *et al.*, 2013, *In Search of Elegance in the Theory and Practice of Computation*. Berlin, Heidelberg, Springer Berlin Heidelberg.

VAQUERO, L. M., RODERO-MERINO, L., CACERES, J., *et al.*, 2009, "A break in the clouds: towards a cloud definition", *ACM SIGCOMM Computer Communication Review*, v. 39, n. 1, pp. 50–55.

VIANA, V., OLIVEIRA, D., OGASAWARA, E., *et al.*, 2011, "SciCumulus-ECM: Um Serviço de Custos para a Execução de Workflows Científicos em Nuvens". In: *Anais do XXVI Simpósio Brasileiro de Banco de Dados*, Florianópolis, Santa Catarina, Brasil.

VÖCKLER, J.-S., JUVE, G., DEELMAN, E., *et al.*, 2011, "Experiences using cloud computing for a scientific workflow application". In: *Proceedings of the 2nd International Workshop on Scientific Cloud Computing*, pp. 15–24, New York, NY, USA.

WARNEKE, D., KAO, O., 2009, "Nephele: efficient parallel data processing in the cloud". In: *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers*, pp. 8:1–8:10, New York, NY, USA.

WEBER, T. Um roteiro para exploração dos conceitos básicos de tolerância a falhas. Disponível em: http://www.inf.ufrgs.br/~taisy/disciplinas/textos/Dependabilidade.pdf. Acesso em: 9 Jan 2014.

WOZNIAK, J. M., ARMSTRONG, T. G., WILDE, M., *et al.*, 2013, "Swift/T: Large-Scale Application Composition via Distributed-Memory Dataflow Processing". In: *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 95–102