RUBENS MASSAYUKI SUGUIMOTO

EXTENSÃO NO MAPEAMENTO DE WORKFLOWS CIENTÍFICOS ABSTRATOS PARA EXECUÇÃO EM AMBIENTES DE NUVENS COM SERVIÇOS DE INFRAESTRUTURA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Luis C. E. de Bona

CURITIBA

2012

RUBENS MASSAYUKI SUGUIMOTO

EXTENSÃO NO MAPEAMENTO DE WORKFLOWS CIENTÍFICOS ABSTRATOS PARA EXECUÇÃO EM AMBIENTES DE NUVENS COM SERVIÇOS DE INFRAESTRUTURA

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Luis C. E. de Bona

CURITIBA

2012

S947e Suguimoto, Rubens Massayuki
Extensão no mapeamento de workflows científicos abstratos para execução em ambientes de nuvens com serviços de infraestrutura / Rubens Massayuki Suguimoto - Curitiba, 2012.
64f. : il. color. ; 30cm.

Dissertação (mestrado) - Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-graduação em Informática, 2012.

Orientador: Luis C. E. De Bona.
Bibliografia: p. 57-64.

1. Fluxo de trabalho. 2. Internet. 3. Processamento eletrônico de dados - Processamento distribuído. I. Universidade Federal do Paraná. II. Bona, Luis Carlos Erpen de. III. Título.

CDD: 004.6782

Ministério da Educação
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Rubens Massayuki Suguimoto, avaliamos o trabalho intitulado, *"Extensão no mapeamento de workflows científicos abstratos para execução em ambientes de infraestrutura como serviço"*, cuja defesa foi realizada no dia 18 de dezembro de 2012, às 10:30 horas, no Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 18 de dezembro de 2012.

Prof. Dr. Luis Carlos Erpen de Bona
DINF/UFPR – Orientador

Prof. Dr. Bruno Richard Schulze
LNCC – Membro Externo

Prof. Dr. Fabiano Silva
DINF/UFPR – Membro Interno

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DO PARANÁ
Setor de Ciências Exatas
Coordenação de Pós-Graduação em Informática

DEDICATÓRIA

Dedico esta dissertação a minha namorada, família e amigos que me apoiaram durante esta jornada.

AGRADECIMENTOS

Agradeço, primeiramente, a minha família, que apesar de não enteder exatamente o que fiz, me ajudaram e me deram suporte para iniciar este mestrado.

Agradeço também a minha namorada Luana que esteve sempre presente para me apoiar e me ajudar a seguir em frente.

Agradeço ao meu orientador Bona pela oportunidade e pela paciência com minhas crises e incertezas.

A todos os professores do Dinf, principalmente aos professores do C3SL, por compartilhar todo o conhecimento que tem e por motivar a seguir em frente no mestrado.

Aos amigos do Hashi, nesses quase 10 anos de amizade.

Aos colegas e amigos da computação pelas convivências e experiências.

Aos colegas e amigos de laboratório Ricardo Coelho, Ailton, João Heugênio, Russo, Jefer, Marquinho, Ricardo Café e, em especial ao Guilherme Galante por ter ajudado muito nas correções e revisões de quase todos os meus trabalhos de mestrado.

Desculpe caso esqueci de alguém, e muito obrigado a todos.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

IaaS - *Infrastructure-as-a-Service* - Infraestrutura como serviço

MPI - *Message Passing Interface*

MV - Máquina virtual

MVID - Código identificador da máquina virtual

PaaS - *Plataform-as-a-Service* - Plataforma como serviço

SaaS - *Software-as-a-Service* - Software como serviço

SGN - Sistema gerenciador de nuvem

SGWfC - Sistema gerenciador de *workflow* científico

TA - Tarefa abstrata

TC - Tarefa concreta

TCN - Tarefa concreta de nuvem

WfC - *Workflow* científico

WfCA - *Workflow* científico abstrato

WfCC - *Workflow* científico concreto

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O desenvolvimento da computação científica e *e-Science* tem exigido cada vez mais recursos computacionais para pesquisas científicas. Com isso novas tecnologias e gerações de infraestrutura computacional, como grades e nuvens, têm recebido atenção novas pesquisas. Outra tecnologia utilizada é o *workflow científico* (WfC) que descreve uma sequência de tarefas em um formato padronizado que facilita o compartilhamento e a reprodução dos experimentos. O WfC faz uso do Sistema Gerenciador de WfC (SGWfC) que auxilia no desenvolvimento do ciclo de vida do WfC. Uma das etapas do ciclo de vida é o mapeamento que associa as tarefas abstratas aos recursos, tornando-as tarefas concretas ou executáveis. Recentemente, a computação em nuvens (*Cloud Computing*) está sendo usada para fornecer recursos virtualizados para execução de aplicações científicas. Devido às vantagens que as nuvens têm a oferecer, o SGWfC está sendo adaptando para fazer uso deste paradigma. No entanto, algumas adaptações apresentam problemas que deixam os recursos virtuais ociosos gerando custos desnecessários e uma má utilização da infraestrutura. Este trabalho propõe uma modificação no mapeamento que faz uso de uma extensão do WfC abstrato para diminuir a ociosidade dos recursos de forma a controlar a instanciação dos recursos durante a execução do WfC. Para avaliação da proposta, foi implementado um módulo do SGWfC Kepler, chamado de *Kloud*, e o mesmo foi testado utilizando um caso de uso com multiplicação de matrizes em MPI com 2, 4 e 8 nodos. Os resultados preliminares sobre o caso de uso mostram que é possível diminuir a ociosidade em aproximadamente 82%.

Palavras-chaves: computação científica, workflow científico, computação em nuvems.

ABSTRACT

The development of scientific computing and e-Science has required more computing resources for scientific researches. New technologies and generations of computational infrastructures, such as grid and clouds, has been used to aquire those high amount of resources. Other tecnology is the scientific workflow (SWf) which describe a sequel of tasks following a pattern to easily share and reproduce the experiments. SWf uses a SWf managment system (SWfMS) that aids the development of SWf life cycle. One of the stages of the SWf life cycle is mapping that associates the abstract tasks to resources, making them concrete or executable tasks. Recently, cloud computing is being used to provide virtualized resources for execution of scientific applications. Due to the advantages that cloud has to offer, many SWfMS is being adapted to make use of this paradigm. However, some adaptations have problems with idle instatiated virtualized resources that generate unnecessary costs and a poor utilization of infrastructure. This work propose a modification to the mapping stage that uses an extension over abstract SWf format to reduce idleness of resources. This format allows the inclusion of concrete tasks to control virtualized resources instantion during the SWf execution. To validate the proposal, a module of SWfMS Kepler, called *Kloud*, was implemented and tested using a MPI matrix multiplication use case with 2, 4 and 8 virtual nodes. Preliminary results show that it is possible to reduce idleness in approximately 82%

Keywords: scientific computing, scientific workflow, cloud computing.

CAPÍTULO 1

INTRODUÇÃO

O desenvolvimento da computação científica e *e-Ciência* (*e-Science*) [24] tem apresentado uma demanda crescente por recursos computacionais para pesquisas científicas. A *e-Ciência* tem como objetivo oferecer uma infraestrutura global de caráter colaborativo a fim de facilitar o desenvolvimento de experimentos científicos fazendo uso das novas tecnologias e gerações de infraestruturas computacionais.

Dentre essas tecnologias estão os *workflow científicos* (WfC) [39, 21, 11, 64]. O *workflow* científico (WfC) é uma especificação formal do processo científico para representar, simplificar e automatizar as etapas de experimentação [64]. O WfC descreve uma sequência de tarefas em um formato padronizado que facilita o compartilhamento e a reprodução dos experimentos de forma a aumentar a confiabilidade de uma pesquisa.

O WfC faz uso do Sistema Gerenciador de WfC (SGWfC) [35, 2, 25, 21], que auxilia no desenvolvimento do ciclo de vida de um WfC. O ciclo de vida consiste nas etapas de composição, mapeamento das tarefas para os recursos, execução das tarefas nos ambientes de execução subjacentes e coleta de meta-dados e proveniência.

A composição do WfC pode ser feita de forma abstrata ou concreta [9]. O WfC abstrato (WfCA) é descrito como um *workflow* cujas tarefas não mencionam os recursos que deverão ser usados durante a execução. Já o WfC concreto (WfCC) se refere ao *workflow* no qual as tarefas estão vinculadas aos recursos para execução. A tarefa do WfCA é chamada de *tarefa abstrata* (TA) e a tarefa do WfCC é chamada de *tarefa concreta* (TC).

Para executar um WfCA é necessário mapear os recursos para as TAs, de forma a gerar um WfCC. O mapeamento faz a seleção de um conjunto de recursos, que são

obtidos dos ambientes de execução subjacentes, e então é feita a ligação de tais recursos com as TAs, gerando as TCs. Essa etapa pode ser feita manualmente, pelo usuário, ou automaticamente sendo possível adicionar políticas para melhorar o desempenho durante execução do WfC.

Recentemente, a computação em nuvens (*Cloud Computing*) tem sido utilizada para fornecer recursos para a execução de aplicações científicas. A computação em nuvens é um novo paradigma de computação distribuída que oferece serviços e recursos computacionais virtualizados sobre a Internet [33, 3].

Para oferecer os serviços de nuvem, os provedores de serviços utilizam Sistemas Gerenciadores de Nuvem (SGN) que gerenciam os recursos físicos, os recursos virtuais e o acesso aos recursos por administradores e usuários. Os SGN podem ser utilizados baseando-se nos modelos de implantação (pública, privada e híbrida) e modelos de serviços (*software*, plataforma e infraestrutura).

Dentre os modelos de serviços, destacamos o modelo de infraestrutura como serviços (*Infrastructure as a Service* - IaaS) que oferece recursos virtuais, tais como memória virtual, disco virtual, processador virtual e rede virtual, que combinados criam uma máquina virtual (MV). As MVs contêm um ambiente de execução personalizado [28], isolado e com recursos escaláveis dinamicamente (elásticos) para execução de aplicações científicas.

Devido às vantagens que a nuvem tem a oferecer, os SGWfC estão sendo adaptados para fazer uso deste novo paradigma [48, 28, 50, 61, 54]. Alguns dos trabalhos [28, 60] utilizam os recursos de nuvem para execução de WfCA, sendo necessário mapear as TAs para os recursos da nuvem.

No entanto, os trabalhos alocam os recursos virtuais antes de iniciar a execução do WfC que geraram custos desnecessários e uma utilização ineficiente da infraestrutura da nuvem. Em um pior caso, um recurso só é utilizado na execução da última tarefa, ficando ocioso durante todo o intervalo em que foi instanciado até o inicio da execução desta última tarefa.

O presente trabalho propõe uma modificação no mapeamento de forma a utilizar uma extensão no WfCA para associar os recursos de nuvem que podem atender os requisitos de execução das TAs. Com o WfCA extendido é possível mapeá-lo para os recursos de nuvem adequados e inserir as TCs de nuvens (TCNs), que instanciam e liberam os recursos, nos momentos adequados para evitar a ociosidade excessiva de recursos.

Para avaliar a modificação proposta, um módulo do SGWfC Kepler [2] foi implementado, chamado de *Kloud*. Para testar a solução, é apresentado um caso de uso de multiplicação de matrizes em *MPI* para computação distribuída nos recursos da nuvem. Os recursos são alocados dinamicamente, durante a execução do WfC de forma a instanciá-los e liberá-los nos momentos adequados.

O trabalho está divido da seguinte forma. No Capítulo 2, é feita uma revisão sobre *workflows científicos* e as implementações existentes de SGWfC. No Capítulo 3, é feita uma revisão sobre Computação em Nuvens, abordando os conceitos, vantagens e uma revisão sobre a integração de WfC com a nuvem. No Capítulo 4, é apresentada a proposta do trabalho. No Capitulo 5, é apresentada a implementação do módulo Kloud e o caso de uso da multiplicação de matrizes em MPI para validação da proposta. Por fim, o trabalho é concluído no Capítulo 6.

CAPÍTULO 2

WORKFLOWS CIENTÍFICOS

O avanço da computação científica e da tecnologia da informação está proporcionando uma aceleração significativa nas descobertas em diversas áreas da ciência (biologia, medicina, física, ecologia, química, geociências, dentre outras). Os experimentos contêm diversas etapas, onde cada etapa integra modelos e fontes de dados desenvolvidos por diferentes grupos de pesquisa [21]. Além dos modelos, as novas tecnologias de infraestrutura computacional estão permitindo a integração de milhares de recursos para execução distribuída de aplicações científicas e gerenciamento dos dados de forma eficiente [1].

Um experimento científico é um conjunto de etapas que inclui movimentar os dados para computadores (para simulação ou análise), iniciar a computação e gerenciar o armazenamento dos resultados [11]. Estas etapas podem se tornar um empecilho para cientistas que possuem pouco conhecimento técnico em informática. Já os cientista que detêm esse conhecimento técnico acabam desenvolvendo soluções específicas (por exemplo *scripts* ou programas) para automatizar o processo de experimentação. Em muitos casos, as soluções específicas desenvolvidas são pouco reutilizáveis ou até mesmo descartadas com as novas tecnologias e propostas de pesquisa.

Uma das soluções de automação no ciclo de experimentação científica é o fluxo de trabalho científico ou *workflow* científico [21, 1, 11, 65, 37, 64]. O *workflow* científico (WfC) é uma especificação formal do processo científico para representar, simplificar e automatizar as etapas de experimentação [64]. No WfC são descritas as tarefa e as dependências entre as mesmas que servem como molde para automatizar a execução de diversas aplicações científicas. Dessa forma é possível obter abstrações sobre o gerenciamento da computação científica para cientistas que possuem pouco conhecimento em informática.

As abstrações ajudam o cientista a redirecionem os esforços para o foco da pesquisa em sí, ao invés de detalhes técnicos de informática. Além disso, a automação sobre a execução permite a reprodução do WfC e, consequentemente, do experimento, a qual é de grande importância para o método científico.

A Figura 2.1 mostra um exemplo de *workflow* que obtém dados da Web e de uma base de dados remota, trabalha sobre os dados (integração, pré-processamento, teste, treinamento, modelagem), faz a validação e, por último, armazena os resultados. Cada etapa do fluxo de trabalho é representada por uma tarefa e a ordem em que cada uma é executada é representada pelas setas.

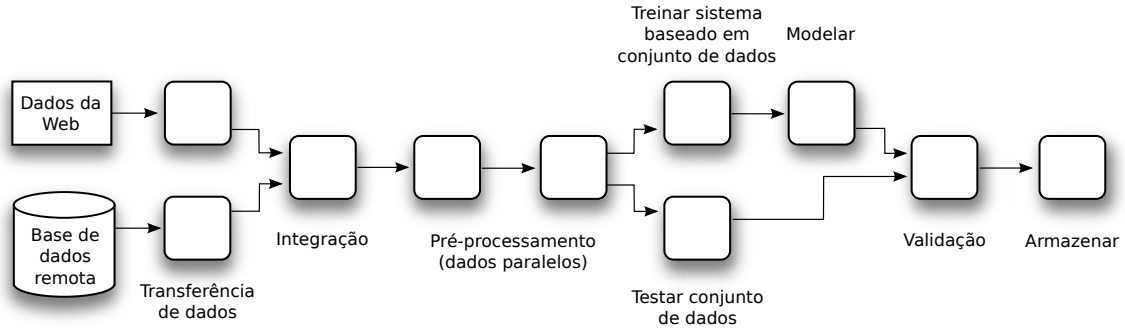


Figura 2.1: *Workflow* exemplo. Adaptado de [8].

Os WfCs são descritos em um formato padronizado que facilite o compartilhamento, reaproveitamento de trabalho e melhor avaliação sobre o processo de obtenção dos resultados de um experimento. Normalmente são usados linguagens de WfCs que contém estruturas para comportar casos diversos de experimentação como Scufl[25], DAG[9] e MoML[2].

Os WfCs são representados com diferentes níveis de abstração sobre as tarefas e as dependências. As tarefas podem assumir um formato *abstrato* ou *concreto* [10, 65] sendo que o primeiro considera as tarefas como unidades funcionais de alto nível, sem vínculo com qualquer recurso em específico, enquanto que no segundo, as tarefas são vinculadas a recursos específicos para execução. As dependências entre as tarefas podem ser de dados e/ou de controle que são passados entre as tarefas, definindo a ordem em que tarefas são executadas.

Os fluxo de trabalhos (abstratos ou concretos) são estruturados em forma de grafo, sendo normalmente representado por grafo dirigido ou direcionado. Neste caso os vértices do grafo representam as tarefas e as arestas representam as dependências [12].

Os WfCs, com as respectivas representações e estruturas, são explorados em vários níveis através de um Sistema de Gerenciamento de WfC (SGWfC) [11, 21]. O SGWfC implementa ferramentas que auxiliam no gerenciamento do ciclo de vida do WfC oferecendo abstrações desde a criação até a obtenção dos resultados. O ciclo de vida de um WfC compreende desde a criação, passando para o mapeamento das tarefas para os recursos, execução das tarefas e a coleta de metadados e informações de proveniência para criação de um histórico dos resultados.

De modo a apresentar uma visão geral sobre os WfCs, o restante do capítulo está organizado da seguinte forma. Na Seção 2.1 são apresentados os critérios de abstração e estruturação do WfC usados nos modelos de representação dos WfCs. A Seção 2.2 apresenta o ciclo de vida do WfC, as etapas de criação, mapeamento, execução e proveniência. Por último, a Seção 2.3 mostra as implementações de SGWfC e o gerenciamento do ciclo de vida do WfC em diferentes sistemas.

2.1 Modelo de representação do *Workflow Científico*

O modelo de representação do WfC depende de fatores como abstração das tarefas (modelo abstrato ou concreto), abstração na dependência entre as tarefas (dados, controle ou híbrido) e estrutura do WfC (sequencial, paralelo, condicional ou iterativo). A Figura 2.2 mostra um diagrama com os fatores que influenciam na modelagem do WfC.

As tarefas podem seguir um modelo abstrato ou concreto, também chamados de WfC abstrato (WfCA) ou WfC concreto (WfCC) respectivamente [10, 65]. No WfCA, as tarefas são consideradas unidades funcionais de alto nível como processos para alinhamento de imagens, agrupamento de dados, pré-processamento, armazenamento de dados. As tarefas no WfCA são chamadas de tarefas abstratas (TA) e não explicitam quais recursos são

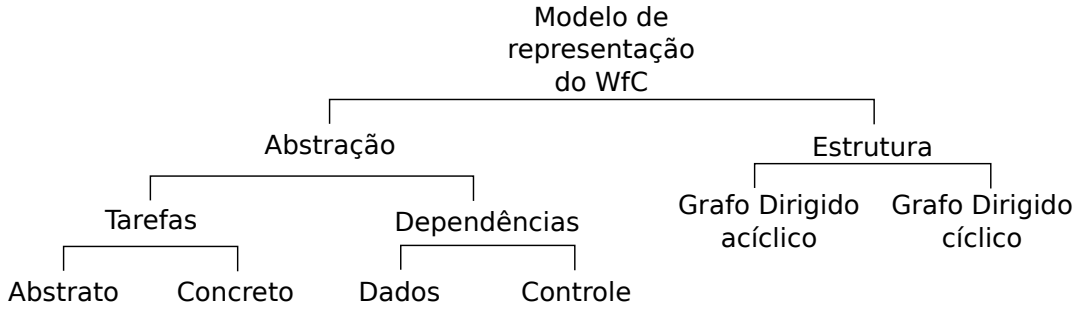


Figura 2.2: Diagrama dos fatores que influenciam na modelagem do WfC.

utilizados para execução do WfC, tais como aplicações, computadores, arquivos e dados, quantidade de memória e capacidade de processamento. Dessa forma o WfCA facilita o uso da computação para cientistas menos experientes pois o fluxo oculta os detalhes técnicos relacionados aos recursos.

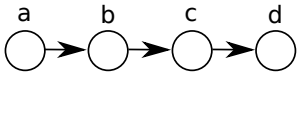
Já no WfC concreto (WfCC), ou também chamado de executável, as tarefas são vinculadas a recursos específicos usados para executá-las tais como as aplicações, os comandos para transferência dos dados, os valores dos parâmetros, os comandos para iniciar as aplicações nos recursos e tratar exceções. As tarefas do WfCC também são chamadas de tarefas concretas e permitem ao cientista ter maior controle sobre a execução do *workflow*.

É possível aplicar abstrações sobre as dependências entre as tarefas, podendo ser a depedência de dados ou dependência de controle. A dependência de dados forma um fluxo de dados representando a transferência de dados gerados por uma tarefa para a tarefa sucessora. Em alguns casos, a dependência de dados também simboliza a transferência de dados entre recursos e/ou a conversão de formatos distintos de dados entre as tarefas.

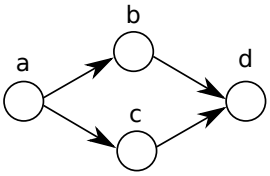
Já a dependência de controle forma um fluxo de controle de execução no qual a tarefa, ao terminar a execução, repassa o controle para a tarefa sucessora Ainda é possível mesclar ambos os modelos de abstração e formar um fluxo híbrido contendo a dependência de dados e controle no mesmo *workflow* [8, 11]. O fluxo híbrido é usado em alguns casos no fluxo de dados quando uma tarefa predecessora gera dados implicitamente, como inserção de dados em um banco de dados, e existe uma tarefa sucessora dependente da tarefa. Assim para prosseguir a execução do WfC é preciso passar o controle de execução a tarefa

seguinte.

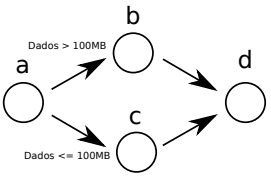
O WfC é estruturado em um formato que pode comportar fluxo de tarefas em sequencial (*pipeline*), fluxo de tarefas em paralelo, fluxo condicional (*choice*) ou fluxo com ciclos dentro do *workflow* [65]. O fluxo sequencial representa a execução em série de tarefas, no sentido em que uma tarefa inicia quando a anterior termina. O fluxo paralelo representa a execução concorrente de duas ou mais tarefas. Já o fluxo condicional representa os desvios de fluxo que ocorrem em tempo de execução. A Figura 2.3 mostra exemplos de WfCs com 4 tarefas estruturado em sequencial (2.3a), paralelo (2.3b), condicional (2.3c) e com ciclo (2.3d).



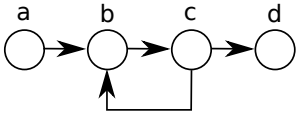
(a) Estrutura com fluxo sequencial.



(b) Estrutura com fluxo paralelo



(c) Estrutura com fluxo condicional.



(d) Estrutura com ciclo

Figura 2.3: Exemplos de estruturas usadas para representar o WfC.

A estrutura mais utilizada para representar o WfC é a de grafo dirigido ou direcionado no qual as tarefas são representadas por nós ou nodos e as dependências são representadas pelas arestas [12]. O grafo direcionado acíclico é simples pois não precisa tratar de estruturas de repetição. Este tipo de grafo é usado para representar os fluxos de tarefas em serial, paralelo e condicional. No entanto, em alguns experimentos, é desejável, ou até mesmo necessário, o uso de estruturas de repetição, devido à grande quantidade de iterações de tarefas. Nesse casos, a estrutura do grafo direcionado cíclico é usada para representar os ciclos ou laços dentro do WfC.

O modelo usado depende do contexto e do experimento. Atualmente existem diferentes modelos de WfC com diferentes níveis de abstração aplicados nos mais diversos experimentos. Assim, ao criar ou utilizar WfC é preciso especificar qual nível de abstração desejada e a quantidade de decisões feitas dentro do *workflow*.

2.2 Gerenciamento do Ciclo de Vida do WfC

O ciclo de vida do WfC é o ciclo das fases ou etapas que compreendem desde a criação até o término da execução do fluxo de trabalho. O ciclo inicia com a criação do *workflow*, passando pelo mapeamento das tarefas nos recursos, execução das tarefas nos recursos e coletas de metadados e informações de proveniência [12]. A Figura 2.4 mostra as etapas de composição, mapeamento, execução e coleta de metadados e proveniência com as respectivas relações entre as mesmas, apresentadas em detalhes nas Seções 2.2.1, 2.2.2, 2.2.3 e 2.2.4, respectivamente.

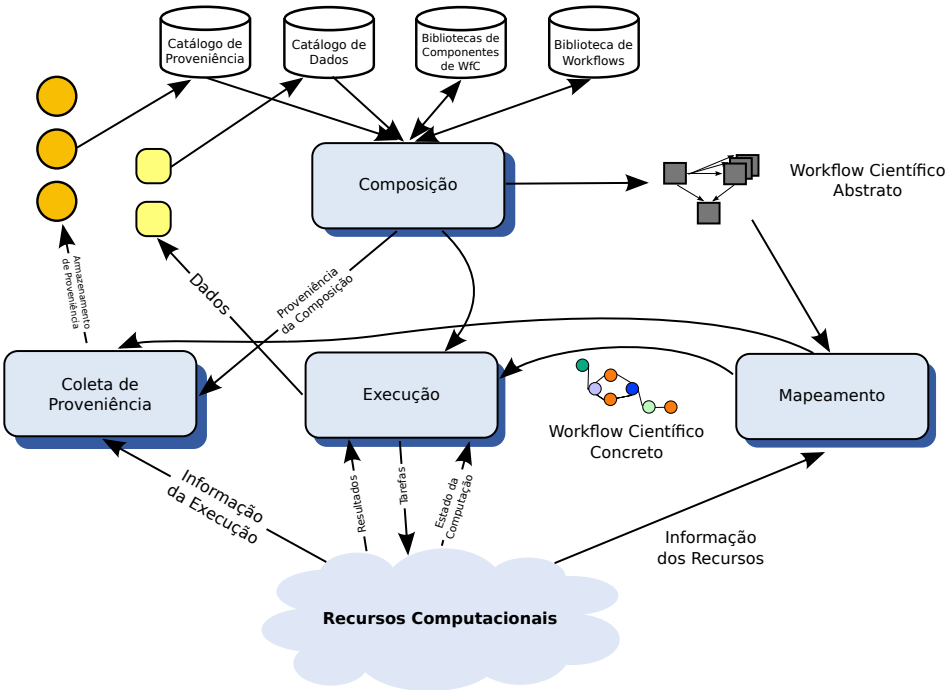


Figura 2.4: Ciclo de vida do WfC com as etapas de composição, execução, mapeamento e coleta de proveniência. Adaptado de [11].

2.2.1 Composição

A etapa de composição é a responsável por oferecer ferramentas aos cientistas para montar o fluxo de trabalho de acordo com a estrutura do grafo direcionado e as abstrações definidas na especificação do experimento. A composição do WfC pode ser feita utilizando uma interface textual ou uma interface gráfica [11]. Na composição textual utiliza-se um editor de texto no qual o cientista descreve o *workflow* diretamente. A interface textual é adequada para composição de *workflows* simples e que incluam somente poucas dezenas de tarefas. Este método de composição é usado de preferência no formato abstrato pois as tarefas abstratas contém poucas descrições.

A composição gráfica faz uso de interface na qual o cientista visualiza e trabalha na criação do *workflow* clicando e arrastando as tarefas e criando as dependências graficamente (*drag-and-drop*). A interface gráfica de composição é amigável aos usuários não especialistas em informática pois através dela é possível visualizar o fluxo de trabalho ao invés da descrição textual. Em contrapartida as interfaces gráficas precisam oferecer suporte ao aninhamento de tarefas, de forma que a visualização de centenas ou milhares de tarefas com as respectivas dependências não fique comprometida.

Os WfC são descritos em uma linguagem de WfC que define um padrão e oferece maior flexibilidade para compor *workflow*. A composição gráfica oferece abstração sobre as linguagens de WfC de forma que o cientista não precisa aprendê-las, já que a própria interface gráfica faz a tradução para o formato textual.

2.2.2 Mapeamento do WfC Abstrato

O mapeamento é a etapa responsável por transformar o WfC abstrato no WfC concreto. O WfC pode assumir um formato abstrato durante a etapa de composição e posteriormente transformado em um formato executável, de forma que as tarefas abstratas sejam associadas aos recursos.

O mapeamento pode ser feito manualmente pelo usuário ou automaticamente através

de um mapeador [11]. O mapeamento manual é feito pelo cientista que faz a escolha dos recursos mais apropriados para execução das tarefas abstratas. Já o mapeador automático utiliza informações sobre os recursos obtidos do ambiente de execução e preferências do cientista, como as aplicações ou conjunto de dados, e então executa um algoritmo de mapeamento.

A definição de recurso varia com o ambiente de execução e cada recurso é descrito de forma distinta podendo variar de descrições simples ou até mesmo descrições complexas que involvem outros recursos. Dentre os recursos podemos citar serviços, computadores e até mesmo um humano [12].

Os serviços descrevem as interfaces definidas por um protocolo (ex. WSDL [5]) e são invocados durante a execução do WfC. Os serviços oferecem abstrações na execução escondendo os detalhes de implementação da aplicação. Assim, para iniciar a computação basta enviar os dados e invocar o serviço utilizando o protocolo de serviço.

Os computadores oferecem o ambiente necessário para executar as aplicações, armazenar os resultados ou recuperar um conjunto de dados. Normalmente, o computador é dividido em categorias de processamento e de armazenamento, sendo a primeira a responsável por efetuar a computação e a segunda, por armazenar os dados de entrada e os resultados.

Por último, a intervenção humana é necessária para verificar se o WfC está progredindo corretamente e para aprovar ou confirmar o uso de um determinado recurso.

2.2.3 Execução do WfC

A execução é a etapa que envolve a orquestração e execução das tarefas nos recursos. A orquestração faz a seleção das tarefas que podem ser executadas de acordo com as dependências resolvidas. A orquestração garante que o WfC siga o fluxo de execução, definido pelo cientista, na etapa de composição.

Em seguida, a tarefa selecionada é preparada e iniciada no recurso alvo através do modelo de execução. O modelo de execução pode ser de serviço ou de *job* [11]. No primeiro, as tarefas são vistas como serviços que oferecem interfaces padronizadas para as aplicações de forma a garantir maior interoperabilidade na execução. Normalmente os serviços são invocados através de protocolos de serviços, por exemplo SOAP [42] e REST [17]).

No segundo caso, as tarefas descrevem como executar a aplicação no recurso. As descrições incluem os dados a serem computados, as aplicações, os parâmetros das aplicações, os comandos para iniciar a aplicação e comandos para coletar os dados gerados após a computação.

A execução é gerenciada pelo executor que implementa as técnicas e comandos inerentes aos ambientes de execução [35]. Os ambientes de execução podem variar desde ambientes com recursos homogêneos e geograficamente próximos (Rede Local e *Clusters*), até ambientes com recursos heterogêneos e geograficamente dispersos (Computação em Grades, Ponto-à-Ponto ou Computação em Nuvens).

2.2.4 Metadados e Proveniência

A proveniência é a etapa responsável por coletar metadados e informações sobre o ciclo de vida do WfC. A proveniência atua nas etapas de composição, mapeamento e execução de forma obtendo informações que permitem entender e, em caso de uma proveniência completa, realizar a reprodução do experimento. Os registros históricos ajudam a traçar o caminho inverso do ciclo de vida do WfC, partindo dos dados resultantes da execução até a composição do WfC.

Uma das áreas mais atuantes no WfC é a proveniência de dados [11]. A proveniência de dados contém registros históricos da centrados na criação de um dado, que incluem informações sobre as aplicações, serviços, computadores, bibliotecas e dados intermediários criados durante a execução do WfC.

Existe também a proveniência da composição (*design*) que mantém um histórico de alterações feitas em um determinado WfC. O histórico de alterações ajuda a entender como um WfC foi alterado para atender as diferentes necessidades ou demandas dos experimentos para a obtenção dos resultados.

2.3 Sistemas Gerenciadores de WfC

Atualmente existem Sistemas Gerenciadores de WfC que atuam em diversas áreas e são usados por diferentes grupos de pesquisas. Alguns dos principais sistemas *software livre* são: Pegasus [9, 12], Kepler [2], Taverna [25], Vistrails [4], Triana [38, 56] e P-Grade [16]. Cada sistema implementa diferentes abordagens para tratar do ciclo de vida do WfC. As implementações são descritas nas seções seguintes para delinear as abordagens e o propósito de cada SGWfC.

2.3.1 Pegasus

O Pegasus [9, 12] é um SGWfC desenvolvido pelo *Information Sciences Institute*, na Universidade da Carolina do Sul. O Pegasus é aplicado na astronomia, bioinformatica, botânica, neurociência, meteorologia, geografia, oceanografia, ciência da computação, em várias outras áreas das ciências da terra, ciências exatas, tecnologia e educação.

O sistema possibilita o uso de WfC abstrato, fazendo o mapeamento para WfC concreto e a execução das tarefas em ambientes distribuídos através de componentes externos ou *middlewares* como Globus [32] e Condor [36]. A composição abstrata é feita pelo *software* externo Chimera [19] que descreve parcialmente o WfC sobre a linguagem *Virtual Data Language* (VDL). O VDL descreve cada tarefa e os respectivos arquivos de entrada e de saída de forma lógica. A descrição lógica é usada para referenciar os arquivos e aplicações que podem conter diferentes nomes no sistema de arquivos ou estar localizados em diferentes recursos.

Agrupando as tarefas lógicas é possível obter o WfC abstrato que, no caso do Pegasus, é descrito em DAX. O mapeamento faz uso do WfCA DAX e de informações sobre os recursos, agrupamento das tarefas, dados de entrada e de saída para a geração do *workflow* concreto no formato DAG.

A execução do DAG é feita no Condor DAGMan [57], que gerencia a execução do WfC. O DAGMan orquestra o *workflow*, selecionando as tarefas concretas para executarem nos recursos. O sistema foi desenvolvido para executar em ambiente distribuído, com suporte a grades computacionais, como Condor-G [20], e a *middlewares* de *clusters*, como o Condor [36]. Dessa forma, a execução da tarefa é feita por um desses *middlewares* externo ao Pegasus.

O Pegasus tem a funcionalidade específica de mapear WfC abstratos para concretos, deixando a cargo de outros sistemas para compor e executar as tarefas no recursos. Isso agrega maior modularidade aos componentes, uma vez que é possível combinar o Pegasus com outros sistemas.

2.3.2 Taverna

O Taverna [25, 47], também denominado de *Taverna Workbench*, é um SGWfC que permite aos cientistas compor e executar WfC através da orquestração de serviços *web*. O sistema foi criado pelo grupo *MyGrid* [6] e financiado através do *Open Middleware Infrastructure Institute* - Reino Unido (OMII-UK).

O sistema foi projetado para auxiliar bioinformatas não especialistas em programação e serviços web. O auxílio se dá através do uso coordenado de serviços web através de WfC de forma a esconder os detalhes de execução dos usuários.

A composição dos WfC é feita pelo usuário através da interface gráfica basedo no *GraphViz* [15]. Os *workflows* são descritos na linguagem *Simplified Conceptual Workflow Language* (Scufl).

Na composição, as tarefas são vistas como serviços que apresentam as interfaces ou portas de entrada (dados de entrada) e de saída (dados de saída). As dependências entre as tarefas são feitas através das ligações entre as portas de saída de uma tarefa e a de entrada de outra.

A execução é feita através do executor *Freefluo*, que faz a orquestração e a chamada de métodos (execução) em serviços web. O *Freefluo* é executado no ambiente local e faz a invocação de serviços através de protocolos de serviços (SOAP, WSDL e REST) e protocolos de serviços de banco de dados remotos (JDBC).

2.3.3 VisTrails

O VisTrails [4] é um SGWfC criado pelo Instituto de Computação Científica e Imagem da Universidade de Utah. O VisTrails tem aplicação em diversas áreas do conhecimento como a medicina, a meteorologia, a astrofísica e a biologia.

O sistema tem como objetivos criar uma infraestrutura para tratar da proveniência, prover um *framework* para execução de *workflows* e oferecer uma interface gráfica para visualização e comparação de resultados.

A composição dos *workflows* é feita no *VisTrails Builder* que utiliza uma interface gráfica baseada no *GraphViz* [15]. A linguagem usada para descrever os *workflows* segue um formato próprio do VisTrails baseado em XML. Cada tarefa do WfC, também chamada de módulo, é vista como uma execução independente. Cada execução contém os comandos e instruções para atender a semântica da tarefa.

As arestas são chamadas de conexões e representam as dependências entre as tarefas. Cada conexão indica o dado, propriamente dito, e o tipo de dado a ser transmitido de uma tarefa a outra. O fluxo de execução do *workflow* segue um modelo de dados.

A execução é feita pelo componente *VisTrails Player* (VP), que faz a orquestração e execução das tarefas. O VP gera uma instância para execução que traduz os módulos

e conexões para as classes de APIs do sistema subjacente (módulos da linguagem de programação Python). Na instância também são aplicados os valores dos parâmetros de cada módulo. Por fim o *workflow* é orquestrado e as tarefas selecionadas são executadas localmente chamando os métodos específicos das classes. É possível utilizar processamento distribuído, através de classes que implementam o acesso a recursos remotos.

Uma das maiores vantagens do VisTrails é o sistema de proveniência. A coleta de proveniência compreende desde a etapa de composição até a obtenção dos resultados finais. Com a coleta é possível armazenar um histórico de composição para mostrar a outros cientistas como o *workflow* foi criado ou alterado para se chegar em um determinado resultado.

2.3.4 Triana

O Triana [38, 56] é um SGWfC criado pela Universidade de Cardiff e é usado por cientistas nas áreas da medicina, biologia, música, ciência da computação e astronomia. O sistema foi concebido para oferecer um ambiente de programação gráfica que provê aos usuários abstrações sobre os serviços em sistemas P2P. As abstrações oferecem um ambiente que facilita a criação e a execução de vários serviços através do *workflow*, sendo possível criar serviços complexos.

Para a composição, é usada uma interface própria do Triana, chamada *Triana GUI*, na qual é possível compor WfC em qualquer linguagem, desde que o componente de escrita ("*writer*") suporte as linguagens desejadas. O sistema suporta as linguagens BPEL4WS ou WS-BPEL [46] e um formato XML próprio do Triana.

A Triana GUI é executada localmente e faz uso de tarefas que se encarregam do processamento remoto. A orquestração das tarefas é feita por um componente interno e a execução é feita por um sub-sistema chamado *Grid Application Prototype* (GAP). O GAP provê uma interface que unifica o acesso aos recursos de forma que as aplicações não precisam tratar das tecnologias subjacentes.

2.3.5 P-Grade

O P-Grade [16] é um SGWfC desenvolvido pelo Laboratório de Sistemas Paralelos e Distribuídos MTA SZTAKI, na Hungria. O sistema é usado em áreas como química, meteorologia, engenharia, matemática, economia, bioinformática.

O sistema contém um portal *web* que permite ao cientista criar, executar, compartilhar e monitorar os *workflows*, exigindo poucos esforços na instalação e configuração dos *softwares*. Nos portais, o cientista utiliza uma conta para obter acesso às ferramentas e às funcionalidades de gerenciamento do WfC.

A composição dos WfC é feita através de uma interface gráfica implementada em Java para *web*, chamada *Workflow Editor*. A composição utiliza o formato concreto e as tarefas são vistas como *jobs*. Os *jobs* são configuráveis sendo possível alterar as propriedades da tarefa assim como adicionar e configurar as portas de comunicação (entrada e saída de dados). A interface traduz o WfC para a linguagem DAGMan.

A execução dos *workflows* é toda feita no portal, deixando para o usuário as operações básicas de iniciar, parar e observar os fluxos. Para isso, o P-Grade contém um sistema de armazenamento de *workflows* no qual é possível armazenar tanto os WfC como os dados gerados pelos mesmos.

Quanto à orquestração é usada o Condor DAGMan para selecionar as tarefas com as dependências de dados atendidas. Com a tarefa selecionada, a mesma é executada, como *job*, nos recursos através de *middlewares* de grades (Globus [32], gLite [13], ARC [44]) ou *clusters* (PBS [43], LSF [66]).

O P-Grade apresenta uma solução mais completa, uma vez que o usuário não precisa tratar da instalação dos componentes de *software* necessários para uso dos WfC. Além disso, faz uso de vários componentes externos bem definidos, que podem ser trocados a fim de manter um maior grau de modularidade.

2.3.6 Kepler

O Kepler [2] é um SGWfC feito na linguagem de programação Java, desenvolvido e mantido pela Universidade da Califórnia (Davis, Santa Barbara e San Diego). O sistema foi desenvolvido sobre o Ptolemy II [14] para composição e execução de WfC. O Kepler tem como objetivo oferecer um ambiente extensível para composição e execução de *workflows* de propósito geral e é aplicado nas áreas de bioinformática, oceanografia e informática (gerência de dados).

Os *workflows* são descritos na linguagem *Modeling Markup Language* (MoML), onde as tarefas são chamadas de atores e as dependências de canais. Os atores contêm portas de entrada, portas de saída e parâmetros que controlam o comportamento do ator. Já os canais podem transmitir dados ou sinais entre os atores, seguindo um fluxo de execução híbrido.

No sistema o usuário pode compor os *workflows* concretos através de uma interface gráfica baseada no *software* Vergil [51]. A linguagem MoML suporta encapsulamento de *workflows*, o que possibilita o agrupamento de tarefas concretas, gerando uma tarefa com maior nível de abstração (*workflow abstrato*).

A Figura 2.5 mostra a interface gráfica do Kepler com um exemplo de *workflow* da equação de Lotka-Volterra usada para descrever as dinâmicas nos sistemas biológicos entre presas e predadores.

O Kepler foi projetado para executar no ambiente local, podendo ter acesso à computação de alto desempenho através de extensões. As extensões são implementadas através dos atores que contêm as informações, bibliotecas e APIs necessárias para ter acesso aos recursos remotos. Dessa forma, o Kepler trata tanto do modelo de execução de serviços como de *jobs*. Cada modelo de execução é implementado dentro de atores específicos que lidam com as diferentes tecnologias e métodos de execução distribuída.

O sistema oferece diferentes modelos de computação do WfC, que são inseridos como um componente no *workflow*, chamado de diretor. Os diretores fazem a orquestração das

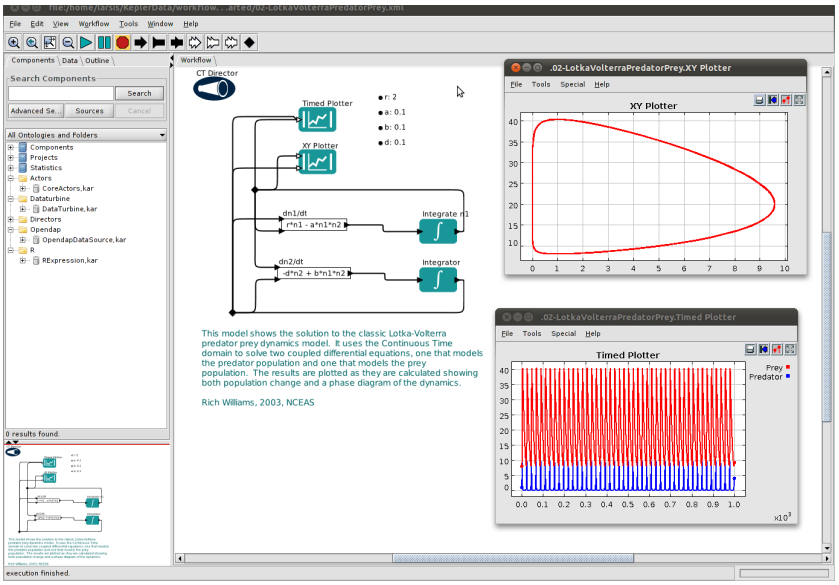


Figura 2.5: Interface gráfica de composição do Kepler.

tarefas e implementam diferentes abordagens, dependendo da semântica do *workflow*. O diretor *Simple Data Flow* (SDF) faz a orquestração sequencial, pois seleciona as tarefas de forma sequencial mesmo que existam tarefas que possam ser executadas em paralelo. O SDF também assume que os dados nos canais (dependências) são estáticos até o final da execução do WfC. Para dados dinâmicos, como no caso de haver laços, existe outro diretor sequencial chamado *Dinamic Data Flow* (DDF). Para seleção paralela, existe o diretor *Process Network* (PN) que faz a seleção de múltiplas tarefas.

Com a tarefa selecionada, a mesma é executada localmente através da chamada de métodos específicos da tarefa ("fire()"), que contêm o código para realizar as ações do ator. Dessa forma, o Kepler é responsável somente por orquestrar o WfC, deixando a cargo das tarefas incluir as funcionalidades desejadas pelo usuário.

As características dos SGWfC estão sumarizadas na tabela 2.1. Dentre as características, encontram-se a linguagem concreta e abstrata para descrição dos WfCs, os componentes para composição, orquestração e execução dos *workflows*; o tipo de fluxo de execução; o ambiente de execução de cada sistema e o suporte à computação remota dos sistemas que funcionam localmente, na máquina do usuário.

Dentre as características apresentadas, o SGWfC Kepler foi escolhido para implemen-

tação e avaliação da proposta do presente trabalho. O Kepler apresenta vantagens na execução do WfC, pois não necessita de instalação de uma infraestrutura complexa de recursos, como é o caso do Pegasus e P-Grade.

Com relação ao Triana e Taverna, o Kepler é mais flexível para computação científica de propósito geral. O Triana e o Taverna são usados em ambientes orientados a serviços e, para se acessar tais recursos, é necessário iniciar uma infraestrutura de serviços, sendo o mesmo caso do Pegasus e P-Grade.

No próximo capítulo é apresentado uma visão geral da computação em nuvem, que é o ambiente de execução explorado no presente trabalho para execução de tarefas do WfC. No capítulo também são abordados estudos relacionados a integração da computação em nuvem com os sistemas de WfC.

SGWfC	Pegasus	Kepler	Taverna	VisTrails	Triana	P-Grade
Interface de composição	Textual (Chimera)	Gráfica (Vergil)	Gráfica (GraphViz)	Gráfica (GraphViz)	Gráfica (Triana GUI)	Gráfica (Workflow Editor)
Orquestrador	Condor DAGMan	PtolemyII (diretor)	Freefluo	Vistrails Player	Triana GUI	Condor DAGMan
Executor	Condor	PtolemyII (atores)	Freefluo	Vistrails Player (módulos)	Grid Application Prototype (GAP)	Condor, Globus
Fluxo de Execução	Dados	Híbrido	Dados	Dados	Dados	Dados
Ambiente de execução	Grades Computacionais e Clusters	Local, Grades Computacionais e Serviços *Web*	Local e Serviços *Web*	Local e Serviços *Web*	Local, Grades Computacionais, P2P, Serivços *Web*	Grades Computacionais e Clusters

Tabela 2.1: Tabela comparativa dos SGWfCs.

CAPÍTULO 3

COMPUTAÇÃO EM NUVENS

O WfC atua na coordenação e gerenciamento da computação e movimentação dos dados entre os recursos de forma a facilitar o uso de recursos computacionais para a experimentação científica. No entanto cada vez mais a computação científica está exigindo maiores quantidades de recursos computacionais para execução de aplicações em tempo hábil. A grande demanda por recursos é justificada pela complexidade dos modelos que incluem processamento distribuído de grande volumes de dados (atualmente, *terabytes* ou *petabytes*) e cálculos complexos.

Uma das formas de oferecer recursos para a computação científica é a computação em nuvens. A computação em nuvens (*Cloud Computing*) é um novo paradigma de computação distribuída que surgiu no meio comercial para oferecer serviços e recursos computacionais virtualizados sobre a Internet [33, 3], mas que recentemente tem sido adotada como ambiente para execução de experimentos científicos [52, 62, 34, 23, 59].

Atualmente existem diversas definições sobre a computação em nuvens [58]. As definições são baseadas em uma determinada perspectiva sobre o uso deste paradigma, como o modelo de negócios, a computação sob-demanda (*utility computing*) ou o gerenciamento da infraestrutura.

Vaquero et al. [58] define a computação em nuvens como um conjunto grande de recursos virtuais acessíveis e de fácil uso (por exemplo *hardware*, plataformas de desenvolvimentos e/ou serviços). Os recursos podem ser configurados dinamicamente com o objetivo de ajustá-los à variação da carga, permitindo uma utilização otimizada dos recursos. Normalmente, o uso dos recursos segue um modelo de pagamento por uso (*pay-per-use*), que é oferecido pelos provedores através de acordos de serviços (*Service Level Agreement*)

personalizados.

Outra definição interessante é a de *Mell e Grance* [40] que define nuvem como um modelo que permite a onipresença (*ubiquitous*), conveniência e acesso sob-demanda a um conjunto compartilhado de recursos computacionais configuráveis (por exemplo redes, servidores, armazenamento, aplicações e serviços). Os recursos podem ser adquiridos e liberados rapidamente com poucos esforços no gerenciamento ou com pouca interação dos provedores da nuvem.

A Figura 3.1 apresenta um exemplo de como a nuvem computacional está organizada. A nuvem contém diferentes serviços que podem ser agrupados em camadas de *software*, plataforma e infraestrutura. A camada de *software* oferece aplicações que são usadas diretamente pelos cientistas. A camada de plataforma oferece plataformas de desenvolvimento para criação e disponibilizaçao de aplicativos. Por último, a camada de infraestrutura contém componentes virtuais de uma infraestrutura computacional, como computadores, discos para armazenamento de dados e redes vituais, que podem ser usados por cientistas e desenvolvedores. Os serviços podem ser acessados através de diferentes dispositivos, podendo abranger celulares, *tablets*, laptops, estações de trabalho e servidores.

A computação em nuvem é composta de cinco características, três modelos de serviços e três modelos de implantação [40]. As características incluem o auto-serviço sob-demanda, amplo acesso à rede, agrupamento de recursos, rápida elasticidade e serviço medido. Os modelos de serviços são classificados em *software* (*software as a service* - SaaS), plataforma (*plataform as a service* - PaaS) e infraestrutura (*infrastructure as a service*) [53, 28]. Já os modelos de implantação são classificados em nuvem privada, híbrida e pública. A Figura 3.2 mostra as classificações dos modelos de serviços, modelos de implantação e algumas características da nuvem.

Para unificar e abstrair o acesso aos recursos computacionais para os diferentes usuários, os provedores da nuvem utilizam um sistema de gerenciamento de nuvem (SGN). O SGN trata de gerenciar os recursos (físicos e virtuais) de forma a otimizar o uso do *hardware*, garantir a qualidade de serviço e controlar o acesso aos recursos.

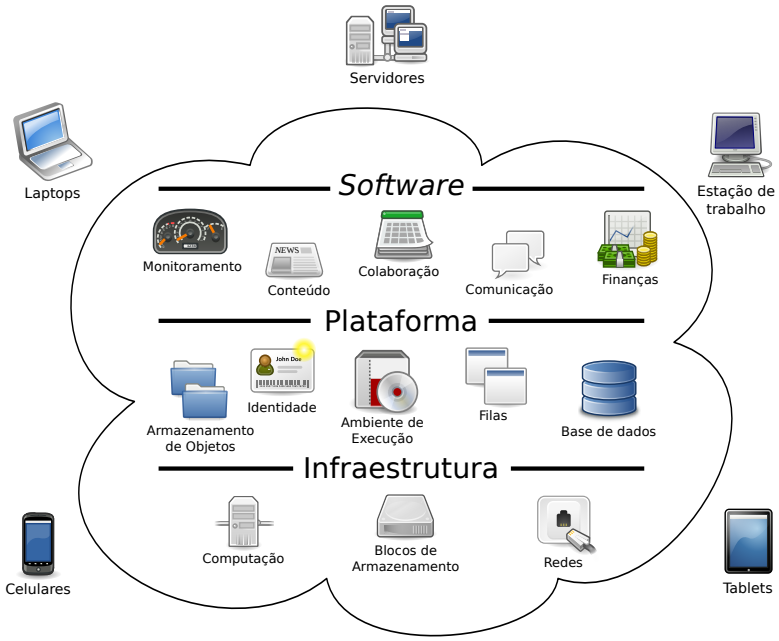


Figura 3.1: Exemplo de nuvem com diferentes serviços oferecidos, que incluem aplicações, plataformas de desenvolvimento e infraestrutura de computadores. Adaptado de [63].

Para apresentar a nuvem em maiores detalhes, este capítulo está dividido da seguinte forma. As características das nuvens são apresentadas na Seção 3.1, os modelos de implantação, na Seção 3.2 e os modelos de serviços, na Seção 3.3. Algumas implementações de sistema de gerenciamento de nuvem são apresentadas na Seção 3.4. Por último, na Seção 3.5, são apresentados trabalhos que abordam a integração de *workflows* científicos e nuvens computacionais.

3.1 Caraterísticas da computação em nuvem

A computação em nuvem compartilha diversas características de outros paradigmas de computação distribuída. No entanto, a nuvem apresenta características que a diferencia de outros paradigmas. Dentre as características estão [40]:

Auto-serviço sob-demanda: um consumidor pode acessar as capacidades computacionais, como uma rede de processamento e armazenamento, quando precisar e de

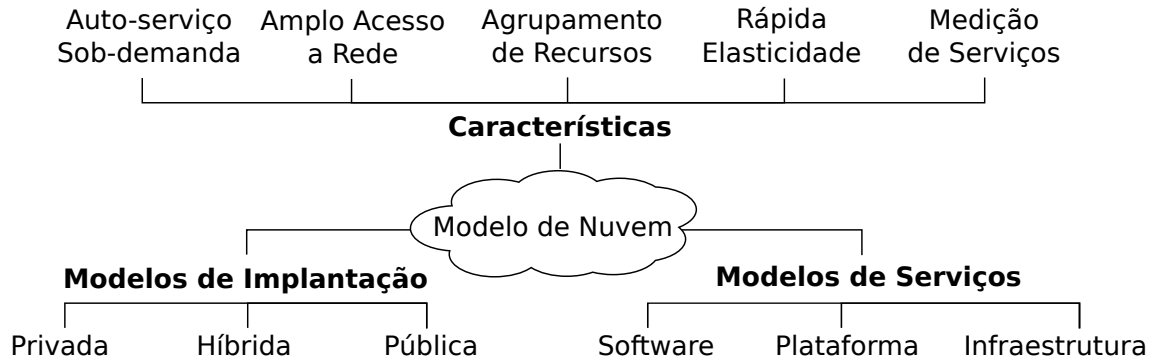


Figura 3.2: Diagrama do modelo de nuvem com as cinco características, os três modelos de implantação e os três modelos de serviços.

forma automática, sem a necessidade de interação humana com cada provedor.

Amplo acesso à rede: as capacidades computacionais estão disponíveis na rede e acessíveis através de mecanismos padronizados que promovem o uso por diferentes plataformas, como celulares, *tablets*, laptops e estações de trabalho.

Agrupamento de recursos: os recursos computacionais são agrupados para atender múltiplos consumidores com diferentes atribuições de recursos físicos e virtuais, dependendo da demanda dos cientistas. Os cientistas não controlam ou não tem conhecimento da localização exata dos recursos obtidos, sendo que em alguns casos é desejável especificar a localização geográfica dos recursos para melhorar o desempenho no uso dos mesmos. Alguns exemplos de recursos incluem armazenamento, processamento, memória e largura de banda de rede.

Rápida elasticidade: os recursos podem ser providos e liberados (elástico) para escalar proporcionalmente com a demanda interna e externa da nuvem. Na visão dos cientistas, os recursos disponíveis para aquisição aparentam ser ilimitados e podem ser obtidos em qualquer quantidade a medida que for necessário.

Medição de Serviços: os sistemas de nuvens controlam e otimizam o uso dos recursos automaticamente através da capacidade de medição dos serviços. O uso dos recursos pode ser monitorado, controlado e reportado, para aumentar a transparência para os provedores e consumidores que utilizam o serviço.

3.2 Modelos de implantação

O modelo de implantação é a forma como a nuvem, ou a infraestrutura da nuvem, é utilizada e organizada para oferecer os serviços. Os modelos de implantação são classificados em [40]:

Privado: a infraestrutura é utilizada exclusivamente por uma organização e pode ser mantida e gerenciada pela própria organização, por outras organizações ou a combinação de ambas.

Público: a infraestrutura é provida para o público geral, podendo ser mantida e gerenciada por organizações privadas, acadêmicas, governamentais ou combinação destas.

Híbrido: a infraestrutura é compostas de duas ou mais infraestruturas de nuvens (pública ou privada) de organizações distintas. As infraestruturas distintas são utilizadas de forma integrada através de padronizações ou com tecnologias proprietárias que permitem a portabilidade de dados e aplicações.

Cada modelo de implantação é aplicado em diferentes contextos, podendo garantir maior ou menor grau de segurança, interoperabilidade e dependabilidade.

3.3 Modelos de serviços

O modelo de serviço especifica os recursos virtuais que são providos para os consumidores, usuários ou desenvolvedores. A especificação dos recursos virtuais delimitam o gerenciamento dos serviços de forma a otimizar o uso da infraestrutura de nuvem. Os modelos de serviços são classificados em: [40]:

Software como Serviço (*Software as a Service - SaaS*): oferece ao usuário aplicações que são executadas na infraestrutura de nuvem. As aplicações são acessadas através de diferentes dispositivos como o navegador *web* (por exemplo, e-mails baseado

na *web*). Nesse serviço, o usuário não gerencia ou controla a infraestrutura de nuvem (rede, servidores, armazenamento) ou até mesmo as aplicações, com algumas exceções de configurações específicas.

Plataforma como Serviço (*Plataform as a Service - PaaS*): oferece mecanismos para implantar aplicações criadas ou adquiridas pelo usuário na infraestrutura da nuvem. A implantação é feita através de linguagens de programação, bibliotecas, serviços e ferramentas suportadas pelo provedor da nuvem. O usuário apenas controla a implantação das aplicações e as possíveis configurações de acordo com o ambiente, deixando o gerenciamento e controle da infraestrutura para os provedores.

Infraestrutura como Serviço (*Infrastructure as a Service - IaaS*): fornece processamento, armazenamento, rede e outros recursos computacionais no qual o usuário pode implantar e executar um *software* arbitrário, incluindo sistemas operacionais e aplicações. O cientista gerencia e controla os sistemas operacionais, o armazenamento, a implantação das aplicações e um limitado controle sobre os componentes de rede, como *firewalls*.

3.4 Sistemas de Gerenciamento de Nuvem

A infraestrutura de nuvem utiliza um sistema de gerenciamento, também chamado de Sistema Gerenciador de Nuvem (SGN) [55, 41, 45, 30], para oferecer os serviços e realizar o gerenciamento dos recursos (físicos e virtuais) na infraestrutura da nuvem. O SGN também implementa ferramentas e interfaces de alto nível aos usuários e administradores para abstrair todo o processo de instanciação de recursos.

As características e capacidades do sistema de nuvem depende do modelo adotado (implantação e serviço). O presente trabalho utiliza o serviço de infraestrutura com implantação privada, sendo o foco no SGN de IaaS para gerenciamento da infraestrutura virtual. Assim as características do SGN de IaaS são [55]:

- Prover uma visão uniforme e homogênea dos recursos virtualizados, sem estar vinculadas as plataformas de virtualização (como Xen, KVM ou VMware).

- Gerenciar o ciclo de vida completo das MVs, incluindo configurar a rede dinamicamente para grupos de MVs e gerenciar os requisitos de armazenamento, como tratar da movimentação das imagens de disco ou criar o ambiente de execução personalizado.

- Suportar a configuração de políticas para alocação de recursos que atendam as necessidades específicas das organizações, como alta disponibilidade ou consolidação de servidores para minimizar o consumo de energia

- Adaptar-se à demanda por recursos, quando os recursos locais não são suficientes para atender a demanda em momentos de pico ou mudanças de recursos como a inserção de novos recursos físicos ou falhas durante o funcionamento dos recursos físicos já existentes.

No caso de IaaS, o SGN oferece recursos virtuais como processadores, memórias, discos e redes virtuais, que combinadas formam máquinas virtuais (MV). O SGN faz uso de outros sistemas, ou sub-sistemas dentro do sistema da nuvem, para controlar e gerenciar cada recurso utilizado pela MV.

Para ilustrar o funcionamento do sistema de nuvem, a Figura 3.3 apresenta um modelo de arquitetura interna genérica do SGN com os sistemas de redes, configurações, imagens e discos que se comunicam com o gerenciador de MVs. O gerenciador de MVs utiliza os recursos virtuais criados nos sub-sistemas e os combina para criar uma descrição da MV. A descrição é utilizada para instanciar uma ou mais MVs no *hardware* ou hospedeiro subjacente. Cada hospedeiro contém um monitor de máquinas virtuais (MMV) que faz a instanciação e monitoramento das MVs localmente.

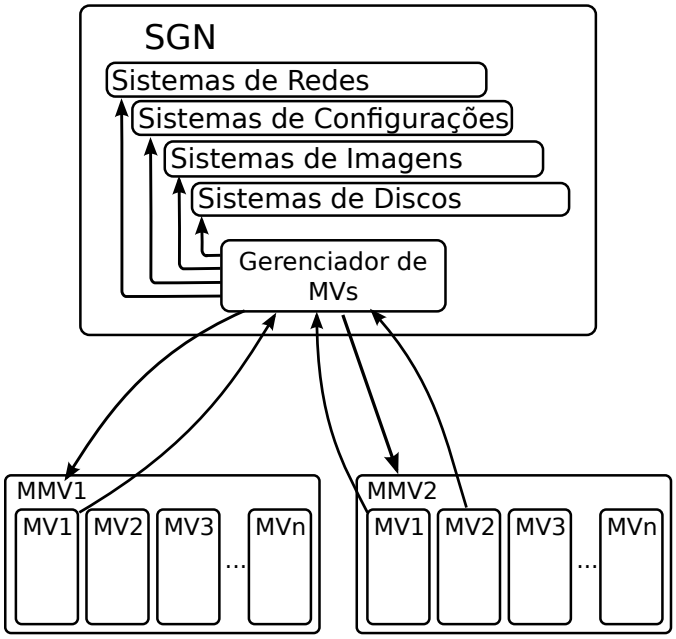


Figura 3.3: Modelo de arquitetura interna genérica de um SGN com os sub-sistemas de redes, configuração, imagem e disco.

Atualmente existem diversas implementações em *software livre* de SGNs para criação de serviços IaaS, dentre os quais podemos citar o OpenNebula [55, 41], Eucalyptus [45], Nimbus [30, 31] e OpenStack [49]. Cada sistema implementa os sub-sistemas descritos anteriormente e também oferece suporte aos protocolos de nuvens públicas como o EC2 [26], S3 [27] e OCCI [22] para gerenciamento de recursos.

O OpenNebula foi o SGN escolhido para uso neste trabalho por ser um sistema inteiramente em *software livre*, suportar diversas linguagens no desenvolvimento de aplicações, facilidade de instalação e configuração dos sistema (sub-sistemas) e suporte aos protocolos públicos de nuvens. A Seção 3.4.1 apresentar e descreve em maiores detalhes o sistema do OpenNebula.

3.4.1 OpenNebula

O OpenNebula /[55] é um sistema que disponibiliza e gerencia MVs, individualmente ou em grupos, que são instanciados na infraestrutura interna (recursos locais) ou em

nuvens externas ou públicas. O sistema oferece suporte para implantação de uma nuvem privada, pública ou híbrida.

A Figura 3.4 apresenta a arquitetura do OpenNebula que contém os componentes de núcleo (*core*), *drivers* e o escalonador de MVs (*scheduler*). O OpenNebula se apresenta como um sistema intermediário entre as infraestruturas de recursos físicos (servidores, *clusters*, supercomputadores) e os sistemas *front-end* (linha de comando ou interface de nuvem), que são usadas por usuários ou desenvolvedores.

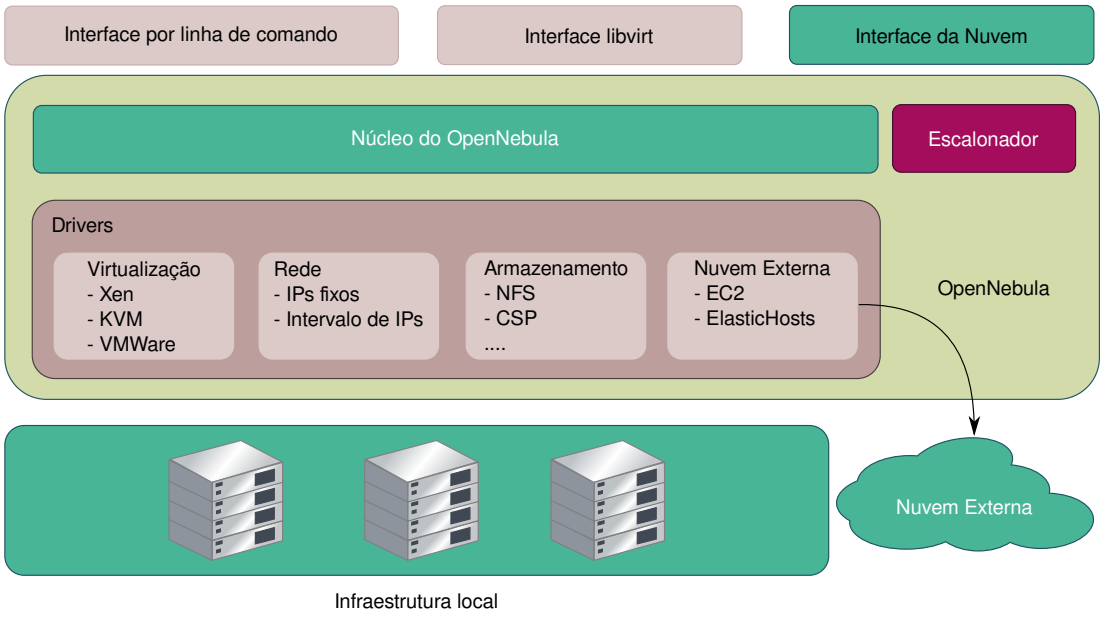


Figura 3.4: Arquitetura do OpenNebula. Adaptado de [55].

O núcleo controla o ciclo de vida da MV através do gerenciamento das redes virtuais, gerenciamento do armazenamento e comunicação com os MMVs. O núcleo também realiza operações específicas de armazenamento, rede ou virtualização com os *drivers*, de forma que estes oferecem abstrações sobre as tecnologias utilizadas na infraestrutura.

Os *drivers* são vistos como módulos que podem ser inseridos ou removidos sem afetar o funcionamento do núcleo. Os módulos implementam os protocolos e comandos relacionados as tecnologias, tornando o sistema do OpenNebula menos acopladas a tecnologias específicas. Alguns dos *drivers* implementados suportam compatibilidade com os monitores de máquinas virtuais Xen, KVM e VMWare.

Por fim, o escalonador é um componente que decide onde uma MV será instanciada de acordo com as informações obtidas dos monitores de recursos físicos e virtuais. O escalonador implementa os algoritmos de escalonamento, como *first-fit* e *round-robin*, e indicam o hospedeiro mais apropriado para instanciação da MV para o núcleo.

A partir do núcleo são desenvolvidos interfaces de gerenciamento externos que são usados para integrar o OpenNebula com outras ferramentas, por exemplo sistemas de contas de usuário ou monitoramento.

3.5 Workflow Científico e Computação em Nuvens de IaaS

Recentemente, nuvens IaaS vem sendo utilizadas para execução de aplicações científicas. Alguns trabalhos investigam o uso deste paradigma [52, 62, 34, 23, 59] e apontam vantagens no controle do recurso virtual por parte do cientista e fácil aquisição de novos recursos específicos exigidos para execução das aplicações legadas.

Normalmente, as MVs obtidas são usadas para processamento distribuído de aplicações de alto desempenho ou sistemas de armazenamento de dados paralelos em larga escala. Em outros casos, as MVs são adquiridas para formar ou incrementar uma infraestrura computacional como criação de *clusters* virtuais, incrementar redes de processamento *Peer-to-Peer* [7] ou aumentar os recursos de uma grade computacional [18].

Os *workflows* científicos (WfC) (apresentado no Capítulo 2) podem fazer uso da computação em nuvem para execução de aplicações científicas. Um WfC compõe um fluxo coordenado de tarefas que descrevem o processo de experimentação através do uso de aplicações ou serviços [11].

A execução dos fluxos de trabalho é feita no sistema de gerenciamento de WfC (SGWfC) que trata de orquestrar e implementar a comunicação e a integração com os ambientes de execução de forma a alocar os recursos necessários para execução das tarefas. Dessa forma, o SGWfC precisa implementar os protocolos e os comandos para requisitar e liberar as máquinas virtuais nos sistemas gerenciador da nuvem (SGN), através das

interfaces de serviço de infraestrutura (IaaS).

O uso das máquinas virtuais oferecem vantagens como um ambiente elástico [3], personalizado [61] e isolado [30, 34]. O ambiente é dito elástico devido a possibilidade de aumentar e diminuir (escalabilidade) a quantidade de recursos computacionais de acordo com a necessidade e com pouco esforço por parte dos provedores e usuários. Por exemplo, é possível adquirir mais recursos virtuais (processador, memória, disco e rede) para execução de aplicações distribuídas através do envio de poucos comandos.

A personalização é a capacidade do ambiente de execução conter os requisitos de execução das aplicações. Assim é possível empacotar os *software* (sistema operacional, bibliotecas, programas, *scripts*, etc) e adquirir o *hardware* (arquitetura do processador, quantidade de memória, quantidade de processadores, quantidade de discos) de acordo com os requisitos das aplicações. Dessa forma, o cientista só precisa instalar e configurar o ambiente de execução uma vez, sendo possível replicá-lo para execução ou reexecução futura.

Por fim, o isolamento é a capacidade do ambiente virtual em restringir as ações dos usuários aos respectivos recursos virtuais, sem afetar os recursos virtualizados de outros usuários. O isolamento oferece maior controle e autonomia ao usuário, dando-lhes acesso e controle total aos ambiente de execução, como obter privilégios de administrador na máquina virtual. Com isso é possível personalizar o ambiente de execução sem afetar o ambiente de outros, como instalar bibliotecas e programas sem gerar conflito de pacotes de *software*. Essa é uma característica importante na experimentação científica, uma vez que as aplicações científicas utilizam pacotes de *software* distintas, desatualizadas e, em alguns casos, sem suporte.

A execução dos WfC na nuvem pode ser feita de diferentes formas. Os SGWfCs foram projetados para executar em ambientes de alta performance como *clusters*, grades computacionais ou supercomputadores. Alguns SGWfC fazem uso de escalonadores ou alocadores de recursos existentes nos ambientes de alta-performance subjacentes e outros implementam mecanismos próprios para escalonar e alocar recursos para execução das

tarefas.

Assim, a integração de WfC com a nuvem computacional pode ser feita adaptando-se o WfC para a nuvem ou adaptando a nuvem para o WfC [28]. No primeiro caso é feito a substituição do componente de execução do SGWfC por um específico da nuvem. Nesse caso, a nuvem implementa o componente de execução de tarefas que fazem uso de filas de tarefas e algoritmos de escalonamento para otimizar o uso da infraestrutura da nuvem.

Já no segundo caso, a nuvem é usada para instanciar diversas MVs para criar um *cluster virtual* ou aumentar os recursos de uma grade computacional. A nuvem incrementa a quantidade de recursos para execução das tarefas, de forma que o componente de execução do WfC não é alterado.

Alguns trabalhos [29, 60] adaptam a nuvem para execução de WfC. Juve *et al.* [29] fazem uma análise de desempenho e custos para executar os WfC na nuvem Amazon EC2 através de diferentes aplicações científicas. As aplicações variam no uso de memória, processador e rede e são executados em diferentes instâncias de MVs que contém custos distintos. Em Vöckler *et al.* [60] descreve-se as experiências no uso de diferentes nuvens para execução de WfC com aplicações reais, ao invés de *benchmarks*. Nesses casos, a nuvem aumenta a quantidade de nodos (*worker nodes*) usados pelo escalonador de tarefas *Condor* [36] para execução de tarefas em paralelo e distribuída.

Já em outros trabalho adaptam o WfC para a nuvem. Em Oliveira *et al.* [48] é proposto um *middleware*, chamado de *SciCumulus*, que explora a execução paralela de aplicações através da troca de parâmetro ou processamento de dados em paralelo. O *middleware* instancia a quantidade de recursos necessários para execução da tarefa e gerencia a execução das aplicações na nuvem. Para usar o *SciCumulus*, o cientista substitui as tarefas do WfC original por tarefas específicas que se comunica com o *middleware* e submetem as aplicações para execução distribuída.

Em Shams *et al.* [54] é proposto um *framework*, chamado *Polyphony*, para facilitar a transferência e o processamento de imagens de Marte. O *Polyphony* utiliza filas de

tarefas distribuídas, que são implementadas na nuvem, e um cliente que faz a submissão das tarefas em tais filas. A execução das tarefas é feita quando existem nodos (*worker nodes*) disponíveis para executá-las. Os nodos requisitam tarefas para uma das filas e, em caso de existir tarefas não processadas, as mesmas são executadas. As tarefas contém descrições das próximas tarefas, de forma que ao terminar a execução de uma tarefa, a mesma adiciona mais tarefas na fila formando o fluxo de tarefas. A execução do WfC termina quando a última tarefa executa, sem adicionar mais tarefas em alguma fila.

O presente trabalho propõe uma adaptação do WfC para execução de tarefas na nuvem através de uma extensão na representação do WfC. A extensão na representação do WfC para uso de serviços IaaS é apresentada no Capítulo 4.

CAPÍTULO 4

EXTENSÃO DE MAPEAMENTO DE *WORKFLOW* CIENTÍFICO PARA EXECUÇÃO NA NUVEM

Conforme apresentado no Capítulo 2, *workflows* científicos (WfC) são usados para representar o processo de experimentação científico. Os WfCs utilizam-se de modelos abstratos ou WfC abstratos (WfCA) para representar o processo em alto nível, sem fazer menção sobre os recursos utilizados para a execução das tarefas. Para executar um WfC em formato abstrato é preciso fazer o *mapeamento* de cada tarefa abstrata para os recursos, gerando um WfC concreto (WfCC).

O mapeamento é uma das etapas envolvidas no ciclo de vida do *workflow* científico e o mesmo faz uso de buscadores de recursos existentes e informações sobre as tarefas abstratas (TA) para mapear o recurso adequado. Um mapeamento tradicional, trasforma as TAs em *jobs* ou serviços e adiciona tarefas concretas (TC) para movimentação das aplicações e dos dados no ambiente de execução. Dentre as TCs encontram-se a criação de diretórios para execução da aplicação, movimentação dos programas, movimentação dos dados de entrada e saída, armazenamento de meta-dados (proveniência) e remoção de arquivos temporários [12, 60].

Dentre as possibilidades para a execução de tarefas do WfC estão as nuvens computacionais (ver Capítulo 3). A computação em nuvem contém serviços de infraestrutura (*Infrastructure as a Service* - IaaS) que disponibilizam recursos virtuais como máquinas virtuais (MVs). As MVs fornecem um ambiente personalizado e isolado para execução de aplicações e são providas elasticamente podendo instanciá-las e liberá-las a medida que for necessário (sob-demanda).

O uso da nuvem exige mudanças no mapeamento tradicional uma vez que os recursos

usados não estão previamente instanciados. Além disso, é necessário controlar a instanciação pois a instanciação em momentos inadequados, para a execução do WfC, geram custos desnecessários e uma utilização ineficiente da infraestrutura da nuvem.

Em alguns trabalhos [29, 60] as MVs são instanciadas antes de iniciar a execução do *workflow*, deixando intervalos de recursos virtuais ociosos. Em um pior caso, os recursos virtuais somente são utilizados na execução da última tarefa, deixando as MVs ociosas por todo o intervalo entre a instaciação até a execução da tarefa.

Assim, o presente trabalho propõe uma modificação na etapa de mapeamento que inclui uma extensão na representação do WfC abstrato para auxiliar a associação das tarefas com recursos de nuvem ou MVs. Com o formato abstrato extendido é feito o mapeamento para o formato concreto de forma a conter TCs que instanciam e liberam os recursos nos momentos adequados, durante a execução do WfC.

Diferente do mapeamento tradicional, os recursos da nuvem não existem até o momento da execução do WfC. Para isso, o mapeamento precisa descobrir qual é a MV com o ambiente adequado para execução das tarefas (quantidade de processador, quantidade de memória, sistema operacional, bibliotecas, programas e *scripts*), gerando o WfCC com as tarefas concretas que instanciam e liberam as MVs nos momentos adequados.

A Figura 4.1 mostra a etapa de mapeamento modificado em duas fases. A primeira é chamada de pré-mapeador que recebe o WfCA, mapeia as tarefas abstratas para os recursos de nuvem e retorna o WfCA extendido. Nessa fase é possível utilizar um catálogo que contém os registros sobre os possíveis recursos de nuvem que realizam a computação descrita pela tarefa. Já na segunda fase, chamada de mapeador, é feito o mapeamento do WfCA extendido para o formato concreto que contém as tarefas de alocação e liberação dos recursos. O mapeador faz uso das informações obtidas dos sistemas de nuvens de definindo em qual nuvem o recurso será instanciado.

Para explicar em detalhes a modificação no mapeamento, este o Capítulo está dividido da seguinte forma. O WfCA extendido é explicado em detalhes na Seção 4.1. As fases do

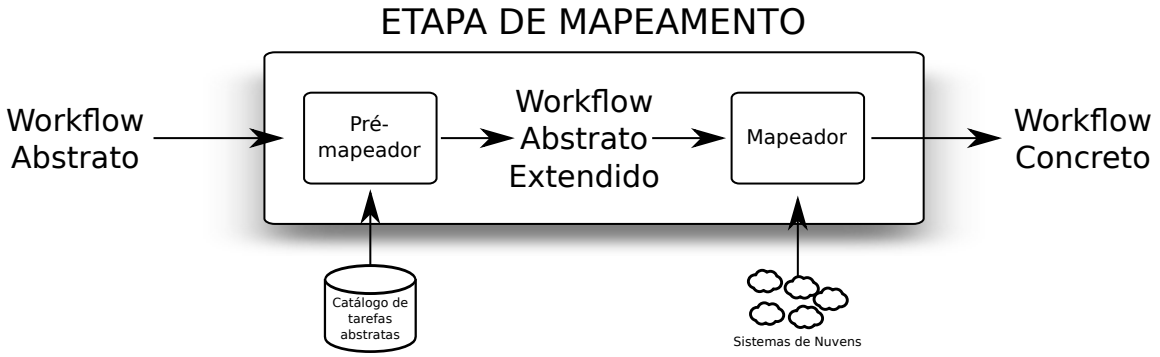


Figura 4.1: Etapa de mapeamento modificada com duas fases.

mapeamento são descritos na Seção 4.2.

4.1 Extensão do *Workflow* Científico Abstrato para Suportar Recursos de Nuvem

Um WfCA normalmente utiliza estrutura de *grafo direcionado acíclico* no qual os nodos são as TAs e as arestas são as dependências de dados entre as TAs. Para suportar os recursos usados pelas TAs, aqui é proposto uma extensão baseada em grafo bipartido que associa as TAs com os recursos virtuais usados para execução.

Neste contexto, os recursos de nuvem descrevem os requisitos de execução de *software* e *hardware* envolvidos na execução da TA. No caso de *jobs* que fazem uso de aplicações, o recurso de nuvem descrevem as bibliotecas, sistema operacional, programas, *scripts* e *frameworks* como os requisitos de *software* e arquitetura do processador, quantidade de memória, quantidade de discos e quantidade de processadores como requisitos de *hardware*. A lista de requisitos (recursos) utilizados pode variar de acordo com o grau de especificidade das aplicações e dos recursos físicos.

Assim, de modo geral o WfCA é representado através da notação de tupla $W = \{T, D\}$, onde T=$\{t_0,t_1,t_2,...,t_n\}$ e $n \in \mathbb{N}$ é o conjunto finito de TAs, e D = $\{(t_i,t_j)/t_i \neq t_j$ e $t_i, t_j \in T\}$ é o conjunto de dependências entre as TAs que não formam circuito (acíclico).

A associação das TAs com os recursos virtuais é representada através da notação em grafo bipartido $N_W = \{T(W) \cup R_N, D_N\}$, onde $T(W)$ é a função que retorna o conjunto de TAs do WfCA W e $T(W) \cap R_N = \emptyset$, R_N é o conjunto finito de recursos de nuvem $\{n_0,n_1,...,n_m\}$, onde $m \in \mathbb{N}$, e D_N é o conjunto que relaciona as TAs com os recursos $\{(t,c)/t \in T(W)$ e $c \in N\}$.

As associações é usado para indicar o intervalo de uso dos recursos virtuais baseado na execução da tarefa. Dessa forma, se uma tarefa utiliza um determinado recurso, o mesmo deve ser alocado antes da execução e liberado após a execução da tarefa. Um recurso que é utilizado por várias tarefas sequenciais pode ser alocado e liberado apenas uma vez.

As associações podem ser feitas de diferentes formas. Aqui apresentamos três associações entre as TAs e os recursos de nuvens sendo de (1) uma TA para um recurso de nuvem, (2) várias TAs para um recurso de nuvem e (3) uma TA para vários recursos.

Relação de uma TA para um recurso de Nuvem é o caso mais simples no qual uma tarefa tem somente um conjunto requisito para execução. Esta relação é representada pelo sub-conjunto $D_N^t = \{(t,n)\}$ de D_N que contém a tupla com a TA t. A Figura 4.2 ilustra a relação $D_N^{t_0} = \{\{t_0, n_0\}\}$ entre uma TA ($T = \{t_0\}$) e um recurso de Nuvem ($R_N = \{n_0\}$).

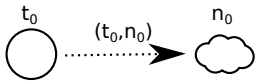


Figura 4.2: Exemplo de relação de uma TA com um recurso de Nuvem.

A segunda associação é da de várias TAs para um recurso de nuvem. Nesse caso, as TAs compartilham de um mesmo requisito ou executam usando o mesmo recurso. Esta relação é representada pelo sub-conjunto $D_N^n = \{(t_0,n),(t_1,n),...,(t_n,n)\}$ de D_N que contém as tuplas com o recurso n. A Figura 4.3 ilustra a relação $D_N^{n_0} = \{\{t_0,n_0\},\{t_1,n_0\},\{t_2,n_0\}\}$ que contém três TAs ($T = \{t_0,t_1,t_2\}$) e um recurso ($R_N = \{n_0\}$).

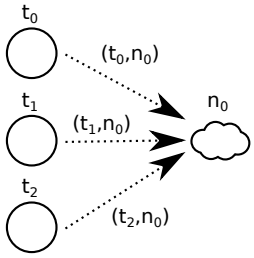


Figura 4.3: Exemplo de relação de várias TAs com um recurso de Nuvem.

A terceira associação é a de uma TA para diversos recursos de nuvem. Esse caso mostra que uma TA pode ser computada através de diferentes recursos ou utilizando vários recursos ao mesmo tempo. Por exemplo, é possível utilizar programas implementados utilizando tecnologias distintas que solucionam o problema descrito pela TA. Assim, a terceira relação é representada pelo sub-conjunto de D_N, $D_N^t = \{(t, n_0), (t, n_1), ..., (t, n_m)\}$ que contém as tuplas com a TA t. A Figura 4.4 ilustra a relação $D_N^{t_0} = \{\{t_0, n_0\}, \{t_0, n_1\}, \{t_0, n_2\}\}$ que contém uma TA ($T = \{t_0\}$) e um recurso ($R_N = \{n_0, n_1, n_2\}$).

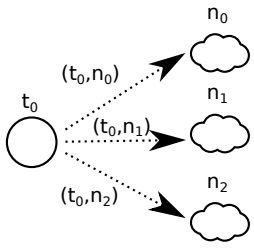


Figura 4.4: Exemplo de relação de uma TA com vários recursos de Nuvem.

Para ilustrar um exemplo do modelo extendido, a Figura 4.5 mostra um exemplo de WfCA que contém um conjunto com quatro TAs $T = \{t_0, t_1, t_2, t_3\}$ que formam um fluxo de dados representado pelo conjunto de dependências $D = \{(t_0, t_1), (t_0, t_2), (t_1, t_3), (t_2, t_3)\}$. A figura também ilustra o conjunto de recursos $R_N = \{n_0, n_1, n_2, n_3, n_4, n_5, n_5, n_6\}$ e a relação entre os requisitos e as TAs pelo conjunto de dependência $D_N = \{(t_0, n_0), (t_0, n_1), (t_1, n_1), (t_1, n_2), (t_2, n_3), (t_3, n_0), (t_3, n_4),$
$(t_3, n_5), (t_3, n_6)\}$. O exemplo mostra os três tipos de relação das TAs com requisitos, de forma que algumas TAs contém os mesmos requisitos que outras.

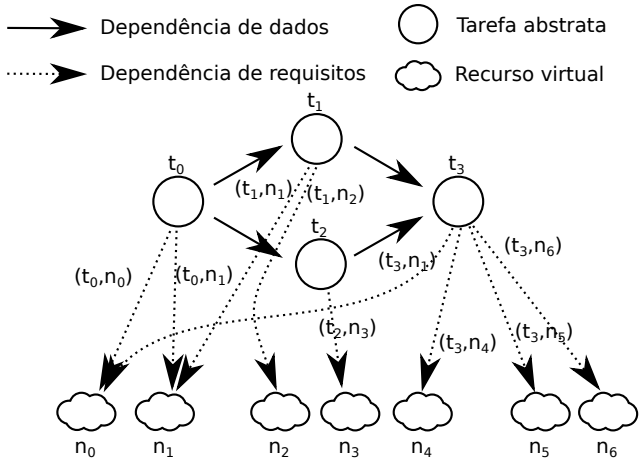


Figura 4.5: Exemplo de WfCA com quatro TAs que são associadas com sete recursos diferentes.

Para utilizar o modelo abstrato extendido proposto é preciso transformá-lo para um formato executável ou o WfCC. A Seção 4.2 descreve a etapa de mapeamento que transforma o WfCA extendido para o WfCC de forma a incluir as TCs que alocam e liberam os recursos de nuvem nos momento adequados.

4.2 Mapeamento do *Workflow* Abstrato Extendido para Execução na Nuvem

O mapeamento tradicional de uma TA consiste em movimentar os dados de entrada e aplicação para o recurso, iniciar a computação no recurso e movimentar ou recuperar os dados de saída (resultados). Cada uma dessas etapas é representada por uma tarefa concreta (TC) que contém descrições ou implementam os comandos que realizam a funcionalidade de cada etapa. Assim, o mapeamento das TAs para as TCs obtem-se um WfCC que descreve o fluxo de execução das aplicações nos recursos.

No entanto, para o mapeamento do modelo de WfCA extendido, proposto na Seção 4.1, os recursos virtuais ou máquinas virtuais (MV) só estarão disponíveis no momento de execução do WfC, não sendo possível associá-los da forma como é feito no mapeamento

tradicional. Com isso é necessário inserir as TCs que instanciam e liberam recursos na nuvem durante a execução, chamados de TC de nuvem (TCN).

No mapeamento para as TCNs é preciso encontrar as imagens de disco para instanciar o ambiente de execução que contêm os *softwares* para execução das tarefas. Dentre os *softwares* estão o sistema operacional, os pacotes, as bibliotecas e as aplicações. Uma imagem de disco pode executar diferentes tarefas, podendo até uma imagem atender o requisitos de execução de todas as TAs de um WfCA. Além disso é preciso também definir o *hardware* virtual usado pelas aplicações tais como memória virtual, processor virtual e rede virtual.

O WfCC do modelo extendido descreve um fluxo de execução com TCNs, que instanciam as MVs na nuvem IaaS, e as TCs tradicionais que iniciam a computação nas MVs. Dessa forma, é instanciar e liberar os recursos da nuvem explicitamente no WfCC.

É importante ressaltar que as TCNs devem ser colocadas corretamente para evitar problemas de ociosidade. Um cenário do problema de ociosidade é quando a primeira e a última TA compartilham de uma mesma imagem de disco, e que por consequência podem utilizar a mesma MV para efetuar a computação. Nesse caso, a alocação da MV ocorreria antes da execução da primeira TA e ficaria ocioso até o início da última TA.

Dessa forma, os algoritmos de mapeamento podem adotar diferentes abordagens para contornar o problema de ociosidade em decorrência da inserção indevida das TCNs. Uma das abordagens libera a MV e instancia novamente ou pode suspender ou pausar o recurso e depois reiniciar quando voltar a utilizá-lo.

A Figura 4.6 mostra o fluxograma de mapeamento com as etapas envolvidas no mapeamento do WfCA extendído (WfCAx) para o WfCC. Inicialmente o mapeamento utiliza as informações contidas no WfCAx para encontrar as imagens de disco virtual e configuração de MV para executar as aplicações das TAs. Para cada TA é cria as TCNs que instanciam e liberam as MVs e então é criado as TCs para movimentar os dados e TCs para executar as aplicações no ambiente virtual. Em seguida é criado as arestas entre as

TCs e TCNs de acordo com o fluxo descrito no WfCA gerando o WfCC que é retornado como resultado do mapeamento.

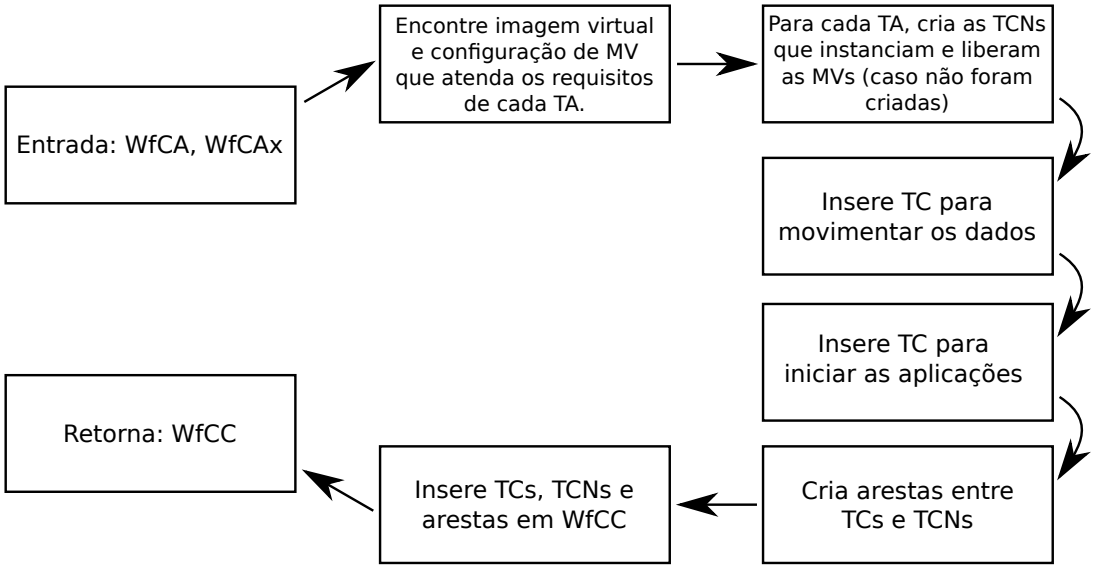


Figura 4.6: Fluxograma do mapeamento com as etapas do mapeamento do WfCA extendído para o WfCC com TCNs.

Para exemplificar um caso simples de mapeamento, a Figura 4.7 mostra um WfCA com as TAs $t_0, t_1, ..., t_n$ que são executadas antes da TA t_c, a qual está associada ao recurso de Nuvem n_0. Assim, o WfCA mapeado ou WfCC, presente na Figura 4.8, mostra a TCN de alocação do recurso n_0 antes da movimentação dos dados para este recurso.

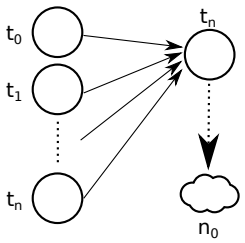


Figura 4.7: WfCA com TA que é executada na Nuvem com várias dependência de dados.

A seguir apresenta-se um caso de mapeado utilizando o Algoritmo **??** sobre o WfCA exemplo da Figura 4.5. Esse exemplo apresenta as relações das TAs com recursos de Nuvens e as múltiplas dependência de dados entre TAs. O mapeamento das TAs (t_0, t_1, t_2, t_3)

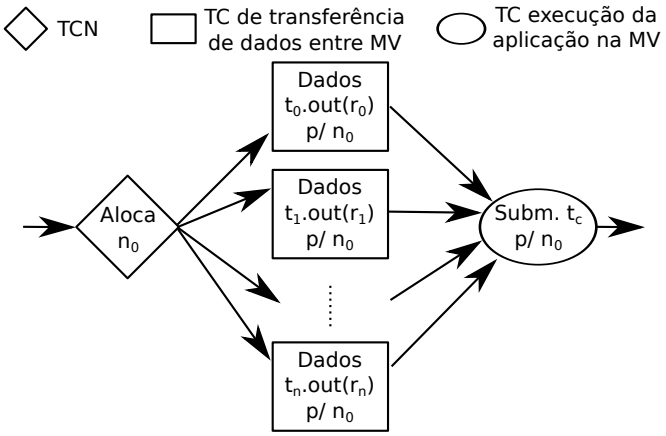


Figura 4.8: WfCC que aloca o recurso e faz a transferência de dados para o recurso alocado.

para TCs são representados nas Figuras 4.9, 4.10, 4.11, 4.12, respectivamente.

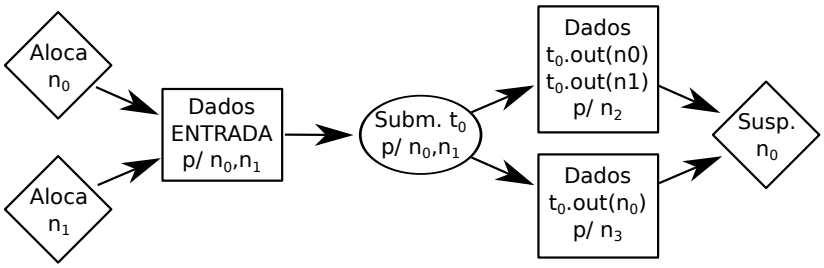


Figura 4.9: Mapeamento da TA t_0 para o modelo concreto.

Como a t_0 é a primeira TA a ser executada, os dados são provenientes da entrada do *workflow*. Da mesma forma com a t_3 no qual os dados são transferidos para a saída do *workflow*, ou seja, um local onde o usuário consegue acessá-los.

O mapeamento de cada TA contém as TCs e TCNs de acordo com as dependência de dados, transformando o WfCA em WfCC. A Figura 4.13 representa o WfCC completo, que contém as TCNs inseridas nos momentos adequados para execução do *workflow* no ambiente da Nuvem.

O mapeamento do WfCA extendído permite explícitar o uso dos recursos de nuvem, possibilitando a inserção de TCNs que alocam e liberam os recursos a medida que for necessário para a execução das tarefas. Para mostrar um caso de uso, da extensão proposta,

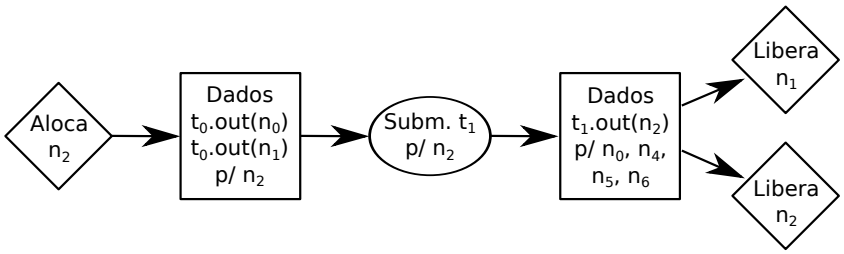


Figura 4.10: Mapeamento da TA t_1 para o modelo concreto.

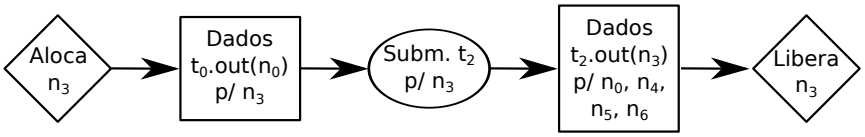


Figura 4.11: Mapeamento da TA t_2 para o modelo concreto.

o Capítulo 5 apresenta uma implementação feita sobre o Sistema Gerenciador de WfC Kepler [2] em conjunto com o Sistema de Gerenciamento de Nuvem OpenNebula [55, 41].

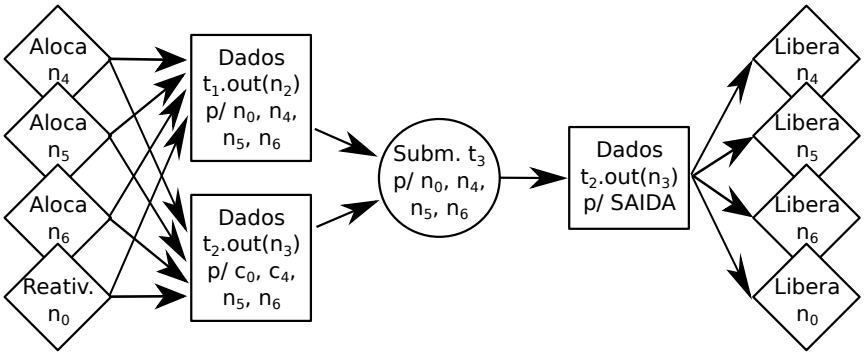


Figura 4.12: Mapeamento da TA t_3 para o modelo concreto.

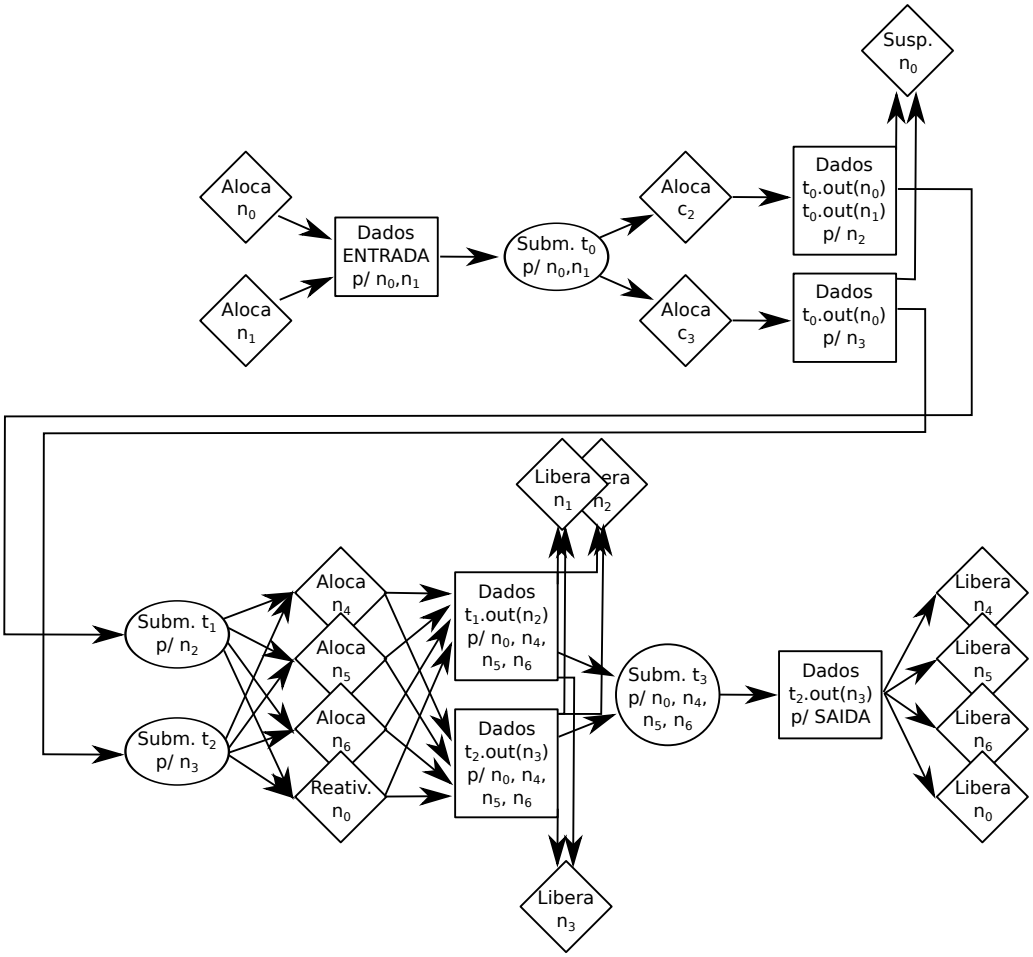


Figura 4.13: WfCC que faz a execução do WfCA do exemplo da Figura 4.5.

CAPÍTULO 5

KLOUD: EXTENSÃO DO SGWFC KEPLER PARA SUPORTAR NUVENS IAAS

Utilizando-se do modelo de *workflow* científico (WfC) abstrato extendido e o mapeamento modificado no Capítulo 4, aqui é apresentado uma implementação de integração do Sistema de Gerenciamento de Workflow Científico (SGWfC) Kepler e o Sistema de Gerenciamento de Nuvem (SGN) OpenNebula para execução das tarefas concretas (TC).

O SGWfC Kepler apresenta vantagens na instalação e configuração do sistema, uma vez que o mesmo não necessita de instalação de uma infraestrutura complexa de recursos. O Kepler também oferece maior flexibilidade para representar os diferentes modelos de computação científica, podendo se adaptar aos diferentes contextos de WfC (ver Capítulo 2).

O OpenNebula foi o SGN escolhido por ser um sistema inteiro em *software livre*, suportar diversas linguagens no desenvolvimento de aplicações, facilidade de instalação e configuração dos sistema e suporte aos protocolos públicos de nuvens. O OpenNebula foi implantado de modo privado em uma infraestrutura com quatro hospedeiro e um *front-end* para gerenciamento remoto da nuvem. Através do *front-end* é possível requisitar recursos na nuvem.

As tarefas concretas de nuvens (TCN) foram implemetadas dentro do módulo de extensão do Kepler, chamado *Kloud*. O *Kloud* contém os atores ou tarefas que implementam os comandos e protocolos de nuvens (EC2, OCCI, S3) para realizar a instanciação e liberação dos recursos virtuais. Os atores de TCN implementam parte dos protocolos e oferecem abstrações não sendo necessário programar ou executar os comandos para requisitar os recursos na nuvem.

O Kepler oferece uma interface gráfica para composição de WfC concreto (WfCC) e abstrato (WfCA). No Kepler os WfC são descritos de forma concreta uma vez que os atores mencionam os recursos (tarefas concretas) utilizados para realizar as funcionalidades das tarefas. No entanto, é possível aninhar (agrupamento) atores gerando atores ou tarefas de alto nível que contém um *sub-workflow*. Em um caso mais extremo de abstração, o ator de alto nível pode ser utilizado na etapa de composição, sem a necessidade de configurar ou mencionar os recursos (tarefas abstratas).

Com os atores de alto nível ou tarefas abstratas (TA) é possível criar um WfCA, sendo que o mapeamento já está determinado no aninhamento e feito de forma manual. Para ajustar quais recursos utilizar é preciso abrir os atores de alto nível e alterar os dados e parâmetros utilizados pelos atores internos ou tarefas concretas (TC). Isso também é aplicado para as TCNs que precisam ter os parâmetros de configuração da nuvem ajustados.

Para implementação das TCNs, ou atores do *Kloud*, a tabela 5.1 apresenta algumas das tarefas relevantes com os respectivos tipos e funcionalidades. As tarefas estão agrupadas em (1) tarefas de instanciação, (2) tarefas de mudança de estado e (3) tarefas auxiliares. As tarefas de instanciação são responsáveis por alocar e liberar as MVs. As tarefas de mudança de estado alteram o estado das MVs. Por último, as tarefas auxiliares são usadas para prover informações auxiliares que podem ser usados no processo de instanciação.

Para exemplificar o uso das TCN implementadas no *Kloud*, a Figura 5.1 mostra um *workflow* que instancia uma MV. O *workflow* faz uso das tarefas Criar (*CreateVM*), Iniciar (*StartVM*), EsperaEstado (*WaitSuccess*), TestaConexão(*TestConnection*) e Informação (*InfoVM*). Para manter a seção e a conexão com o SGN, é usada a tarefa de Sessão de Cloud (*CloudSession*).

A tarefa de sessão de Cloud trata da autenticação do usuário na Nuvem. Tem como entrada o endereço da Nuvem (*URL*), o usuário, a senha, as chaves, a *hash* de autenticação, dentre outros. Como saída a tarefa retorna uma *string* de autenticação de forma que as outras TCNs possam utilizá-la para realizar as operações na nuvem.

Tarefa	**Tipo**	**Função**
Criar	Instanciação	Cria uma instância de MV, mas não inicia a MV em algum hospedeiro.
Destruir	Instanciação	Força o desligamento da MV e remove a instância de MV.
Informação	Instanciação	Obtém informação sobre uma instância de MV (nome, descrição, rede virtual, disco virtual, hardware virtual, etc.).
Iniciar	Mudar Estado	Inicia a instância de MV em algum hospedeiro.
Desligar	Mudar Estado	Desliga a MV.
Hibernar	Mudar Estado	Salva o estado de uma MV e desliga a instância.
Pausar	Mudar Estado	Pausa a MV e a mantém instanciada no hospedeiro.
Suspender	Mudar Estado	Salva o estado da MV, pausa a instância e remove-a do hospedeiro.
EsperaEstado	Auxiliar	Espera por um estado específico no processo de instanciação (por exemplo o estado de executando (*running*) ou estado de erro).
TestaConexão	Auxiliar	Testa a conexão com uma instância de MV.
Sessão de Cloud	Auxiliar	Cria e mantém uma sessão com SGN a fim de executar as tarefas em uma determinada nuvem.

Tabela 5.1: Tabela com as TCNs com os respectivos tipos e funcionalidades para instanciação e liberação das MVs.

A tarefa de criar (*CreateVM*) instancia uma MV e recebe como entrada um *template* de MV e a autenticação da sessão. Na versão utilizada do OpenNebula (2.0), é possível utilizar um *template* que descreve a configuração da instância de MV com os recursos virtuais como número de processadores, quantidade de memória, discos e rede. Como resultado, a tarefa aloca uma instância e retorna o identificador da MV (MVID) instanciada. A partir do MVID, é possível localizar a instância na nuvem para realizar operações para obter informações e alterar o estado da máquina.

As tarefas auxiliares como o EsperaEstado (*WaitSuccess*) e TestaConexão (*TestConnection*) tratam respectivamente por esperar o estado, por exemplo de sucesso, e de conexão com a MV. A MV contém diferentes estados durante o processo de instanciação

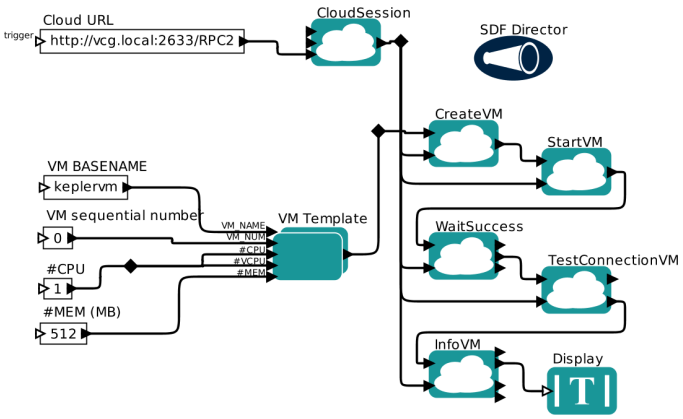


Figura 5.1: *Workflow* no SGWfC Kepler que instancia e inicia um MV através das tarefas do Kloud.

dentre os quais estão o estado de preparação, escalonamento, sucesso e falha. Ambas as tarefas auxiliares são usadas para garantir que a MV foi instanciada corretamente antes de iniciar a execução das aplicações. Por último, a tarefa de informação (*InfoVM*) busca informações sobre a MV e retorna o nome da máquina, endereço IP e o estado da MV.

Note que o MVID gerado pela tarefa "*Criar*" é repassada por cada uma das TCN de instanciação, de forma a gerar um fluxo de controle que garante a ordem na qual as TCN são executadas e qual é a instância de MV na qual está sendo utilizada.

Com as tarefas do *Kloud* definidas, a Seção 5.1 apresenta um caso de uso com um *workflow* que instancia diferentes quantidades de MV para execução de uma aplicação em paralelo. A aplicação faz a múltiplicação de matrizes em paralelo utilizando MPI.

5.1 Multiplicação de matrizes em MPI

Para testar o Kloud foi usado um caso de uso de uma aplicação que faz a multiplicação de matrizes quadradas através de MPI com nodos de processamento instanciados na nuvem. As matrizes usadas para o cálculo tem tamanho quadrado variando de 100 até 1500, aumentando de 100 em 100. As tarefas são explicadas com maiores detalhes a seguir. A Figura 5.2 mostra o *workflow* do caso de uso com 5 tarefas abstratas sendo a primeira a tarefa "*Pre-data*" que prepara os dados que vão ser processados. As tarefas "*MM_ 8n*",

"*MM_4n*" e "*MM_2n*" configuram e distribuem a computação na nuvem utilizando 8, 4 e 2 nodos respectivamente. Cada tarefa de multiplicação de matrizes retornam um arquivo com duas colunas sendo a primeira o tamanho da entrada e a segunda o tempo gasto para executar todas as entrada. Por último está a tarefa "*Plot_result*" que trata dos dados resultante e gera um gráfico para comparação de resultados.

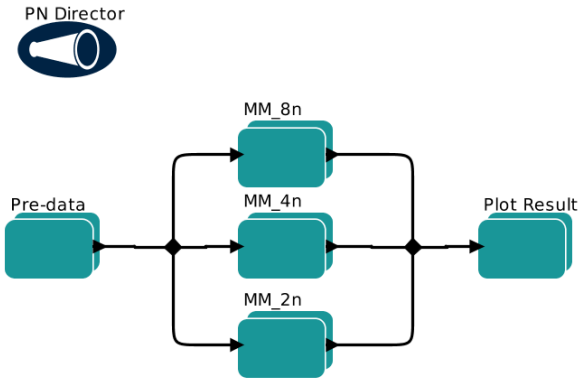


Figura 5.2: WfCA de caso de uso que faz a multiplicação de matrizes usando 2, 4 e 8 nodos de processamento.

A Figura 5.3 apresenta a tarefa "MM_2n" vista internamente. Esta tarefa contém um *sub-workflow* com 3 tarefas. A primeira tarefa "*Create 2 VMs*" instancia dois nodos na nuvem e tem como saída os endereços de rede (IP) de cada nodo. A tarefa "*MM-mpi-2-nodes*" contém outro *sub-workflow* que faz a distribuição dos dados e aplicação, configura os nodos MPI e inicia a computação. A tarefa tem como saída o resultado da múltiplicação das matrizes. Por último, a tarefa "*Release 2 VMs*" libera os dois nodos na nuvem após a execução da aplicação.

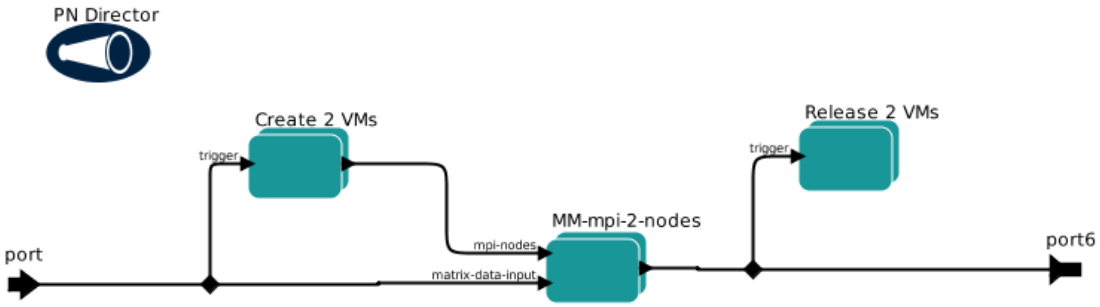


Figura 5.3: Tarefa "MM_2n" visto como um *sub-workflow* com 3 TA.

A tarefa de instanciação "Create 2 VMs" é uma derivação do *workflow* da Figura 5.1. Para ilustrar o maior caso com 8 instanciação a Figura 5.4 mostra a tarefa de criação de 8 nodos. A tarefa contém um *sub-workflow* que consiste nas etapas (1) manter sessão com o SGN, (2) cria o template da MV, (3) instanciar as MVs, (4) iniciar as MVs, (5) espera por estado de sucesso, (6) espera por conexão com as MVs, (7) obtem o endereço IP de cada MV e (8) gerar uma lista de IPs para retornar como resultado da instanciação.

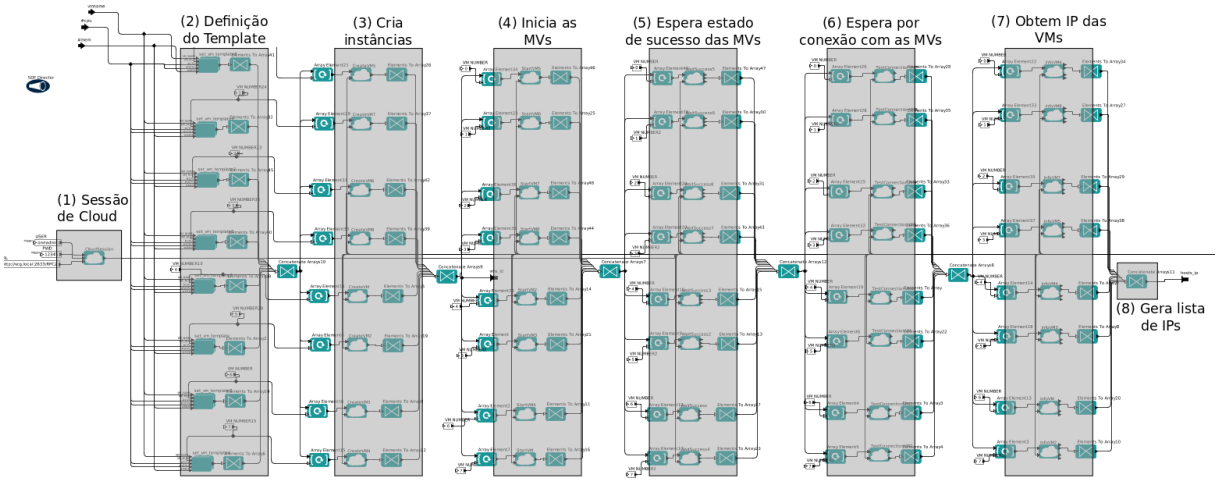


Figura 5.4: Tarefa de criação que instancia 8 MVs.

A tarefa "*MM-mpi-2-nodes*" consiste em enviar a aplicação MPI, em conjunto com os dados, executar o comando para iniciar a aplicação e por fim recuperar os resultados.

A tarefa recebe como a lista de IPs da MVs, o usuário que irá executar a aplicação na MV, o caminho da aplicação MPI, o comando para iniciar a computação e o caminho, na máquina local, onde será armazenado os resultados.

A Figura 5.5 mostra o *workflow* que trata de (1) selecionar as MVs que vão ser utilizadas, (2) copiar o pacote que contém a aplicação e os dados, (3) descompacta o pacote, (4) executa a aplicação com os parâmetros e os dados de entrada e (5) movimenta os resultado para ambiente local. O *workflow* utiliza de somente 2 MVs, sendo que nos outros dois casos com 4 e 8 nodos, o fluxo de seleção das MVs, copiar pacote e descompactar o pacote são replicado 4 e 8 vezes respectivamente. Para executar a aplicação é assumido que o MPI mestre é o primeiro IP da lista de IPs. Para este é criado uma lista de MPI escravos que são usados para distribuir os dados para execução da aplicação.

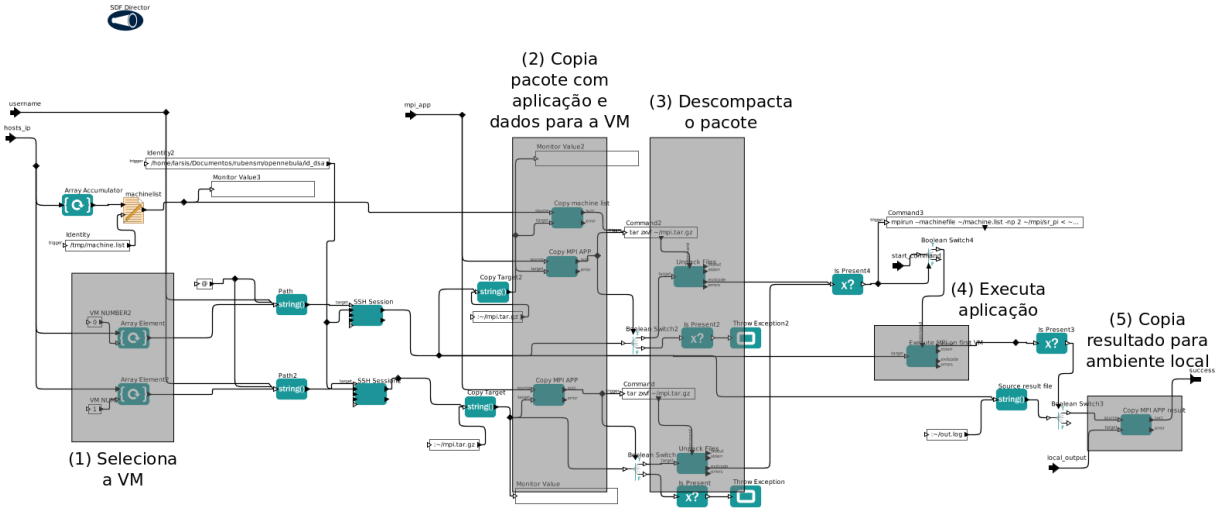


Figura 5.5: Tarefa que executa a aplicação de multiplicação de matrizem com 2 nodos ou MVs.

A tarefa "*Release 2 VMs*" recebe a lista de MVIDs e libera os recursos virtuais na nuvem. Para ilustrar o funcionamento da tarefa a Figura 5.6 mostra um *sub-workflow* que libera as 8 MVs.

Após executar as aplicações os dados resultantes é repassado para a tarefa "*Plot_result*" que trata os dados brutos, gera uma tabela em um formato reconhecido pelo gerador de gráfico nativo do Kepler.

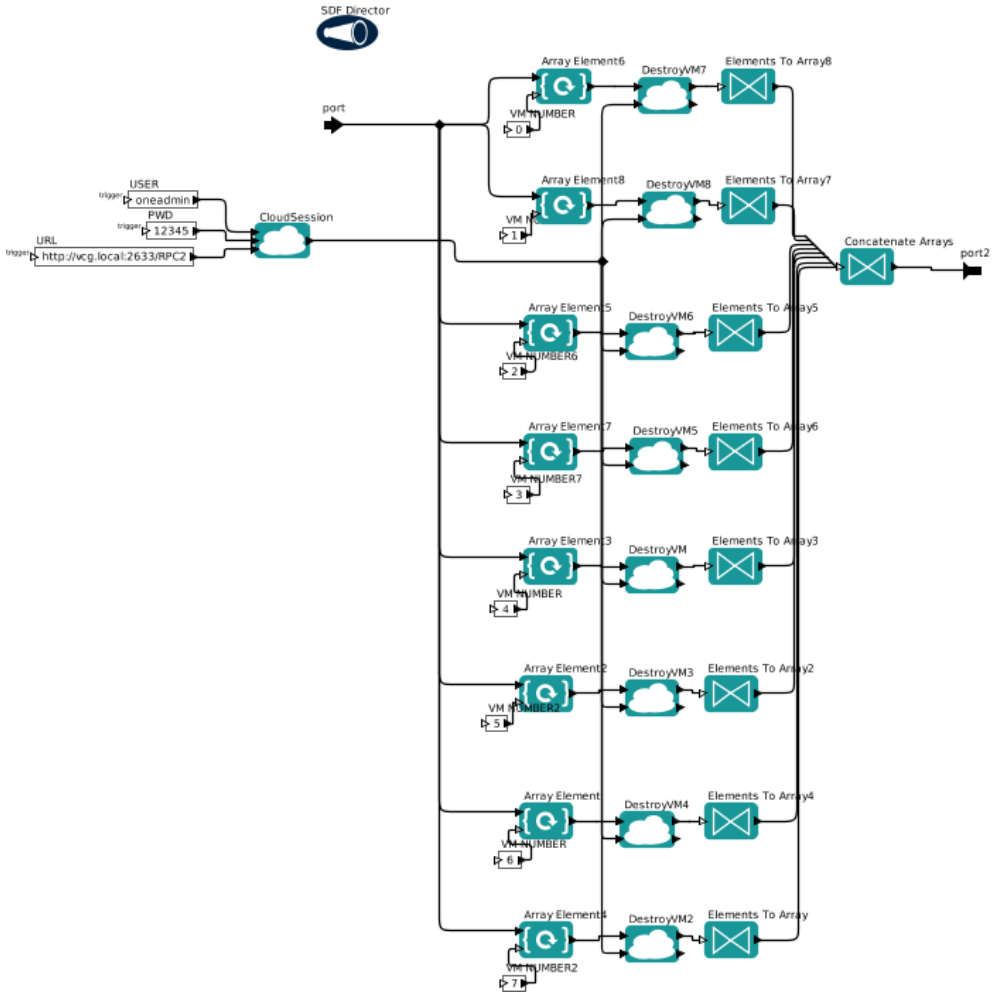


Figura 5.6: Tarefa para liberar 8 nodos ou MVs na nuvem após a execução da aplicação.

Cada tarefa que executa a aplicação de multiplicação de matrizes tem um intervalo de uso dos recursos que diminui com a quantidade de recursos adicionais. A Figura 5.7 mostra o tempo de uso dos recursos por cada tarefa. O tempo para executar com 8 nodos foi de 184 segundos, com 4 nodos foi de 191 segundos e com 2 nodos foi de 251 segundos.

Na "Pré-instanciação"os recursos são todos instanciados antes da execução do *workflow*. O tempo de pré-instanciação pode variar entre instanciar os nodos e executar o *workflow*. Da mesma forma, o tempo para liberar os recursos pode variar entre o término da execução do *workflow* e liberação dos recursos. Aqui assumimos que o intervalo é inexistente pois como o tempo pode ser arbitrário não podemos assumir um valor confiável. Assim, para definir um valor assumimos que o pior tempo é o tempo da pior quantidade de instancias ("*MM_ 2n*"). Assim o tempo total de uso dos recursos foi de 3514 (251*14)

segundos.

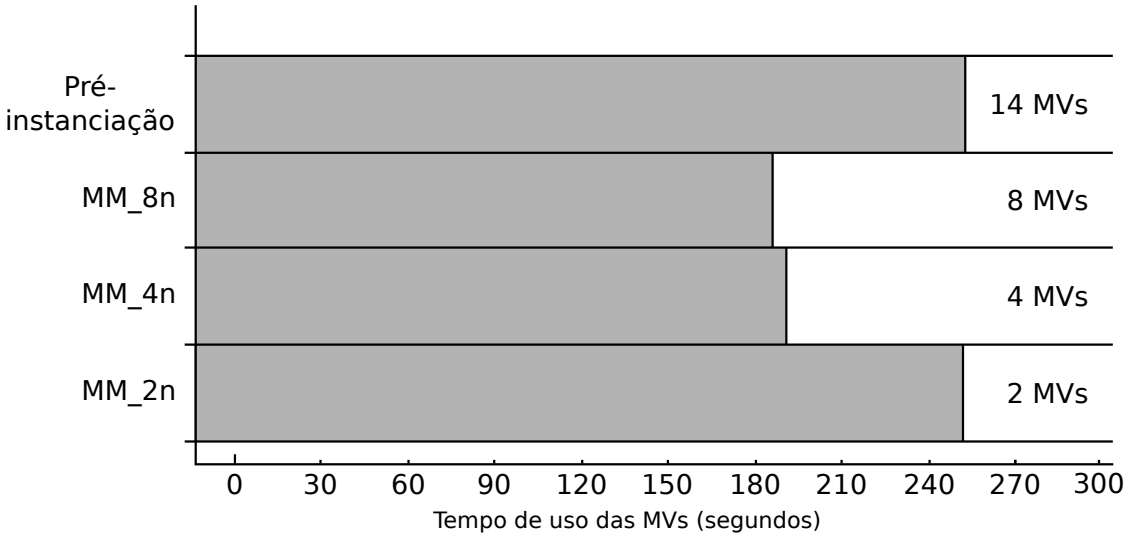


Figura 5.7: Gráfico com o tempo usado para a multiplicação usando 2, 4 e 8 nodos MPI.

O tempo de uso total dos recursos, utilizando a abordagem deste trabalho, foi de 626 segundos, contra os 3514 segundos. De acordo com este resultados é possível observar uma queda de ociosidade dos recursos em aproximadamente 82%. No entanto este é apenas um resultado preliminar com um exemplo simples de multiplicação de matrizes. Como proposta futura, o trabalho será testado em aplicações científicas usadas na comunidade para verificar com maior precisão a melhora oferecida pela proposta.

Neste Capítulo ilustramos um caso de uso da proposta do presente trabalho através do *Kloud* que implementa as TCNs com funcionalidades para instanciar e liberar os recursos de nuvem (MVs) a medida que for necessário para executar as aplicações científicas.

CAPÍTULO 6

CONCLUSÃO

Computação em nuvens está sendo utilizada para disponibilizar recursos virtualizados para execução de experimentos científicos. Dentre os serviços que a nuvem oferece está o serviço de infraestrutura que fornece máquinas virtuais (MV) como recursos virtualizados.

Os *workflows* científicos (WfC) estão sendo adaptandos para utilizar as nuvens computacionais para execução de aplicações científicas. No entanto, a instanciação inadequada de recursos na nuvem geram gastos desnecessários e um uso ineficiente da infraestrutura da nuvem.

O presente trabalho propôs uma modificação na etapa de mapeamento do WfC e uma extensão no WfC abstrato para associar os recursos de nuvem que atendam os requisitos de execução das tarefas abstratas. Com a extensão é possível associar os recursos virtuais que executam determinadas tarefas abstratas de forma a diminuir a ociosidade dos recursos com a instanciação e liberação das MVs nos momentos adequados.

Para validar a proposta do trabalho, uma extensão no Sistema de Gerenciamento de WfC Kepler foi implementada, chamada de *Kloud*. O *Kloud* implementa as tarefas de concretas de nuvens, que instanciam e liberam recursos na nuvem sendo possível combiná-las para obter tarefas de nuvens de alto nível ou abstratas.

Para testar a implementação, foi utilizada uma aplicação de multiplicação de matrizes em MPI com 2, 4 e 8 nodos instanciados na nuvem privada OpenNebula (2.0). Os resultados preliminares mostram que é possível utilizar a abordagem proposta e diminuir o tempo de ociosidade dos recursos em até 82%.

A abordagem de inserir tarefas concretas que explicitam o uso dos recursos virtuais ainda não resolve completamente o problema da ociosidade dos recursos. A inserção de

tarefas de forma inadequada pode gerar ociosidade de recursos da mesma forma que foi apontada no presente trabalho. Assim, é necessário maiores estudos para um algoritmo de mapeamento de forma a inserir as tarefas adequadamente para os mais variados cenários de experimentação científica.

BIBLIOGRAFIA

[1] I. Altintas, O. Barney, e E. Jaeger. *Provenance Collection Support in the Kepler Scientific Workflow System.* Springer Berlin / Heidelberg, 2006.

[2] I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, e S. Mock. Kepler: an Extensible System for Design and Execution of Scientific Workflows. *Proceedings of the 16th International Conference on Scientific and Statistical Database Management*, páginas 423–424. IEEE Computer Society, 2004.

[3] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, e M. Zaharia. Above the Clouds: A Berkeley View of Cloud Computing. Relatório técnico, EECS Department, University of California, Berkeley, 2009.

[4] L. Bavoil, S.P. Callahan, P.J. Crossno, J. Freire, C.E. Scheidegger, C.T. Silva, e H.T. Vo. VisTrails: Enabling Interactive Multiple-View Visualizations. *Proceedings of IEEE Visualization*, páginas 135–142. IEEE Computer Society, 2005.

[5] E. Christensen, F. Curbera, G. Meredith, e S. Weerawarana. Web Services Description Language (WSDL) 1.1, W3C Note, março de 2001.

[6] W. Cirne, D. Paranhos, L. Costa, E. Santos-Neto, F. Brasileiro, J. Sauve, F.A.B. Silva, C.O. Barros, e C. Silveira. Running Bag-of-Tasks applications on computational grids: the MyGrid approach. *Proceedings of International Conference on Parallel Processing*, páginas 407–416. IEEE Computer Society, 2003.

[7] J. Crowcroft, M. Pias, R. Sharma, e S. Lim. A Survey and Comparison of Peer-to-Peer Overlay Network Schemes. *Journal of IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005.

[8] V. Curcin e M. Ghanem. Scientific workflow systems - can one size fit all? *2008 Cairo International Biomedical Engineering Conference*, páginas 1–9. IEEE, dezembro de 2008.

[9] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. H. Su, K. Vahi, e M. Livny. Pegasus: Mapping Scientific Workflows onto the Grid. M. Dikaiakos, editor, *Grid Computing*, volume 0086044, páginas 131–140. Springer Berlin / Heidelberg, 2004.

[10] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G.g Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, e S. Koranda. Mapping Abstract Complex Workflows onto Grid Environments. *Journal of Grid Computing*, 1(1):25–39, 2003.

[11] E. Deelman, D. Gannon, M. Shields, e I. Taylor. Workflows and e-Science: An overview of workflow system features and capabilities. *Journal of Future Generation Computer Systems*, 25(5):528–540, maio de 2009.

[12] E. Deelman, G. Singh, M.H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G.B. Berriman, J. Good, A. Laity, J. C. Jacob, e D. S. Katz. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Journal of Scientific Programming*, 13(3):219–237, 2005.

[13] EGEE e EMI. gLite - Lightweight Middleware for Grid Computing, 2012. Disponível em: <http://glite.web.cern.ch/glite/>. Acessado em: 01/10/2012.

[14] J. Eker, J.W. Janneck, E.a. Lee, J. Ludvig, S. Neuendorffer, e S. Sachs. Taming heterogeneity - the Ptolemy approach. *Journal of Proceedings of the IEEE*, 91(1):127–144, janeiro de 2003.

[15] J. Ellson, E. Gansner, L. Koutsofios, S. North, e G. Woodhull. Graphviz - Open Source Graph Drawing Tools. *Graph Drawing*, volume 2265, páginas 594–597. Springer Berlin / Heidelberg, 2002.

[16] Z. Farkas e P. Kacsuk. P-GRADE Portal: A Generic Workflow System to Support User Communities. *Journal of Future Generation Computer Systems*, 27(5):454–465, maio de 2011.

[17] R. T. Fielding e R. N. Taylor. Principled design of the modern Web architecture. *ACM Transactions on Internet Technology*, 2(2):115–150, maio de 2002.

[18] I. Foster e C. Kesselman. Computational grids. *Vector and Parallel Processing (VECPAR 2000)*, 12:3–37, março de 2001.

[19] I. Foster, J. Vöckler, M. Wilde, e Y. Zhao. Chimera: Avirtual data system for representing, querying, and automating data derivation. *Proceedings of the 14th International Conference on Scientific and Statistical Database Management*, páginas 37–46, 2002.

[20] J. Frey, T. Tannenbaum, M. Livny, I. Foster, e S. Tuecke. Condor-G: A computation management agent for multi-institutional grids. *Journal of Cluster Computing*, 2002.

[21] Y. Gil, E. Deelman, M. Ellisman, T. Fahringer, G. Fox, D. Gannon, C. Goble, M. Livny, L. Moreau, e J. Myers. Examining the Challenges of Scientific Workflows. *Journal of Computer*, 40(12):24–32, dezembro de 2007.

[22] OCCI Working Group. Open Cloud Computing Interface, 2011. Disponível em: <http://occi-wg.org/about/specification/>. Acessado em: 01/10/2012.

[23] S. Hazelhurst. Scientific Computing Using Virtual High-Performance Computing. *Proceedings of the 2008 annual research conference of the South African Institute of Computer Scientists and Information Technologists on IT research in developing countries riding the wave of technology (SAICSIT '08)*, páginas 94–103, New York, NY, USA, 2008. ACM Press.

[24] T. Hey e A. E. Trefethen. The UK e-Science Core Programme and the Grid. *Journal of Future Generation Computer Systems*, 18(8):1017–1031, outubro de 2002.

[25] D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, e T. Oinn. Taverna: a tool for building and running workflows of services. *Journal of Nucleic Acids Research*, 34(Web Server issue):729–732, julho de 2006.

[26] Amazon Inc. Amazon Elastic Compute Cloud (Amazon EC2), 2008. Disponível em: <http://aws.amazon.com/ec2/>. Acessado em 01/10/2012.

[27] Amazon Inc. Amazon Simple Storage Service (Amazon S3), 2008. Disponível em: <http://aws.amazon.com/s3/>. Acessado em 01/10/2012.

[28] G. Juve e E. Deelman. Scientific Workflows and Clouds. *Magazine Crossroads*, 16(3):14–18, março de 2010.

[29] G. Juve, E. Deelman, K. Vahi, G. Mehta, B. Berriman, B. P. Berman, e P. Maechling. Scientific workflow applications on Amazon EC2. *Proceedings of 5th IEEE International Conference on E-Science Workshops*, páginas 59–66. IEEE Computer Society, dezembro de 2009.

[30] K. Keahey, I. Foster, T. Freeman, X. Zhang, e D. Galron. Virtual Workspaces in the Grid. *Euro-Par 2005 Parallel Processing*, páginas 627–627. Springer Berlin / Heidelberg, 2005.

[31] K. Keahey, T. Freeman, J. Lauret, e D. Olson. Virtual Workspaces for Scientific Applications. *Journal of Physics: Conference Series*, 78:012038, julho de 2007.

[32] C. Kesselman e I. Foster. Globus: A Metacomputing Infrastructure Toolkit. *Journal of International Journal of Supercomputer Applications*, 11:115—-128, 1996.

[33] N. Leavitt. Is Cloud Computing Really Ready for Prime Time? *Journal of Computer*, 42(1):15–20, janeiro de 2009.

[34] C. A. Lee. A Perspective on Scientific Cloud Computing. *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing (HPDC '10)*, páginas 451–459, 2010.

[35] K. Lee, N. W. Paton, R. Sakellariou, E. Deelman, A. A. A. Fernandes, e G. Mehta. Adaptive Workflow Processing and Execution in Pegasus. *Proceedings of 3rd International Conference on Grid and Pervasive Computing Workshops*, páginas 99–106, maio de 2008.

[36] M.J. Litzkow, M. Livny, e M.W. Mutka. Condor-a hunter of idle workstations. *Journal of Computing Systems*, 1988.

[37] B. Ludäscher e C. Goble. Guest editors' introduction to the special section on scientific Workflows. *SIGMOD Officers, Committees, and Awardees*, 34(3):3–4, 2005.

[38] S. Majithia, M. Shields, I. Taylor, e I. Wang. Triana: a graphical Web service composition and execution toolkit. *Journal of IEEE International Conference on Web Services*, páginas 514–521, 2004.

[39] T. McPhillips, S. Bowers, D. Zinn, e B. Ludäscher. Scientific workflow design for mere mortals. *Journal of Future Generation Computer Systems*, 25(5):541–551, maio de 2009.

[40] P. Mell e T. Grance. The NIST Definition of Cloud Computing. Relatório técnico, National Institute of Standards and Technology, 2011.

[41] D. Milojič, I. M. Llorente, e R. S. Montero. OpenNebula: A Cloud Management Tool. *Journal of IEEE Internet Computing*, 15(2):11–14, março de 2011.

[42] He. F. Nielsen, N. Mendelsohn, J. J. Moreau, M. Gudgin, e M. Hadley. {SOAP} Version {1.2} Part 1: Messaging Framework. {W3C} recommendation, W3C, junho de 2003.

[43] B. Nitzberg, J. M. Schopf, e J. P. Jones. PBS Pro: Grid Computing and Scheduling Attributes. Jarek Nabrzyski, Jennifer M Schopf, e Jan Weglarz, editors, *Grid resource management*, capítulo PBS Pro: G, páginas 183–190. Kluwer Academic Publishers, Norwell, MA, USA, 2004.

[44] NORDUGRID. ARC: Advanced Resource Connector, 2012. Disponível em: <http://www.nordugrid.org/arc/>. Acessado em: 01/10/2012.

[45] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, e D. Zagorodnov. The Eucalyptus Open-Source Cloud-Computing System. *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID '09)*, páginas 124–131. IEEE Computer Society, maio de 2009.

[46] Web Services Business Process Execution Language (WSBPEL) Technical Committee OASIS. *Web Services Business Process Execution Language ({WS-BPEL}) Version 2.0*. Organization for the Advancement of Structured Information Standards (OASIS), abril de 2007.

[47] T. Oinn, M. Greenwood, M. Addis, M. N. Alpdemir, J. Ferris, K. Glover, C. Goble, A. Goderis, D. Hull, D. Marvin, P. Li, P. Lord, M. R. Pocock, M. Senger, R. Stevens, A. Wipat, e C. Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Journal of Concurrency and Computation: Practice and Experience*, 18(10):1067–1100, agosto de 2006.

[48] D. Oliveira, E. Ogasawara, F. Baião, e M. Mattoso. SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. *2010 IEEE 3rd International Conference on Cloud Computing*, páginas 378–385, julho de 2010.

[49] Openstack.org. OpenStack Cloud Framework, 2012. Disponível em: <http://www.openstack.org/date>. Acessado em: 01/10/2012.

[50] S. Pandey, D. Karunamoorthy, e R. Buyya. Workflow Engine for Clouds. *Cloud Computing: Principles and Paradigms*, capítulo 12, páginas 321–344. John Wiley & Sons, Inc., Hoboken, NJ, USA, 2011.

[51] Ptolemy-Project. Vergil. Disponível em: <http://ptolemy.eecs.berkeley.edu/java/vergil/>. Acessado em: 01/10/2012.

[52] J. Rehr, F. Vila, J. Gardner, L. Svec, e M. Prange. Scientific Computing in the Cloud. *Journal of Computing in Science & Engineering*, 99(PrePrints), 2010.

[53] B. P. Rimal, E. Choi, e I. Lumb. A Taxonomy and Survey of Cloud Computing Systems. *Proceedings of the 5th International Joint Conference on INC, IMS and IDC*, páginas 44–51. IEEE Computer Society, agosto de 2009.

[54] K. S. Shams, M. W. Powell, T. M. Crockett, J. S. Norris, R. Rossi, e T. Soderstrom. Polyphony: A Workflow Orchestration Framework for Cloud Computing. *Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, páginas 606–611. IEEE Computer Society, 2010.

[55] B. Sotomayor, R. S. Montero, I. M. Llorente, e I. Foster. Virtual Infrastructure Management in Private and Hybrid Clouds. *Journal of IEEE Internet Computing*, 13(5):14–22, setembro de 2009.

[56] I. Taylor, M. Shields, I. Wang, e A. Harrison. The Triana Workflow Environment: Architecture and Applications. I. J. Taylor, E. Deelman, D. B. Gannon, e M. Shields, editors, *Book of Workflows for e-Science*, páginas 320–339. Springer London, 2007.

[57] Condor Team. DAGMan: A Directed Aciclyc Graph Manager, 2005. Disponível em: <http://research.cs.wisc.edu/condor/dagman/>. Acessado em: 01/10/2012.

[58] L. M. Vaquero, L. Rodero-Merino, J. Caceres, e M. Lindner. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review*, 39(1):50–55, 2008.

[59] C. Vecchiola, S. Pandey, e R. Buyya. High-Performance Cloud Computing: A View of Scientific Applications. *Proceedings of the 10th International Symposium on Pervasive Systems, Algorithms, and Networks (ISPAN '10)*, páginas 4–16. IEEE Computer Society, dezembro de 2009.

[60] J. Vöckler, G. Juve, E. Deelman, M. Rynge, e B. Berriman. Experiences using cloud computing for a scientific workflow application. *Proceedings of the 2nd international*

workshop on Scientific cloud computing - ScienceCloud '11, páginas 15, New York, New York, USA, 2011. ACM Press.

[61] M. A. Vouk. Cloud computing - Issues, research and implementations. *Proceedings of the 30th International Conference on Information Technology Interfaces (ITI '08)*, páginas 31–40. IEEE Computer Society, junho de 2008.

[62] L. Wang, J. Tao, M. Kunze, A. C. Castellanos, D. Kramer, e W. Karl. Scientific Cloud Computing: Early Definition and Experience. *Proceedings of the 10th IEEE International Conference on High Performance Computing and Communications (HPCC '08)*, páginas 825–830. IEEE Computer Society, setembro de 2008.

[63] Wikipedia.org. Cloud Computing, 2011. Disponível em: <http://en.wikipedia.org/wiki/Cloud_computing. Acessado em: 01/10/2012.

[64] P. Yang e Z. Yang. Formal modeling and analysis of scientific workflows using hierarchical state machines. *e-Science and Grid Computing, IEEE*, páginas 619–626, 2007.

[65] J. Yu e R. Buyya. A Taxonomy of Scientific Workflow Systems for Grid Computing. *Journal of ACM SIGMOD Record*, 34(3):44–49, setembro de 2005.

[66] S. Zhou, J. Wang, X. Zheng, e P. Delisle. Utopia: a Load Sharing Facility for Large, Heterogeneous Distributed Computer Systems. Relatório técnico, Computer Systems Research Institute, University of Toronto, 1993.