Capítulo 1

Introdução à Modelagem e Execução de Workflows Científicos

Kelly Rosa Braghetto e Daniel Cordeiro

Resumo

Os sistemas de gerenciamento de workflows permitem que um cientista descreva um experimento científico como um conjunto de tarefas a serem realizadas. Um cientista define o seu workflow como um modelo usando uma notação de compreensão bastante intuitiva. A partir desse modelo, o sistema de gerenciamento é capaz de executar o experimento de forma automática, com pouca ou nenhuma intervenção do cientista, utilizando, para isso, a infraestrutura computacional disponível. Com isso, os workflows científicos tornaram a automação de experimentos e o uso de plataformas de computação de alto desempenho acessíveis a pesquisadores de diversas áreas do conhecimento. Este texto faz uma introdução a conceitos fundamentais de modelagem e execução de workflows científicos.

Abstract

Workflow management systems allow scientists to describe a scientific experiment as a set of tasks that must be performed by a computer. A scientist can model his/her workflow using an intuitive and easy-to-understand notation. Using the model, a management system is able to execute the experiment in an automated way, with almost no human intervention, on the available computational platform. Scientific workflows made the use of high performance computing platforms available to researchers from different fields of knowledge. In this course, we will introduce the main concepts of the modeling and execution of scientific workflows. We will present the state of the art in scientific workflow management systems and their underlying algorithms.

1.1. Introdução

O modo de se fazer ciência passou por grandes transformações nas últimas décadas. A primeira grande mudança foi o aparecimento de uma vertente *computacional* em pesquisas em diversas áreas. O uso de computação permitiu estudos e simulações de fenômenos complexos, inexequíveis até então.

Mas há uma mudança mais profunda em andamento. Nos últimos anos, observa-se o aparecimento de um quarto paradigma, a exploração de grandes quantidades de dados, muitas vezes chamado de e-science (e-ciência), que unifica teoria, experimentos e simulação, ao mesmo tempo em que lida com uma quantidade enorme de informação [Cordeiro et al., 2013].

Em muitas áreas da ciência, principalmente nas ciências naturais, as novas tecnologias criaram novas possibilidades (ou "tipos") de pesquisa. Criou-se uma nova metodologia de pesquisa em que dados experimentais, coletados por meio de instrumentos ou gerados por simulação, são processados por sistemas de software complexos, e só então a informação (ou o conhecimento) resultante é armazenada em computadores. Os cientistas só analisam os dados no final do processo.

A Ciência da Computação exerce um papel fundamental nesse novo processo científico. Mas para que cientistas de áreas tão distintas possam interagir com a Ciência da Computação, foi necessário a criação de novas formas de expressão. Utilizando uma linguagem compreensível por cientistas de diferentes áreas, é possível descrever um processo experimental de forma programática, como *fluxos de trabalho*, mais conhecidos como *workflows*.

Os *workflows* têm recebido atenção crescente da comunidade de pesquisa desde suas primeiras aparições nos anos 80. Hoje, eles são amplamente encontrados nos domínios tanto científico quanto industrial. Os *workflows científicos* são definidos como "redes de processos tipicamente utilizadas como 'pipelines de análises de dados' ou ainda para comparar dados observados ou previstos, e que podem incluir uma vasta gama de componentes, e.g. para consultar bancos de dados, para transformar ou minerar dados, para executar simulações em computadores de alto desempenho, etc." [Ludäscher et al., 2006]. Portanto, um workflow científico é uma automação de um experimento ou de um processo científico, expressa em termos das atividades a serem executadas e, principalmente, das dependências dos dados manipulados [Yu and Buyya, 2005, Cuevas-Vicenttin et al., 2012].

Workflows científicos tendem a ser computacionalmente intensivos mas, principalmente, são fortemente voltados à manipulação de grandes volumes de dados complexos. A utilização intensiva de dados caracteriza os workflows científicos como workflows baseados em fluxos de dados (*data-driven workflows*), ou seja, as conexões entre as atividades do workflow representam basicamente o fluxo dos dados, com o fluxo de controle aparecendo como uma representação complementar.

Um workflow científico tem seu ciclo de vida composto por quatro fases, ilustradas pela Figura 1.1 [Ludäscher et al., 2009]:

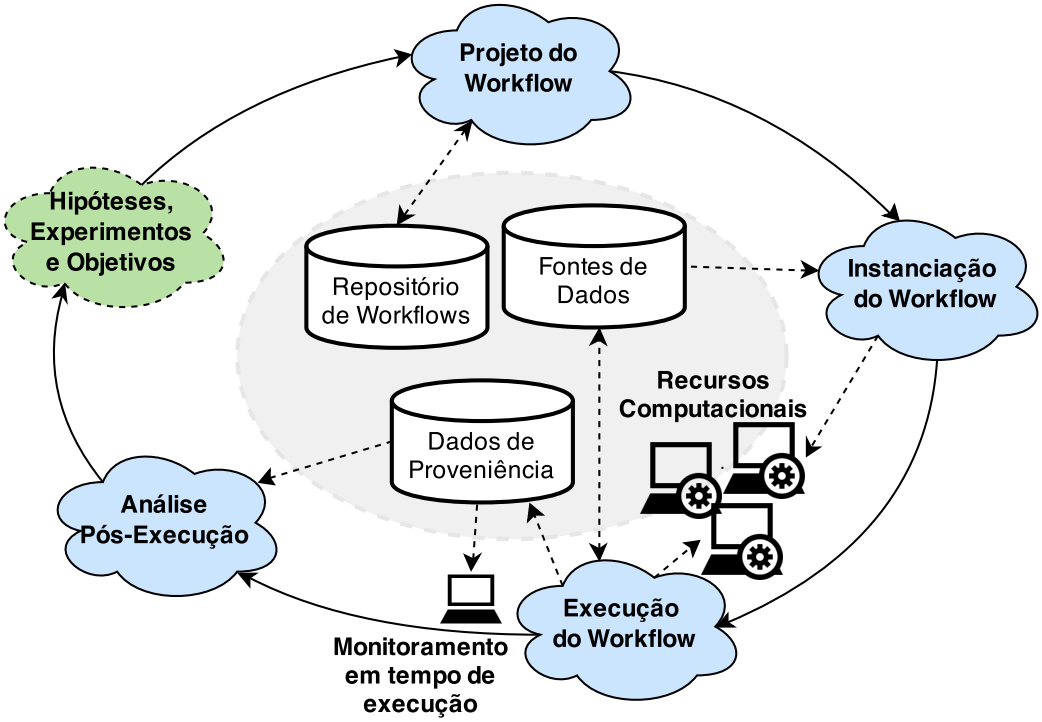


Figura 1.1: Fases do ciclo de vida de um workflow científico (adaptado de [Ludäscher et al., 2009]).

1. *Projeto*: consiste na construção de um modelo para o workflow a partir de uma hipótese científica a ser testada ou de um objetivo experimental específico a ser alcançado. Existem diferentes representações e abordagens que podem ser usadas na modelagem de workflows científicos. Além disso, o projeto de um workflow pode se embasar em outros workflows já existentes. Por essa razão, é conveniente criar representações digitais dos modelos de workflows, que também podem ser armazenadas em repositórios para reuso e até compartilhadas com outros pesquisadores posteriormente.

2. *Instanciação*: corresponde à preparação necessária para uma execução particular do workflow. A cada nova execução do workflow, é necessário indicar quais são os dados de entrada e os parâmetros definidos pelo cientista para a execução. Nessa fase, pode já haver alguma seleção e alocação dos recursos computacionais que serão usados na execução. Em alguns casos, é necessário transferir os dados de entrada para os computadores onde a execução será realizada.

3. *Execução*: corresponde à execução das atividades que compõem o workflow nos recursos computacionais disponíveis. As atividades processam os dados de entrada e produzem novos dados que, por sua vez, podem ser processados por outras atividades. Durante a execução, é desejável que os dados intermediários gerados e as *informações de proveniência* sejam registrados, para que o cientista possa consultar o estado atual e até mesmo identificar possíveis problemas da execução.

   As informações de proveniência geralmente incluem, além dos dados de entrada e

os parâmetros definidos pelo cientista na instanciação do workflow, o registro de todas as atividades já executadas na instância, seus respectivos tempos de início e de término, os recursos empregados em sua execução, as referências para os dados usados como entrada e saída de cada atividade, etc. Essas informações possibilitam que um dado experimento seja reproduzido tal como ele foi realizado da primeira vez. Elas possibilitam também que, em caso de falha na execução do workflow, a execução possa ser retomada do ponto onde parou.

4. *Análise Pós-Execução*: corresponde à inspeção e interpretação dos resultados obtidos a partir da execução do workflow. Por exemplo, a análise dos dados de saída gerados pode levar à rejeição da hipótese científica. E a análise de informações de proveniência, por exemplo, permite identificar as atividade que são gargalos na execução do workflow.

   Os resultados obtidos a partir da análise podem conduzir a uma revisão da hipótese ou do objetivo experimental inicial e, portanto, a um reprojeto do workflow (ou seja, uma nova iteração do seu ciclo de vida).

As próximas seções deste texto fazem uma introdução dos conceitos fundamentais relacionados à modelagem (projeto) e execução de workflows científicos – fases que constituem o núcleo do ciclo de vida desses workflows. A Seção 1.2 descreve as estruturas geralmente empregadas na representação de workflows científicos, considerando também diferentes perspectivas e objetivos de modelagem. A Seção 1.3 discute algoritmos de escalonamento que são usados para definir a forma por meio da qual um determinado ambiente de computação de alto desempenho pode ser usado para executar aplicações científicas eficientemente. A Seção 1.4 exemplifica como a modelagem e o escalonamento de workflows são apoiados na prática por sistemas de gerenciamentos de workflows científicos. Na Seção 1.5, ambientes computacionais comumente usados na execução de workflows científicos são brevemente caracterizados. Finalmente, a Seção 1.6 destaca desafios atuais relacionados à pesquisa na área de workflows científicos.

1.2. Modelagem de Workflows Científicos

Como introduzido na Seção 1.1, um workflow científico é composto principalmente por atividades de processamento de dados e as conexões existentes entre essas atividades. Nesta seção, serão apresentadas as estruturas comumente empregadas na representação abstrata dos workflows científicos.

A terminologia básica usada na representação de workflows científicos é a seguinte:

- *Modelo de workflow* — define as atividades que precisam ser realizadas no workflow para que um determinado objetivo científico seja alcançado. Uma função importante do modelo é estabelecer a ordem (ainda que parcial) na qual as atividades do workflow devem ser executadas.

  Um modelo de workflow é comumente representado por meio de um arcabouço formal ou de uma linguagem de especificação de workflows. Enquanto estas últimas

permitem, de modo geral, a especificação de processos em um nível de detalhamento mais próximo ao requerido para a execução do processo, o uso de arcabouços formais possibilitam a análise de propriedades dos processos que neles são modelados.

- *Atividade* — é uma unidade atômica (ou seja, indivisível) de trabalho em um modelo de workflow. Na literatura sobre workflows, é possível encontrar vários sinônimos para o termo *atividade*. Exemplos: *tarefa*, *processo*, *ação* e *transição*.

  Uma atividade em um workflow pode ser classificada como *manual*, *automática* ou *semi-automática*. Uma *atividade manual* é realizada por pessoas, enquanto que uma *atividade automática* é realizada completamente por computadores ou outros tipos de máquinas, sem intervenção humana. Já uma atividade *semi-automática* depende tanto de pessoas quanto de máquinas para ser realizada.

- *Conector* — é usado para definir alguma relação de precedência ou algum tipo de transferência de dados entre atividades em um modelo de workflow.

  Toda linguagem de modelagem de workflows possui o seu próprio conjunto de conectores para a composição de atividades. São esses conectores que definem a expressividade da linguagem, ou seja, a variedade e quantidade de modelos de workflows que a linguagem é capaz de representar.

- *Instância de workflow* — é uma execução particular de um dado workflow ou, em outras palavras, é uma instanciação de um modelo de workflow.

  Duas instâncias de um mesmo workflow não necessariamente executarão a mesma sequência de atividades, já que um workflow pode ter em seu modelo ramos de execução alternativos ou atividades que só serão realizadas caso determinados tipos de dados estejam disponíveis.

1.2.1. Workflows de Negócio × Workflows Científicos

Desde os anos 90, os sistemas de workflow vêm sendo usados em grande escala para automatizar processos industriais e gerenciais, os chamados *processos de negócio*. Os workflows aplicados a esse domínio são comumente designados como *workflows de negócio*.

Muitos modelos, técnicas e ferramentas computacionais já foram desenvolvidos para apoiar o ciclo de vida de workflows de negócio. Comunidades e coalizões, envolvendo indústria e academia, foram formadas com o objetivo de criar documentos, modelos de referência e padrões relacionados à gestão de workflows de negócio.

Já o uso de workflows em domínios científicos só se popularizou nos anos 2000. E, apesar das muitas relações de similaridade que podem ser estabelecidas entre os workflows científicos e os de negócio, a comunidade científica não adotou os padrões desenvolvidos para a área de workflows de negócio. Isso ocorreu porque o domínio científico possui particularidades que justificam a criação de técnicas e ferramentas específicas para o gerenciamento de seus workflows [Barga and Gannon, 2007].

1.2.1.1. Padrões de Representação e Interoperabilidade entre Ferramentas

Como processos de negócio com frequência envolvem processos inter-organizacionais, muitos esforços já foram feitos para padronizar as linguagens de modelagem de workflows de negócio, visando a interoperabilidade das ferramentas desenvolvidas para a área de gestão de processos de negócio. Exemplos bem sucedidos de iniciativas de padronização nesse domínio são o modelo de referência da *Workflow Management Coalition* [WfMC, 1999] e a *Business Process Model and Notation* (BPMN) [BPMN, 2013], uma notação padrão para a representação gráfica de processos de negócio criada pela *Business Process Management Initiative* (BPMI) em 2004 e hoje mantida pelo *Object Management Group* (OMG).

Não existem representações padrão para os modelos de workflows científicos. De forma geral, cada sistema de gerenciamento possui a sua própria linguagem de modelagem, sem a preocupação de garantir interoperabilidade entre diferentes ferramentas. Isso ocorre porque ainda são poucos os domínios científicos que fazem uso sistematizado de ferramentas de apoio à gestão de workflows científicos. Entre os domínios que se destacam por fazer uso dessa tecnologia estão a Bioinformática e a Astronomia.

Os modelos de workflows de negócio geralmente são desenvolvidos por especialistas em TI. As atividades desses workflows são implementadas como componentes em formatos padronizados e portáveis (como os serviços Web), que facilitam o reúso e a construção dos processos inter-organizacionais.

Laboratórios de pesquisa nem sempre têm à sua disposição especialistas em TI para especificar os seus workflows científicos e implementar as atividades de processamento de dados seguindo as boas práticas de programação. Com frequência, os próprios cientistas implementam as atividades de processamento de dados como *scripts* contendo comandos para os softwares matemáticos ou estatísticos (como o *Matlab* ou o *R*) que possuem em seus laboratórios. E, nesses casos, os workflows científicos são definidos apenas como *scripts shell* que coordenam a execução de outros *scripts*. É por essa razão que a proposição de padrões relacionados à gestão de workflows para o domínio científico é uma atividade mais complicada do que foi para o domínio de negócios.

1.2.1.2. Natureza das Atividades

Os workflows de negócio comumente possuem atividades automáticas e manuais. Por exemplo, um workflow para o tratamento de reclamações de clientes de uma empresa de TV a cabo pode envolver atividades manuais relacionadas à inspeção (por eletrotécnicos) da integridade física de cabeamento de uma dada localidade e atividades automáticas como as de notificação de reestabelecimento do serviço.

Já os workflows científicos não costumam envolver atividades manuais. Suas ferramentas de apoio não oferecem suporte a esse tipo de funcionalidade. Assim, os cientistas usam workflows científicos para controlar apenas as etapas automatizáveis de seus experimentos.

1.2.1.3. Integridade da Execução do Workflow

Uma preocupação crucial em workflows de negócio é a integridade das sequência de atividades realizadas por eles. Workflows de negócio geralmente representam transações comerciais que podem envolver contratos financeiros e prestação de serviços. Por essa razão, workflows de negócio geralmente são workflows transacionais — em caso de falhas durante sua execução, é preciso garantir maneiras de se desfazer as atividades já realizadas (mesmo que as atividades só possam ser desfeitas logicamente, por meio de atividades compensatórias). Por exemplo, uma falha na entrega de um produto adquirido por um cliente em uma livraria online segundo um workflow de compra pode ser compensada por meio da devolução ao cliente do valor pago pelo livro ou ainda por meio da atribuição de créditos para a compra de outros produtos na mesma livraria.

Em workflows científicos, essa preocupação é menos frequente [Barga and Gannon, 2007]. Workflows científicos que têm como alvo o processamento e a análise de dados geralmente não demandam tratamento transacional. No caso da ocorrência de uma falha durante a execução, de forma geral, remove-se os resultados intermediários produzidos pela atividade onde a falha ocorreu e então reinicia-se a execução do workflow a partir do ponto onde ela foi interrompida ou do seu início.

Mesmo quando a reexecução de uma instância de workflow científico que apresentou falha não envolve complicações técnicas (como a necessidade de compensação de atividades realizadas antes da falha), é importante enfatizar que essa reexecução pode ter um custo alto em termos de tempo e de recursos computacionais consumidos. O tempo de execução de instâncias de workflows que processam grandes volumes de dados pode variar de algumas horas até vários dias.

1.2.1.4. Fluxo de Controle × Fluxo de Dados

Um workflow pode ser modelado sob diferentes perspectivas, sendo que as mais relevantes são:

- *Fluxo de controle* — descreve a ordem (parcial) de execução das atividades pertencentes ao workflow por meio de diferentes construtores de composição. O fluxo de controle define a transferência de controle entre atividades conectadas, habilitando uma atividade para execução quando as atividades que a precedem são concluídas.
- *Fluxo de dados* — descreve a forma como os dados são transferidos entre as atividades do workflow, ou seja, estabelece as dependências das atividades com relação aos dados que elas manipulam, com atividades produzindo dados que serão consumidos por outras atividades.
- *Organizacional* (também chamada de *recursos*) — atrela ao workflow uma estrutura organizacional, por meio da definição de papéis (desempenhados por pessoas ou equipamentos) responsáveis pela execução das atividades.
- *Tratamento de exceções* — lida com as causas das exceções e as ações que precisam ser tomadas nos seus tratamentos.

Grande parte dos trabalhos teóricos e ferramentas desenvolvidos para a modelagem de workflows de negócio focam a perspectiva de fluxo de controle. A prevalência dessa perspectiva nesse domínio é compreensível, dado que o fluxo de controle fornece uma visão sobre a especificação de um processo de negócio que é essencial para a avaliação de sua efetividade. As demais perspectivas assumem um papel secundário, uma vez que elas oferecem uma visão complementar da estrutura do processo.

Entretanto, as aplicações científicas são, por natureza, intensivas em dados. Nesse tipo de aplicação, a geração de dados derivados a partir de dados brutos é o objetivo principal. Por essa razão, no domínio dos workflows científicos a predominância é da modelagem de fluxos de dados. Além disso, essa perspectiva de modelagem é mais intuitiva para cientistas, que rotineiramente já descrevem seus experimentos científicos em termos dos dados brutos coletados neles e dos processos de filtragem, transformação e análise aos quais esses dados são submetidos para a obtenção dos dados derivados que sustentarão suas hipóteses científicas.

De qualquer forma, é importante enfatizar que as quatro perspectivas de modelagem listadas acima não são excludentes. Com frequência, ferramentas desenvolvidas para a gestão de workflows de negócio combinam as perspectivas de fluxo de controle e de tratamento de exceções (visando garantir integridade à execução dos workflows, como discutido na Seção 1.2.1.3). E muitos sistemas de gerenciamento de worflows científicos possuem linguagens de modelagem híbridas, que apoiam a descrição de fluxos tanto de dados quanto de controle.

1.2.2. Estruturas para Representação de Fluxos de Controle

A Figura 1.2 ilustra um workflow modelado sob a perspectiva de fluxo de controle. Em linguagens de modelagem gráficas, as atividades geralmente são representadas como retângulos e as relações de precedência são estabelecidas por meio de setas. No modelo da figura, os conectores foram representados por elipses. Cada elipse aparece rotulada com a função que o conector desempenha no fluxo de controle.

O modelo mostrado na Figura 1.2 exemplifica o uso de seis construções de fluxo de controle que são consideradas essenciais para a modelagem de workflows [van der Aalst et al., 2003, WfMC, 1999]:

- *Sequência*: ocorre quando uma atividade só pode ser executada após o término de uma outra atividade do workflow.

  Por exemplo, no workflow da Figura 1.2 a atividade `B` só estará pronta para execução após o término da execução de `A`.

- *Paralelismo (Divisão-E)*: um ponto onde uma linha do fluxo do workflow se divide em duas ou mais linhas que podem ser executadas em paralelo.

  Por exemplo, no workflow da Figura 1.2, após o término da execução da atividade `B`, a linha de fluxo do workflow é dividida em três linhas paralelas.

- *Sincronização (Junção-E)*: um ponto onde duas ou mais linhas de fluxo paralelas são sincronizadas, convergindo em uma única linha de fluxo.

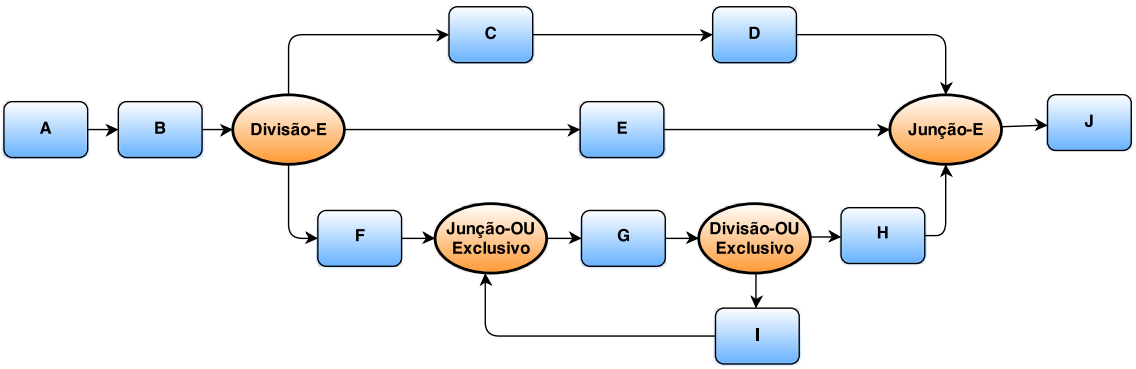


Figura 1.2: Exemplo de workflow modelado usando as construções básicas de fluxo de controle.

Por exemplo, no workflow da Figura 1.2 a atividade `J` somente estará pronta para execução após o término das três linhas de fluxo que são a entrada do conector de `Junção-E` que precede `J` no modelo.

- *Escolha Exclusiva (Divisão-OU-Exclusivo)*: um ponto onde uma linha de fluxo é escolhida para execução dentre duas ou mais linhas possíveis. Essa escolha baseia-se em uma decisão ou em um dado de controle do workflow.

  Por exemplo, no workflow da Figura 1.2, após a execução da atividade `G`, apenas uma das atividades `H` ou `I` será escolhida para execução.

- *Junção (Junção-OU-Exclusivo)*: um ponto onde dois ou mais fluxos alternativos convergem em um fluxo único, sem sincronização.

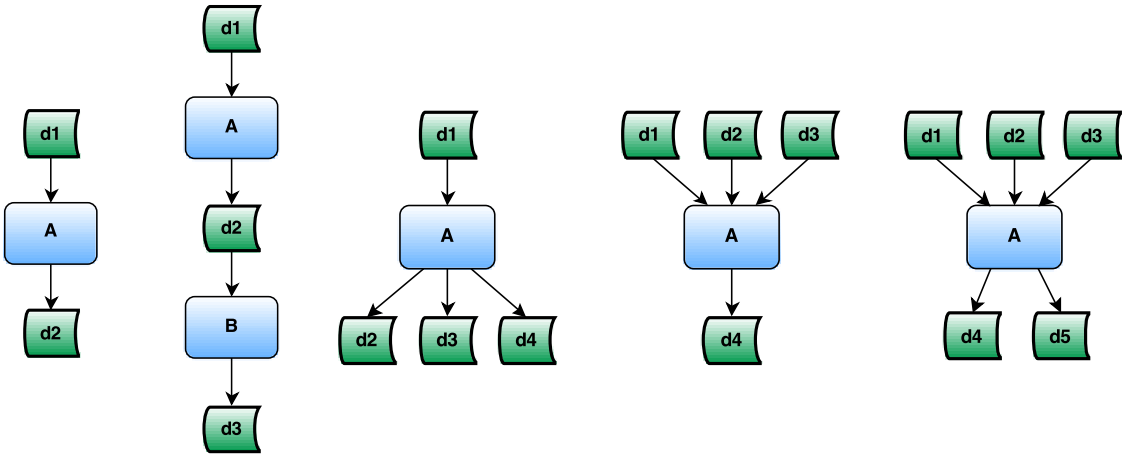
  Por exemplo, no workflow da Figura 1.2, após o término da execução de uma das atividades `F` ou `I`, a atividade `G` estará pronta para a execução.

- *Ciclo* (ou *Iteração*): execução repetida de uma ou mais atividades do workflow.

  Por exemplo, no workflow da Figura 1.2, as atividades *G* e *I* podem ser executadas repetidas vezes.

Embora seja possível definir uma classe vasta de modelos de workflows usando apenas as construções básicas de fluxo de controle, algumas linguagens de modelagem possuem outras estruturas mais complexas de ramificação e junção de fluxos. Pesquisadores da *Eindhoven University of Technology* e da *Queensland University of Technology*, em 1999, criaram uma iniciativa conjunta — a *Workflow Patterns* [1] — para o estudo sistemático das linguagens de modelagem usadas pelas ferramentas de gestão de workflows existentes na época. A iniciativa identificou um vasto conjunto de construções frequentemente encontradas nas ferramentas e, a partir delas, definiu um conjunto de padrões empregados na modelagem de workflows. Foram definidos padrões para todas as perspectivas de modelagem listadas na Seção 1.2.1.4. A análise mais recente sobre construções para fluxo de controle publicada pela *Workflow Patterns* [Russell et al., 2006] contabilizou um total de 43 padrões diferentes.

[1] *Workflow Patterns Initiative*: `http://www.workflowpatterns.com/`



(a) (b) (c) (d) (e)

Figura 1.3: Construções básicas para a modelagem de fluxos de dados: (a) Processamento; (b) *Pipeline*; (c) Distribuição de dados; (d) Agregação de dados; (e) Redistribuição de dados.

Sob a perspectiva do fluxo de controle, workflows são um tipo especial de sistemas concorrentes. Eles podem ser compostos por um grande número de atividades e o relacionamento de precedência existente entre elas define uma ordem parcial que deve ser respeitada nas execuções das instâncias dos workflows. Uma mesma atividade pode aparecer em diferentes lugares de um mesmo modelo. Atividades podem ser repetitivas; um modelo de workflow pode conter ciclos estruturados ou não-estruturados.

A maioria das linguagens de modelagem que priorizam a perspectiva de fluxo de controle permitem que workflows sejam modelados de forma estrutural, como a interação de subworkflows mais simples. Linguagens que possuem essa importante característica são chamadas de *composicionais*.

1.2.3. Estruturas para a Representação de Fluxos de Dados

Sob a perspectiva de fluxo de dados, as seguintes construções (ilustradas na Figura 1.3) são as consideradas básicas para a modelagem de wokflows científicos [Bharathi et al., 2008]:

- *Processamento*: é a construção que representa o processamento realizado por uma atividade sobre dados de entrada para produzir dados de saída. É a construção mais simples que um fluxo de dados pode conter.
- *Pipeline*: combina um ou mais processamentos sequencialmente, de forma que cada atividade do *pipeline* processe os dados produzidos pela atividade que a precedeu e que a sua saída seja usada como entrada para a próxima atividade do *pipeline*.
- *Distribuição de dados*: é feita por uma atividade que produz dois ou mais conjuntos de dados de saída que são recebidos como entrada por duas ou mais atividades. A distribuição de dados pode ser feita para dois propósitos distintos: (i) para produzir

novos conjuntos de dados para serem consumidos por múltiplas atividades; ou (ii) para particionar um grande conjunto de dados recebido como entrada em subconjuntos menores, para que esses possam ser processados por outras atividades no workflow. A distribuição de dados usada para esse segundo propósito é também chamada simplesmente de *particionamento*.

É importante observar que uma distribuição de dados pode resultar na criação de linhas de execução paralelas dentro de uma instância do workflow.

- *Agregação de dados*: é feita por uma atividade que agrega e processa dados de saída de duas ou mais atividades, gerando uma combinação dos dados como saída.

  Uma agregação de dados pode resultar em uma sincronização de linhas de execução paralelas dentro de uma instância do workflow.

- *Redistribuição de dados*: é feita por uma atividade que combina a função de agregação de dados à função de distribuição dos dados. Como a atividade desse tipo recebe como entrada múltiplos conjuntos de dados e produz como saída também múltiplos conjuntos de dados, a atividade pode tanto sincronizar linhas de execução paralelas já existentes quanto criar novas linhas de execução paralelas.

Na Figura 1.3, uma atividade é expressa graficamente como um retângulo, ligado por meio de setas aos seus conjuntos de dados de entrada e de saída. A Figura 1.4 mostra um exemplo de workflow — o *Montage* — modelado sob a perspectiva de fluxo de dados, usando as construções listadas acima. A Seção 1.2.3.1 explica o que esse workflow faz e como ele está estruturado.

A *Workflow Patterns Initiative* também identificou padrões empregados na representação de fluxos de dados [Russell et al., 2005]. Entretanto, esses padrões se diferem bastante das construções listadas acima. A razão disso é que os padrões foram definidos a partir da análise de ferramentas para workflows de negócio, e as manipulações de dados realizadas nesses workflows, em geral, aparecem no modelo apenas como um aspecto complementar à perspectiva principal, que é a de fluxo de controle. Portanto, as construções listadas acima se aproximam melhor daquilo que é oferecido pelas linguagens para a modelagem de workflows puramente baseadas em fluxos de dados, ou pelas linguagens de modelagem associadas aos sistemas de gerenciamento de workflows científicos, que muitas vezes são híbridas: possibilitam igualmente a modelagem de fluxos de dados e fluxos de controle.

É importante ressaltar que, mesmo em um workflow cujo modelo é descrito puramente por meio de fluxo de controle, pode haver (e geralmente há) manipulações de dados. Em workflows desse tipo, supõe-se que cada atividade sabe quais são os dados necessários para sua execução e sabe também como obtê-los. Os dados usados ou criados pelo workflow podem ficar armazenados em um repositório (por exemplo, um banco de dados) acessível às atividades pertencentes ao workflow.

1.2.3.1. O Workflow *Montage*

O workflow *Montage* [2] é uma ferramenta computacional de código aberto desenvolvida pelo IRSA (*NASA/IPAC Infrared Science Archive*) e usada na área de Astronomia para criar mosaicos personalizados do céu a partir de imagens no formato FITS (*Flexible Image Transport System*). Ele permite criar imagens de regiões do céu que são grandes demais para poderem ser produzidas por câmeras astronômicas. Além disso, ele também permite criar imagens compostas a partir de imagens com diferentes configurações ou coletadas por meio de diferentes equipamentos.

Para gerar o mosaico, o *Montage* calcula a geometria da imagem de saída a partir das imagens de entrada. Depois disso, ele transforma as imagens de entrada, para garantir que elas estejam na mesma escala e rotação espacial. Na sequência, os planos de fundo das imagens são normalizados e então as imagens são compostas para formar a imagem final do mosaico.

O *Montage* foi modelado como um workflow científico com uma estrutura que facilita a sua execução em grades computacionais. A Figura 1.4 mostra uma representação para o *Montage* proposta por Bharathi *et al.* [Bharathi et al., 2008]. Essa representação baseia-se em fluxo de dados e possui relativamente poucas atividades.

A estrutura do modelo do *Montage* precisa variar de acordo com a quantidade de imagens fornecidas como entrada para a composição do mosaico. O modelo da Figura 1.4 considera que o mosaico é formado a partir de apenas quatro imagens de entrada. Caso mais imagens sejam fornecidas como entrada, então é preciso incluir mais atividades de processamento no modelo. A quantidade de atividades `mProjectPP` (que transformam as imagens de entrada) deve ser igual ao número de imagens de entrada a serem processadas. A saída de cada uma dessas atividades é a imagem transformada, mais uma imagem contendo a fração da imagem transformada que irá compor a imagem final do mosaico. Essas saídas são então processadas conjuntamente nos passos seguintes. A atividade `mDiffFit` calcula uma diferença entre cada par de imagens que se sobrepõem. Na sequência, as imagens de diferença são ajustadas pela atividade de agregação `mConcatFit`, que implementa um algoritmo de mínimos quadrados. Depois disso, a atividade `mBgModel` determina qual correção deve ser aplicada a cada imagem para que o melhor ajuste global seja obtido. A correção é aplicada individualmente em cada imagem pelas atividades `mBackground`. Assim, a atividade `mBgModel` atua como uma particionadora de dados. A atividade `mImgTbl` agrega os metadados de todas as figuras e cria uma tabela que é usada pela atividade `mAdd` para unir as imagens transformadas em uma única imagem no formato FITS, que já é a imagem do mosaico. Por fim, o tamanho da imagem do mosaico é reduzido pela atividade `mShrink`, por meio do cálculo de médias de blocos de pixels.

O *Montage* é frequentemente usado como base de comparação na avaliação de técnicas e ferramentas de modelagem e execução de workflows científicos. Vários trabalhos de pesquisa o utilizam como estudo de caso na avaliação de desempenho de sistemas de gerenciamento de workflows ou na avaliação da expressividade de linguagens de modelagem.

[2] Projeto *Montage: An astronomical image engine*: `http://montage.ipac.caltech.edu`.

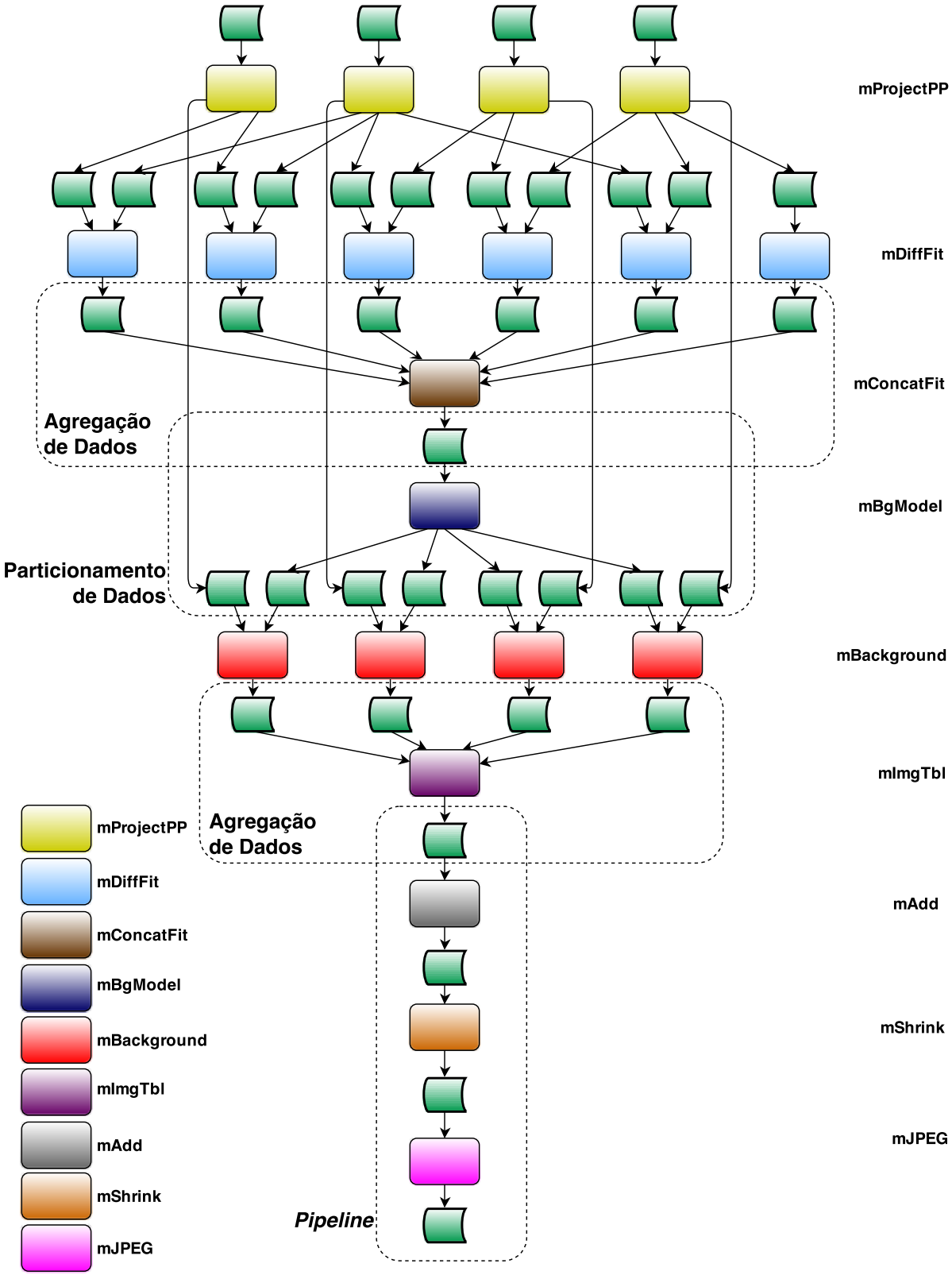


Figura 1.4: Workflow *Montage*, modelado sob perspectiva de fluxo de dados (adaptado de [Bharathi et al., 2008]).

1.2.4. Modelagem Formal de Workflows

As duas classes de formalismos mais apropriadas para a representação de workflows são as *Álgebras de Processos* e as *Redes de Petri*, sendo que essas últimas são as mais empregadas devido à sua notação gráfica que é de interpretação intuitiva.

Nas *Redes de Petri* (RdP), o comportamento dos sistemas são descritos por meio de grafos bipartidos, em que os nós são *lugares* (que modelam os elementos estáticos do processo, como um estado, dados ou condições) ou *transições* (que modelam os elementos dinâmicos do sistema, ou seja, as atividades). Um arco no grafo liga um lugar de entrada a uma transição — associando uma atividade a uma *pré-condição* de execução — ou liga uma transição a um lugar de saída — associando uma atividade a uma *pós-condição*. Os lugares possuem fichas e o arcos possuem pesos. O estado do sistema em um dado momento é definido pela *marcação da rede*, ou seja, o número de fichas existentes em cada um dos lugares da rede.

A Figura 1.5 mostra um exemplo de workflow – o SIPHT – descrito em RdP. Em uma rede de Petri, os lugares são graficamente representados como círculos, enquanto que as transições são expressas como retângulos ou quadrados. As fichas são expressas como pequenos círculos preenchidos ou números dentro dos lugares. Os detalhes sobre o workflow SIPHT são descritos na Seção 1.2.4.1.

A semântica operacional das RdP é definida por meio da regra de *disparo* das transições. Uma transição só está habilitada para o disparo quando a quantidade de fichas que há em cada um dos seus lugares de entrada é superior ao peso do arco que a conecta ao lugar. O disparo de uma transição consome fichas dos seus lugares de entrada e produz fichas em seus lugares de saída; a quantidade de fichas consumidas ou produzidas em cada lugar é definida pelo peso do arco que liga a transição ao lugar (ou vice-versa). A evolução da marcação da rede de acordo com os disparos das transições descreve o comportamento dinâmico do processo modelado.

Cada uma das marcações que podem ser obtidas a partir da marcação inicial e de uma sequência de disparos de transições de uma RdP representa um possível estado do sistema modelado. A partir desses estados e da estrutura da rede, é possível verificar propriedades do sistema. Por exemplo, se existe em um modelo de RdP uma transição que não é habilitada por nenhuma das possíveis marcações da rede, então essa transição é *morta* (ou seja, ela nunca será disparada) e, portanto, isso pode indicar a existência de um erro no projeto do sistema. Também pode-se querer saber se um dado sistema pode atingir um determinado estado indesejado. Nesse caso, pode-se identificar qual é a marcação da RdP que representa esse estado e caso essa marcação esteja entre as que podem ser obtidas a partir da marcação inicial da rede, então estará confirmado que o estado indesejado pode ocorrer no sistema.

Modelos formais de workflows podem ser analisados sob duas perspectivas distintas: a qualitativa e a quantitativa. Podemos citar como exemplos de análise qualitativa a *validação* e a *verificação*. A primeira avalia se um sistema modelado se comporta como o esperado (correção semântica), enquanto a segunda avalia se um sistema modelado respeita alguns critérios estruturais (correção sintática). Na análise qualitativa de workflows é possível, por exemplo, verificar se dois modelos de workflows são equivalentes, ou ainda

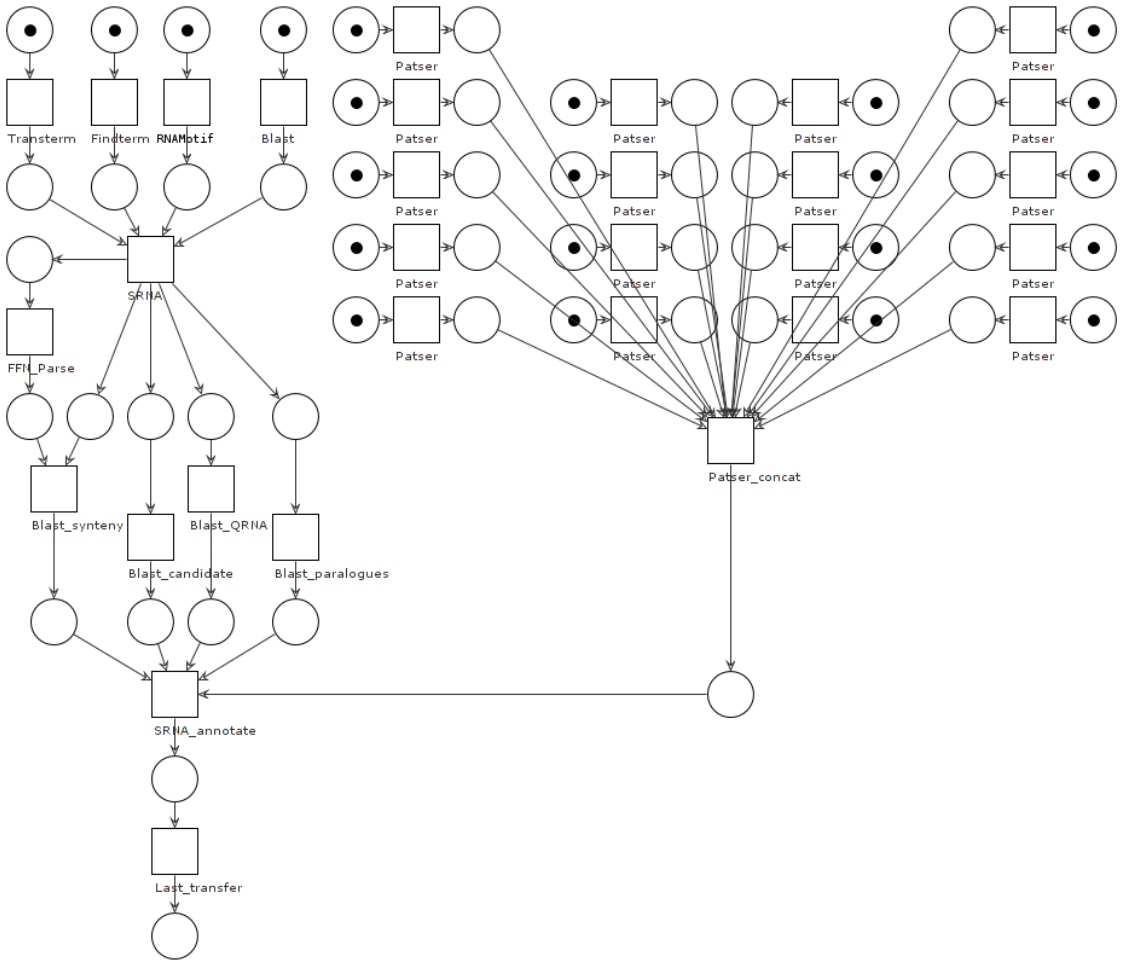


Figura 1.5: Exemplo de workflow modelado como uma Rede de Petri.

se um dado workflow possui pontos de impasse (*deadlocks*).

A *análise de desempenho* é o tipo mais empregado de avaliação quantitativa de sistemas. A partir de modelos estocásticos (como os baseados em cadeias de Markov), é possível realizar análises quantitativas preditivas sobre o desempenho de um workflow, como, por exemplo, obter o tempo de resposta esperado para as atividades e para o workflow como um todo, obter a probabilidade de execução de uma dada sequência de atividades do workflow ou ainda identificar possíveis gargalos na execução do workflow. Tanto as Álgebras de Processos quanto as Redes de Petri possuem extensões estocásticas que podem ser usadas para a modelagem de workflows visando análises quantitativas.

1.2.4.1. O Workflow SIPHT

O workflow representado na Figura 1.5, assim como o *Montage*, é um workflow real que é usado como base de comparação de ferramentas e linguagens de modelagem de workflows científicos [Bharathi et al., 2008].

O SIPHT (*sRNA identification protocol using high-throughput technologies*) é um projeto de bioinformática conduzido pela universidade de Harvard com o objetivo de buscar por *pequenos ARN (ácidos ribonucleicos) não codificados* (*sRNA*) que regulam processos de secreção ou virulência em bactérias.

O protocolo de identificação de sRNA usa workflows para automatizar a busca de genes que codificam um sRNA para todos os replicons bacterianos cadastrados no banco de dados do *National Center for Biotechnology Information* (NCBI) americano.

Todos os workflows SIPHT possuem uma estrutura quase idêntica. Workflows maiores podem ser criados pela composição de workflows independentes menores. A única diferença na estrutura de quaisquer duas instâncias está no número de atividades *Patser*, que depende da quantidade de entradas. Os resultados dessas atividades *Patser* são concatenados pela atividade *Patser_concat*, que é uma atividade de agregação de dados. O workflow possui várias atividades *BLAST* (*Basic Local Alignment Search Tool*) que comparam diferentes combinações de sequências. A atividade *BLAST* inicial (aquela que não depende de nenhuma outra atividade) e a atividade *Blast_QRNA* atuam em todas as combinações de regiões inter-gênicas do genoma (IGRs) e de possíveis replicons. Mesmo que não apareçam explicitamente na Figura 1.5, essas atividades *BLAST* atuam em centenas de arquivos de dados e podem, portanto, serem consideradas como atividades de agregação. Há três atividades no workflow que procuram por terminações das unidades de transcrição: *Find-Term*, *RNAMotif* e *Transterm*. A predição de sRNA é realizada pela atividade *SRNA*, que analisa as saídas das atividade precedentes e a saída da atividade *BLAST* inicial. A saída produzida por essa atividade é utilizada por todas as outras atividade *BLAST*. Portanto, a atividade SRNA pode ser considerada como uma atividade de redistribuição de dados. A atividade *SRNA_Annotate* anota informações sobre a bactéria nos locais onde os sRNA candidatos foram encontrados. Portanto, essa atividade pode ser considerada uma atividade de agregação.

1.3. Escalonamento de Workflows Científicos

Cientistas desenvolvem workflows científicos construindo modelos abstratos, nos quais atividades e dados são descritos sem referência direta ao seu instanciamento físico nos recursos computacionais disponíveis. Essa representação abstrata simplifica o processo de construção dos workflows e permite que o cientista se concentre apenas em descrever a aplicação, sem levar em conta detalhes de execução. Os workflows modelados definem a ordem de execução das atividades e a movimentação dos dados envolvidos, mas não indicam que recursos computacionais devem ser usados na execução. Assim, os modelos precisam ser convertidos para uma forma executável por meio do mapeamento das atividades e dados nos recursos de armazenamento e computação disponíveis. Esse processo de mapeamento é conhecido como *escalonamento*.

Problemas de escalonamento são problemas de otimização combinatória em que, dado um conjunto de recursos (α) e um conjunto de atividades e suas restrições (β), pretende-se encontrar uma alocação (no tempo) de recursos a atividades que minimize algum critério de otimização (γ). Problemas de escalonamento são denotados, em geral, utilizando-se a notação $\alpha \mid \beta \mid \gamma$, introduzida por [Graham, 1966].

Dependendo do contexto, recursos e atividades podem significar coisas diferentes

[Leung, 2004]. Por exemplo, recursos podem ser máquinas de uma linha de montagem; CPU, memória e dispositivos de E/S em um sistema computacional; pistas de pouso e decolagem em um aeroporto; mecânicos em uma oficina mecânica; etc. Atividades, nesses contextos, poderiam ser as operações de um processo de manufatura, a execução das tarefas que compõem uma aplicação concorrente; pousos e decolagens em um aeroporto; o reparo de automóveis em uma oficina mecânica e assim por diante.

Este texto se restringe ao contexto de sistemas de computação de alto desempenho e nos concentrar em como a execução de workflows científicos pode ser modelada utilizando Teoria do Escalonamento.

Existem diferentes critérios de otimização que são estudados em problemas de escalonamento. No contexto de problemas de computação de alto desempenho, o critério de otimização mais estudado é denominado *makespan* (denotado também por $C_{\max}$), que indica o instante de tempo em que a última tarefa que compõe uma aplicação termina de ser executada.

O conjunto de tarefas e suas relações de precedência usualmente são descritos a partir de um digrafo acíclico (DAG) $G = (V,E)$, onde V é o conjunto de vértices e E é o conjunto de arcos do grafo. Cada tarefa i é representada por um vértice em V. Existe um arco com ponta inicial em i e ponta final em j (denotado por $i \rightarrow j$) se a tarefa i for uma *predecessora imediata* de j, ou seja, a tarefa j não pode ser iniciada antes do término da execução da tarefa i. Os arcos em um grafo de dependências tipicamente representam ou uma *dependência de controle* (como as definidas nos modelos de fluxo de controle descritos na Seção 1.2.2) ou uma *dependência de dados* (como as definidas nos modelos de fluxo de dados descritos na Seção 1.2.3) [Banerjee et al., 1993].

Quando o número de recursos computacionais disponíveis (m) é conhecido previamente e deseja-se minimizar o *makespan* — tipicamente o caso em problemas de escalonamento em aglomerados (*clusters*) e grades computacionais — o problema é fortemente NP-completo [Garey and Johnson, 1979]. Isso significa que não é possível construir um algoritmo polinomial para resolver o problema, a não ser que $P = NP$. Na prática, sistemas de gestão de recursos computacionais utilizam heurísticas que, apesar de não produzirem o resultado ótimo, são muito rápidas e, por vezes, possuem desempenho matematicamente garantido.

O restante dessa seção apresentará alguns dos algoritmos de escalonamento mais utilizados para a execução de workflows em plataformas de computação paralela e/ou distribuída.

1.3.1. Algoritmos de Lista

A Teoria do Escalonamento estuda a complexidade computacional e a eficiência de algoritmos para diferentes configurações de $\alpha \mid \beta \mid \gamma$. Encontrar o escalonamento ótimo geralmente é uma operação demorada. Entretanto, existem algoritmos capazes de computar soluções com um bom desempenho (matematicamente garantido) gastando uma quantidade de tempo razoável na prática. O exemplo mais proeminente de tais algoritmos são os chamados *algoritmos de lista* [Graham, 1966, Graham et al., 1979].

A classe dos algoritmos de lista é composta por heurísticas rápidas e eficientes

que permitem o escalonamento de tarefas em computadores paralelos. Esses algoritmos possuem uma garantia de aproximação de $\left(2-\frac{1}{m}\right)$ para o cálculo do *makespan* (no pior caso), mas produzem resultados próximos do ótimo na prática, principalmente quando a razão entre o número de tarefas e o número de recursos computacionais disponíveis é grande.

O princípio básico do funcionamento dos algoritmos de lista é, dada uma lista ordenada de tarefas, escolher de forma gulosa uma tarefa e designá-la ao processador que estiver menos sobrecarregado, executando-a o mais cedo possível. A ordem da lista de tarefas é definida usando algum esquema de prioridade escolhido de acordo com o objetivo de desempenho a ser otimizado e de acordo com as restrições de execução das tarefas.

O algoritmo LPT (*Longest Processing Time first*) cria a lista de tarefas em ordem não-crescente do tempo de processamento. Dessa forma, o maior percentual do trabalho a ser realizado (dado pelas tarefas de longa duração) é alocado de forma ótima. Uma análise mais fina permite mostrar que o LPT produz soluções no máximo $\frac{4}{3}-\frac{1}{3m}$ vezes pior que a solução ótima, garantia melhor do que a obtida por algoritmos de lista genéricos.

Nem sempre as aplicações estão interessadas em otimizar apenas o *makespan*. Para otimizar o tempo médio de término das tarefas — o caso típico de aplicações em que os resultados parciais podem ser utilizados imediatamente pelo usuário — deve-se usar o algoritmo SPT (*Shortest Processing Time first*). O SPT prioriza as tarefas em ordem não-decrescente de tempo de execução. O SPT produz soluções ótimas quando se quer minimizar o tempo médio de término, ao mesmo tempo em que seu $C_{\max}$ é limitado por $\left(2-\frac{1}{m}\right)$ do ótimo.

O algoritmo de lista mais comumente utilizado por sistemas de gerenciamento de workflows é o HEFT (*Heterogeneous Earliest Finish Time*) [Topcuoglu et al., 2002]. Proposto para aplicações que devem ser executadas em um número limitado de processadores heterogêneos, a execução do algoritmo possui duas etapas. A *etapa de priorização de tarefas* calcula a prioridade de cada uma das tarefas baseada nos seus custos de computação e comunicação. A *etapa de seleção dos processadores* avalia as tarefas em uma ordem topológica de prioridades, que respeita as relações definidas pelo grafo de precedências, e escalona cada tarefa no processador que minimizará o seu tempo de término (*earliest finish time first)*.

O HEFT não é o único algoritmo de lista usado para minimizar o *makespan* de aplicações modeladas como workflows científicos. O algoritmo *Myopic* [Wieczorek et al., 2005] é uma adaptação do esquema clássico de priorização FIFO (*first in, first out*), em que cada tarefa é alocada, em ordem de chegada, na máquina que a executará mais rapidamente. Os algoritmos *Min-Min* e *Max-Min* implementam uma heurística que atribui a prioridade de escalonamento às tarefas de acordo com o seu tempo de execução estimado. Os algoritmos listam todas as atividades a serem mapeadas e associam a cada atividade os tempos de execução observados ou previstos em cada um dos recursos disponíveis.

Os algoritmos *Min-Min* e *Max-Min* diferem apenas na política de priorização das tarefas. O *Min-Min* prioriza a atividade que possuir a menor estimativa para seu tempo

de execução e a mapeia para o recurso no qual o menor tempo foi obtido (daí o nome *Min-Min*). De modo similar, o algoritmo *Max-Min* prioriza atividades com o *maior* tempo previsto para a execução. Grosso modo, é possível relacionar as vantagens do uso de *Min-Min* e *Max-Min* com as vantagens dos algoritmos SPT e LPT, respectivamente, aplicadas a cenários heterogêneos.

Algoritmos com outras heurísticas, suas classificações e descrições podem ser encontrados nos levantamentos bibliográficos realizados por [Dong and Akl, 2006] e por [Yu and Buyya, 2005].

1.3.2. Algoritmos com Múltiplos Objetivos

Algoritmos como os descritos na seção anterior são utilizados até hoje pelos sistemas de gerenciamento de workflows científicos. Contudo, eles foram desenvolvidos para serem executados em plataformas de execução estáticas e bem conhecidas, tais como aglomerados (*clusters*) de computadores e grades computacionais clássicas como o Grid'5000[3], onde as máquinas são reservadas e obtidas antes da execução da aplicação. Em geral, os algoritmos de escalonamento para tais aplicações eram concebidos para que produzissem um escalonamento realizável (i.e., que respeita todas as restrições definidas pelo problema) e com um único objetivo, normalmente o de minimizar o tempo máximo de execução das aplicações (seu *makespan*).

Os novos ambientes de execução para sistemas distribuídos são formados por recursos computacionais disponibilizados de forma colaborativa e/ou são mais dinâmicos. Nesse contexto, novos objetivos de otimização passam a ser valorizados. Se antes os processadores eram reservados para uso exclusivo de uma aplicação, agora os recursos computacionais são disputados por diversos usuários, cada qual com seus próprios objetivos de desempenho [Cohen et al., 2011]. Minimizar o *makespan* da aplicação de um usuário pode ser impossível sem prejudicar o desempenho de outro.

Além disso, o advento de novas tecnologias de gestão de sistemas distribuídos nos permite acessar uma quantidade praticamente ilimitada de recursos sob demanda. A chamada *Computação em Nuvem* permite o aprovisionamento sob demanda de um conjunto de recursos de computação configuráveis (por exemplo, redes, servidores, dispositivos de armazenamento, aplicações e outros serviços). Entretanto, o uso desses recursos está condicionado a custos extras como o preço do aluguel dos serviços, ou a custos operacionais da gestão dessa grande quantidade de recursos. Ou seja, em alguns problemas pode ser necessário degradar um objetivo para otimizar o outro.

O aumento da complexidade dos workflows, combinado às características dos sistemas distribuídos modernos estão transformando os problemas que antes tinham um único objetivo em problemas com diversos objetivos que devem ser considerados simultaneamente. A essa classe de problemas dá-se o nome de *problemas multiobjetivo*.

Problemas multiobjetivo possuem uma noção diferente do significado de uma solução ótima. Tipicamente, um problema multiobjetivo não possui uma única solução que otimiza todos os objetivos ao mesmo tempo, mas sim uma multitude de soluções com diferentes compromissos. No caso de um problema com um único objetivo, é fácil com-

[3]Grid'5000 testbed: `https://www.grid5000.fr`

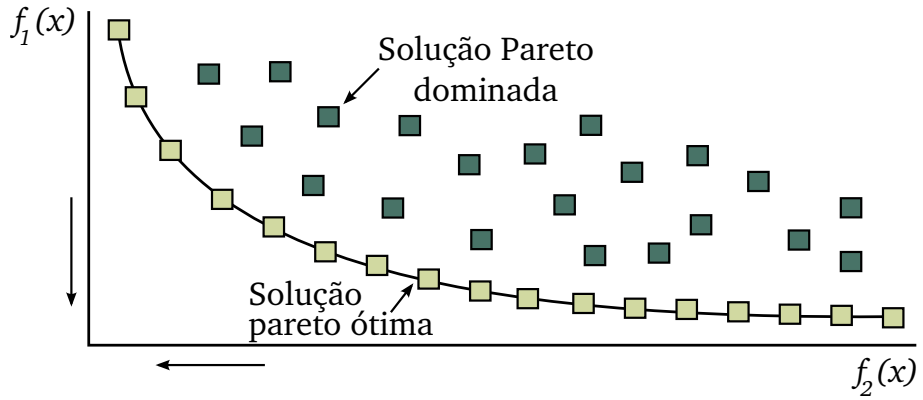


Figura 1.6: Exemplo de um conjunto de soluções de Pareto para um problema bi-objetivo.

parar as diferentes soluções. A ordem total induzida pela função objetivo é suficiente para definir qual solução é melhor. Em problemas multiobjetivo isso não é possível. Para representar o compromisso entre os diferentes objetivos, o conceito de *dominância de Pareto* [Voorneveld, 2003] é usado.

A intuição por trás do conceito técnico é que uma solução S é Pareto dominante em relação a uma solução S' se S é tão boa quanto S' em todos os objetivos menos em um, onde a solução S é estritamente melhor do que S'. Portanto, em problemas multiobjetivo deseja-se computar soluções que não são Pareto-dominadas por nenhuma outra solução. Essas soluções são chamadas de *soluções Pareto ótimas*. Essas noções estão ilustradas na Figura 1.6.

Workflows científicos possuem características especiais que são exploradas por algoritmos de escalonamento especializados e podem variar de acordo com a aplicação e/ou tipo de plataforma onde serão executados. Os problemas de escalonamento nesse contexto são naturalmente problemas multiobjetivo.

[Wieczorek et al., 2009] criaram uma taxonomia para problemas de escalonamento de workflows com múltiplos objetivos. Ao se analisar um problema de escalonamento de workflow, deve-se considerar certos fatores que influenciam de forma decisiva a escolha do algoritmo de escalonamento mais apropriado para o caso. Cada fator descreve o problema de escalonamento de uma perspectiva diferente. Os autores propõem cinco facetas principais que devem ser consideradas: (1) o modelo do workflow; (2) os critérios de escalonamento; (3) o processamento do escalonamento; (4) o modelo de recursos; e (5) o modelo de tarefas. Essas facetas são detalhadas a seguir.

1. **Modelo do workflow**

   (a) modelo de componentes:

   - orientado a tarefas (considera-se apenas a atividade na hora de realizar o escalonamento);
   - orientado a tarefas e transferência de dados (a transferência de dados também é considerada uma tarefa).

(b) estrutura:
- DAG: possui um digrafo de dependências entre as atividades;
- Digrafo estendido: estruturas como laços e condicionais também são permitidas;
- DAG simplificado: a estrutura do workflow é regular e pode ser descrita como um subconjunto de todos os DAGs possíveis. Por exemplo: seção paralela, sequência, arborescências, *fork-join*, etc.

(c) dinamismo da estrutura:
- fixa: os nós (atividades ou tarefas de transferência de dados) de um workflow não podem ser modificados;
- configurável: nós podem ser adicionados, removidos ou modificados durante o processo de escalonamento.

(d) processamento de dados:
- entrada simples: o workflow é executado uma única vez para um único conjunto de dados;
- *pipelined*: o workflow é executado múltiplas vezes, com fluxos diferentes de dados como entrada.

2. **Critérios de escalonamento**

(a) modelo de otimização:
- orientado ao workflow: o critério de otimização é definido pelo usuário que executa o workflow (ex.: tempo de execução);
- orientado à grade computacional: o critério é definido para a plataforma de execução (ex.: utilização dos recursos, justiça (*fairness*), etc.

(b) dependência da estrutura do workflow:
- dependente da estrutura (ex.: tempo de execução);
- independente da estrutura (ex.: custo financeiro da execução).

(c) impacto na otimização:
- objetivo: o objetivo da otimização é encontrar o melhor custo para um dado critério (ex.: tempo de execução);
- restrição: uma restrição imposta ao resultados (ex.: um prazo).

(d) método de cálculo:
- aditivo (ex.: custo financeiro);
- multiplicativo (ex.: qualidade dos dados);
- côncavo (ex.: largura de banda).

(e) flexibilidade do modelo de custo:
- fixo: o custo parcial dos serviços é dado como entrada para o escalonamento;
- adaptativo: o custo parcial dos serviços pode ser ajudado durante a execução (ex.: leilões ou negociações).

(f) intradependência:

- intradependente: a alteração do custo de uma tarefa pode implicar em alterações no custo das outras (ex.: tempo de execução);
- independente: os custos das tarefas são independentes entre si (ex.: confiabilidade).

(g) interdependência:

- interdependente: os critérios não são mutuamente independentes (ex.: tempo de execução e disponibilidade);
- independentes: tempo de execução e custo financeiro.

3. **Processamento do escalonamento**

(a) número de objetivos:

- mono-objetivo;
- multiobjetivo.

(b) multiplicidade do workflow:

- workflow simples: a execução de um único workflow é otimizada durante o processo de escalonamento;
- múltiplos workflows: as execuções de múltiplos workflows podem ser otimizadas simultaneamente em um único processo de escalonamento.

(c) dinamismo:

- *just-in-time*: a tomada de decisão é adiada até que uma tarefa esteja pronta para ser executada;
- planejamento *a priori*: o escalonamento é computado antes da execução do workflow (estático);
- híbrido: uma combinação dos dois anteriores.

(d) reserva dos recursos:

- com reserva: o escalonador se ocupa da obtenção dos recursos necessários para a execução;
- sem reserva: o escalonador utiliza os recursos já disponibilizados anteriormente.

4. **Modelo de recursos**

(a) diversidade:

- homogêneo: todo os recursos possuem as mesmas características;
- heterogêneo: os recursos possuem características diversas.

(b) execução de tarefas:

- sequencial: o escalonador só pode executar uma tarefa em um recurso em um dado instante de tempo;
- paralelo: o escalonador pode designar um mesmo recurso para a execução de múltiplas tarefas em um dado instante de tempo.

5. **Modelo de tarefas**

   (a) mapeamento dos recursos:

   - rígido: uma tarefa utiliza um número fixo de recursos;
   - moldável: o número de recursos a ser utilizado é definido pelo escalonador dinamicamente, antes da execução da tarefa;
   - maleável: o escalonador pode alocar e remover recursos de uma tarefa em tempo de execução.

   (b) migração:

   - com migração: a tarefa pode ser suspensa e seu estado transferido para outro processador;
   - sem migração: a tarefa executa até o final no mesmo recurso designado pelo escalonador no início de sua execução.

Cada uma das possíveis combinações das características levantadas por [Wieczorek et al., 2009] implica em problemas de escalonamento de natureza diferente, em que as técnicas e algoritmos para a solução do problema devem ser consideradas caso a caso. Por exemplo, o problema de escalonamento de workflows do tipo série-paralelo, com o objetivo de minimizar o consumo de energia, foi objeto de estudo de [Benoit et al., 2011]. Mostrou-se que o problema possui um algoritmo polinomial para algumas instâncias de DAGs simplificados, mas que para o caso geral o problema é NP-difícil. Ou seja, segundo a taxonomia apresentada acima, o problema pode ter sua solução ótima encontrada em tempo polinomial se a estrutura do modelo do workflow for a de um DAG simplificado. Porém, para DAGs genéricos, os autores propõem o uso de heurísticas gulosas e aleatorizadas para obter resultados aproximados.

1.4. Sistemas de Gerenciamento de Workflows Científicos

Um *sistema de gerenciamento de workflows científicos* (SGWCs) é um sistema que permite definir, modificar, gerenciar, executar e monitorar workflows científicos. Ou seja, é um sistema computacional que executa aplicações científicas compostas por atividades cuja ordem de execução é definida por uma representação digital da lógica do workflow [Lin et al., 2009]. Os sistemas de gerenciamento considerados neste texto gerenciam apenas workflows compostos exclusivamente por atividades científicas que podem ser implementadas e executadas programaticamente por um sistema computacional.

Sistemas de gerenciamento de workflows científicos auxiliam cientistas a projetar, executar e analisar experimentos computacionais utilizando um nível de abstração que oculta as tecnicalidades envolvidas nessas tarefas. Em geral, os SGWCs fornecem aos cientistas as seguintes funcionalidades:

- interface de usuário personalizável e interativa;
- ferramentas de apoio à reprodutibilidade do experimento;
- integração a ferramentas e serviços distribuídos heterogêneos;

- apoio à execução em sistemas computacionais de alto desempenho;
- monitoramento da execução de um workflow e tratamento de falhas na execução.

Essas funcionalidades estão presentes na maioria dos grandes SGWCs. Para tanto, esses sistemas proveem uma implementação para os subsistemas computacionais básicos [Lin et al., 2009] descritos a seguir:

- *Subsistema de Projeto de Workflows*: permite a criação e a modificação de workflows científicos. Esse subsistema produz especificações de workflows científicos em formato digital, descritas em uma linguagem de modelagem de workflows. A interoperabilidade entre diferentes SGWCs pode ser estabelecida por meio desse subsistema, em geral com o uso de linguagens de representação padronizadas.
- *Subsistema de Visualização e Apresentação*: permite visualizar o workflow e os dados produzidos por ele em diferentes formatos. É essencial que esse subsistema forneça uma apresentação que facilite a análise dos dados e de sua proveniência, de forma que o cientista consiga extrair conhecimento a partir do grande volume de dados e metadados gerados pela execução de diversos experimentos;
- *Subsistema de Execução de Workflows*: fornece as ferramentas computacionais para o gerenciamento e execução dos workflows. Esse subsistema cria e executa instâncias de workflows de acordo com o seu modelo digital, que define as transições de estado e as atividades computacionais que os compõem. Além disso, é responsabilidade desse subsistema escalonar as atividades nos recursos computacionais disponíveis. Em geral, os SGWCs controlam a execução de múltiplas instâncias de workflows ao mesmo tempo, que podem ser executadas concorrentemente.
- *Subsistema de Monitoramento*: monitora o status das instâncias de workflows durante sua execução. Também provê as ferramentas necessárias para a detecção e o tratamento das falhas que ocorrem durante a execução das instâncias.
- *Subsistema de Gerenciamento de Atividades*: lida com as questões ligadas à heterogeneidade do sistema distribuído e provê o gerenciamento e o ambiente necessário para a execução das atividades do modelo do workflow.
- *Subsistema de Gerenciamento de Proveniência*: responsável pela representação, armazenamento e consulta dos dados e metadados de proveniência;
- *Subsistema de Gerenciamento de Dados*: é responsável por gerenciar os dados usados e produzidos pelas atividades na execução do workflow. Deve lidar com problemas relacionados à heterogeneidade e ao grande volume dos dados manipulados pelas atividades. O grande volume pode resultar na necessidade de se distribuir os dados em várias máquinas. Isso torna o acesso aos dados e a movimentação dos mesmos um problema importante de desempenho (que ainda hoje é considerado um problema de pesquisa em aberto).

Neste texto, são apresentadas as características de quatro sistemas de gerenciamento que são amplamente adotados pela comunidade científica e que ainda possuem grupos de desenvolvimento ativos. Esses SGWCs resultaram de diferentes motivações de desenvolvimento, público-alvo específico e decisões técnicas particulares a cada projeto, o que faz com que se diferenciem consideravelmente um do outro e que exemplifiquem diferentes aspectos relacionados à execução e à modelagem de workflows. Os sistemas que serão descritos são: ASKALON [Fahringer et al., 2007], Pegasus [Deelman et al., 2005], Taverna [Wolstencroft et al., 2013] e Kepler [Ludäscher et al., 2006].

1.4.1. ASKALON

O ASKALON é um sistema de gerenciamento de workflows criado pelo grupo de Sistemas Distribuídos e Paralelos do Instituto de Ciência da Computação da Universidade de Innsbruck, na Áustria. O ASKALON foi criado com o objetivo principal de simplificar a execução de aplicações de workflow em grades computacionais.

A linguagem de modelagem utilizada no ASKALON é a AGWL (*Abstract Grid Workflow Language*), baseada em XML. Os usuários podem especificar seus modelos de workflows de forma textual, usando diretamente a AGWL, ou podem especificá-lo graficamente, usando uma interface gráfica baseada em diagramas UML que é distribuída junto com o ASKALON. A partir da representação gráfica do workflow, o ASKALON gera uma especificação textual em AGWL automaticamente. A linguagem AGWL é composicional, ou seja, permite que os workflows sejam modularizados e reutilizados na construção de outros workflows.

A AGWL possui construções para a representação tanto de fluxos de controle quanto de fluxos de dados. Dentre as construções de AGWL para a representação de fluxos de controle é possível destacar: (i) a construção `sequence`, que permite elencar várias atividades a serem executadas sequencialmente; (ii) o elemento `parallel`, que permite realizar paralelismo e sincronização de linhas de fluxo; (iii) o elemento `switch`, usado na modelagem de escolhas exclusivas e junções simples; (iv) os elementos `while`, `for` e `forEach`, que possibilitam a realização de iterações no workflow, sendo que o `forEach` define iterações que percorrem conjuntos de dados; (v) os elementos `parallelFor` e `parallelForEach`, que criam iterações que são executadas em processos paralelos.

No que se refere à representação de fluxos de dados, a AGWL permite que, na definição de uma atividade, a origem dos dados de entrada seja a saída de outra atividade do workflow, possibilitando a construção de um *pipeline*. Uma atividade de distribuição de dados é definida em AGWL como uma atividade que possui várias saídas. De modo análogo, uma atividade de agregação é definida como uma atividade que possui várias entradas de dados, enquanto que uma atividade de redistribuição é definida como uma atividade que possui múltiplas entradas e múltiplas saídas de dados. A AGWL também permite a definição de dados de entrada e saída para construções mais complexas, como as iterações e os condicionais.

Em AGWL, é possível definir os tipos de dados das saídas das atividades. Os tipos de dados possíveis são os tradicionais dos sistemas de computação, como `integer`, `float`, `boolean` e `string`. A AGWL possui ainda os tipos `file`, que possibilita o

```
<cgwd name="Wien2K">
  <cgwdInput>
    <dataIn name="startInput" type="agwl:file"
            source="gsiftp://.../.../WIEN2K/atype/STARTINPUT.txt"/> ...
  </cgwdInput>
  <cgwdBody>
    <while name="whileConv">
      <dataLoops>
        <dataLoop name="overflag" type="xs:boolean"
                  initSource="Wien2K/overflag" loopSource="MIXER/overflag"/>
      </dataLoops>
      <condition> whileConv/overflag </condition>
      <loopBody>
        <activity name="LAPW0" type="wien:LAPW0">
          <dataIns>
            <dataIn name="startInput" type="..." source="Wien2K/startInput"/> ...
          </dataIns>
          <dataOuts>
            <dataOut name="kpoints" type="xs:integer"/> ...
          </dataOuts>
        </activity>
        <parallelFor name="pforLAPW1">
          <loopCounter name="lapw1Index" type="..." from="1" to="LAPW0/kpoints"/>
          <loopBody>
            <activity name="LAPW1" type="wien:LAPW1" .../>
          </loopBody>
          <dataOuts .../>
        </parallelFor>
        <activity name="LAPW1_FERMI" type="wien:LAPW1_FERMI" .../>
        <parallelFor name="pforLAPW2" .../>
        <activity name="MIXER" type="wien:MIXER" .../>
      </loopBody>
      <dataOuts>
        <dataOut name="overflag" type="xs:boolean" source="MIXER/overflag"/>
      </dataOuts>
    </while>
  </cgwdBody>
  <cgwdOutput>
    <dataOut name="overflag" type="xs:boolean" source="whileConv/overflag"
             saveto="gsiftp://.../.../WIEN2K/result/..."/>
  </cgwdOutput>
</cgwd>
```

Figura 1.7: Exemplo de trecho de workflow definido em AGWL (extraído de [Fahringer et al., 2007]).

acesso a um arquivo por meio de uma URL, e o `collection` para a criação de coleções de dados que podem ser acessadas diretamente ou separadamente, em blocos.

A Figura 1.7 mostra um trecho de um arquivo de definição de workflow em AGWL. O workflow em questão é a aplicação WIEN2k [4], que realiza cálculos relacionados à estrutura eletrônica de sólidos usando a Teoria do Funcional da Densidade, com os métodos *(linearized) augmented plane-wave* ((L)APW) e *local orbital* (lo).

Os modelos de workflows descritos em AGWL são completamente independentes de recursos de execução, ou seja, eles não trazem associações diretas entre as atividades e os recursos que serão usados em sua execução. O workflow representado em AGWL é interpretado pelos serviços de *middleware* do ASKALON, para escalonamento e execução em grades ou nuvens. Os serviços principais do sistema são [Fahringer et al., 2007]:

[4] Pacote de programas WIEN2k: `http://www.wien2k.at/`.

- o *Escalonador (Scheduler)*, que define o mapeamento do workflow para execução na grade;
- o *Motor de Execução (Enactment Engine)*, que gerencia a execução e garante confiabilidade;
- o *Gerenciador de Recursos (Resource Manager)*, que controla os componentes de software e hardware disponíveis;
- a *Predição de Desempenho (Performance Prediction)*, que mantém dados a respeito do tempo de execução das atividades e das transferências de dados, obtidos a partir de métodos estatísticos e de observações de execuções feitas em fases de treinamento dos workflows.

O ASKALON implementa um conjunto de algoritmos de escalonamento que difere dos outros sistemas de gerenciamento por sua diversidade de tipos. O sistema de gerenciamento provê tanto algoritmos com desempenhos garantidos, quanto heurísticas de Computação Evolucionária.

O algoritmo padrão do ASKALON – o HEFT (descrito na Seção 1.3) – é um algoritmo de lista com desempenho garantido. O mapeamento utilizando o HEFT leva em conta informações coletadas em tempo de execução que são utilizadas como entrada de um algoritmo simples de predição de desempenho. Ou seja, os pesos utilizados no HEFT são provenientes de informações coletadas de execuções passadas. O ASKALON também implementa outro algoritmo de lista denominado *Myopic*. Esse algoritmo, apesar de simples, é muito rápido (o que reduz a sobrecarga de desempenho causada pelo escalonador).

Por fim, o ASKALON também provê um conjunto de *algoritmos genéticos*. Os algoritmos genéticos são utilizados para obter (em tempo hábil) mínimos locais em um espaço de busca multidimensional (ou seja, quando diversos objetivos devem ser minimizados ao mesmo tempo). Algoritmos genéticos procuram por soluções emulando operações que ocorrem na evolução de cadeias cromossômicas como, por exemplo, mutações e cruzamentos. Dessa forma, novas soluções são encontradas a partir de combinações ou de pequenas mudanças aleatórias.

1.4.2. Taverna

O Taverna, desenvolvido como parte do projeto *myGrid* [5], é um ambiente para criação e execução de workflows científicos amplamente utilizado por cientistas de vários campos da ciência, mas teve sua motivação nas necessidades específicas de cientistas da área de Ciências Biológicas que geralmente não possuem conhecimento profundo de programação, mas fazem uso de componentes programáveis e executáveis (como scripts *ad hoc* e serviços Web) para execução de experimentos.

O Taverna permite a construção de workflows a partir de componentes (tanto remotos quanto locais) que são disponibilizados publicamente pela comunidade científica na forma de serviços. Um workflow Taverna é especificado como um grafo direcionado

[5]Projeto myGrid: `http://www.mygrid.org.uk/`

onde os nós, chamados na ferramenta de *processadores*, são as atividades implementadas como serviços. Um nó de atividade consome dados que chegam em suas entradas e produz dados em suas saídas. Cada arco no grafo conecta um par entrada-saída e denota uma dependência de dados a partir da saída da atividade de origem para a entrada da atividade de chegada. Adicionalmente, ligações de controle podem ser definidas entre atividades para determinar que uma atividade só pode ser executada depois que outra atividade tenha concluído seu processamento [Oinn et al., 2006, Tan et al., 2009].

O processo de construção e execução de workflows no Taverna se inicia com a descoberta de serviços e a composição do modelo no *Workbench* (Figura 1.8 [6]), que é a interface gráfica para o acesso aos serviços disponíveis e a construção do workflow. Um workflow definido graficamente é representado textualmente na linguagem Scufl (*Simplified Conceptual Workflow Language*), baseada em XML. Finalmente, a execução do workflow ocorre quando o arquivo codificado em Scufl é interpretado pelo *Motor de Execução* do Taverna.

A Scufl é uma linguagem que foca a representação de fluxos de dados. A construção de *pipelines* é a atividade básica do Taverna; as construções de Distribuição, Agregação e Redistribuição de dados podem ser modeladas no *Workbench* como múltiplas entradas e saídas ligadas às atividades conforme necessário. Cada saída de uma atividade pode ter um tipo de dados associado a ela. Com isso, assim como ocorre na AGWL, é possível que uma atividade possua múltiplos tipos de saída. O gerenciador Taverna suporta diferentes interfaces de serviços e seus respectivos tipos de dados, possibilitando o uso de estruturas de dados definidas pelos próprios serviços. O Taverna permite também o uso de estruturas de dados como listas e árvores, que podem organizar os dados em coleções.

A Scufl possui ainda algumas construções para a representação de fluxos de controle. Uma sequência é definida em Scufl por meio da *ligação de coordenação*, que impede que uma atividade inicie seu processamento até que uma outra atividade indicada pela ligação complete sua execução. Também é possível definir na linguagem paralelismo e sincronização de atividades. Entretanto, Scufl não possui outras construções para fluxo de controle. Portanto, escolhas, junções simples e iterações só podem ser expressas por meio de dependências de dados entre as atividades. Também não há na linguagem construções para desvios condicionais, mas eles podem ser implementados por meio de serviços fornecidos pelo próprio Taverna, como as atividades `FailIfFalse` e `FailIfTrue`.

Assim como AGWL, a Scufl é uma linguagem composicional. Ela permite que um workflow seja usado como uma atividade na definição de outros workflows.

O modelo de um workflow para o Taverna pode possuir algumas especificações para apoiar a sua execução. Por exemplo, pode-se definir no modelo a quantidade de tentativas de reinício para um dado serviço em caso de falha, ou ainda, associar a ele serviços alternativos. Também pode-se definir o número máximo de invocações paralelas que podem ser feitas a uma atividade quando a mesma possuir vários dados de entrada prontos para execução em um dado instante do tempo.

[6] Figura extraída do manual do Taverna: `www.taverna.org.uk`.

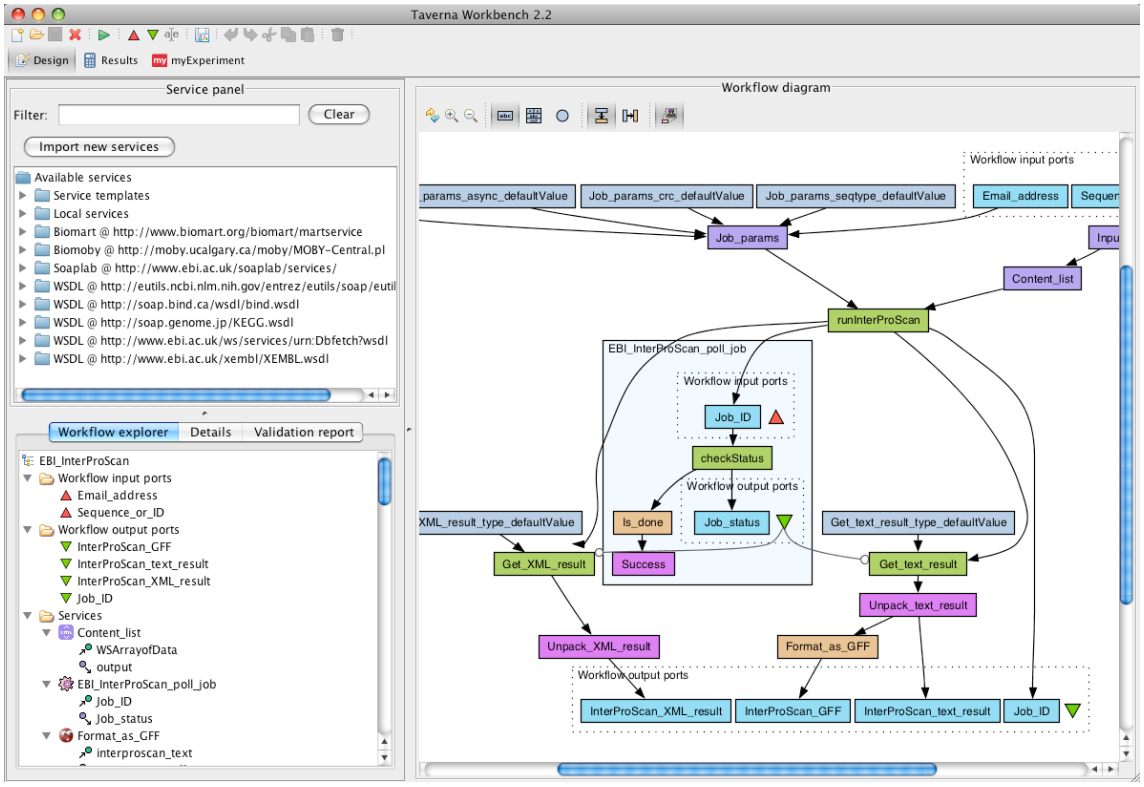


Figura 1.8: Tela principal do Taverna.

Diferentemente dos SGWCs tradicionais, que usam em seus subsistemas de execução algoritmos clássicos de escalonamento, o motor de execução do Taverna emprega uma estratégia de execução bastante simples, baseada em um modelo *multi-thread*. Durante a execução de uma instância de workflow, o Taverna inicia a execução de uma atividade em uma nova *thread* tão logo as entradas da atividade possuam ao menos um item de dado disponível. O Taverna usa essa estratégia porque seus workflows se baseiam em serviços que estão ligados a recursos computacionais fixos, enquanto que os outros SGWCs precisam lidar com o problema de mapear a execução de suas atividades em um conjunto variável de recursos computacionais.

1.4.3. Pegasus

O SGWC Pegasus (*Planning for Execution in Grids*) [Deelman et al., 2005, Deelman et al., 2012] é um arcabouço que permite o mapeamento de instâncias de workflow em um conjunto de recursos distribuídos, como uma grade. O Pegasus recebe uma descrição do workflow e a informação sobre os recursos disponíveis, e faz o mapeamento para gerar um workflow executável (que consiste no workflow original acrescido de todas as informações de recursos necessárias para execução).

A descrição do workflow é fornecida como entrada na forma de um DAG em formato XML, na linguagem DAX (*Directed Acyclic Graph in XML*) [Deelman et al., 2005], e representa o fluxo de processamento e as dependências entre as tarefas. A linguagem DAX descreve modelos de workflows como grafos de atividades, onde cada atividade possui um código de identificação e informações e metadados sobre seus arquivos de en-

trada e saída. A linguagem DAX é puramente orientada a fluxo de dados. Um exemplo de workflow descrito em DAX pode ser visto na Figura 1.9.

No Pegasus, as informações sobre os recursos são fornecidas pelo conjunto de serviços de grade *Globus Toolkit* [Foster and Kesselman, 1996].

O sistema é formado por 3 componentes principais:

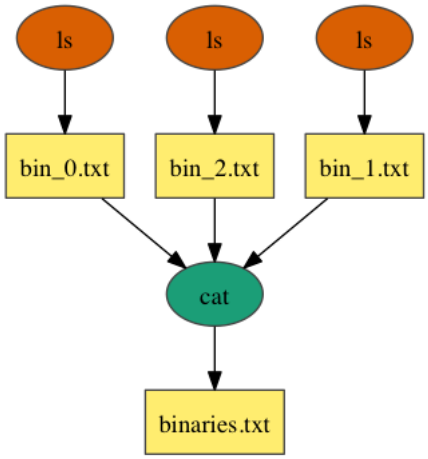
- *Mapeador* (*PegasusMapper*), que gera um workflow executável baseado no modelo de workflow fornecido pelo usuário ou por um sistema de composição de workflow. É função do mapeador encontrar o software, os dados e os recursos computacionais requeridos para a execução do workflow. O Mapeador pode também reestruturar o workflow para otimizar o desempenho e adicionar transformações para o gerenciamento de dados e geração de informações de proveniência;
- *Motor de Execução* (*DAGMan*), é o componente responsável por analisar a descrição do workflow, interagir com os recursos remotos, monitorar o estado das atividades em execução e identificar as novas atividades que ficaram prontas para execução, como resultado das atividades que foram concluídas;
- *Gerenciador de Tarefas* (*Condor-G*): que recebe as requisições do *DAGMan* e gerencia as tarefas do workflow individualmente, supervisionando sua execução nos recursos locais e remotos.

O Pegasus distingue qual o nó do sistema distribuído que é responsável pela submissão de novas tarefas para o ambiente da grade computacional. O *submit host* deve executar os serviços do Pegasus, do DAGMan e do Condor-G, que submetem tarefas para a grade computacional gerenciada pelo Globus. Esse nó também é o responsável por manter informações sobre os componentes computacionais executáveis instalados em outros nós remotos e sobre os locais de execução, por meio de *catálogos*, e serve também como plataforma de execução ou depuração local de workflows.

A combinação do *DAGMan*, *Condor-G* e *Globus* permite a execução do workflow executável gerado pelo Pegasus em um ambiente de grade composto de recursos heterogêneos. O Pegasus submete o workflow descrito em DAX para o *DAGMan*, que envia as tarefas para o *Condor-G* no momento em que elas estão prontas para execução, respeitando as dependências descritas no workflow. O Condor-G se comunica com o Globus para ter acesso aos recursos distribuídos, e as tarefas são executadas.

O escalonamento das tarefas e a otimização do modelo do workflow é realizada pelo *PegasusMapper*. O mapeador do Pegasus é capaz de realizar otimizações baseadas no modelo abstrato do workflow e no estado atual de sua execução. Se uma atividade for responsável por gerar um dado intermediário que já está presente no sistema, o mapeador tem a liberdade de decidir não reexecutar a atividade. Além disso, o mapeador pode decidir agrupar algumas atividades e forçar sua execução em um mesmo nó para diminuir a movimentação de dados. Isso torna a execução do workflow mais eficiente.

O mapeador implementa as seguintes heurísticas de escalonamento, sendo a primeira a heurística padrão:



(a) Exemplo de modelo de workflow como um DAG.

```
<adag xmlns="http://pegasus.isi.edu/schema/DAX"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://pegasus.isi.edu/schema/DAX
http://pegasus.isi.edu/schema/dax-3.4.xsd" version="3.4" name="merge">
        <job id="ID0000001" name="ls">
                <argument>-l /bin</argument>
                <stdout name="bin_0.txt" link="output"/>
                <uses name="bin_0.txt" link="output"/>
        </job>
        <job id="ID0000002" name="ls">
                <argument>-l /usr/bin</argument>
                <stdout name="bin_1.txt" link="output"/>
                <uses name="bin_1.txt" link="output"/>
        </job>
        <job id="ID0000003" name="ls">
                <argument>-l /usr/local/bin</argument>
                <stdout name="bin_2.txt" link="output"/>
                <uses name="bin_2.txt" link="output"/>
        </job>
        <job id="ID0000004" name="cat">
                <argument>
                        <file name="bin_0.txt"/>
                        <file name="bin_1.txt"/>
                        <file name="bin_2.txt"/>
                </argument>
                <stdout name="binaries.txt" link="output"/>
                <uses name="bin_2.txt" link="input"/>
                <uses name="bin_0.txt" link="input"/>
                <uses name="binaries.txt" link="output" register="false" transfer="true"/>
                <uses name="bin_1.txt" link="input"/>
        </job>
        <child ref="ID0000004">
                <parent ref="ID0000001"/>
                <parent ref="ID0000002"/>
                <parent ref="ID0000003"/>
        </child>
</adag>
```

(b) Descrição do workflow na linguagem DAX.

Figura 1.9: Exemplo de workflow que concatena dados representado graficamente e na linguagem DAX.

- *Random*: as tarefas são atribuídas aos nós disponíveis de forma aleatória;
- *RoundRobin*: os nós disponíveis são selecionados para receberem tarefas de forma sequencial;
- *Group*: grupos de tarefas (definidas manualmente pelo desenvolvedor na linguagem DAX) são atribuídas obrigatoriamente a um mesmo nó. Atividades que não são incluídas em nenhum grupo são escalonadas utilizando a heurística *Random*;
- *HEFT*: as tarefas são escalonadas seguindo o algoritmo de escalonamento HEFT, descrito na Seção 1.3.

1.4.4. Kepler

O Kepler adota o paradigma *orientado a atores* para a modelagem e a execução de workflows científicos. Cada ator é projetado para desempenhar uma atividade específica, que pode ser atômica ou composta. Os atores em um workflow podem conter portas para consumir ou produzir dados, e se comunicar por meio de canais com outros atores no workflow.

Com o Kepler é possível criar workflows hierárquicos, possibilitando que atividades complexas sejam definidas a partir de componentes mais simples. Uma particularidade do Kepler é que a ordem de execução dos atores em um workflow é determinada por uma entidade independente, chamada *diretor* [Ludäscher et al., 2006]. É o diretor quem define como os atores são executados e como eles se comunicam entre si. O projetista do workflow é quem escolhe qual(is) diretor(es) será(ão) usado(s), por meio da interface gráfica disponibilizada pelo Kepler para a modelagem do workflow. Cada tipo de diretor possui associado a si um tipo de modelo de computação específico, que define, por exemplo, se os atores serão executados de forma sequencial ou paralela.

Um diferencial do Kepler é a facilidade ao acesso a serviços implementados como serviços Web. O sistema disponibiliza um ator genérico denominado `WebService`. Dada a URL do descritor WSDL (*Web Service Definition Language*) do serviço Web, é possível incorporar o serviço ao workflow e utilizá-lo como se ele fosse um componente local. Em particular, as entradas e saídas definidas pelo WSDL são explicitamente mapeadas ao mecanismo de entrada e saída dos atores.

Além disso, o Kepler se integra facilmente às grades computacionais que utilizam o Globus [Foster and Kesselman, 1996]. Uma atividade implementada como um `Job` do Globus pode ser facilmente executada em qualquer workflow usando o ator `RunJobGridClient`. Arquivos podem ser armazenados e recuperados de computadores pertencentes à grade com o uso dos atores `FileFetcher` e `FileStager` (que armazenam e recuperam os arquivos, respectivamente).

Uma aplicação local pode ser integrada ao workflow com o uso do ator `CommandLine`. Qualquer aplicação acessível pela linha de comando pode ser executada usando esse ator.

Outra característica interessante é que o sistema pode notificar o usuário sobre alterações no workflow de forma assíncrona. O ator `Email` oferece uma forma simples e conveniente de enviar, via e-mail, notificações sobre a execução de um workflow.

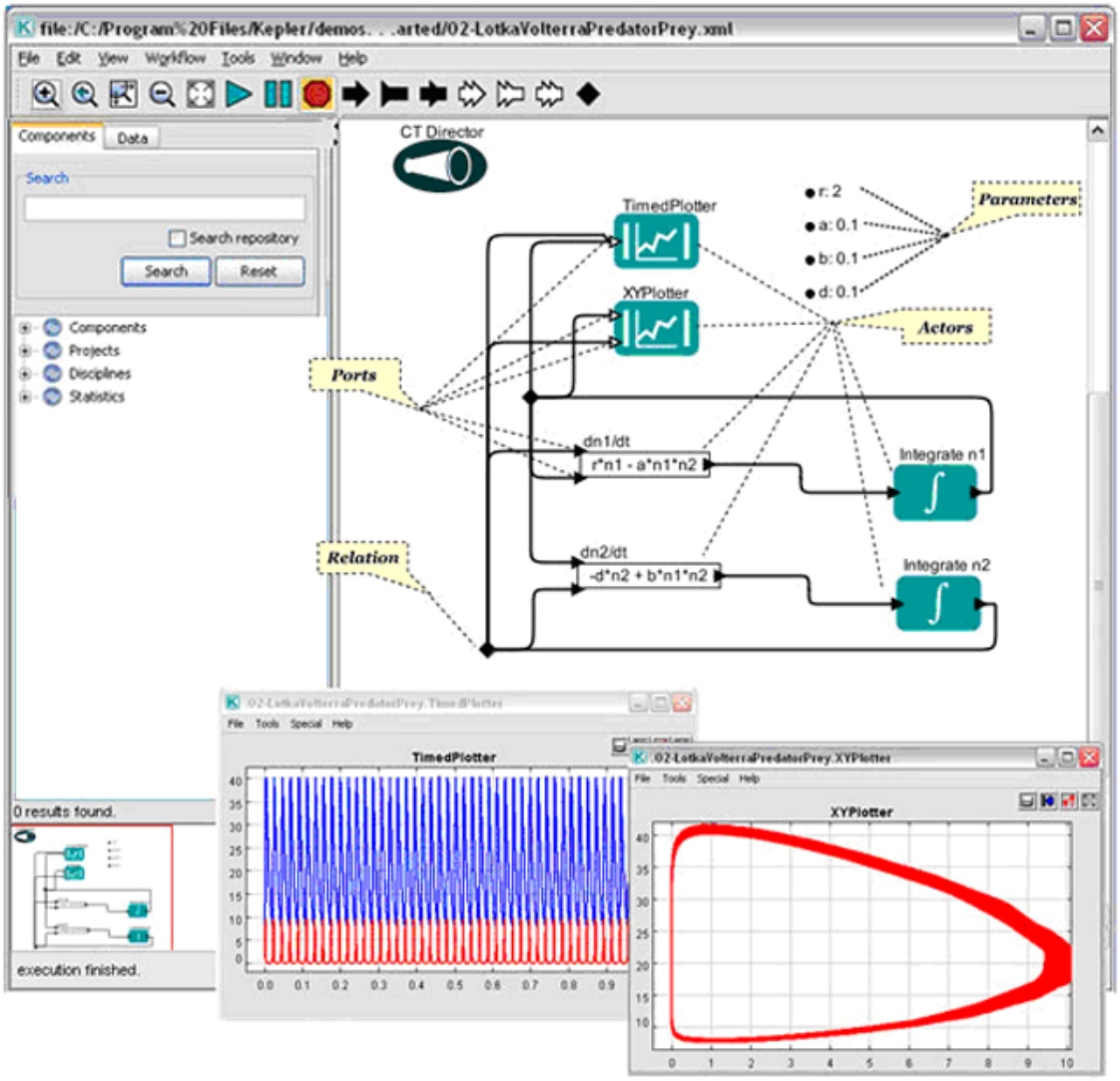


Figura 1.10: Exemplo de workflow modelado com o Kepler

O Kepler também oferece acesso direto a dados científicos mantidos em vários repositórios públicos, como o do *Knowledge Network for Biocomplexity* (KNB).

A Figura 1.10 mostra um exemplo de workflow extraído do manual[7] do Kepler.

1.4.5. Ferramentas Complementares

O uso de sistemas de gerenciamento de workflows possibilitou o estabelecimento de novos meios de comunicação e cooperação entre cientistas. Portais da Web como o *MyExperiment* [8] permitem que workflows e experimentos *in silico* sejam divulgados e compartilhados entre diferentes grupos de pesquisa. Os workflows compartilhados são categorizados de acordo com critérios como linguagem de modelagem (sistema de gerenciamento),

[7]Exemplos de workflows disponibilizados pelo Kepler: `https://kepler-project.org/users/sample-workflows`.

[8]Sítio Web do *MyExperiment*: `http://www.myexperiment.org/`.

funcionalidades, grupo de pesquisa criador, etc.

Além disso, é cada vez mais frequente que grupos de pesquisa disponibilizem bases de dados e outros artefatos digitais. Bancos com dados coletados de experimentos por pesquisadores de diferentes áreas do conhecimento não só permitem que suas pesquisas possam ser reproduzidas por outros pesquisadores, mas também fomentam que novas pesquisas sejam desenvolvidas a partir dos dados disponibilizados.

1.5. Execução de Workflows Científicos em Plataformas de Computação de Alto Desempenho

Em workflows científicos, de maneira geral, é comum a manipulação de grandes volumes de dados por um número elevado de atividades, o que requer grande quantidade de tempo de processamento e de espaço de armazenamento. Para atender essa necessidade, plataformas de computação de alto desempenho como aglomerados, grades e, mais recentemente, nuvens de computadores, têm sido empregadas para permitir aos cientistas desenvolverem e executarem workflows científicos complexos e obterem resultados em tempo razoável.

Um *aglomerado* (*cluster*) é definido como um conjunto de *nós de processamento* interconectados por uma rede de alta velocidade, trabalhando de forma integrada e vistos como um sistema único que abrange todos os nós [Baker and Buyya, 1999]. Um nó de processamento pode ser um sistema multiprocessado ou de processador único, e um aglomerado típico geralmente se refere a dois ou mais nós conectados por uma rede local.

Plataformas de *Computação em Grades* (*Grid Computing*) [Foster and Kesselman, 2003] são plataformas computacionais cooperativas. Elas são formadas por recursos computacionais heterogêneos oriundos de diferentes organizações, possivelmente geograficamente distribuídas. Uma organização (um laboratório, uma universidade, etc.) contribui com seus recursos computacionais, e em troca pode executar suas próprias aplicações em máquinas ociosas pertencentes a outras organizações.

Plataformas de Computação em Nuvem (do inglês *cloud computing* [Mell and Grance, 2011]) oferecem computação como um serviço que é entregue sob demanda, disponibilizando conjuntos de recursos virtualizados e facilmente acessíveis. São definidas como um modelo que permite o acesso, via rede, a um conjunto de recursos computacionais configuráveis (como servidores, armazenamento, aplicações e serviços) de forma ubíqua, conveniente e sob demanda. Esses recursos podem ser rapidamente provisionados e liberados com um mínimo esforço de gerenciamento ou interação com o provedor de serviço. Plataformas de Computação em Nuvem podem ser classificadas quanto à sua estrutura ou quanto ao seu modelo de serviço. Quanto à estrutura, nuvens podem ser:

- *Privadas*: formadas com recursos computacionais próprios;
- *Comunitárias*: formadas com recursos provenientes de várias organizações;
- *Públicas*: disponibilizadas por fornecedores comerciais, como é o caso dos serviços

Amazon Elastic Compute Cloud [9] e *Microsoft Azure Services* [10];

- *Híbridas*: formadas por combinações das estruturas anteriores.

O uso de novas técnicas de computação paralela e distribuída como a Computação em Nuvem promove a democratização do acesso ao poder computacional [Cordeiro et al., 2013]. Oportunidades de pesquisa que antes eram restritas ao seleto grupo dos que tinham acesso a supercomputadores agora podem ser exploradas por milhares de pesquisadores.

O gerenciamento de recursos em SGWCs deve lidar com as características intrínsecas à cada tipo de plataforma computacional[Yu et al., 2008]. Dentre as características mais importantes, os SGWCs devem ser capazes de lidar com problemas relacionados a:

- *Compartilhamento de recursos*: vários usuários competem ao mesmo tempo pelos mesmos recursos;
- *Controle distribuído*: não há um controle centralizado, e os recursos podem estar submetidos a controladores diferentes em cada domínio com restrições e regras próprias;
- *Heterogeneidade*: os recursos podem ser altamente heterogêneos, com tecnologias variadas e desempenho desigual para tarefas diferentes;
- *Dinamicidade*: a disponibilidade dos recursos é variável, e os controladores devem adequar seu comportamento à disponibilidade dos recursos;
- *Comunicação*: o custo de movimentação de dados entre os recursos deve ser considerado, principalmente em aplicações intensivas em dados como os workflows científicos.

Todos os sistemas de gerenciamento apresentados na Seção 1.4 são capazes de utilizar aglomerados e grades de computação. Nessas plataformas, os recursos computacionais se apresentam como uma lista estática de recursos com suas características conhecidas *a priori*.

As versões mais recentes desses gerenciadores possuem algum tipo de adaptação para a execução em plataformas de Computação em Nuvem. Via de regra, as novas versões introduziram algum componente de software que permite a alocação de uma quantidade fixa de recursos de um provedor de Computação em Nuvem e permitem o acesso aos serviços de armazenamento oferecidos por ela. Após a inicialização da plataforma, os recursos alocados são manipulados da forma habitual, como se pertencessem a um aglomerado de computadores.

Essa abordagem simples possui duas vantagens: (i) o custo com a alocação dos recursos (se houver) é controlado e pode ser facilmente previsto; ao escolher os tipos e

[9] Amazon AWS: `http://aws.amazon.com`
[10] Microsoft Azure: `http://azure.microsoft.com`

quantidades de recursos que serão disponibilizados para a execução do workflow, pode-se controlar a quantidade de recursos que será utilizada, e (ii) não são necessárias alterações nem no código dos programas, nem no modo como o usuário interage com o SGWC; como as máquinas são pré-alocadas, não há diferença prática na utilização de nuvens ou aglomerados do ponto de vista do utilizador. Um bom exemplo de SGWC que utiliza essa abordagem é o Tavax [Abouelhoda et al., 2012].

Entretanto, essas abordagens ignoram uma das características mais promissoras de plataformas de Computação em Nuvem: a chamada *elasticidade*. Em uma plataforma de Computação em Nuvem, um novo recurso computacional pode ser adquirido em alguns minutos (ao contrário de um aglomerado, onde a compra de um computador novo pode demorar semanas). Um sistema de gerenciamento de recursos avançado poderia monitorar a execução de uma instância do workflow e decidir quando alocar ou remover recursos para acelerar a execução do workflow. A consequência imediata é que o custo e o número de recursos computacionais é variável e depende do modelo de workflow a ser executado.

Há um claro compromisso entre o desempenho da execução e o seu custo. Pesquisas recentes [Fard et al., 2012, Zheng and Sakellariou, 2013] descrevem algoritmos que avaliam a execução de workflows levando em consideração esse compromisso. A alocação de recursos tendo em vista vários objetivos é explicada em detalhes na Seção 1.3.

1.6. Desafios Atuais para a Pesquisa na Área de Workflows Científicos

No gerenciamento de workflows científicos intensivos em dados, uma preocupação importante é o impacto das transferências de dados entre as atividades no tempo de execução do workflow. Os SGWCs vêm combinando diferentes estratégias de gerenciamento de dados e escalonamento, para garantir execuções eficientes dos workflows em plataformas computacionais distribuídas, mas que possuem uma quantidade fixa de recursos. Nesse contexto, os SGWCs da atualidade possuem duas deficiências não-negligenciáveis:

- Eles não estão aptos a gerenciar fluxos de dados envolvendo volumes enormes de dados (e.g., *big data*) – a maioria dos SGWCs só é capaz de gerenciar conjuntos de dados que podem ser armazenados como arquivos comuns, que cabem em uma única máquina, e que podem ser facilmente transmitidos de um nó para outro quando necessário;

- Eles não foram adaptados ainda para explorar a capacidade total das plataformas de Computação em Nuvem – esses sistemas vêm usando as nuvens como grades; ainda não se beneficiam da elasticidade e da escalabilidade dos recursos nas nuvens.

Ambos os problemas requerem soluções avançadas de gerenciamento de recursos que otimizem mais de um critério de desempenho ao mesmo tempo — por exemplo, minimizar o tempo de execução e a quantidade de dados transferidos no primeiro caso, ou minimizar o tempo de execução e o custo total de alocação dos recursos no segundo caso. Problemas multiobjetivo são difíceis de otimizar. Não existe uma solução ótima para tais problemas, mas sim um conjunto de soluções que oferecem algum tipo de compromisso

entre os objetivos. Grande parte das pesquisas realizadas atualmente propõem heurísticas que ou agregam os múltiplos objetivos em um único objetivo (o que nem sempre é possível), ou tentam otimizar um dos objetivos impondo algum limite máximo para os outros objetivos. Isso torna possível encontrar soluções próximas às soluções ótimas de Pareto em tempo hábil para que o escalonamento dos recursos não se torne um gargalo no sistema.

Além disso, há ainda questões em aberto sobre a utilização de plataformas colaborativas compartilhadas entre vários usuários, como é o caso da Computação em Nuvem. A imprevisibilidade do desempenho (causada pelo compartilhamento de recursos físicos entre diferentes máquinas virtuais) e a falta de algoritmos adaptativos para a execução de workflows (capazes de ajustar o número de recursos em utilização para a execução dos workflows) ainda é um desafio a ser superado para a realização de computação de alto desempenho em tais plataformas.

Espera-se que esses problemas sejam mitigados com a criação de novos algoritmos de escalonamento e de sistemas de gerenciamento de workflows científicos otimizados para essas novas plataformas.

1.7. Agradecimentos

Os autores gostariam de agradecer as valiosas ideias e contribuições de Eduardo Cotrin Teixeira (UTFPR) a esse trabalho.

Esse trabalho foi parcialmente financiado pela Fundação de Amparo à Pesquisa do Estado de São Paulo (processos FAPESP #2011/24114-0 e #2012/03778-0).

Referências

[Abouelhoda et al., 2012] Abouelhoda, M., Issa, S. A., and Ghanem, M. (2012). Tavaxy: Integrating taverna and galaxy workflows with cloud computing support. *BMC Bioinformatics*, 77(13).

[Baker and Buyya, 1999] Baker, M. and Buyya, R. (1999). Cluster computing at a glance. *High Performance Cluster Computing: Architectures and Systems*, 1:3–47.

[Banerjee et al., 1993] Banerjee, U., Eigenmann, R., Nicolau, A., and Padua, D. A. (1993). Automatic program parallelization. *Proceedings of the IEEE*, 81(2):211–243.

[Barga and Gannon, 2007] Barga, R. and Gannon, D. (2007). Scientific versus business workflows. In Taylor, I., Deelman, E., Gannon, D., and Shields, M., editors, *Workflows for e-Science*, pages 9–16. Springer London.

[Benoit et al., 2011] Benoit, A., Melhem, R., Renaud-Goud, P., and Robert, Y. (2011). Energy-aware mappings of series-parallel workflows onto chip multiprocessors. In Gao, G. R. and Tseng, Y.-C., editors, *International Conference on Parallel Processing*, ICPP, pages 472–481.

[Bharathi et al., 2008] Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.-H., and Vahi, K. (2008). Characterization of scientific workflows. In *Workflows in Support of Large-Scale Science, 2008*, pages 1–10. IEEE Computer Society.

[BPMN, 2013] BPMN (2013). *Business Process Model and Notation*. Object Management Group, 2.0.2 edition.

[Cohen et al., 2011] Cohen, J., Cordeiro, D., Trystram, D., and Wagner, F. (2011). Multi-organization scheduling approximation algorithms. *Concurrency and Computation: Practice and Experience*, 23(17):2220–2234.

[Cordeiro et al., 2013] Cordeiro, D., Braghetto, K. R., Goldman, A., and Kon, F. (2013). Da ciência à e-ciência: paradigmas da descoberta de conhecimento. *Revista USP*, 97:71–81.

[Cuevas-Vicenttin et al., 2012] Cuevas-Vicenttin, V., Dey, S. C., Köhler, S., Riddle, S., and Ludäscher, B. (2012). Scientific workflows and provenance: Introduction and research opportunities. *Datenbank-Spektrum*, 12(3):193–203.

[Deelman et al., 2012] Deelman, E., Mehta, G., Singh, G., Su, M.-H., and Vahi, K. (2012). *Workflows for e-Science: Scientific Workflows for Grids*, chapter Pegasus: Mapping Large-Scale Workflows to Distributed Resources. Springer.

[Deelman et al., 2005] Deelman, E., Singh, G., Su, M.-H., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Vahi, K., Berriman, B., Good, J., Laity, A., Jacob, J. C., and Katz, D. S. (2005). Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237.

[Dong and Akl, 2006] Dong, F. and Akl, S. G. (2006). Scheduling algorithms for grid computing: state of the art and open problems. Technical Report 2006-504, School of Computing, Queen's University, Kingston, Ontário, Canadá.

[Fahringer et al., 2007] Fahringer, T., Prodan, R., Duan, R., Hofer, J., Nadeem, F., Nerieri, F., Podlipnig, S., Qin, J., Siddiqui, M., Truong, H.-L., Villazón, A., and Wieczorek, M. (2007). ASKALON: A Development and Grid Computing Environment for Scientific Workflows. In Taylor, I. J., Deelman, E., Gannon, D. B., and Shields, M., editors, *Workflows for e-Science*, chapter 27, pages 450–471. Springer.

[Fard et al., 2012] Fard, H. M., Prodan, R., Barrionuevo, J. J. D., and Fahringer, T. (2012). A multi-objective approach for workflow scheduling in heterogeneous environments. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (Ccgrid 2012)*, CCGRID '12, pages 300–309, Washington, DC, USA. IEEE Computer Society.

[Foster and Kesselman, 2003] Foster, I. and Kesselman, C. (2003). *The Grid 2: Blueprint for a new computing infrastructure*. Elsevier.

[Foster and Kesselman, 1996] Foster, I. T. and Kesselman, C. (1996). Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128.

[Garey and Johnson, 1979] Garey, M. R. and Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company.

[Graham, 1966] Graham, R. L. (1966). Bounds for certain multiprocessing anomalies. *Bell System Technical Journal*, 45(9):1563–1581.

[Graham et al., 1979] Graham, R. L., Lawler, E. L., Lenstra, J. K., and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics*, 5:287–326.

[Leung, 2004] Leung, J. Y. T., editor (2004). *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapman & Hall/CRC Press, Boca Raton, FL, USA.

[Lin et al., 2009] Lin, C., Lu, S., Fei, X., Chebotko, A., Pai, D., Lai, Z., Fotouhi, F., and Hua, J. (2009). A reference architecture for scientific workflow management systems and the VIEW SOA solution. *IEEE Transactions on Services Computing*, 2(1):79–92.

[Ludäscher et al., 2006] Ludäscher, B., Altintas, I., Berkley, C., Higgins, D., Jaeger, E., Jones, M., Lee, E. A., Tao, J., and Zhao, Y. (2006). Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice and Experience*, 18(10):1039–1065.

[Ludäscher et al., 2009] Ludäscher, B., Weske, M., McPhillips, T., and Bowers, S. (2009). Scientific workflows: Business as usual? In Dayal, U., Eder, J., Koehler, J., and Reijers, H., editors, *Business Process Management*, volume 5701 of *Lecture Notes in Computer Science*, pages 31–47. Springer Berlin Heidelberg.

[Mell and Grance, 2011] Mell, P. M. and Grance, T. (2011). The NIST definition of cloud computing. Technical Report SP 800-145, National Institute of Standards & Technology, Gaithersburg, MD, United States.

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

[Russell et al., 2006] Russell, N., ter Hofstede, A., van der Aalst, W., and Mulyar, N. (2006). Workflow control-flow patterns: A revised view. Technical Report BPM-06-22, BPM Center.

[Russell et al., 2005] Russell, N., ter Hofstede, A. H. M., Edmond, D., and van der Aalst, W. M. P. (2005). Workflow data patterns: Identification, representation and tool support. In *Proceedings of the 24th International Conference on Conceptual Modeling*, ER'05, pages 353–368, Berlin, Heidelberg. Springer-Verlag.

[Tan et al., 2009] Tan, W., Missier, P., Madduri, R., and Foster, I. T. (2009). Building scientific workflow with Taverna and BPEL: A comparative study in caGrid. In Feuerlicht, G. and Lamersdorf, W., editors, *Service-Oriented Computing - ICSOC 2008 Workshops*, pages 118–129. Springer.

[Topcuoglu et al., 2002] Topcuoglu, H., Hariri, S., and Wu, M.-Y. (2002). Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Transactions on Parallel and Distributed Systems*, 13(3):260–274.

[van der Aalst et al., 2003] van der Aalst, W. M. P., ter Hofstede, A. H. M., Kiepuszewski, B., and Barros, A. P. (2003). Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51.

[Voorneveld, 2003] Voorneveld, M. (2003). Characterization of pareto dominance. *Operations Research Letters*, 31(1):7–11.

[WfMC, 1999] WfMC (1999). Workflow management coalition – terminology & glossary. Technical Report WfMC-TC-1011, Workflow Management Coalition, Winchester, UK.

[Wieczorek et al., 2009] Wieczorek, M., Hoheisel, A., and Prodan, R. (2009). Towards a general model of the multi-criteria workflow scheduling on the grid. *Future Generation Computer Systems*, 25(3):237–256.

[Wieczorek et al., 2005] Wieczorek, M., Prodan, R., and Fahringer, T. (2005). Scheduling of scientific workflows in the ASKALON grid environment. *ACM SIGMOD Record*, 34(3):56–62.

[Wolstencroft et al., 2013] Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., Bhagat, J., Belhajjame, K., Bacall, F., Hardisty, A., Nieva de la Hidalga, A., Balcazar Vargas, M. P., Sufi, S., and Goble, C. (2013). The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic Acids Research*, 41(W1):W557–W561.

[Yu and Buyya, 2005] Yu, J. and Buyya, R. (2005). A taxonomy of scientific workflow systems for grid computing. *SIGMOD Record*, 34(3):44–49.

[Yu et al., 2008] Yu, J., Buyya, R., and Ramamohanarao, K. (2008). Workflow scheduling algorithms for grid computing. In Xhafa, F. and Abraham, A., editors, *Metaheuristics for Scheduling in Distributed Computing Environments*, pages 173–214. Springer.

[Zheng and Sakellariou, 2013] Zheng, W. and Sakellariou, R. (2013). Budget-deadline constrained workflow planning for admission control. *Journal of Grid Computing*, 11(4):633–651.