VARIAÇÕES NO DESENHO DE WORKFLOWS CIENTÍFICOS VISANDO AO AUMENTO DE DESEMPENHO EM EXECUÇÕES EM LARGA ESCALA

Silvia Benza Bareiro

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

Orientadoras: Marta Lima de Queirós Mattoso
Kary Ann del Carmen Ocaña Gautherot

Rio de Janeiro

Julho de 2017

VARIAÇÕES NO DESENHO DE WORKFLOWS CIENTÍFICOS VISANDO AO AUMENTO DE DESEMPENHO EM EXECUÇÕES EM LARGA ESCALA

Silvia Benza Bareiro

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

Examinada por:

_____
Prof. Marta Lima de Queirós Mattoso, D.Sc.

_____
Prof. Kary Ann del Carmen Ocaña Gautherot, D.Sc.

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

_____
Prof. Leonardo Gresta Paulino Murta, D.Sc.

_____
Prof. Carla Osthoff Ferreira de Barros, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
JULHO DE 2017

Bareiro, Silvia Benza

Variações no desenho de workflows científicos visando ao aumento de desempenho em execuções em larga escala/ Silvia Benza Bareiro. – Rio de Janeiro: UFRJ/COPPE, 2017.

XII, 59 p.: il.; 29,7 cm.

Orientadoras: Marta Lima de Queirós Mattoso

Kary Ann del Carmen Ocaña Gautherot

Dissertação (mestrado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2017.

Referências Bibliográficas: p. 60-64.

1. *Workflows* Científicos. 2. Álgebra de *Workflows*. 3. Proveniência de Dados. 4. SciCumulus. I. Mattoso, Marta Lima de Queirós *et. al.* II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Titulo

Agradecimentos

Após finalizar esta etapa, gostaria de agradecer principalmente aos meus pais, Carmen e Francisco, sem eles nada disto seria possível. Apesar da distância nunca cansaram nem desanimaram em me oferecer todo o apoio que eu precisei nesta empreitada. Agradeço também aos meus irmãos, Francisco, Bruno Ale e Martín, pelo amor incondicional.

Esta dissertação foi um desafio, conquistado com muita ajuda dos professores e colegas do PESC. Gostaria de agradecer especialmente às minhas orientadoras. Professora Marta, foi um prazer trabalhar com a senhora ao longo do mestrado, agradeço eternamente pela ajuda e dedicação, sempre me guiando para realizar um melhor trabalho. Professora Kary, obrigada pela paciência e empenho, poder trabalhar junto a senhora foi um privilégio.

A todos os amigos dentro e fora da universidade, obrigada pela amizade e por manter meu espírito em forma, vocês são a verdadeira maravilha desta cidade. Dentre todas as pessoas maravilhosas que me ajudaram, Thais e Vanessa, obrigada por tudo o que fizeram para este mestrado concluir. Aos meus colegas Renan e Victor, conseguimos!

À CAPES, pela concessão da bolsa de mestrado entre 2014 e 2016 que me permitiram realizar o trabalho.

Agradeço.

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

VARIAÇÕES NO DESENHO DE WORKFLOWS CIENTÍFICOS VISANDO AO AUMENTO DE DESEMPENHO EM EXECUÇÕES EM LARGA ESCALA

Silvia Benza Bareiro

Julho/2017

Orientadoras: Marta Lima de Queirós Mattoso

Kary Ann del Carmen Ocaña Gautherot

Programa: Engenharia de Sistemas e Computação

Experimentos científicos em larga escala são considerados complexos devido à própria modelagem na definição do *workflow* ou por envolver a manipulação de dados científicos volumosos e heterogêneos. A execução de *workflows* científicos em ambientes paralelos e distribuídos é requerida, mas também é preciso registrar a proveniência desses experimentos. Os dados de proveniência, quando acrescidos de dados de desempenho da execução do *workflow* permitem uma noção sobre os custos computacionais tanto do *workflow* total como das suas atividades e dos programas atrelados a elas. Dessa maneira podem ser avaliadas melhoras potenciais no desempenho do *workflow* e comparar alternativas de desenho, caso o *workflow* apresente variações que levem à diminuição no tempo e custo computacional da execução. Esta dissertação tira proveito do desenho de *workflows* por meio de linhas de experimentos para recomendar variabilidades de desenhos que visam melhorias no desempenho. A abordagem utilizada foi baseada em técnicas de otimização de consultas junto com informações retiradas da base de dados de proveniência por meio do uso de sistemas de gerência de *workflows* científicos. Técnicas como fragmentação de dados a serem consumidos e implementação de filtros com o fim de reduzir os dados antes do processamento são propostas como variabilidades na linha de experimento. Com esta representação fazendo parte da linha de experimentos é possível recomendar ao cientista derivações que possam levar a uma redução no tempo de execução do *workflow*.

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

VARIATIONS IN THE DESIGN OF SCIENTIFIC WORKFLOWS AIMING AT INCREASING PERFORMANCE IN LARGE-SCALE EXECUTIONS

Silvia Benza Bareiro

July/2017

Advisors: Marta Lima de Queirós Mattoso

Kary Ann del Carmen Ocaña Gautherot

Department: Systems and Computer Engineering

Large-scale scientific experiments are considered complex due to the workflow design during the specification and because it involves the manipulation of massive and heterogeneous scientific data. The execution of scientific workflows in parallel and distributed environments is required, but it is also necessary to manage the provenance data of these experiments. The provenance data, when enhanced with performance data of the workflow execution, allows a notion about the computational costs of both the workflow as a whole and its activities or programs related. In this way, potential improvements in the workflow performance can be evaluated and design alternatives can be compared, in cases when the workflow presents variations that lead to a decrease in execution time or computational cost. This dissertation takes advantage of the design of workflows through experiment lines to recommend variabilities in the designs, aiming to improve performance. The approach was based on query optimization techniques with provenance data information extracted from databases through scientific workflow management systems. Techniques such as data fragmentation and the filter are proposed as variabilities in the experiment line. With this representation as part of the experiment line, it is possible to recommend to scientists several derivations that lead to a reduction of the workflow execution time.

Índice

Índice de Figuras

Índice de Tabelas

LISTA DE SIGLAS

PAD – Processamento de Alto Desempenho

SGWfC – Sistema de Gerência de *Workflows* Científicos

NGS – Da sigla em inglês, *Next-Generation Sequencing*,

DAG – Grafo Acíclico dirigido, da sigla em inglês, *Directed Acyclic Graph*

XML – Da sigla em inglês, *eXtensible Markup Language*

NCBI – Da sigla em inglês, *National Center for Biotechnology Information*

UniProt – Da sigla em inglês, *Universal Protein Resource*

PDB – Da sigla em inglês, *Protein Databank*

PERL – Da sigla em inglês, *Practical Extraction and Report Language*

SSH – Da sigla em inglês, Secure SHell

UFRJ – Universidade Federal do Rio de Janeiro

COPPE – Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia

NIH - Da sigla em inglês, *National Institutes of Health*

Capítulo 1 - Introdução

Um experimento científico pode ser visto como "uma situação criada em laboratório que visa a observar, sob condições controladas, o fenômeno de interesse" (Jarrard 2001). Nas últimas décadas, com a evolução da computação, novos tipos de experimentos científicos baseados em simulação (Deelman *et al*. 2009) têm sido amplamente explorados.

Pesquisas desenvolvidas em ambientes acadêmicos para o tratamento de grandes quantidades de dados (*big data*) (Bertino *et al*. 2011, Lynch 2008) realizam a exploração e análise, em determinados casos, por meio de experimentos científicos em larga escala.

Um cientista deve cumprir inúmeras atividades para satisfazer os requisitos de um experimento científico simulado e algumas destas atividades podem representar o encadeamento de programas usados durante tais simulações. Modelar o encadeamento destes programas não é trivial: na realidade pode se tornar uma tarefa tão complexa que é vista como uma barreira técnica para alcançar modelos mais sofisticados (Gil *et al*. 2007). Também, para realizar o processamento de grandes quantidades de dados em um tempo razoável, o cientista deve ser capaz de realizar a simulação do experimento utilizando técnicas de paralelismo/distribuição em ambientes de processamento de alto desempenho (PAD). Ainda assim, o volume de dados pode ser tão grande que mesmo em ambientes de PAD o experimento científico pode levar dias ou semanas, o que pode sugerir outro tipo de otimizações tanto na modelagem como na execução.

Workflows científicos são utilizados como uma abstração para desenhar o fluxo de atividades e de dados em um experimento de maneira estruturada. Segundo Taylor (Taylor 2007a), os *workflows* se tornaram um padrão para a modelagem de experimentos na comunidade científica. Eles podem ser definidos como a especificação formal de um processo científico que representa os passos a serem executados em um determinado experimento científico (Deelman *et al*. 2009). Cada atividade corresponde à invocação de um programa, e as dependências representam o fluxo de dados entre as atividades envolvidas na execução. Portanto, o cientista consegue modelar e executar experimentos científicos por meio de simulações computacionais complexas em larga escala em ambientes de PAD por meio dos *workflows* científicos.

Sistemas de Gerência de *Workflows* Científicos (SGWfC) (Deelman *et al*. 2009) foram desenvolvidos para apoiar a execução de *workflows* científicos, estes sistemas permitem que cientistas especifiquem os experimentos científicos como *workflows*, incluindo as suas atividades e dependências de dados. Os SGWfC são os responsáveis pela coordenação das ativações dos programas; eles proveem a infraestrutura para o desenho, execução e monitoramento de *workflows* científicos e são utilizados amplamente em distintos campos de estudos como a Biologia, Física, Química, Ecologia, Geologia, Astronomia e Engenharias em geral (Cavalcanti *et al*. 2005). Existem inúmeras soluções de SGWfC, cada uma apresentando uma representação específica para o desenho e execução de *workflows*, dentre elas podem-se citar: o VisTrails (Callahan *et al*. 2006), o Kepler (Altintas *et al*. 2004), o Taverna (Hull *et al*. 2006), o Pegasus (Deelman *et al*. 2007), o Swift (Zhao *et al*. 2007), o Triana (Taylor *et al*. 2007b) e o SciCumulus/C^2 (Silva *et al*. 2014).

Em contrapartida, o apoio dado ao cientista pelos SGWfC atuais no levantamento e organização das atividades na especificação e desenho do experimento científico ainda são limitados. Desta maneira, o cientista deve desenhar um *workflow*, reaproveitando ou se inspirando a partir de *workflows* similares, ou inclusive começar a modelagem de zero. Alguns SGWfC, como Pegasus, Swift, Triana e SciCumulus/C^2 possuem a capacidade de criar um plano de execução eficiente, com paralelismo, levando em consideração as dependências entre as distintas atividades especificadas, no entanto, cabe ao cientista elaborar a estrutura do *workflow* definindo quais atividades fazem parte do *workflow*, o encadeamento dessas atividades e o fluxo de dados. No entanto, alternativas mais eficientes que demandem o conhecimento do cientista, não podem ser automaticamente incorporadas ao plano de execução gerado pelo SGWfC. Por exemplo, o cientista escreve um programa que é executado para cada dado de entrada de um determinado arquivo. O cientista sabe que essas execuções do programa podem ser executadas em paralelo, de modo independente. No entanto, o SGWfC não tem essa informação, nem saberia como ler o arquivo para obter os dados separadamente para então invocar as execuções paralelas do programa. Em outras situações, alguns programas são bem mais rápidos que outros, porém, menos seletivos, por exemplo. Caso o cientista não tenha escolhido o programa mais rápido, o SGWfC não pode fazer a substituição sem a autorização do usuário. Por outro lado, o cientista pode não saber que essa alternativa poderia ter sido usada no desenho do *workflow*. Por estes motivos, a

especificação do *workflow* pelo cientista é um ponto fundamental a ser definido, pois dependendo dessa especificação, o tempo de execução demandado pode se tornar muito custoso. Este fato se torna um desafio, pois o cientista se depara com a necessidade de pesquisar sobre novas técnicas tanto de otimização e paralelismo, ou inclusive programas similares que levem à redução do tempo de execução, focando em conservar as mesmas características qualitativas (ou similares) nos resultados. Cabe ressaltar que, inclusive para os cientistas que trabalham com *workflows* científicos, o conhecimento de quais atividades estão relacionadas entre si ainda é tácito, o que leva à necessidade de executar um número alto de simulações de diferentes fluxos de atividades (Ogasawara 2009b).

Em decorrência disto, novas configurações e estruturas de *workflows* dentro do mesmo experimento podem ser desenhadas, o que dificulta o seu controle mesmo através dos SGWfC; isto induz a que o próprio cientista tente gerenciar as variações desse experimento manualmente. Este problema se intensifica, quando o desenho do *workflow* visa a melhora no desempenho, devido ao fato de que são dois fatores que devem ser gerenciados, a modelagem e a execução paralela. Um fato a ser tomado em conta é que cientistas das áreas de aplicação que trabalham com *workflows* científicos podem não ter um conhecimento abrangente de tecnologias e ambientes de PAD, pelo que focam inicialmente no desenho semântico do experimento ao invés da execução paralela. Por último, os cientistas se deparam com um novo cenário de complexidade, além de realizar o desenho e a execução dos *workflows* em ambientes de PAD, eles devem gerenciar a variabilidade desses *workflows* (análogos) gerados para a mesma solução.

Para minimizar esse problema Ogasawara et. al (2009) definiram o conceito de linhas de experimentos (LE) que visam a organizar as distintas estruturas dos *workflows* científicos baseados nos conceitos de linhas de produtos de *software* (SPL da sigla em inglês *software product lines*) (Northrop 2002). Uma linha de experimento pode ser definida como o encadeamento de atividades dentro de um *workflow* abstrato, em que cada atividade representa um componente desta linha. Cada atividade abstrata pode representar uma série de atividades concretas, isto é, uma lista de programas específicos que realizam a função equivalente ou similar dentro do componente, o que é chamado de variabilidade. Assim, uma atividade opcional é a atividade abstrata que pode ser incluída ou suprimida do *workflow* derivado da linha de experimento e uma atividade mandatória é a atividade abstrata que não pode ser removida do *workflow* derivado,

devido a que precisa estar presente em todas as derivações possíveis da linha. A partir da linha de experimento, é possível derivar *workflows* concretos (para diferentes SGWfC) optando pelas variabilidades desejadas.

Nesta dissertação, procura-se assistir o cientista no desenho de *workflows*, tendo como ênfase a redução do tempo de execução. Para tanto, a estratégia proposta tira proveito do desenho de *workflows* por meio de linhas de experimentos para recomendar variabilidades de desenhos que possam fornecer melhorias no desempenho. A abordagem utilizada se baseia em técnicas de otimização de consultas aplicadas em sistemas de gerência de *workflows* científicos. Técnicas como fragmentação de dados a serem consumidos e de implementação de filtros que reduzem os dados antes da execução são propostas como variabilidades na linha de experimento. Com esta representação fazendo parte da linha de experimentos é possível recomendar ao cientista derivações que possam levar a uma redução no tempo de execução do *workflow*. A ideia é que a linha de experimento contemple, além das variações semânticas, aquelas variabilidades de *workflows* que levem a uma redução no tempo de execução. Este apoio se dará por meio de recomendações baseadas no desempenho das variabilidades dos *workflows* apoiadas por linhas de experimentos.

A aplicação deste conjunto de recomendações tem como objetivo permitir que o cientista possa explorar o experimento modelado baseado nas diferentes estruturas (variabilidades), podendo com isto, escolher uma estrutura dentre os *workflows* derivados a partir da linha de experimento. No entanto, cabe ressaltar que não existe ainda referenciado na literatura, um SGWfC que gerencie os *workflows* científicos derivados como linha de experimentos, se baseando em possíveis melhorias no desempenho. Portanto, todas as alterações são realizadas no momento de especificação do *workflow* abstrato e não na derivação. Além disso, esta proposta é realizada no nível de linha de experimento, o que permite que os cientistas possam utilizar qualquer SGWfC, *i.e.*, esta abordagem independe de um SGWfC específico para a posterior execução dos *workflows* derivados.

Nesta dissertação, é apresentado o experimento da Metagenômica como caso de estudo, propondo a Linha de Experimento de Metagenômica denominada SciMG, de maneira a validar as recomendações propostas. O SGWfC SciCumulus/C^2 (Silva *et al*. 2014) foi usado na modelagem e gerenciamento das execuções dos *workflows* científicos

derivados do SciMG, as quais foram realizadas em ambientes de PAD, neste caso foi usado o supercomputador Lobo Carneiro (LoboC) do NACAD/UFRJ.

1.1 Caracterização do Problema

De acordo com o que foi explicitado na subseção anterior, o problema que esta dissertação propõe é:

"Apoiar o cientista na derivação de workflows científicos que visem a melhoria de desempenho, de maneira que os workflows mantenham uma semântica coerente e baseada em linha de experimentos"

1.2 Hipótese

Como hipótese geral desta dissertação é possível tirar proveito de linhas de experimento (Ogasawara 2009a) para adicionar alternativas de derivação de *workflows* que obedeçam às regras propostas para a linha. Tais alternativas visam a redução de dados por meio de opcionalidades como a inserção de filtros e substituição de execuções sequenciais por paralelismo de dados. Devido a que os *workflows* científicos são centrados em dados por natureza, a redução e paralelização nos dados a serem processados indicam oferecer uma melhoria de desempenho. Tais alterações devem permitir que os *workflows* derivados se mantenham semanticamente corretos, porém gerando ganhos no tempo de execução.

Linhas de experimentos vêm ganhando aceitação no apoio ao desenho de *workflows,* no entanto, não foram encontrados trabalhos que incorporam alternativas de *workflows* com ênfase em desempenho e ambientes PAD. Apesar de os SGWfC permitirem a execução paralela de *workflows*, esta execução está limitada ao paralelismo que pode ser gerado automaticamente pela máquina de execução. Em algumas situações, o cientista especifica o *workflow* para consumir um conjunto de dados dentro de uma estrutura de arquivo caixa preta, devido a que o SGWfC não tem como fragmentar esses dados para realizar a respectiva execução paralela. Quando as atividades necessárias para o paralelismo de dados são definidas como alternativas ao *workflow* "sequencial", é possível gerar um *workflow* que permita ao SGWfC gerar um plano de execução com paralelismo. Da mesma forma, ao sugerir a inclusão de um filtro, que mantenha as propriedades de correção da linha (Marinho *et. al.* 2017), o usuário diminui a quantidade de dados a serem processados e consequentemente obtém um resultado com

menos dados processados e com um menor tempo de execução. Tal inclusão de filtro, pode até já fazer parte da linha, mas cabe ao sistema de recomendação proposto dar uma informação quanto ao potencial de redução no tempo total de execução.

Deste modo, o objetivo principal desta dissertação é propor recomendações de derivação de *workflows* científicos de forma a que exista uma melhora no desempenho do experimento como um todo, mas que não afete a qualidade dos resultados obtidos e que as possíveis alterações nos *workflows* sejam coerentes com a especificação proposta original, ou seja, que se mantenha semanticamente correto.

1.3 Organização da Dissertação

Esta dissertação está organizada em seis capítulos, incluindo este capítulo introdutório. Ao longo do Capítulo 2 são apresentados conceitos importantes para a compreensão desta dissertação, como experimentos científicos, *workflows* científicos, álgebra de *workflows* científicos, linha de experimento e proveniência de dados. O Capítulo 3, referente à abordagem proposta, desenvolve as recomendações aplicáveis no desenho do experimento científico para apoiar a execução paralela de *workflows* científicos. O Capítulo 4 apresenta a linha de experimento SciMG proposta como caso de estudo nesta dissertação, e o Capítulo 5 apresenta as avaliações experimentais realizadas e os resultados obtidos. Finalmente, o Capítulo 6 apresenta as conclusões e contribuições.

Capítulo 2 - Fundamentação Teórica

Este Capítulo apresenta os conceitos referentes a experimentos científicos suportados por meio de *workflows* científicos, de maneira a expor a fundamentação teórica necessária para a compreensão desta dissertação. A Seção 2.1 apresenta o conceito de experimentos científicos baseados em simulação. A Seção 2.2 define o ciclo de vida de um experimento científico. A Seção 2.3 apresenta o conceito de *workflows* científicos. A Seção 2.4 define a álgebra de *workflows* científicos. A Seção 2.5 apresenta o SciCumulus/C^2, um sistema de gerência de *workflows* científicos. A Seção 2.6 apresenta o conceito de linha de experimento. A Seção 2.7 discute a proveniência de dados no contexto de *workflows* científicos. Por fim, a Seção 2.8 expõe os trabalhos relacionados.

2.1 Experimentos Científicos Baseados em Simulação

Experimentos científicos podem ser definidos como "testes realizados em ambientes controlados para demonstrar a veracidade de um fato, examinar a validade de uma hipótese ou determinar a eficácia de algo ainda não testado" (Soanes e Stevenson 2003). Assim, um experimento científico é "a modelagem de etapas a serem executadas para produzir um determinado resultado" (Mattoso *et al*. 2009).

A exploração de experimentos costuma ser realizada em bancadas de laboratórios ou em ambientes reais, contudo a utilização de recursos computacionais tem se tornado mais comum para a realização de pesquisas científicas (Deelman *et al.*2009). Existem diversas classificações para os experimentos científicos, a saber: *in vivo, in vitro, in virtuo* e *in silico* (Travassos e Barros 2003). A categoria de interesse para esta dissertação é a composta pelos experimentos científicos baseados em simulação nomeados *in silico,* termo utilizado para referenciar aqueles experimentos que são simulados em ambientes computacionais. Segundo Pereira e Travassos (2009), a exploração do mundo real por meio de simulações pode trazer benefícios como um maior controle do ambiente, diminuir o custo associado aos esforços dos cientistas necessário para realizar a pesquisa, assim como prever possíveis complicações decorrentes da mesma.

Para Travassos e Barros (2003), experimentos científicos baseados em simulação, ou *in silico,* são caracterizados pela ausência de interação humana na realização dos mesmos, isto é, tanto o ambiente como os sujeitos da pesquisa são tratados como modelos

computacionais. Esta abstração é utilizada nos mais diversos domínios científicos: bioinformática (Ocaña *et al*. 2013, Oliveira *et al*. 2011, 2013), estudos fisiológicos (Porto *et al*. 2011), prospecção de petróleo em águas profundas (Martinho *et al*. 2009), astronomia (Ball e Brunner 2009), dinâmica de fluidos computacional (Guerra e Rochinha 2009, Guerra *et al*. 2009, 2012, Lins *et al*. 2009), previsão de precipitação (Evsukoff *et al*. 2011), mapeamento dos corpos celestes (Hey *et al*. 2012), ecologia (Hartman *et al*. 2010), monitoramento aquático (Pereira e Ebecken 2011), entre vários outros.

Estes experimentos científicos são modelos computacionais representados através do encadeamento de programas, onde uma grande quantidade de dados é produzida e consumida ao longo da simulação. Cada programa é executado com parâmetros e dados de entrada específicos, obtendo resultados com semântica e sintaxe definidos.

Nos mais diversos domínios científicos, estes experimentos podem também se caracterizar, entre outras coisas, como experimentos em larga escala devido à exploração de grandes quantidades de dados (*big data*) (Bertino *et al*. 2011, Lynch 2008).

Experimentos científicos em larga escala possuem um gerenciamento especialmente complexo devido à manipulação e produção de grandes quantidades de dados e à exploração de diferentes programas envolvidos na simulação, o que leva a um alto custo computacional e de tempo (Oliveira 2012). Isto torna necessária a utilização de ambientes de processamento de alto desempenho (PAD) como *clusters*, grades computacionais, supercomputadores e nuvens de computadores.

2.2 Ciclo de Vida do Experimento Científico

Existem diferentes modelos para representar o ciclo de vida de um experimento científico; nesta dissertação utilizaremos o proposto por Mattoso *et al*. (2010b). O ciclo de vida de um experimento científico descreve as tarefas que um cientista deve realizar no momento da implementação, execução e interpretação de dados de um experimento.

O modelo proposto por Mattoso *et al*. (2010b) possui três fases principais: composição, execução e análise, como apresentadas na Figura 1 e que serão melhor detalhadas nos parágrafos subsequentes. Cada fase possui uma subfase autônoma que aparece em momentos distintos do curso do experimento. Esse modelo foi criado com a finalidade

de assistir no gerenciamento da informação manipulada ao longo do experimento científico.

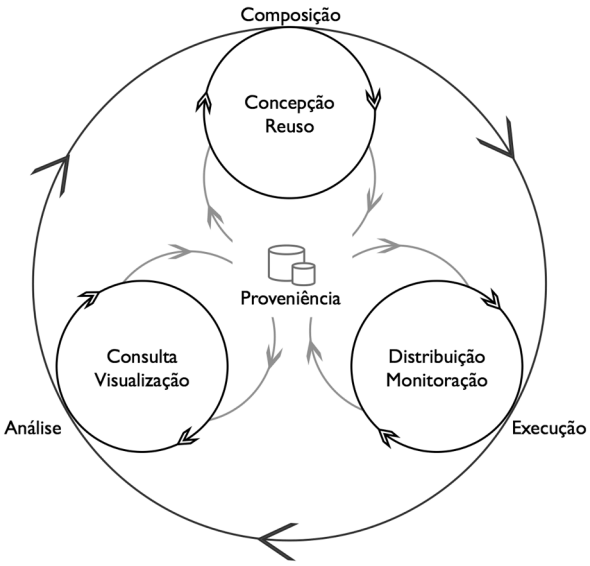


Figura 1 - Ciclo de vida do experimento científico. Adaptado de Mattoso *et al.* (Mattoso *et al.* 2010b).

A primeira fase, chamada de composição, possui um alto nível de abstração e é responsável pela estruturação e criação de todo o experimento. Nesta fase é realizada a definição, edição e manipulação dos *workflows* abstratos (definidos na seção seguinte). Mais especificamente, é nesta fase que é estabelecida a sequência lógica das atividades, os tipos de dados das entradas, os parâmetros que devem ser fornecidos e os tipos de dados de saída que serão gerados. Esta fase, por sua vez, possui duas subfases: concepção e reuso. A concepção se encarrega da criação e estruturação dos experimentos a serem definidos e modelados como *workflows*. Por outro lado, o reuso serve como apoio à primeira subfase, utilizando experimentos previamente criados que podem ajudar na criação de outros experimentos, permitindo a realização de adaptações ou simplesmente preparando o experimento para ser executado novamente (Cardoso *et al*. 2002, Ogasawara *et al*. 2009, Oliveira *et al*. 2010c, 2008).

Na fase de execução, o *workflow* abstrato desenhado ou adaptado na fase anterior é consolidado, transformando em um *workflow* concreto e executável em determinadas infraestruturas computacionais, tais como *clusters*, grades e nuvens de computadores (Dantas 2005). Ao longo desta fase são definidos os valores exatos dos parâmetros e listados os dados de entrada para cada execução. Esta fase é a responsável também pelo armazenamento de todas as informações da execução que poderão ser utilizadas nas

demais fases. A fase de execução por sua vez pode ser decomposta em distribuição e monitoramento. A distribuição organiza as atividades a serem executadas, atendendo a necessidade da execução em ambientes PAD, escolhendo quais atividades serão executadas em paralelo e distribuindo as cargas de cada máquina no ambiente distribuído. Já o monitoramento está encarregado de gerenciar as execuções, conferindo e recompilando informações do estado da execução, ou seja, verifica periodicamente o estado atual da execução do experimento. Sendo assim, sabendo que algumas execuções podem ser demoradas, levando até meses para concluir, o monitoramento auxilia na verificação do andamento do experimento.

Na terceira fase de análise são utilizados os dados obtidos nas fases anteriores para assim realizar inferências com base nos mesmos. Esta etapa possui duas subfases: consulta e visualização. Na subfase de consulta o cientista realiza consultas nos dados de proveniência. Os dados de proveniência contêm informações tanto da execução do experimento como dos resultados obtidos pelo mesmo. Na visualização, o cientista pode optar por realizar a análise através de gráficos ou mapas que resumam as informações das consultas. Com estas informações os cientistas podem confirmar ou refutar as hipóteses levantadas na criação do experimento; recomeçar um novo ciclo com a possibilidade de somente realizar mudanças nos parâmetros para verificar o seu funcionamento em diferentes cenários ou recriar todo o experimento com a geração de até novos *workflows*.

2.3 *Workflows* Científicos

Workflow é definido como "a automação de um processo de negócio, completo ou apenas parte dele, através do qual documentos, informações ou tarefas são transmitidos de um participante a outro por ações, de acordo com regras procedimentais" pelo *Workflow Management Coalition* (WfMC) (WfMC 2009).

Os *workflows* científicos são as abstrações que modelam e permitem o gerenciamento dos experimentos científicos de maneira estruturada (Mattoso *et al*. 2010b). Um *workflow* científico também pode ser definido como a especificação formal de um processo científico que representa os passos a serem executados em um determinado experimento científico (Deelman *et al*. 2009). Ao longo dos últimos anos os *workflows* científicos se tornaram um padrão para a modelagem de experimentos científicos que são baseados em simulação computacional (Mattoso *et al*. 2010b).

A estrutura do *workflow* é definida pelas atividades que representam os artefatos, programas e *scripts* a serem executados, onde as atividades estão entrelaçadas dentro do fluxo de execução do *workflow*. Nesta estrutura, as atividades geram resultados intermediários que podem servir como dados de entrada em uma ou mais atividades subsequentes.

O *workflow* científico pode ser definido em dois níveis de abstração: o abstrato e o concreto. O *workflow* abstrato é utilizado para descrever o experimento em alto nível, onde o foco está na estrutura do *workflow*, observando por exemplo, as atividades a serem executadas e sua ordenação, verificando a existência de dependências entre elas, sem a preocupação de detalhes do planejamento de sua execução.

Já um *workflow* concreto especifica os programas a serem utilizados por cada atividade, definindo os parâmetros e dados de entrada consumidos pela atividade, assim como os recursos computacionais a serem utilizados no momento da execução.

O *workflow* pode ser visto como um grafo acíclico dirigido (DAG, da sigla em inglês *Directed Acyclic Graph*) (Cormen *et al*. 2009) definido como *W(A, Dep)*, em que os nós ($A=\{a_1, a_2, ..., a_n\}$) correspondem a todas as atividades do *workflow* a ser executado, isto é, representam a invocação de programas ou *scripts*. A execução de cada atividade pode ser sequencial ou paralela e as arestas (*Dep*) estão associadas à dependência de dados entre as atividades do conjunto A.

Portanto, para cada atividade $a_i \mid (1 \leq i \leq n)$, é considerado como $I = \{i_1, i_2, .., i_m\}$ o conjunto possível de dados de entrada, então $Input(a_i) \supset I$. Também, é considerado O como sendo o conjunto possível de dados de saída produzidos por a_i, o que indica que $Output(a_i) \supset O$. Uma atividade especifica a_i é modelada como um a_i (*time*) onde *time* é o tempo total de execução de a_i. A dependência entre duas atividades é modelada como $dep(a_i, a_j, ds) \leftrightarrow \exists\ O_k \in Input(a_j) \mid O_k \in Output(a_i)$ e ds é o volume de dados transferidos entre essas duas atividades, independente da unidade de medida utilizada.

2.4 Álgebra de *Workflows*

A álgebra, como parte da matemática, estuda a combinação de elementos e figuras para representar números, de maneira abstrata, em formulas e equações. Esta dissertação foca na proposta apresentada por Ogasawara *et al.* (2011), a álgebra centrada em dados, que é utilizada para representar os dados e as atividades por relações e operações

algébricas. Aplicando a semântica no desenho do experimento, ela provê aos cientistas a possibilidade da reescrita de *workflows* científicos.

Na álgebra de *workflows*, os dados são uniformemente representados por meio de relações, de forma análoga ao conceito de relacionamentos no contexto do modelo relacional, e as atividades de *workflows* são regidas por operações algébricas que possuem semântica sobre produção e consumo destes dados (Ogasawara 2011). Os dados de entrada e saída de um *workflow* são análogos às tuplas na representação do modelo relacional, e as atividades executadas no *workflow* são vistas como caixas pretas encarregadas de consumir as relações de entrada e posteriormente de gerar relações de saída.

Portanto, com a álgebra de *workflows*, os dados são padronizados na forma de relações, e a semântica é adicionada às atividades que processam esses dados. Com isso, o sistema responsável pela execução do *workflow* pode interpretar as atividades que o representam, antecipando o fluxo de dados e relações a serem geradas.

As operações algébricas (ϕ) especificam a razão entre consumo e geração das tuplas por cada atividade, as operações algébricas são: *Map, SplitMap, Reduce, Filter* e *Query,* explicadas em detalhe ao longo desta seção.

Isto é:

$$\phi \in \{Map, SplitMap, Reduce, Filter, Query\}$$

Podemos representar uma dada relação de saída T com esquema τ da seguinte maneira:

$$T \leftarrow \phi\ (\ Y, \{\omega\}, \{R\})$$

onde $\{R\} = \{R_1\ , R_2\ , ..., R_n\}$ é o conjunto de relações de entrada com esquemas $\{\Re_i, .., \Re_n\}$, $\omega = \{\omega_1, \omega_2, ..., \omega_n\}$ é o conjunto opcional de operandos adicionais e Y é a atividade do *workflow*. Nós representamos $Y <\{\Re_i, .., \Re_j\}, \tau_i, UC>$ como uma atividade que consome um conjunto de relações compatíveis com os esquemas $\{\Re_i, .., \Re_n\}$ e executam uma dada unidade computacional (UC) para produzir uma relação de saída compatível com o esquema τ.

A razão de consumo e geração de tuplas por cada atividade está definida pelo operador algébrico associado à mesma. A Tabela 1 representa de maneira resumida os tipos de

atividades, e a razão de consumo e geração das operações, que são melhor detalhadas ao longo desta seção.

Operação	Tipo de Atividade Operada	Razão (Consumo/Geração de Tuplas)
Map	Programa	1:1
SplitMap	Programa	1:m
Reduce	Programa	n:1
Filter	Programa	1:[0-1]
Query	Expr. da álgebra relacional	n:m

Tabela 1 - Razão dos operadores da álgebra de *workflows*

As operações *Map, SplitMap, Reduce* e *Filter* são utilizadas para o apoio na execução de programas, que podem ser de terceiros, e que não são capazes de compreender a representação relacional de dados, de maneira intrínseca. A operação *Query* é utilizada para consumir dados da álgebra relacional.

2.4.1 Operação de Mapeamento (*Map*)

A operação mapeamento apoia atividades que, para cada tupla consumida do relacionamento R, produzem uma única tupla de saída no relacionamento *T*:

$$T \leftarrow Map\,(Count.exe, R)$$

A razão desta operação é 1:1, isto é, para cada tupla da relação R, é gerada exatamente uma na relação T:

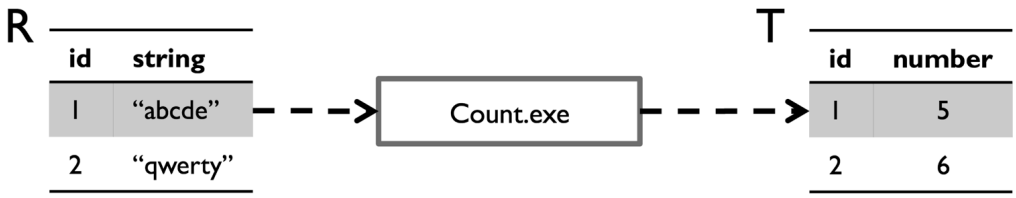


Figura 2 - Exemplo de operador *Map*

O programa Count.exe, que conta o número de caracteres de uma *string*, consome a tupla de entrada ('abcde') e gera uma tupla de saída (5). Caso exista recurso disponível para a execução em paralelo, a máquina de *workflows* pode simultaneamente instanciar outra ativação de Count.exe que, por sua vez, consome a tupla 'qwerty' para produzir a tupla 6.

2.4.2 Operação de Fragmentação de Dados (*SplitMap*)

A operação de fragmentação de dados está associada às atividades que, para cada tupla de entrada, produzem múltiplas tuplas de saída:

$$T \leftarrow SplitMap\ (SplitVogals.exe, R)$$

A razão desta operação é 1:n, isto é, para cada tupla da relação R, pode ser gerada mais de uma tupla na relação T:

Figura 3 – Exemplo de operador *SplitMap*

No exemplo apresentado na Figura 3, o programa SplitVogals.exe consome uma tupla "abcde" e separa as vogais das consoantes, gerando duas tuplas de saída: "bcd" e "ae".

2.4.3 Operação de Redução (*Reduce*)

A operação de redução também pode ser vista como uma operação de agregação, onde um conjunto de tuplas gera somente uma saída. Esta redução ou agrupamento é realizada através de um conjunto de atributos previamente definidos pelo cientista.

$$T \leftarrow Reduce\ (CountType.exe, R)$$

A razão desta operação é n:1, similar ao *Reduce* do paradigma *MapReduce* como pode ser visualizado no exemplo da Figura 4.

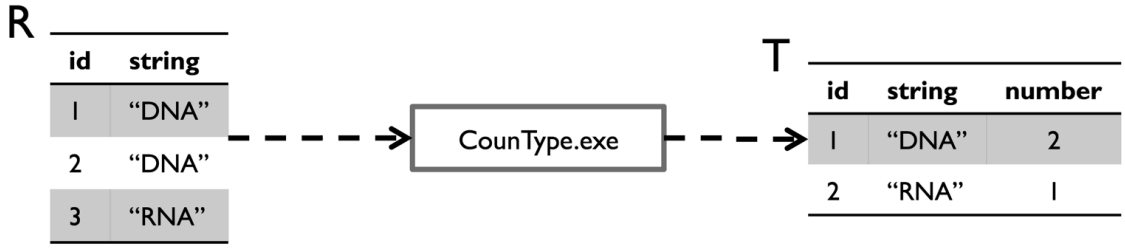


Figura 4 – Exemplo da operação *Reduce*

Na Figura 4, o programa CountType.exe, responsável por contar o número de ocorrências de cada uma das *strings* de entrada, consome três tuplas e gera duas tuplas na saída, com informações de agregação das tuplas consumidas.

2.4.4 Operação de Filtro (*Filter*)

A operação de filtro está associada a atividades que, para cada tupla de entrada, produzem uma ou nenhuma tupla de saída. Isto acontece quando a ativação gera um resultado que, dependendo de características definidas pelo cientista, tem a permanência no *dataflow* avaliado.

$$T \leftarrow Filter\ (FilterProteins.exe, R)$$

Como visto anteriormente, a razão desta operação é 1:[0-1]:

Figura 5 – Exemplo da operação *Filter*

No exemplo apresentado na Figura 5, o programa FilterProteins.exe consome três tuplas de entradas que são então filtradas, fazendo com que apenas duas delas sejam apresentadas na saída.

2.4.5 Operação de Consulta (*Query*)

Finalmente, a operação de consulta possui uma única ativação, que processa as relações de entrada como consultas tradicionais da álgebra relacional, onde a razão desta operação é n:m. O resultado obtido nesta operação é armazenado na relação de saída:

$$T \leftarrow Query\ (\Pi_{id,\ string}\ (\sigma_{evalue\ >1e\text{-}10}\ (R)), R)$$

Isto é, através de uma consulta da álgebra relacional realizada diretamente na relação R, é obtido o relacionamento T. No exemplo Figura 6 pode ser observado que somente as tuplas que possuem *evalue* acima de 1e-10 poderão continuar no *dataflow*.

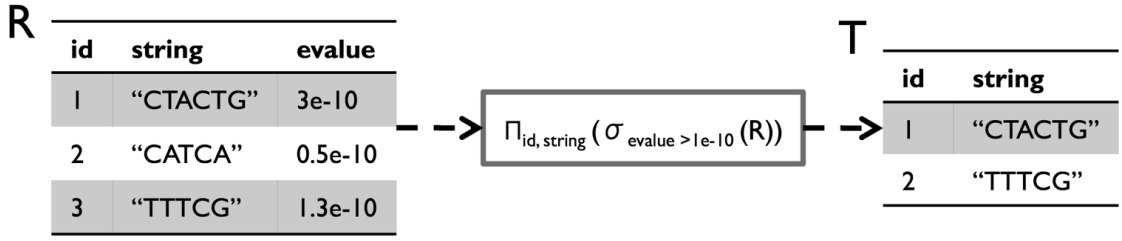


Figura 6 – Exxemplo da operação *Query*

2.5 Sistemas de Gerência de *Workflows* Científicos.

Sistemas de Gerência de *Workflows* Científicos (Deelman e Chervenak 2008) (SGWfC) são responsáveis por especificar, executar e monitorar os *workflows* científicos definidos pelos cientistas. Com isto, os cientistas podem definir a ordem das atividades executadas pelo *workflow*, assim como a definição dos parâmetros e arquivos de entrada e saída de cada uma dessas atividades. Dentre os SGWfC mais usados atualmente podemos citar: VisTrails (Callahan *et al*. 2006), Kepler (Altintas *et al*. 2004), Taverna (Hull *et al*. 2006), Pegasus (Deelman *et al*. 2007) e Swift (Zhao *et al*. 2007). Cada um destes sistemas tem um algoritmo, definição e caraterísticas próprias.

Alguns destes SGWfC permitem o monitoramento e armazenamento de dados de proveniência ao longo do ciclo de vida do experimento e não somente durante a execução. Além disso, existem SGWfC que possuem recursos para a execução paralela de atividades, os quais gerenciam com sucesso a varredura de parâmetros (Walker e Guiang 2007). Ogasawara *et al*. (2009) indica que a varredura de parâmetros é caracterizada pela execução conjunta de uma atividade específica, onde cada atividade a_i do *workflow Wf* é executada em paralelo com um conjunto de valores diferentes definidos como parâmetros de entrada.

Para esta dissertação, foi escolhido o SciCumulus/C^2 (Silva *et al*. 2014), inspirado nos SGWfC Chiron (Ogasawara *et al.* 2011) para *clusters* e SciCumulus para nuvem de computadores (Oliveira *et al*. 2010a), os quais proporcionam o gerenciamento da execução de *workflows* científicos.

O SciCumulus/C^2 foi desenvolvido para o desenho, execução e analise de *workflows* científicos, e apoia a execução paralela de *workflows* científicos quando executados em paralelo no ambiente de computação em nuvem, como Amazon EC2 (Amazon EC2 2010), e na sua versão atual em ambiente de PAD de *clusters* e grades computacionais. Ele é o responsável pelo gerenciamento e execução das atividades do *workflow* científico (ou do *workflow* como um todo), orquestrando a sua execução em um conjunto de processadores disponíveis no ambiente de PAD ou em máquinas virtuais (MV) disponíveis no ambiente em nuvem. O SciCumulus/C^2 simplifica o trabalho do cientista, pois lida com toda a complexidade que decorre do gerenciamento de execuções em paralelo, considerando a quantidade de arquivos de entrada para uma atividade, gerando uma série de tarefas para a mesma que serão executadas em paralelo

(*e.g.*, uma tarefa por *core* ou MV). A proveniência de dados (definição a ser detalhada ao longo deste capítulo) é coletada e armazenada em tempo real, oferecendo o histórico de execução do *workflow,* podendo ser utilizado para realizar consultas ao longo da execução.

O SciCumulus/C^2 possui quatro componentes ou camadas principais: cliente, distribuição, execução, e dados (Oliveira *et al*. 2010a, 2010b, 2011, 2012). A camada cliente se encarrega de distribuir as atividades a serem executadas em paralelo e seus componentes podem ser instalados nas máquinas dos cientistas. A camada de distribuição gera as tarefas a serem executadas. Já a camada de execução é a responsável pela execução destas tarefas e de todos os programas necessários no experimento. É nesta camada que os dados de proveniência são coletados. Finalmente, a camada de dados é a responsável por alocar os dados de entrada e saída (dados consumidos e gerados) durante a execução, assim como os dados de domínio do experimento. Esta camada possui toda a informação sobre o ambiente distribuído onde está sendo executado o experimento.

2.6 Linha de Experimento

Linhas de produtos de *software* (LPS) são métodos, ferramentas e técnicas da engenharia de *software* utilizados para criar uma coleção de sistemas de *software* semelhantes a partir de um conjunto compartilhado de ativos de *software* usando meios de produção comuns. (Northrop 2002). As linhas de experimento são adaptações das LPS utilizadas para representar experimentos baseados em simulação (Ogasawara *et al.*2009).

As linhas de experimento (LE), similares aos *workflows* abstratos*,* não especificam quais programas e *scripts* serão utilizados para o experimento, e sim a finalidade de cada atividade a ser executada junto com o fluxo de dados. Isto é, as linhas de experimento são o encadeamento de atividades conceituais com o maior nível de abstração. No escopo desta dissertação, estas atividades serão chamadas de Atividades da Linha de Experimento (ALE) (Marinho *et. al.* 2017). Portanto, cada ALE corresponde a um escopo a ser realizado ao longo do *workflow* abstrato.

Dentro do contexto de linhas de experimento, as definições de variabilidade e opcionalidade das ALEs são importantes para entender o comportamento de cada componente. Variabilidade, ou ponto de variação, é o mapeamento de uma ALE em

mais de uma atividade abstrata, característica que define a magnitude da linha de experimento. Por exemplo, uma ALE variável é mapeada em três atividades abstratas, aonde cada atividade abstrata virá a representar um programa, *script* ou algoritmo alternativo que cumpre a mesma finalidade que a ALE correspondente.

Por outra parte, uma ALE é vista como opcional quando a remoção da mesma não altera o resultado obtido pelo experimento. Este tipo de atividade pode ser, por exemplo, um algoritmo que busque a melhoria no desempenho da atividade subsequente. Com a definição de opcionalidade, a obrigatoriedade é adicionada às demais ALE, portanto as atividades não opcionais precisam estar presentes na derivação do *workflow* abstrato. Além das características de opcionalidade e variabilidade, no momento da definição da linha de experimento, o fluxo de dados também é representado, pelo que as dependências entre as atividades através das entradas e saídas de cada ALE precisam ser informadas.

Após a definição das linhas de experimento, é realizado o processo de derivação do mesmo, podendo obter como resultado um ou mais *workflows* abstratos. Este processo possui duas etapas a serem realizadas. Primeiramente, as variabilidades devem ser resolvidas, isto é, para cada ponto de variação deve ser definida qual atividade permanece após derivação. Consequentemente, caso existam atividades com opcionalidade, a escolha pela inclusão deve ser também definida.

A Figura 7 exemplifica o processo de derivação, onde a atividade B possui um ponto de variação e a atividade F tem a característica de opcionalidade. Na primeira derivação, a atividade b1 é escolhida e a opcionalidade é eliminada. Já na derivação 2, pode ser observado que a atividade B foi derivada com a variação b2 e a opcionalidade foi adicionada ao *workflow* abstrato. A partir desta derivação a atividade F fará parte obrigatoriamente do *workflow* derivado 2.

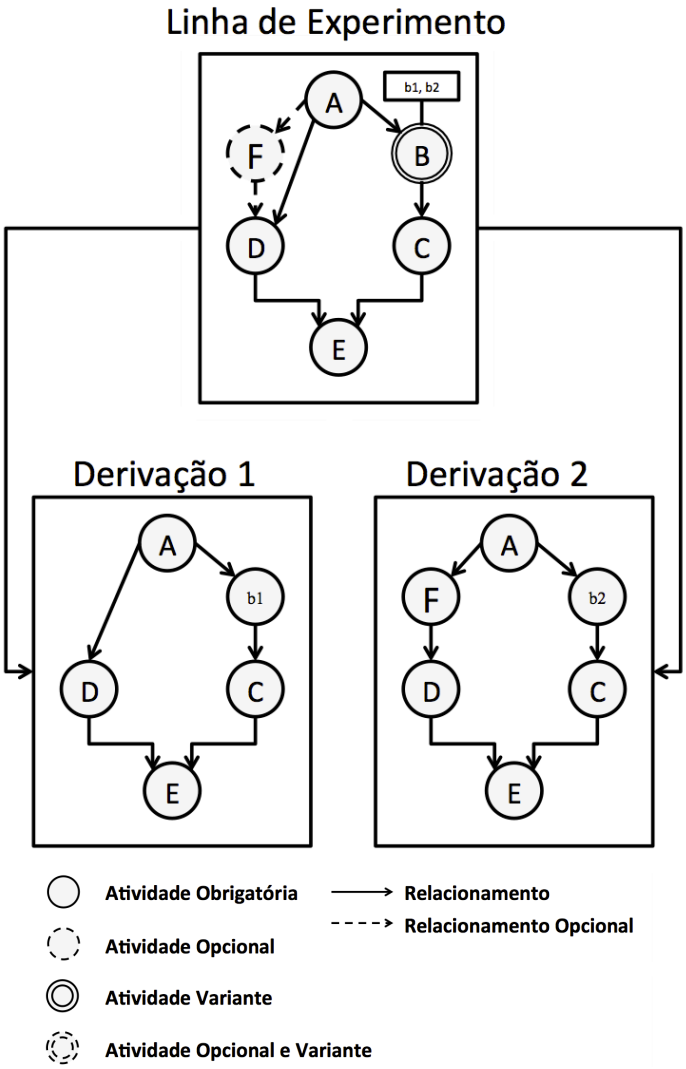


Figura 7 - Derivação de uma linha de experimento

Para obter um *workflow* concreto a partir do nível abstrato obtido na derivação é realizado o processo de instanciação. Após a instanciação do *workflow* concreto, o experimento pode ser executado em ambientes de PAD utilizando SGWfC apropriados.

2.7 Proveniência

A proveniência de dados, também conhecida como linhagem ou *pedigree*, representa a história de um objeto (Freire *et al*. 2008, Mattoso *et al*. 2010b). Explorando os experimentos científicos através da utilização de *workflows*, os metadados associados a cada *workflow* são chamados de dados de proveniência. Estes metadados são armazenados para guardar a história do experimento, registrando toda a informação ao longo do ciclo de vida do mesmo, desde sua composição até sua execução.

Existem inúmeras definições do termo "proveniência", uma delas feita pelo Dicionário Oxford de Inglês foca na origem, isto é, "o registro de propriedade de uma obra de arte, indicando o autor da mesma e todos os compradores ao longo do tempo. Tal registro é utilizado como uma guia de autenticidade ou qualidade da obra".

Dentro do contexto dessa dissertação, é fundamental lembrar que os experimentos podem ser reexecutados inúmeras vezes, dentro de condições controladas, registrando resultados similares. Alcançar tal capacidade os caracteriza como experimento científico. A reprodução de um determinado experimento é utilizada para a própria validação por parte de terceiros. Por tanto, algumas informações (*e.g.*, dados de entrada/saída, parâmetros de configuração do ambiente, execução, erros) relacionadas ao experimento precisam ser armazenadas com o intuito de serem usadas *a posteriori* nas reproduções, assim como nas análises e inferências dos resultados.

Trabalhando com *workflows* científicos, os sistemas que se encarregam do gerenciamento reúnem as informações de cada experimento, tais como informações estatísticas e erros na execução dos experimentos, assim como a interrupção por parte dos próprios pesquisadores. Estes dados de proveniência apoiam os cientistas na análise *a posteriori* do experimento, ou até na reprodução do mesmo, variando o ambiente (infraestrutura computacional) ou os parâmetros de entrada.

No contexto de *workflows* científicos, existem duas distinções para o termo proveniência: a prospectiva e a retrospectiva. A proveniência prospectiva (Davidson e Freire 2008) está interessada em coletar as informações da estrutura do *workflow*: a ordem das atividades, os tipos de parâmetros de entrada e configurações do ambiente onde será executado o experimento. Isto é, a proveniência prospectiva se encarrega de salvar os dados na fase de composição do ciclo de vida do experimento.

Por sua vez, a proveniência retrospectiva (Freire *et al*. 2008) armazena as informações ao longo da fase de execução do *workflow* concreto. A medida que cada atividade é executada, informações do tempo inicial e final, erros de execução e resultados parciais são disponibilizadas para realizar consultas. Assim, o cientista pode interromper uma execução caso o experimento não gere resultados de acordo com o esperado ou apresente erros de execução. Uma ação de interrupção por parte do pesquisador pode ser beneficial ao experimento como um todo, evitando tempo e custo computacional

excessivo em execuções que não estejam apresentado algum tipo de vantajem ao mesmo.

2.8 Trabalhos Relacionados

Em pesquisas realizadas durante o levantamento bibliográfico para esta dissertação, foi verificado que têm diversos estudos que se relacionam de alguma maneira com a otimização no desempenho de experimentos científicos centrados a dados. Existem abordagens que focam na otimização de desempenho no nível de *workflow* concreto. Entretanto, nenhuma destas abordagens realizam a otimização por meio de linha de experimento.

Por este motivo, os trabalhos a seguir foram organizados em duas categorias, as soluções existentes no nível de *workflow* concreto e no nível de linha de experimento.

Na primeira classificação, são apresentados trabalhos no nível de *workflow* concreto:

- Em Liu e Iftikhar (2015) é apresentado um *framework* baseado em fragmentação de dados que visa aumentar o paralelismo da análise de fluxo de dados ETL, do inglês *Extract Transform Load*. Liu e Iftikhar focam em otimizar as análises ETL de grande escala, propondo que seja realizada a fragmentação dos dados de entrada. Porém, este *framework* foca somente em execuções de fluxo de dados ETL.
- Em Hueske *et. al* (2012) é proposta a otimização no nível abstrato de desenho para a análise do fluxo de dados. Este protótipo utiliza uma abordagem na qual o fluxo de dados a ser analisado é especificado com funções definidas pelo usuário (UDF, do inglês *user-defined functions*) através de um DAG. Realizando uma reordenação da sequência de funções e propõe, quando possível, a fragmentação das entradas de determinadas UDF, buscando aumentar a paralelização da análise. No entanto, este otimizador somente pode ser utilizado em análises de fluxo de dados baseadas em UDF definidas no sistema *Stratosphere.*
- Deshpande e Hellerstein (2012) trabalham com otimização de consultas, eles assumem que existem *n* filtros em uma consulta, onde cada filtro será executado em um determinado nó de execução. Porém cada nó possui um limite de taxa de transferência e o foco é reordenar os filtros existentes no *workflow* da consulta buscando a maximização dessa taxa. No entanto este trabalho não apresente uma

relação direta com o objetivo desta dissertação, o mesmo mostrou-se como a utilização correta de filtros, o que poderia trazer uma melhoria no desempenho das consultas.

- Rheinländer *et. al* (2015) apresentam o otimizador SOFA para análise de fluxo de dados em larga escala. Ele foca em UDF e sua reordenação, onde são inseridas tarefas de filtro após análises que realizam seleções de dados com o intuito de reduzir a quantidade de tuplas a serem consumidas nas atividades posteriores. Este trabalho é o mais relevante em termos de similaridade ao apresentado nesta dissertação, já que o otimizador também visa o aumento de paralelização com a inserção de atividades de fragmentação de dados de entrada. Porém, este otimizador não trabalha com execuções de experimentos científicos genéricos, foi desenvolvido em cima do sistema *Stratosphere,* a proposta desta dissertação não está presa a nenhuma ferramenta de desenho de *workflow*.

Todas as abordagens mencionadas anteriormente focam na otimização de análises fluxos de dados em cima de ferramentas exclusivas, enquanto que nesta dissertação são realizadas as propostas de inserção de atividades em cima da Linha de Experimento, isto é, não está amarrada a nenhuma solução existente, definindo o experimento científico no nível mais abstrato para ser transportado para as distintas ferramentas que o cientista escolher.

É importante mencionar alguns SGWfC (seção 2.5) que trabalham com a representação e execução de *workflows* científicos apoiam a execução paralela das atividades, por exemplo tanto o Pegasus (Deelman *et al*. 2007), o SciCumulus (Oliveira *et al*. 2010a), e o Swift (Zhao *et al*. 2007), entre outros, permitem que os cientistas possam desenhar *workflows* paralelos para serem executados em ambiente de PAD, como clusters e grades, porém, esses sistemas possuem estratégias de otimização internas à máquina de execução do *workflow*. Estes SGWfC se baseiam na especificação do *workflow* concreto e desconhecem alternativas baseadas em pontos de variabilidade e opcionalidade da linha de experimento que poderiam ser aplicadas na etapa de composição do *workflow*, visando à geração de um desenho já com potencial de redução de dados e ganhos no desempenho.

Já a segunda classificação foca no desenho de linhas de experimento:

- Em Marinho *et al*. (2017) é apresentada a ferramenta GExpLine, que permite o desenho de linhas de experimentos (seção 2.6), através de uma interface gráfica. Esta ferramenta permite por sua vez, que a linha de experimento especificada nela seja derivada em *workflows* concretos.

Esta ferramenta é complementar ao trabalho proposto nesta dissertação, já que serve exclusivamente para o desenho, e não possui recomendações que visem a melhoria no desempenho do experimento.

Capítulo 3 - Recomendações para Apoiar a Execução Paralela de *Workflows* Científicos

Neste capítulo são apresentadas recomendações que buscam apoiar o cientista no momento da especificação do *workflow* científico. A motivação por trás destas recomendações foi baseada na dificuldade que os cientistas enfrentam ao desenhar a estrutura dos *workflows*, mas tendo como foco visar na melhoria no desempenho. Quando aplicadas as recomendações, são realizadas alterações diretamente na estrutura do *workflow,* inicialmente especificado pelo cientista, de maneira a obter uma linha de experimento onde os *workflows* derivados se mantem semanticamente corretos, mas procurando gerar ganhos no tempo de execução.

É importante ressaltar que esta proposta não foca no desenvolvimento de *workflows* concretos, isto é, as recomendações buscam assistir no desenho de maneira abstrata sem estarem vinculadas a algum SGWfC. As recomendações são realizadas assumindo que inicialmente o cientista especifica o *workflow* abstrato focando na semântica do experimento, de uma maneira abstrata e que poderia ser otimizada.

As recomendações foram baseadas em técnicas utilizadas na otimização de consultas em banco de dados: na utilização de filtros para diminuir o número de dados a serem analisados e na fragmentação de dados para aumentar a distribuição e paralelização dos mesmos.

Inicialmente a seção 3.1 foca na especificação do experimento, realizando o levantamento das atividades e suas dependências, propondo a procura por paralelização entre as atividades independentes do experimento, preparando o *workflow* por meio de linhas de experimentos de maneira a aplicar as recomendações. Na seção 3.2 é apresentada ao cientista uma recomendação com foco no tempo de execução e no tamanho dos dados, propondo uma paralelização maior dentro da atividade. Na seção 3.3 é apresentada uma recomendação focada no fluxo de entrada de dados ao longo das atividades da linha*,* e propõe o uso de filtros para reduzir o volume de dados processados pelas atividades.

3.1 Especificação do *Workflow* por Meio de Linhas de Experimentos

Na seção 2.3 discutiu-se que um *workflow* científico pode ser visto como um DAG. Dessa forma, as atividades do *workflow* científico possuem dependências de dados com outras atividades, definindo uma ordem para a execução dessas atividades de acordo com as restrições de dependência especificadas pelo DAG. A tese de Marinho (Marinho 2016) propõe um ferramental algébrico para a linha de experimentos e define formalmente um conjunto de condições para determinar se um *workflow* é válido, considerando que:

- Cada atividade deve estar associada a um operador algébrico (seção 2.4);
- A cardinalidade de entrada e saída de cada atividade deve estar de acordo com o operador algébrico especificado; e
- As atividades e as suas relações de dados devem obedecer às restrições do DAG (*i.e.*, dependência de dados entre atividades).

Enquanto que a paralelização dada no nível de conjunto de entradas (ou paralelismo de dados) é gerenciada e manipulada em tempo de execução por SGWfC (seção 2.5), os cientistas devem decidir a ordem e as dependências de dados entre as atividades no momento da modelagem do experimento científico como um *workflow* (antes da execução pelo SGWfC), o que inclui decisões importantes nas estratégias que proporcionarão a paralelização dos dados e da execução das atividades pelos SGWfC em ambientes de PAD. Nesse sentido, duas ou mais atividades podem realizar suas execuções concorrentemente, caso elas sejam independentes entre si, isto é, a entrada de uma atividade não depende da saída da outra.

Portanto, para modelar um *workflow* científico por meio de uma álgebra de *workflows* (Ogasawara *et al.* 2011), o cientista deve identificar o conjunto de dados de entrada I_i e o conjunto de dados de saída O_i de cada atividade a_i do conjunto A de atividades (seção 2.3). Em seguida, para cada atividade deve ser indicado o operador algébrico, por exemplo, uma atividade que consome uma única tupla de entrada, executa o programa científico e produz apenas uma tupla de saída, seria recomendado a especificação do operador *Map* para reger o comportamento dessa atividade. Mais especificamente, no caso de dependência de dados entre atividades, o conjunto de dados de saída O_i deve ser utilizado como o conjunto de dados de entrada I_j de outra atividade para expressar que a

atividade a_i depende da atividade a_j. Assim, o conjunto com as dependências de dados D deve contemplar a dependência entre a_i e a_j, ou seja, Dep(a_i, a_j).

Porém, para esta dissertação a especificação é no nível de linha de experimento (seção 2.6), para a qual o cientista deve identificar, em conjunto com as informações citadas anteriormente, as variabilidades e opcionalidades que farão parte da linha. Dado o conjunto A={a, b, c} de atividades conceituais, onde a atividade **b** é uma atividade variante, isto é, a atividade pode ser derivada em mais de uma atividade abstrata. Também é dado o conjunto de tuplas D=(Dep(a, c), Dep(b, c)), junto com o conjunto F = (IF(I_1, a), IF(I_1, b), IF(I_2, c)), onde I_2 e I_3 representam as saídas das atividades a e b respectivamente. Na Figura 8 vista a linha de experimento obtida junto com duas das derivações possíveis:

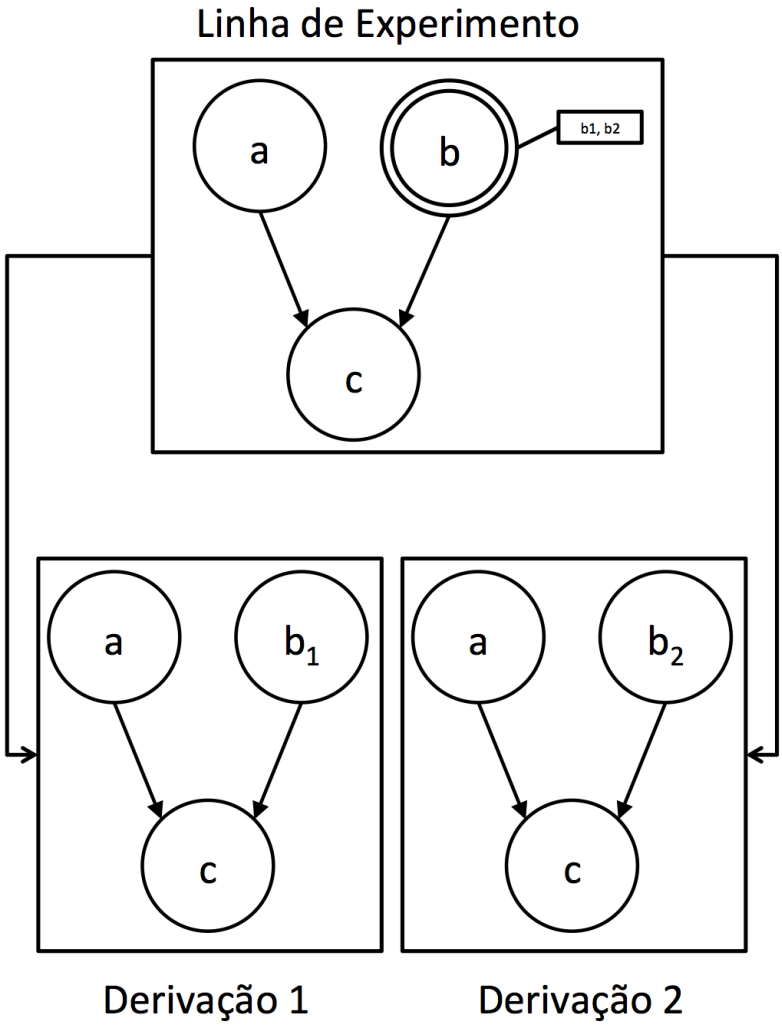


Figura 8 – Exemplo do desenho da linha de experimento e suas derivações.

Como já foi mencionado ao longo desta dissertação, existem SGWfC que criam e

gerenciam o plano de execução de *workflows* concretos quando definidos nos diferentes sistemas com suas respectivas sintaxes, porém as recomendações aqui apresentadas focam na estrutura abstrata do *workflow,* propondo as mesmas sem depender de um SGWfC específico, isto é, sem sintaxe ou algoritmos específicos.

3.2 Paralelização dos Dados

Geralmente, o cientista escolhe as ferramentas, programas ou *scripts* a serem utilizados no experimento baseando-se no conhecimento e na disponibilidade dos mesmos. Geralmente o cientista é quem possui o conhecimento sobre o uso, parametrização e o tempo médio de execução dessas ferramentas. Desta maneira, é possível ter um conhecimento *a priori* do tempo de execução para determinados tamanhos de arquivos de entrada, já para um estudo em larga escala pode ser inviável sem o conhecimento de tecnologias e ambientes de SGWfC e PAD.

Por outro lado, como já mencionado os SGWfC gerenciam a paralelização dos dados de entrada, isto é, para cada arquivo de entrada uma execução será inicializada no recurso computacional disponível. As atividades de um experimento podem possuir quantidades de entrada diferentes entre elas, e no momento da escolha da quantidade de recursos alocados para o experimento como um todo, é possível que alguns recursos computacionais fiquem ociosos enquanto executam as atividades sem dados de entrada suficientes para tirar proveito do paralelismo.

Sabendo isto, a ociosidade dos recursos computacionais pode ser aproveitada para aumentar a paralelização de uma única atividade. Isto é, sabendo que as coleções de dados de entrada das atividades podem ser fragmentadas e executadas em paralelo sem perder a qualidade do resultado. Assim, o cientista pode adicionar, uma atividade que fragmente essas entradas na definição do *workflow*.

Olhando novamente para a linha especificada na Figura 8 da seção anterior, temos o conjunto de atividades A={a, b, c}, com tempo médio de execução TM={tm_a, tm_{b1}, tm_{b2}, tm_c}, onde $tm_{b1} > tm_a + tm_c$, e dependências D=(Dep(a, c), Dep(b, c)).

Com a informação do tempo de execução, é possível ver que o tempo de execução do *workflow* como um todo depende em maior medida da atividade **b**. Portanto implementar a paralelização nesta atividade pode ser vantajoso para reduzir o tempo de execução do *workflow*.

Uma forma de deixar essa alternativa pronta para ser usada é modelar esse *workflow* como derivação de uma linha de experimento. A atividade que realiza uma fragmentação dos arquivos de entrada da atividade $\mathbf{b_1}$ seria incluída como uma nova variabilidade da atividade **b** como indicado na Figura 9. A execução do *workflow* com essa variabilidade consistirá em diversas invocações do programa científico, uma para cada fragmento de dados gerado em diferentes recursos computacionais, a fim de tirar proveito do paralelismo, reduzindo o tempo de execução do *workflow*.

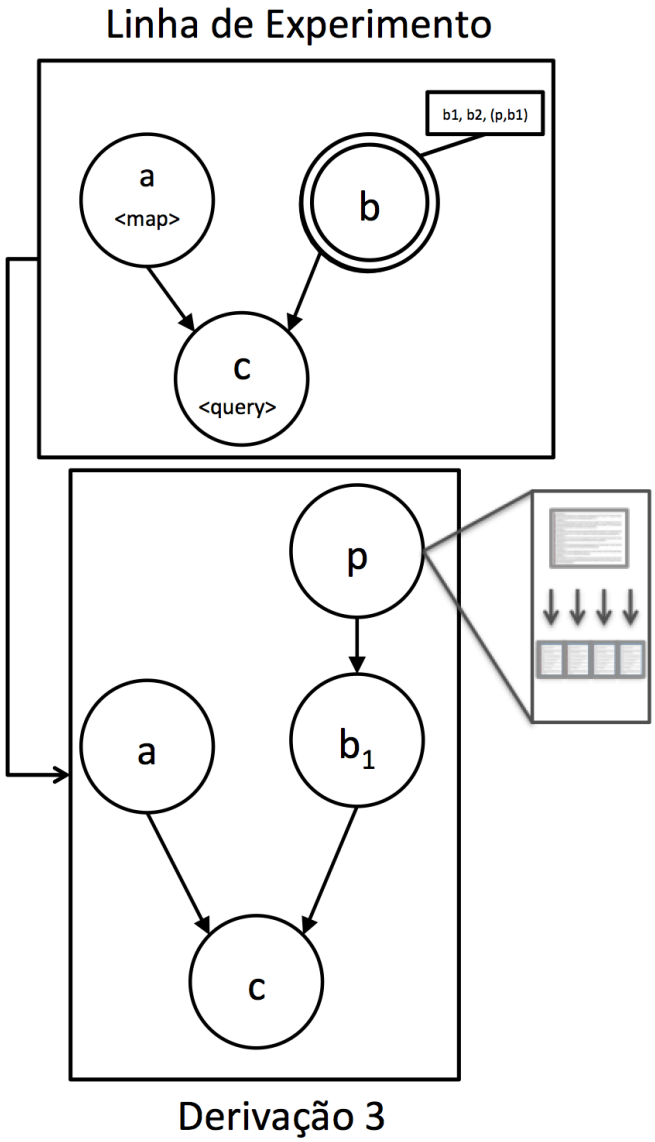


Figura 9 – Exemplo do desenho da linha de experimento e a nova derivação após inserir a atividade p de fragmentação de arquivo

Portanto, o tempo médio T_m de execução da atividade pode ser reduzido em função de técnicas de paralelismo de dados. Além disso, essa atividade é um ponto de variação, e cabe ao cientista a escolha da invocação do programa científico com ou sem

fragmentação dos dados de entrada.

3.3 Filtragem dos Dados de Entrada

Os operadores algébricos das atividades da linha de experimento deste capítulo, são *Map, Map* e *Query*, como indicado na Figura 10. A atividade c realiza uma operação de *Query* e, como visto na seção 2.4.5, esta operação possui uma única ativação, na qual opera com o resultado das atividades das quais ela depende, neste caso as atividades a e b.

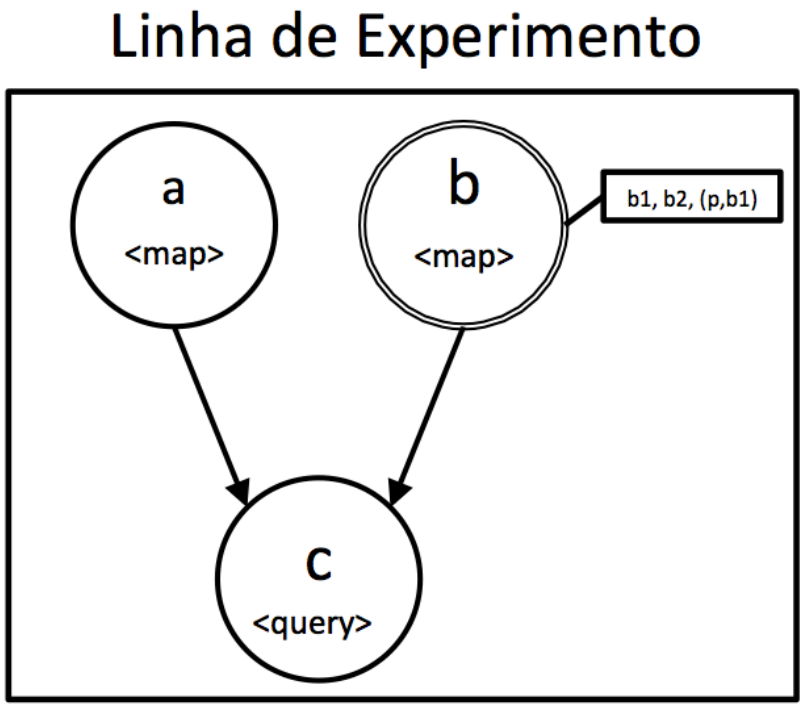


Figura 10 - Linha de experimento com operadores algébricos

Porém, quando levada em consideração a relação entre o conjunto de dados de entradas das atividades e a influência delas no experimento, pode se observar o conjunto denominado IF=((I_i, a_j)), onde cada I_i é o conjunto I = {i_1, i_2, .., i_m} de entradas da atividade a_i. O cientista deve obter o conjunto F = (IF(I_1, a), IF(I_1, b), IF(I_2, c) , IF(I_3, c)), onde I_2 e I_3 representam as saídas das atividades a e b respectivamente, com isto é verificado que o conjunto de dados de entrada para as atividades a e b é o mesmo, representado na figura 11.

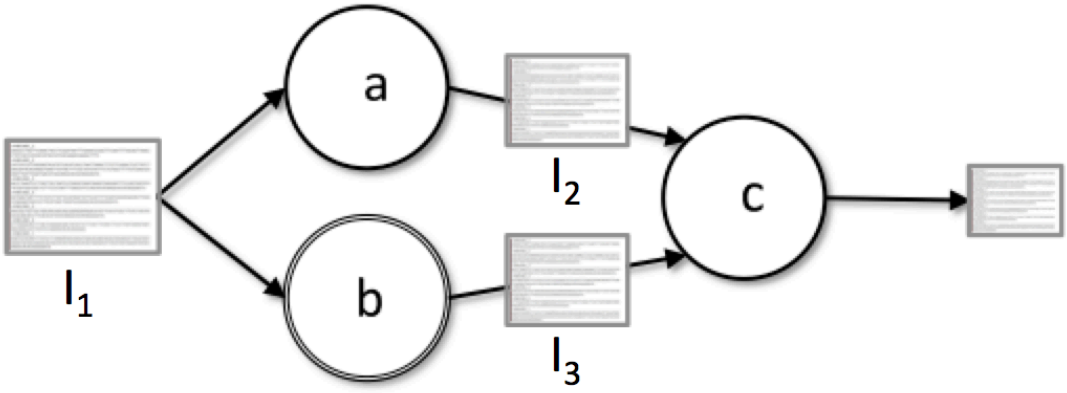


Figura 11 – Representação das entradas das atividades da linha de experimento

Isto indica que a atividade c não somente depende do resultado das atividades a e b, mas indiretamente também do mesmo conjunto I_1 de entrada. Em alguns casos, a atividade c realiza uma comparação das diferenças obtidas nas atividades anteriores, o que pode sugerir que somente parte do conjunto I_1 é interessante para o experimento.

Portanto, a partir do conjunto de dados I_1, uma atividade regida pelo operador algébrico de filtro pode ser adicionada para reduzir tanto o número de tuplas como o tamanho do arquivo de entrada a ser processado pelas próximas atividades, o que reduz, consequentemente, o tempo de execução das atividades posteriores.

Na seção anterior foi visto que o tempo médio de execução das atividades é TM={tm_a, tm_{b1}, tm_{b2}, tm_c}, onde $tm_{b1} > tm_a + tm_c$, incluem uma atividade que realiza a filtragem dos arquivos de entrada da atividade $\mathbf{b_1}$, sendo que a mesma poderá ser incluída como um novo ponto de opcionalidade antes da atividade **b** como indicado na Figura 12.

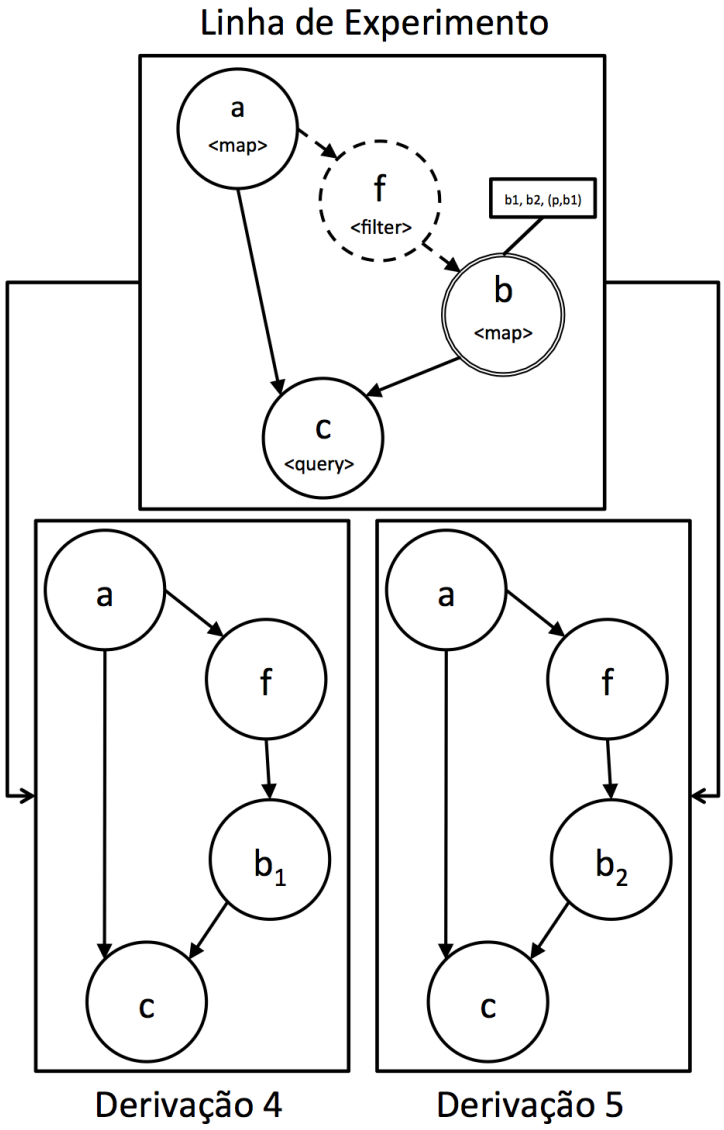


Figura 12 – Exemplo do desenho da linha de experimento e as novas derivações após inserir a atividade opcional f de filtro de arquivo

Considerando a possibilidade de inclusão da atividade com a filtragem de dados nas derivações 4 e 5 (Figura 12) e após a execução do programa **a**, as atividades $\mathbf{b}_1$ e $\mathbf{b}_2$ somente executarão com as entradas I_1 que forem selecionadas pela atividade **f**.

A inclusão desta atividade realiza uma redução no arquivo de entrada, o que pode evitar a execução desnecessária de dados não relevantes, reduzindo assim o tempo de execução do *workflow* derivado. A atividade de filtro recomendada é combinada com a variabilidade **b1** sendo as **f** e **b1** representadas como um ponto de opcionalidade antes da atividade **b**. Cabe ao cientista a escolha da invocação do programa científico com ou sem filtragem dos dados de entrada.

Capítulo 4 - Caso de Estudo

Este capítulo apresenta o experimento da bioinformática onde foram aplicadas as recomendações apresentadas no Capítulo 3. Na seção 4.1. é dada uma breve introdução sobre bioinformática e a área de metagenômica, escolhida como caso de estudo deste campo. A seção 4.2 apresenta a linha de experimento de metagenômica SciMG onde foram aplicadas as recomendações. Finalmente, na seção 4.3 são apresentados os quatro *workflows* derivados desta linha.

4.1 Metagenômica na Bioinformática

Nas últimas décadas, as tecnologias de sequenciamento de nova geração (do inglês *next-generation sequencing,* NGS) (Zhang *et al*. 2011) fizeram possível a geração de uma grande quantidade de dados biológicos. Como resultado, a taxa de submissão das sequências em bancos de dados especializados como o NCBI (Pruitt *et al*. 2009) (National Center for Biotechnology Information), o UniProt (*The UniProt Consortium* 2010) (*Universal Protein Resource*), e o PDB (Rose *et al*. 2013) (*Protein Databank*) é incrementado anualmente. Entretanto, a capacidade de interpretação dos dados não acompanha tal crescimento, pelo qual é preciso de tecnologias eficientes e escaláveis (Ho-Sik Seok *et al.* 2014).

Segundo Lengauer, a área que une a tecnologia da informação com a biologia é descrita pelos termos de Bioinformática e Biologia Computacional. O *National Institute of Health*[1] (NIH) define a bioinformática como a "pesquisa, desenvolvimento ou aplicação de ferramentas e abordagens computacionais para a expansão do uso de dados biológicos, médicos, comportamentais ou de saúde, incluindo aqueles usados para adquirir, armazenar, organizar, analisar ou visualizar esses dados".

---

[1] www.nih.gov

O NIH é a maior fonte de financiamento para a investigação médica em todo o mundo, com milhares de cientistas em universidades e instituições de pesquisa em toda a América e ao redor do mundo.

Existem inúmeras áreas na Bioinformática, sendo que atualmente as análises estão centradas nos dados de experimentos genômicos, metagenômicos, proteômicos e transcriptômicos. A metagenômica é "a análise independente do cultivo dos genomas coletados de micróbios diretamente de um determinado meio ambiente utilizando abordagens de sequenciamento e análise de última geração" (Meiring *et al.* 2011). Também pode ser definida como "a análise genética direta nos genomas contidos dentro de uma amostra ambiental" (Thomas *et al*.2012). A metagenômica provê o acesso à composição genética funcional das comunidades microbianas. Como a genômica, o termo metagenômica é utilizado para descrever tanto o campo da pesquisa como o conjunto que de técnicas de utilizadas na análise específica desses dados.

No escopo desta dissertação, será estudada a área da metagenômica focada em um aspecto específico na análise dos dados conhecida como predição de genes. Este tipo de experimento científico é considerado complexo e computacionalmente intensivo, pelo que mostrou ser um bom candidato para ser apoiado por tecnologias de *workflows* científicos (apresentado nos Capítulo 2, subseções 2.3 e 2.4).

4.1.1 Especificação do Experimento da Metagenômica

O caso de estudo da presente pesquisa é baseado no experimento da metagenômica de comunidades microbianas marinas brasileiras, mais especificamente na identificação e predição funcional de novos genes presentes nessas comunidades.

As sequências coletadas são moléculas de DNA, RNA ou proteínas, extraídas com técnicas de sequenciamento de alta geração. O experimento visa a identificação de novas proteínas dentre as moléculas coletadas, e a reanotação daquelas que foram identificadas em outros organismos. O banco utilizado para esta pesquisa é o *National Center for Biotechnology Information* (NCBI).

O experimento de metagenômica para a identificação funcional de genes é dividido em quatro atividades principais, como apresentado na Figura 13.

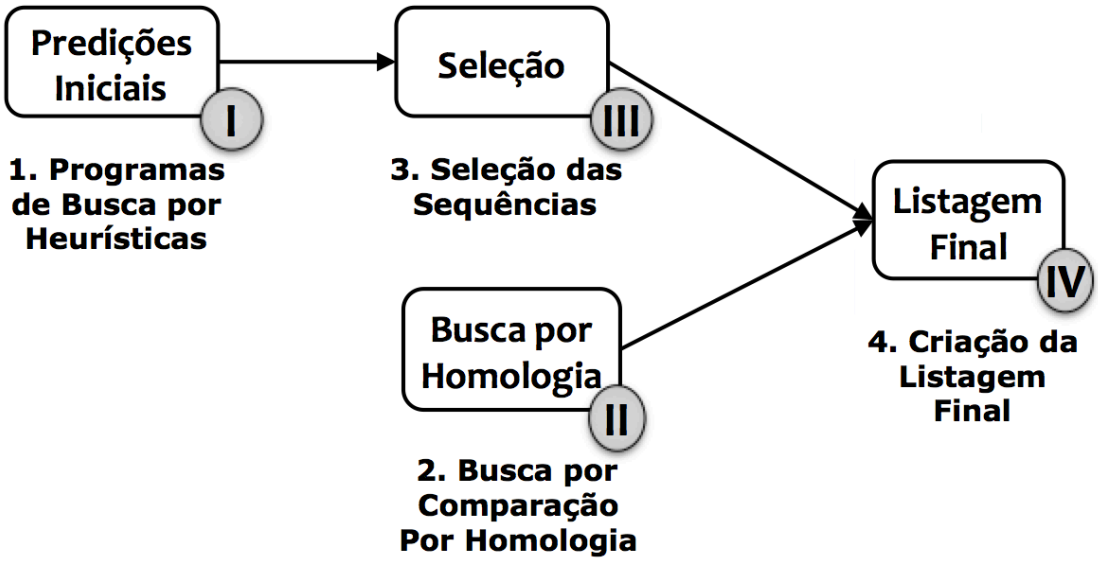


Figura 13 - Experimento da Metagenômica para a Identificação Funcional de Genes

A primeira atividade, a informação biológica contida nos arquivos de entrada é analisada para predizer possíveis proteínas, mediante a identificação por diversos programas baseados em métodos heurísticos (*ab initio*). No escopo desta dissertação, foram utilizados três programas deste tipo a modo de aumentar a confiabilidade na identificação da proteína, portanto, somente serão consideradas aquelas que forem reconhecidas pelos três programas (consideradas como de confiabilidade 3).

A segunda atividade, realiza uma busca por homologia, onde cada sequência de entrada é comparada com as sequências contidas no banco de dados de proteínas do NCBI.

A atividade III visa unificar as saídas da atividade I, pelo que realiza uma seleção baseada na análise dos dados de proveniência extraídos e retorna uma listagem preliminar com todas as sequências apresentando uma confiabilidade 3. Finalmente, a atividade IV utiliza os dados de saída extraídos da atividade II em conjunto com as sequências selecionadas pela atividade III e retorna uma listagem final com todas as sequências reconhecidas ainda não anotadas no banco de dados NCBI.

Para este experimento os seguintes programas de bioinformática com parâmetros padrão foram executados: o Prodigal 2.6.1, FragGeneScan 1.30 e MetaGeneMark 3.26 (para a procura por heurísticas) e o pacote BLASTn 2.2.28 (para a procura por homologia). Dado que este experimento foca na gerência de dados de proveniência obtidos de cada atividade, as atividades não foram tratadas como caixas fechadas. Sendo assim, as saídas dos distintos programas foram analisadas e os dados de proveniência extraídos

em tempo de execução. Desta maneira, este experimento usa de forma iterativa tanto os dados de domínio-específicos do experimento metagenômica como do próprio *workflow* modelado em sim.

4.2 SciMG: Linha de Experimento de Metagenômica

Nesta seção é apresentada a composição da linha de experimento SciMG assim como as recomendações baseadas nessa linha (baseada na seção 2.6), definindo as atividades abstratas, seus respectivos operadores e dependências.

A Tabela 2 apresenta as atividades com suas respectivas cardinalidades, assim como a dependência de dados de entrada e saída entre essas atividades.

Atividade	Cardinalidade	Entrada	Saída
I - Busca por heurísticas	1:1	Arquivo Fasta	Sequências extraídas I
II - Busca por homologia	1:1	Arquivo Fasta	Sequências extraídas II
III - Seleção	n:m	Sequências extraídas I	Sequências selecionadas
IV - Listagem Final	n:m	Sequências selecionadas e sequências extraídas II	Sequências finais

Tabela 2 - Conjunto de entradas e saídas das atividades do experimento da Metagenômica

As atividades I e II possuem o operador *Map* que é utilizado quando uma dada entrada gera 1 saída (apresentado na seção 2.4.2). A cardinalidade da saída nesta atividade é dependente do arquivo de saída gerado por cada programa, onde é realizada a extração dos dados de domínio para cada uma das atividades. Finalmente, nas atividades III e IV utilizam o operador *Query* realizando consultas na base de proveniência, e realizando assim a união das saídas dos distintos programas. Finalmente, podemos observar que a cardinalidade das atividades condiz com os seus respectivos operadores.

4.2.1 Aplicação das Recomendações para o Desenho do Experimento

Com as atividades definidas, pode ser verificada a possibilidade de paralelização das atividades (seção 3.1). Como foi indicado na seção anterior, as atividades I e II são independentes, mesmo recebendo o mesmo conjunto de entradas.

A atividade I contém os três programas de predição *ab initio* e é considerada como uma atividade abstrata variante, onde o cientista pode escolher uma ou mais atividades

dependendo da confiabilidade requerida para o experimento. A linha de experimento SciMG é apresentada na Figura 14.

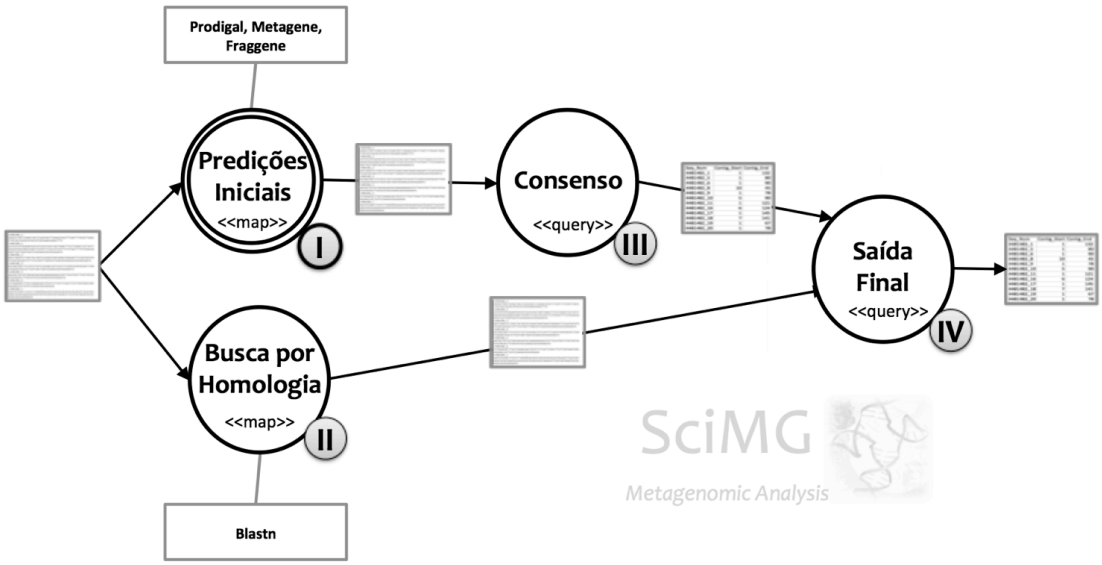


Figura 14 - Linha de experimento SciMG

É importante indicar que nesta Linha de Experimento, as atividades I e II podem se beneficiar da paralelização criada pelo plano de execução do SGWfC utilizado. Isto é, quando o *workflow* derivado desta linha for executado com SGWfC que suportam paralelismo em cima de coleções de dados tanto a atividade I e II serão executadas em paralelo tendo em conta a quantidade de recursos disponíveis e o número de entradas em cada conjunto para cada atividade.

Por outro lado, pela recomendação da seção 3.2, o tempo de execução das atividades deve ser analisado. A Tabela 3 apresenta o tempo médio das ferramentas utilizadas no experimento. Os programas Prodigal, MetaGeneMark e FragGene apresentam valores similares, já o pacote BLASTn apresenta um tempo 20 vezes maior, para o mesmo arquivo multi-fasta de 500 sequências.

Programa	Tempo Médio (segundos)
Prodigal	5.6
MetaGeneMark	6.2
FragGeneScan	6.1
BLASTn	122.0

Tabela 3 - Tempo médio de execução das ferramentas utilizadas nas atividades da linha SciMG

Devemos ressaltar que a execução do programa BLASTn realiza a análise independente para cada sequência dentro de cada um dos arquivos multi-fasta de entrada. Portanto, tratar um arquivo ou vários fragmentos do mesmo levaria ao mesmo resultado em termos de tempo computacional, porém se a execução de cada fragmento é paralela e distribuída, a tendência é de obter uma redução no desempenho na execução.

Neste caso, a inclusão de uma atividade que vise a paralelização da atividade BLASTn pode ser realizada a nível de fragmentação do arquivo de entrada. Para isto é utilizado um *script* da biblioteca Python chamada PyFasta que divide o arquivo multi-fasta original em um dado número *n*, sem afetar a informação biológica contida. O operador para esta atividade é o *SplitMap.*

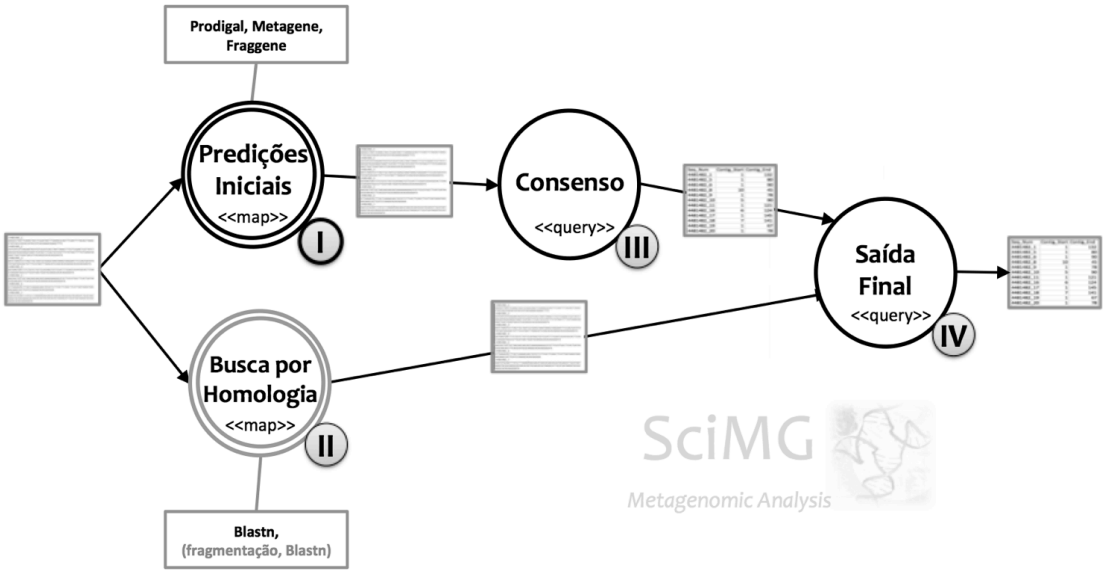


Figura 15 – Linha de experimento SciMG após a inclusão da atividade de fragmentação de arquivo

Na Figura 15, a atividade de fragmentação é incluída no desenho da linha de experimento. Esta atividade foi adicionada como ponto de variabilidade na atividade II, tornando-a uma atividade variável, isto é, somente será adicionada aos *workflows* derivados que o cientista definir.

A última recomendação analisa o fluxo de dados do experimento como um todo, a fim de verificar aquelas atividades que sejam dependentes entre sim. Desta forma, se visa incluir uma nova atividade de filtro para diminuir o conjunto de entradas de uma determinada atividade. Porém, o ponto principal é que o cientista observe aquelas atividades que não sejam dependentes entre sim, procurando atividades que unifiquem os resultados das execuções anteriores, que para a álgebra de *workflows* ocorre com atividades com o operador *Query*.

Nesta linha de experimento, duas atividades apresentam o operador *Query*. Na atividade III, só as sequências com confiabilidade 3 (dos programas *ab initio*) são consideradas significativas e filtros encima delas podem ser adicionados para filtrar as sequências. Porém analisando o tempo médio (arquivos de 500 sequências), a adição do filtro não apresenta ganhos significativos para o experimento, como um todo, devido a que o tempo da atividade BLASTn tem uma ordem de grandeza muito superior aos programas *ab initio*. Já para atividade IV, que compara os resultados obtidos das atividades II e III, pode ser esperada uma diferença considerável no tempo de execução, após a inserção do filtro para reduzir o número de entradas na atividade II.

A Figura 16 apresenta a linha de experimento com a atividade que realiza a filtragem. Foi utilizado um *script* para analisar as sequências selecionadas da atividade III e gerar um novo arquivo multi-fasta, no qual só aquelas sequências com confiabilidade 3 são incluídas. Similar à atividade de fragmentação, a filtragem só será adicionada aos *workflows* derivados indicados pelos cientistas.

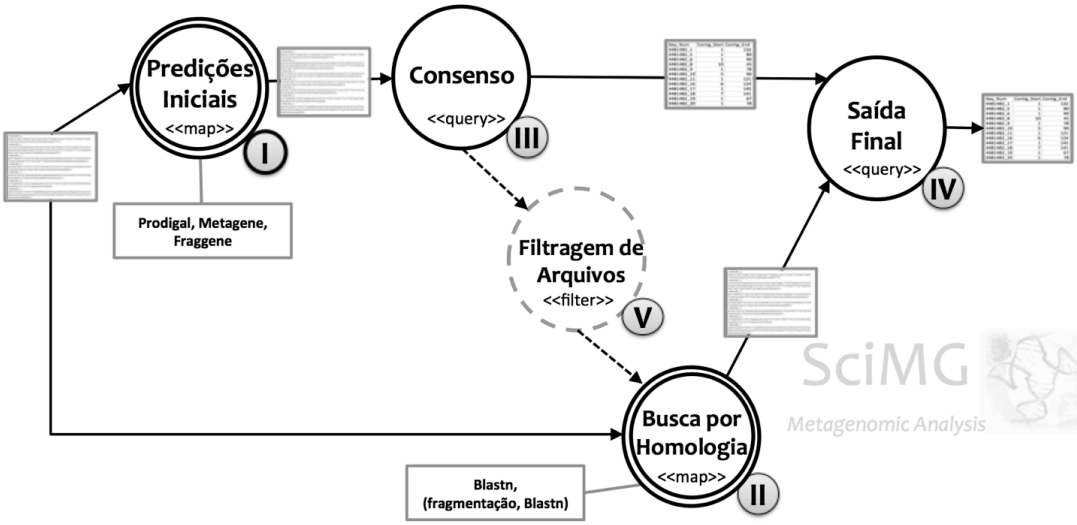


Figura 16 – Linha de experimento SciMG após a inclusão da atividade de filtro de arquivo

4.3 Derivações da Linha de Experimento SciMG

Quatro variações de *workflows* concretos são apresentadas baseadas nas recomendações da Figura 16. É importante lembrar que a especificação de uma linha de experimento pode ser utilizada no apoio ao desenho de *workflows*, a qual é independente ao SGWfC usado. Para o escopo desta pesquisa foi escolhido o SGWfC SciCumulus/C^2.

4.3.1 *Workflow* Derivado 1 – *Baseline*

O primeiro *workflow* concreto é chamado *baseline* e é composto pelas seguintes macro-atividades: predição inicial, busca por homologia, saída de consenso e saída final. Cada macro atividade pode estar composto por uma ou mais atividades como apresentado na Figura 17.

(I) Predição Inicial	atividades (1a,1b,1c)
(II) Busca por Homologia	atividade (2)
(III) Consenso	atividade (3)
(IV) Saída Final	atividade (4)

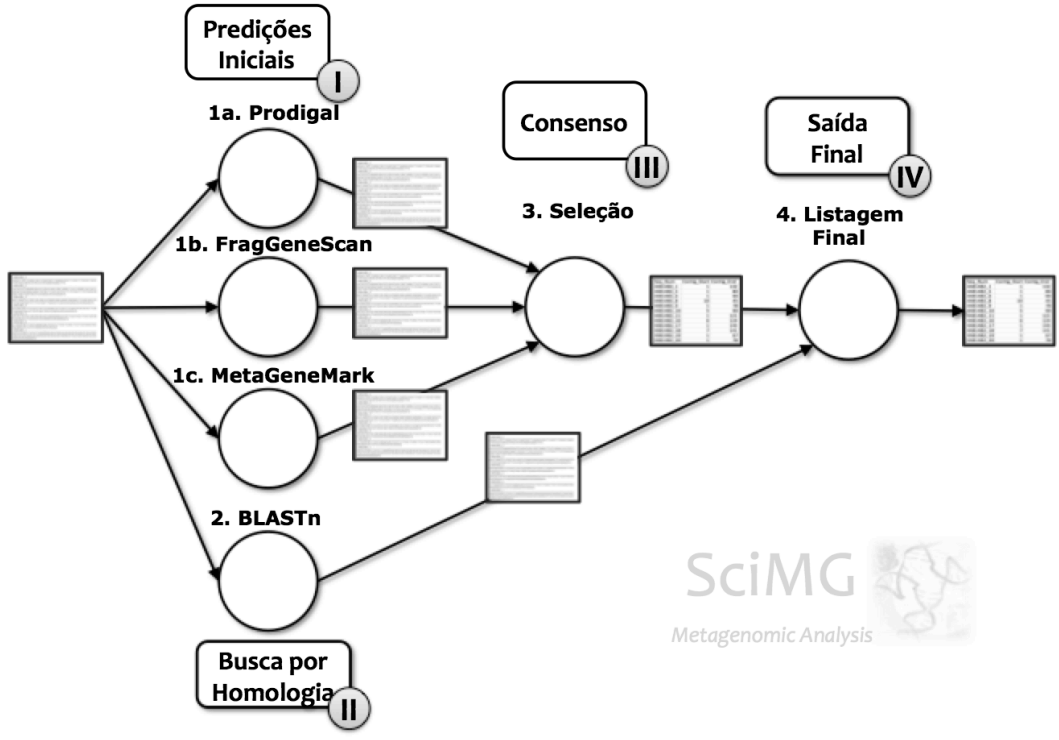


Figura 17 - Modelo conceitual do *workflow* concreto *baseline*

Estas atividades executam os seguintes programas de bioinformática com parâmetros

padrão: Prodigal 2.6.1 (1a), FragGeneScan 1.30 (1b), MetaGeneMark 3.6 (1c), BLASTn 2.2.28 (2), assim como *scripts* próprios para extração de dados de domínio. Como mencionado anteriormente, as derivações da linha de experimento foram realizadas com o SGWfC eleito, o SciCumulus/C^2. As atividades abstratas III e IV (geram a listagem de sequências reconhecidas) ao serem derivadas sofrem uma alteração, cada uma agrupando duas atividades concretas. Na atividade III são realizadas consultas na base de proveniência para agrupar os arquivos que contêm as sequências extraídas da atividade I utilizando o operador *Query*. Em seguida, a atividade *Map* realiza a seleção das sequências reconhecidas e gera um arquivo com as sequências reconhecidas para entrada inicial. Já na atividade IV, o operador *Query* realiza um processamento similar, unindo a saída das atividades II e III e posteriormente uma atividade com o operador *Map* retorna todas as sequências existentes na saída da atividade III, que não estejam na saída da atividade II. Esta derivação será utilizada a modo de comparação com as outras derivações, já que nenhuma atividade opcional obtida a partir das recomendações foi adicionada.

4.3.2 *Workflow* Derivado 2 – *Split*

A segunda derivação é chamada de *workflow split* como apresentada na Figura 18 e possui as seguintes macro-atividades.

(I) Predições Iniciais	atividades (1a,1b,1c)
(V) Fragmentação de Arquivos	atividade (5)
(II) Busca por Homologia	atividade (2)
(III) Consenso	atividade (3)
(IV) Saída Final	atividade (4)

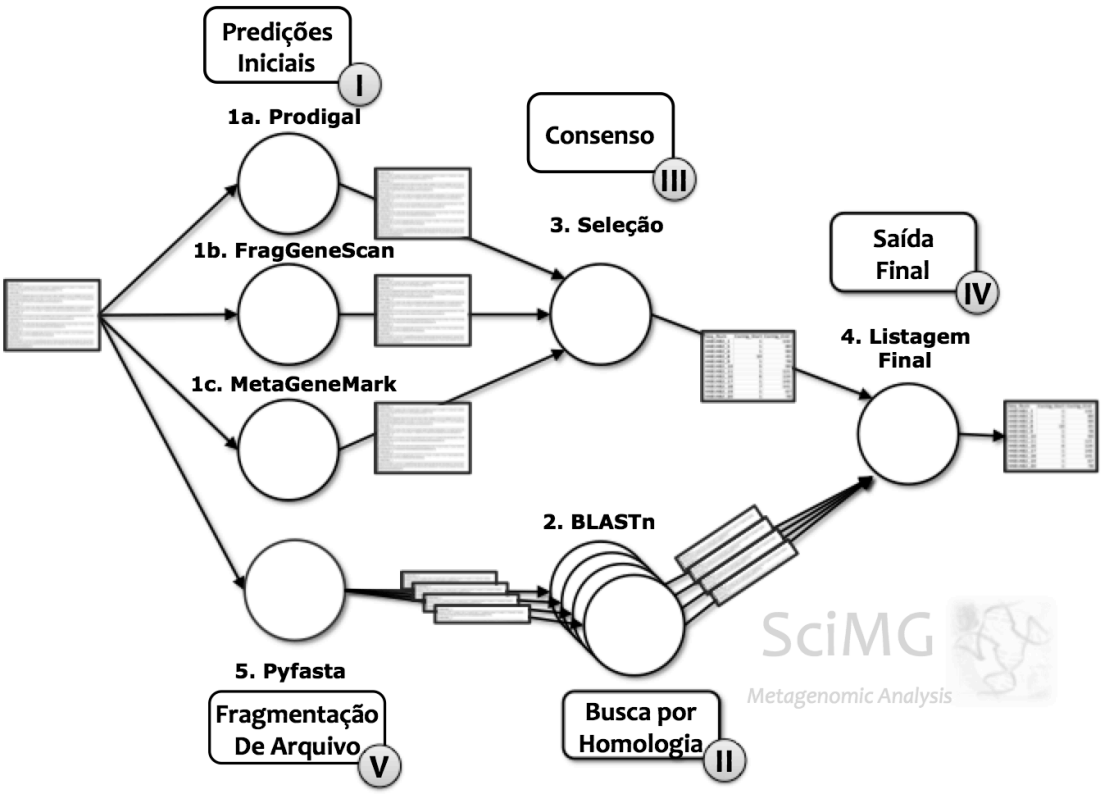


Figura 18 - Modelo conceitual do *workflow* concreto *split* aplicando paralelização e fragmentação do arquivo de entrada.

Nesta derivação, é vista que a atividade V que é opcional e de fragmentação. As atividades possuem novamente, os mesmos programas de bioinformática com parâmetros padrão: Prodigal 2.6.1 (1a), FragGeneScan 1.30 (1b), MetaGeneMark 3.6 (1c), BLASTn 2.2.28 (2). Estas atividades executaram *scripts* próprios para extração de dados de domínio; já nas atividades III e IV foram realizadas as mesmas derivações apresentadas no *workflow baseline*, com operadores do SciCumulus/C^2. Finalmente, a atividade V executa o *script* da biblioteca PyFasta.

4.3.3 *Workflow* Derivado 3 – *Filter*

A derivação chamada *workflow Filter* é apresentada na Figura 19 e possui as seguintes macro-atividades.

(I) Predições Iniciais	atividades (1a,1b,1c)
(III) Consenso	atividade (3)
(VI) Filtro de Arquivos	atividades (6,7)
(II) Busca por Homologia	atividade (2)

(IV) Saída Final atividade (4).

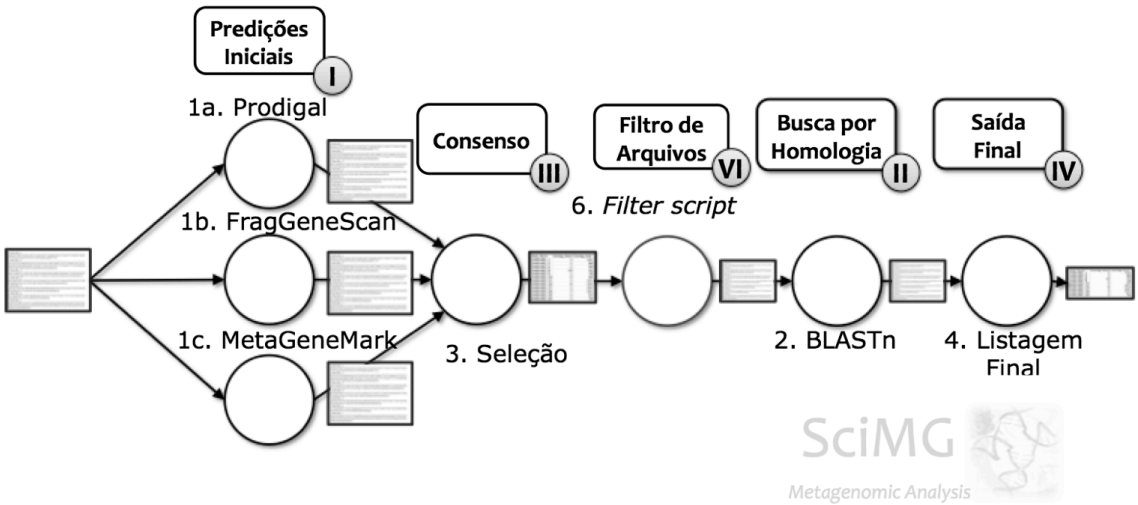


Figura 19 - Modelo conceitual do *workflow* concreto *filter* aplicando paralelização e filtro do arquivo de entrada.

As atividades obrigatórias foram derivadas com parâmetros padrão similares ao *workflow baseline*, porém para a atividade VI (filtro de arquivos) foi criado um *script* próprio que trata das sequências extraídas na atividade III e o arquivo multi-fasta original. Esse *script* gera um novo arquivo multi-fasta contendo só as sequências reconhecidas por todos os programas *ab initio* a ser utilizado como entrada pela atividade BLASTn.

4.3.4 *Workflow* Derivado 4 – *SplitFilter*

A última derivação é *splitfilter*, a qual aplica todas as recomendações (Figura 20) e possui as seguintes macro atividades.

(I) Predições Iniciais atividades (1a,1b,1c)

(III) Consenso atividade (3)

(VI) Filtro de Arquivos atividades (6,7)

(V) Fragmentação de Arquivos atividade (5);

(II) Busca por Homologia atividade (2).

(IV) Saída Final atividade (4).

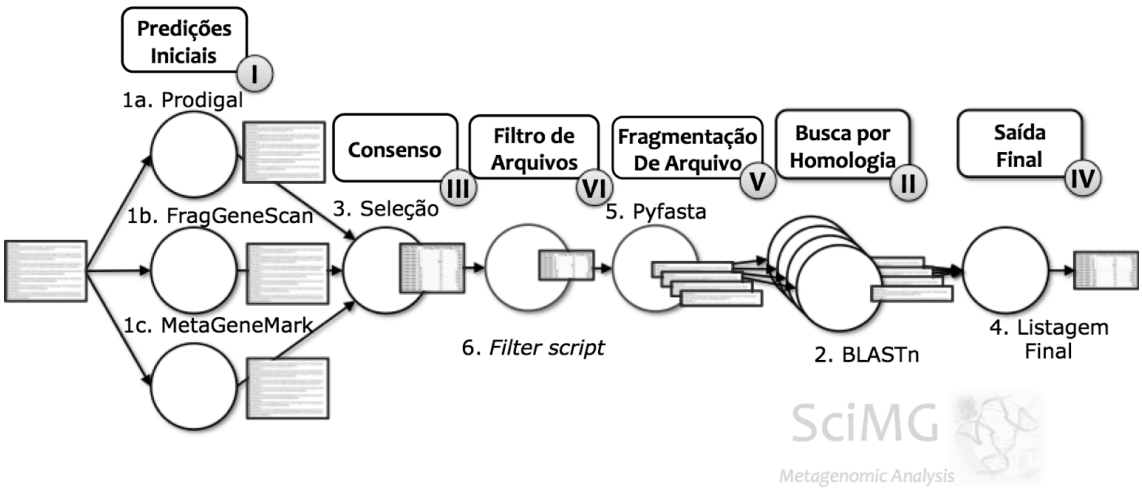


Figura 20 - Modelo conceitual do *workflow* concreto *splitfilter* aplicando paralelização, filtro e fragmentação do arquivo de entrada.

Nessa derivação, ambas as recomendações foram utilizadas em sequência para verificar se a combinação delas oferece vantagens para a redução do tempo de execução. O objetivo é visar o aumento do paralelismo da atividade BLASTn, fragmentando os arquivos multi-fasta obtidos após o filtro do arquivo original.

Capítulo 5 - Avaliação Experimental

Esse capítulo apresenta as configurações utilizadas, a avaliação experimental e os resultados obtidos ao executar os *workflows* derivados da linha de experimento SciMG. O objetivo principal deste capítulo é avaliar o desempenho e a escalabilidade das execuções desses *workflows*.

Para realizar a avaliação experimental inicialmente foram caracterizados dois cenários, nos quais foram executadas as quatro derivações modeladas com o SciCumulus/C^2 e executados em um ambiente PAD.

Este capítulo foi dividido da seguinte maneira: a seção 5.1 apresenta a configuração do ambiente utilizado para a execução dos experimentos. A seção 5.2 traz as configurações para a execução destas derivações; enquanto que a seção 5.3 apresenta a análise de desempenho e escalabilidade.

5.1 Configuração do Ambiente

Para os experimentos executados nesta dissertação, foi utilizado o supercomputador Lobo Carneiro (LoboC) do Instituto Alberto Luiz Coimbra de Pós-Graduação e Pesquisa de Engenharia (COPPE), o LoboC possui a seguinte configuração:

- 504 CPUs Intel Xeon E5-2670v3 totalizando 6048 núcleos.
- Cada CPU com 24 núcleos de Processamento (reais) e 48 núcleos de processamento com Hyper-Threading (HT).
- 64 GBytes de memória por núcleo de processamento.
- 16 TBytes Total de Memória RAM distribuída:
- O sistema operacional instalado é o Suse Linux Enterprise (SLE)
- Os compiladores instalados são Intel, PGI e GNU (Fortran-90 e C/C++)
- MPI: MPT, Intel MPI, MVAPICH e OpenMPI

Para esta análise, foram utilizados o SGWfC SciCumulus/C^2, o SGBD PostgreSQL 9.5.3, os programas e pacotes de bioinformática BLASTn 2.6.0, FragGeneScan 1.30, MetaGeneMark 3.26, Prodigal 2.6.3, Pyfasta 0.5.2 e os *scripts* desenvolvidos para a extração de dados de domínio.

5.2 Configuração do Experimento

Para executar a linha de experimento SciMG em paralelo com o SciCumulus/C^2, a base de comparação foi o *workflow baseline* obtido a partir da Derivação 1 da linha de experimento SciMG. As outras três derivações propostas serão chamadas de *filter*, *split* e *splitfilter* respectivamente. O conjunto de dados de entrada é constituído por arquivos multi-fasta com sequências reais de proteínas.

Foram estabelecidos dois cenários de execução, o primeiro cenário realiza a execução dos *workflows* derivados, de maneira a corroborar o comportamento das recomendações em condições controladas. Para isto, foi utilizado como conjunto de entrada, arquivos de tamanhos similares. Foi realizada a comparação dos tempos dos *workflows* derivados que possuem atividades inseridas baseadas nas recomendações previamente levantadas.

O segundo cenário busca analisar as execuções das derivações dos *workflows* com condições mais próximas àquelas de um experimento real de metagenômica, com arquivos originais obtidos após o sequenciamento dos metagenomas. Ambas as comparações dos cenários 1 e 2 foram realizadas contra o *workflow* derivado *baseline* (sem recomendações).

No primeiro cenário, o conjunto de dados de entrada é constituído por 200 arquivos multi-fasta coletadas de ambientes reais. Esses arquivos originais foram preparados de maneira a que cada arquivo seja constituído por uma média de 497 sequências biológicas (fasta), sendo que o menor arquivo possui 465 sequências e o maior arquivo 544 sequências. O nível de fragmentação adoptado pela atividade é de 2.

Já no segundo cenário, o conjunto de dados de entrada é constituído por 3 arquivos multi-fasta sem alteração, onde os arquivos possuem 18.291, 29.404 e 53.310 sequências respectivamente. O nível de fragmentação adotado pela atividade é de 200.

5.3 Análise de Desempenho da Execução

Esta seção apresenta a avaliação experimental da comparação das diferentes recomendações aplicadas. Assim foram executadas as 4 derivações do experimento SciMG: *baseline, filter, split, splitfilter*.

5.3.1 Análise de Desempenho dos *Workflows* derivados da Linha de Experimento SciMG no Cenário 1

Para o primeiro cenário considerou-se um conjunto de dados de entrada de 200 arquivos multi-fasta, com uma média de 497 sequências por arquivo. O cenário deste experimento pode ser considerado como de larga escala, sendo que o tempo de execução sequencial pode chegar a uma média de, no mínimo, 7 horas e 50 minutos.

Para este cenário, as derivações da Linha de Experimento SciMG foram executadas no ambiente paralelo do *cluster* LoboC com 230 núcleos. A execução em paralelo do *workflow baseline* foi realizada em 5,35 minutos.

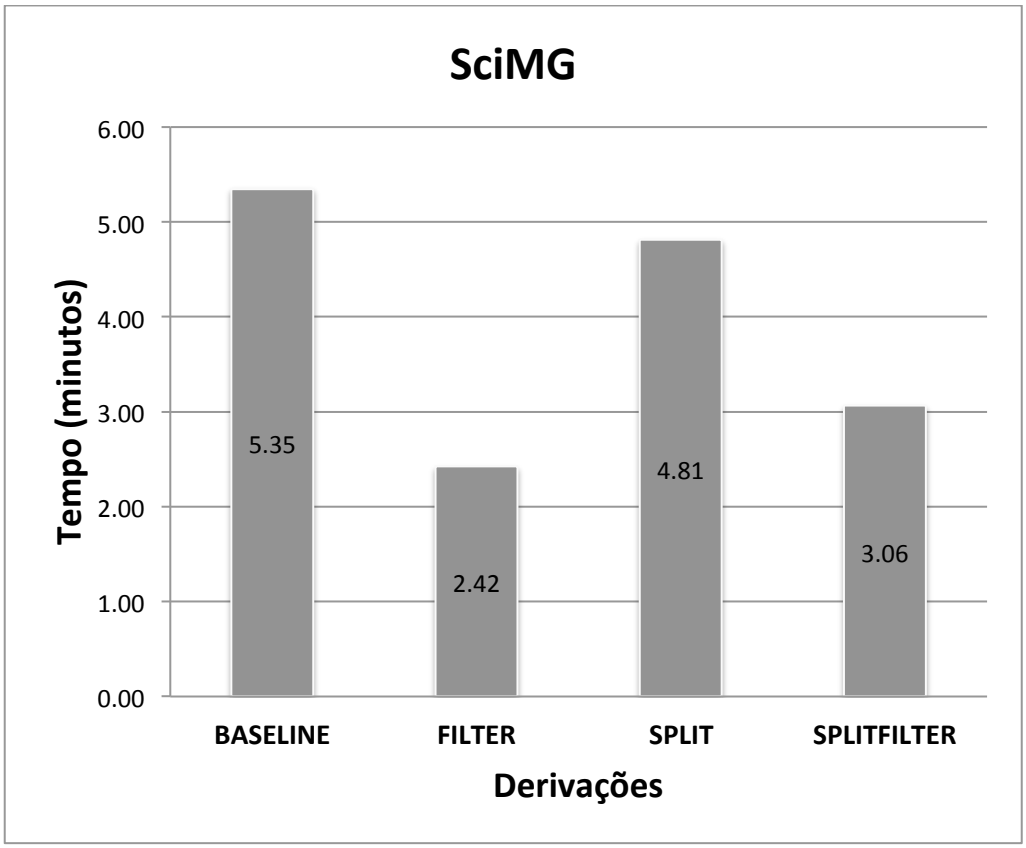
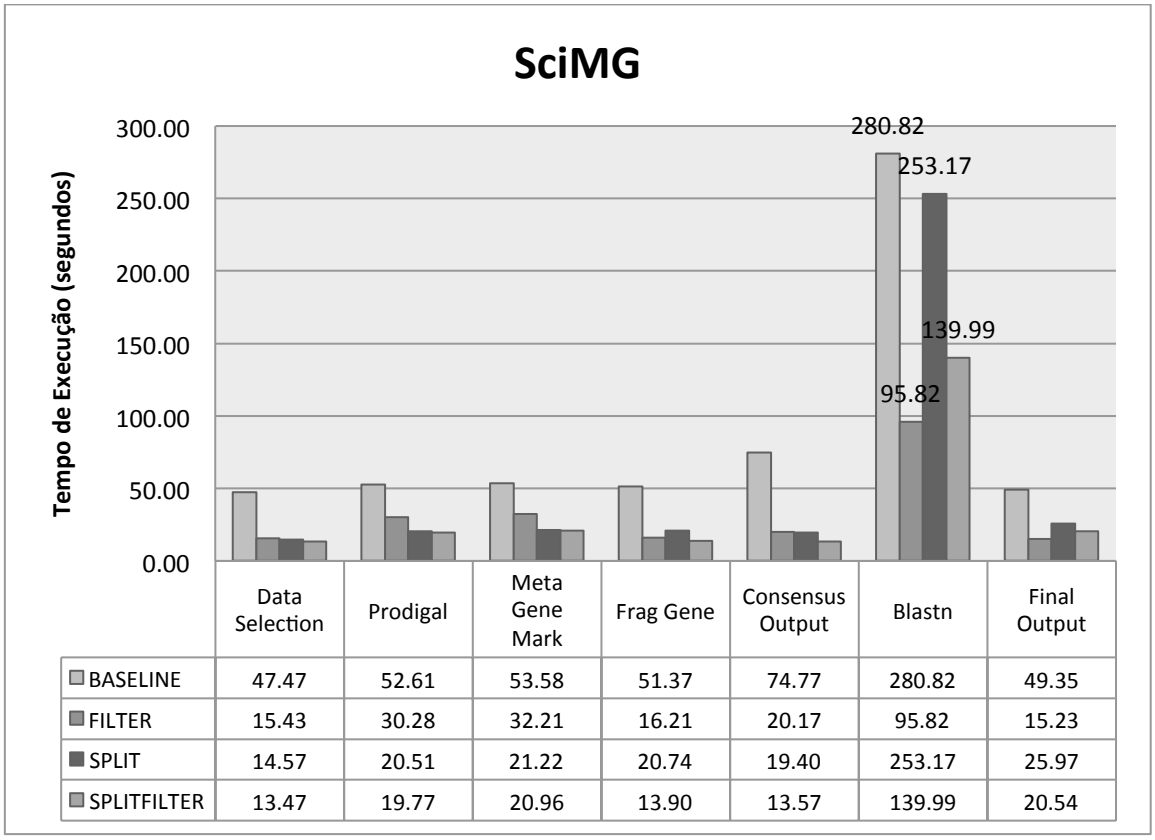


Figura 21 - Tempo médio dos *workflows* derivados da Linha de Experimento SciMG no Cenário 1

A Figura 21 apresenta o tempo de execução desse cenário para cada *workflow* derivado obtido. A derivação *filter* foi a que apresentou o melhor desempenho. Os *workflows filter*, *split* e *splitfilter* foram executados em 2,42 minutos, 4,81 minutos, e 3,06 minutos, respectivamente. Comparando-os com o *workflow baseline*, o *filter* e o *splitfilter* obtiveram uma redução no tempo de execução de 54,7%, e 42,7%, enquanto que a derivação *split* teve uma redução de apenas 10%.

A Figura 22 descrimina o tempo de execução de cada atividade, onde as atividades opcionais de filtro e partição de dados foram incluídas no cálculo do tempo da atividade BLASTn, quando presentes na derivação. Pode ser observado que para a atividade BLASTn, os *workflows filter* e *splitfilter* apresentam uma diminuição considerável no tempo de execução de 65,8% e 50,1% respectivamente, enquanto o *workflow split* apenas apresenta, uma redução de 9,84%.



	Data Selection	Prodigal	Meta Gene Mark	Frag Gene	Consensus Output	Blastn	Final Output
BASELINE	47.47	52.61	53.58	51.37	74.77	280.82	49.35
FILTER	15.43	30.28	32.21	16.21	20.17	95.82	15.23
SPLIT	14.57	20.51	21.22	20.74	19.40	253.17	25.97
SPLITFILTER	13.47	19.77	20.96	13.90	13.57	139.99	20.54

Figura 22 - Tempo médio das atividades dos *workflows* derivados

Porém nesta derivação foi feito um experimento com um aumento de entradas da atividade BLASTn após a execução da atividade de fragmentação. Isto resultou no aumento da quantidade de ativações na atividade BLASTn, que inicialmente era de 200 arquivos nas derivações com a inserção da atividade de fatiamento. Agora a atividade BLASTn passa a receber 400 arquivos. Vale ressaltar que a quantidade de recursos computacionais se manteve em 230 núcleos. Com isto, o número de recursos computacionais disponíveis era menor que a quantidade de tuplas de entrada, assim a paralelização não estaria sendo utilizada corretamente.

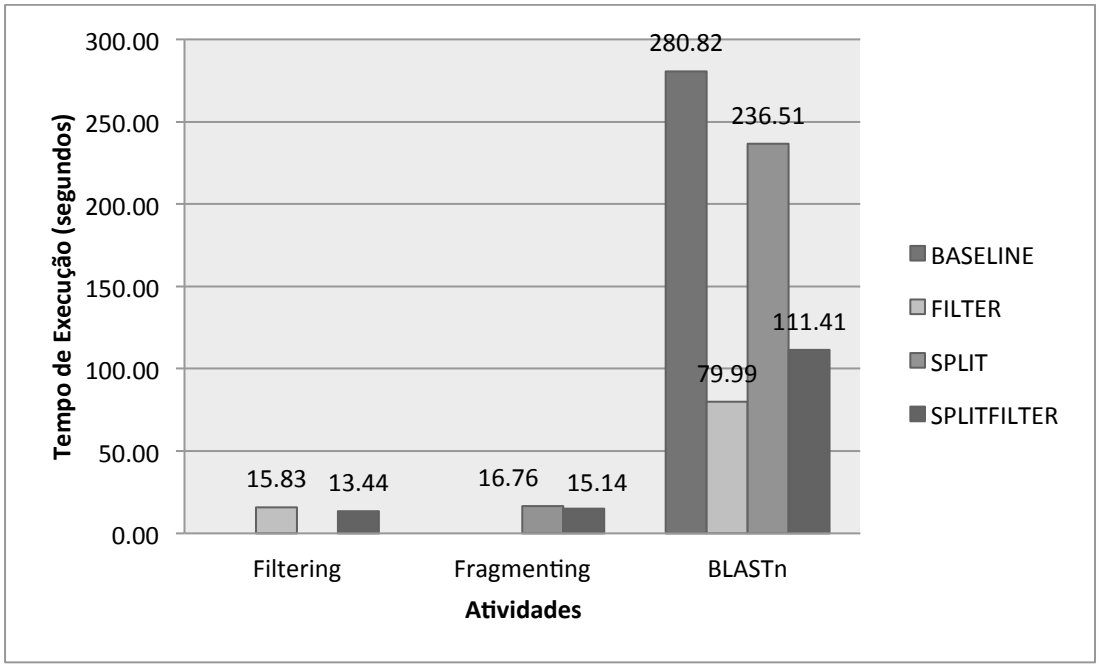


Figura 23 - Impacto das atividades de filtro e fragmentação em cima do BLASTn no Cenário 1

A Figura 23 apresenta o impacto das atividades inseridas relacionadas à atividade BLASTn. Podemos observar que para a derivação *split*, a atividade de fragmentação é a responsável por uma parte da sobrecarga de tempo nesse *workflow*. Além disso, a Tabela 4 apresenta o tempo de execução do BLASTn para distintas quantidades de sequências, o que indica que essa sobrecarga poderia estar diretamente relacionada ao custo de inicialização do BLAST para cada tarefa do *workflow*.

Apesar de reduzir consideravelmente o número de tuplas processadas por uma tarefa (arquivo de 250 sequências, fatiado em 2), observamos que a atividade BLASTn apresenta um tempo de execução muito próximo àquele executado com o conjunto de dados original (sem fatiar). Contudo, vale ressaltar que esse comportamento não é observado em arquivos com sequências maiores (acima de 4.000 sequências), uma vez

que o custo de inicialização do BLAST torna-se desprezível em relação ao tempo de processamento das consultas.

Sequências	Tempo (segundos)
120	39
240	40
484	79
968	201
1.936	225
3.872	258
7.744	326
15.488	568

Tabela 4 - Tempo de execução do BLASTn para diferentes tamanhos de sequências

A paralelização de dados é recomendada quando o cientista possui os recursos computacionais necessários para alocar todas as ativações em paralelo. Para analisar melhor esta recomendação, foi realizada a execução do mesmo cenário variando a quantidade de núcleos de 115 a 690 núcleos, como apresentado na Figura 24.

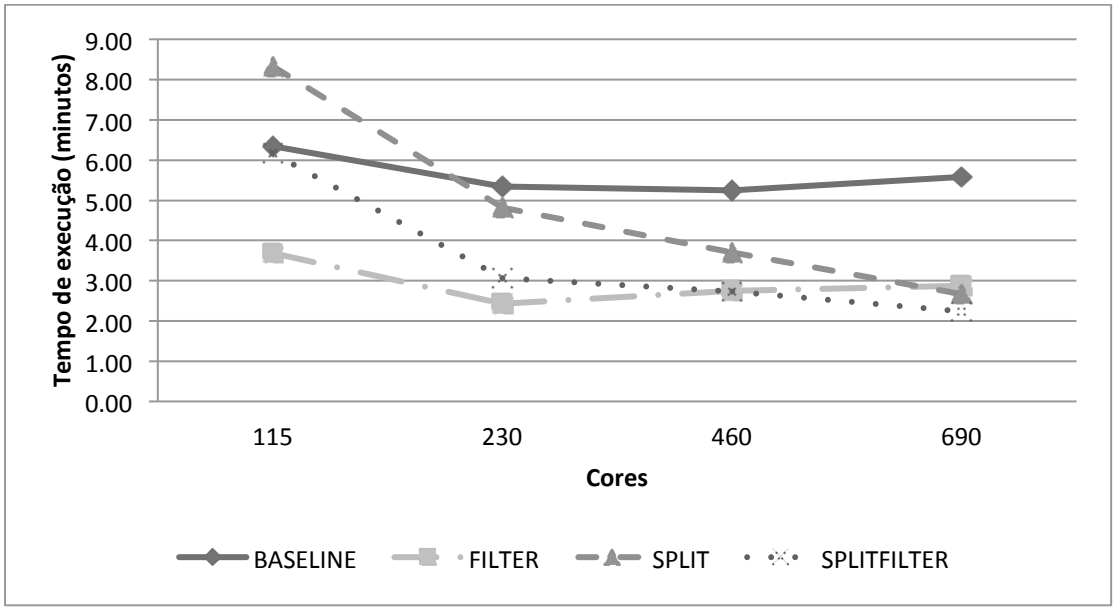


Figura 24 - Gráfico com tempos totais das execuções dos *workflows* derivados do SciMG, para o cenário 1

A Figura 25 mostra a comparação da derivação *filter* com a *baseline*. A derivação *filter* apresenta uma redução no tempo de execução de 41,9%, 54,7%, 47,5% e 48,4% para 115, 230, 460 e 690 núcleos respectivamente.

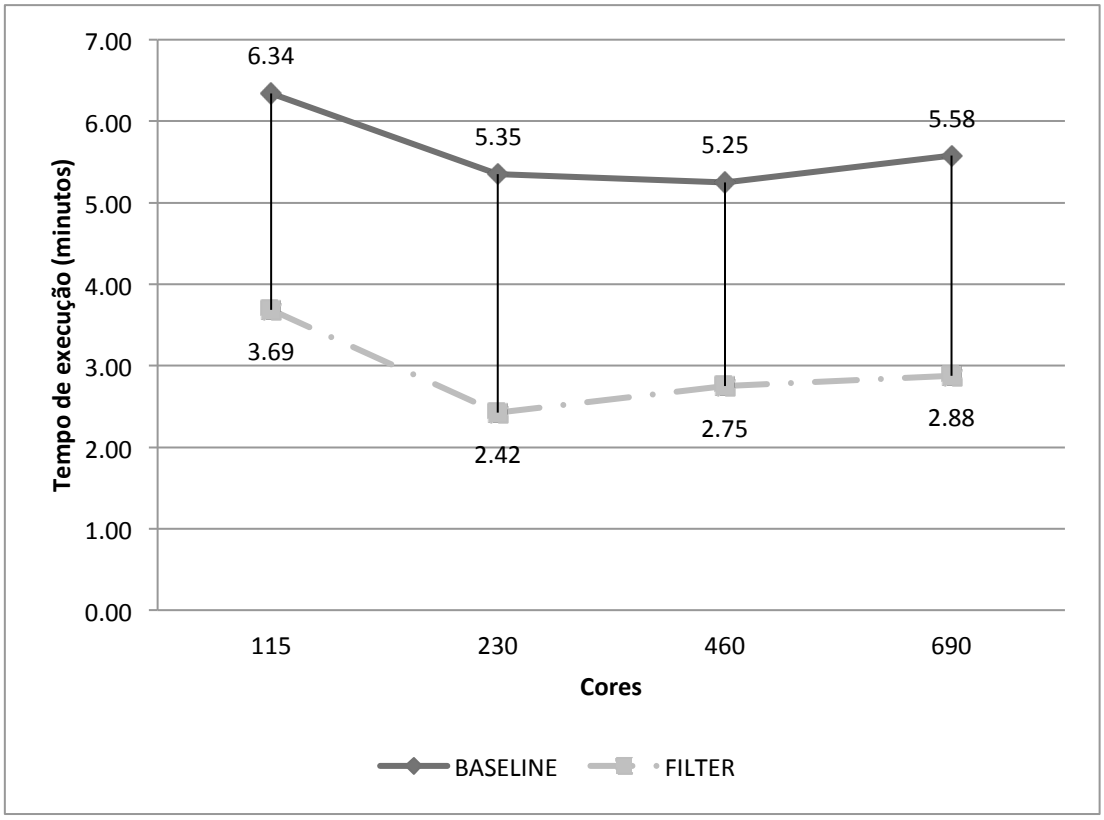


Figura 25 – Comparação da derivação *filter* com a *baseline*

A derivação *filter* se mostrou melhor quando os recursos disponíveis são limitados e quando incrementado o número de núcleos; isto não influencia esta derivação, pois o número de ativações é constante. Realizando a comparação da derivação *split* com a *baseline* na Figura 26, pode ser observado que o comportamento mencionado anteriormente (fragmentação de arquivos) somente é vantajoso quando existem recursos disponíveis para a alocação de todas as ativações. Com 230 núcleos a derivação não possui grande ganho no tempo de execução, porém com o aumento dos recursos disponíveis, esse tempo é muito menor em comparação com o *baseline*.

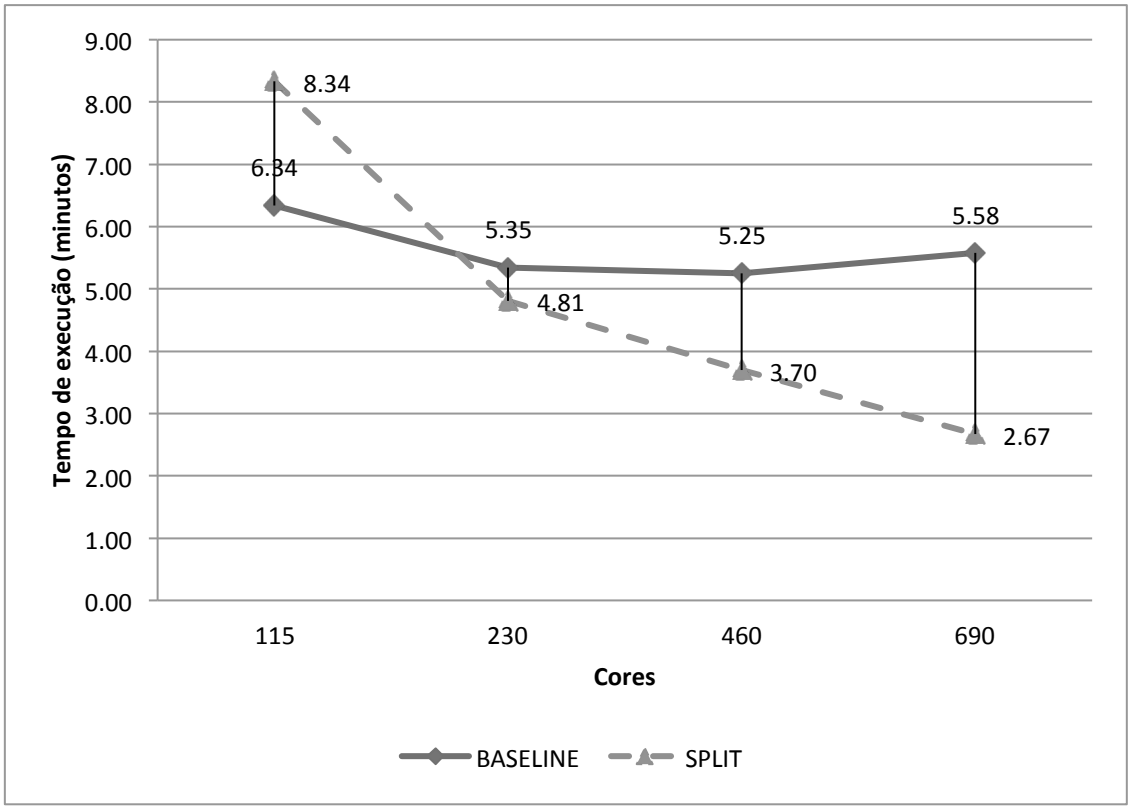


Figura 26 – Comparação da derivação *split* com a *baseline*

Ao serem alocados menos recursos na derivação *split,* pode ser observado um aumento no tempo de execução. Com 115 núcleos, esta derivação tem um aumento no tempo de execução de 31,4%, mas a mesma mostrou uma redução no tempo de execução de 10,1%, 29,6% e 52,2% para 230, 460 e 690 núcleos, respectivamente.

Finalmente, a Figura 27 apresenta a comparação entre a derivação *splitfilter* e a *baseline.* Pode ser observado um comportamento similar à derivação *split*, assim existe uma melhora no tempo quando o número de recursos computacionais é incrementado. Já quando executado com 115 núcleos, esta derivação teve uma diminuição no tempo de execução de apenas 2,8%, sendo que a mesma mostrou uma redução no tempo de execução de 42,7%, 47,9% e 59,9% para 230, 460 e 690 núcleos, respectivamente.

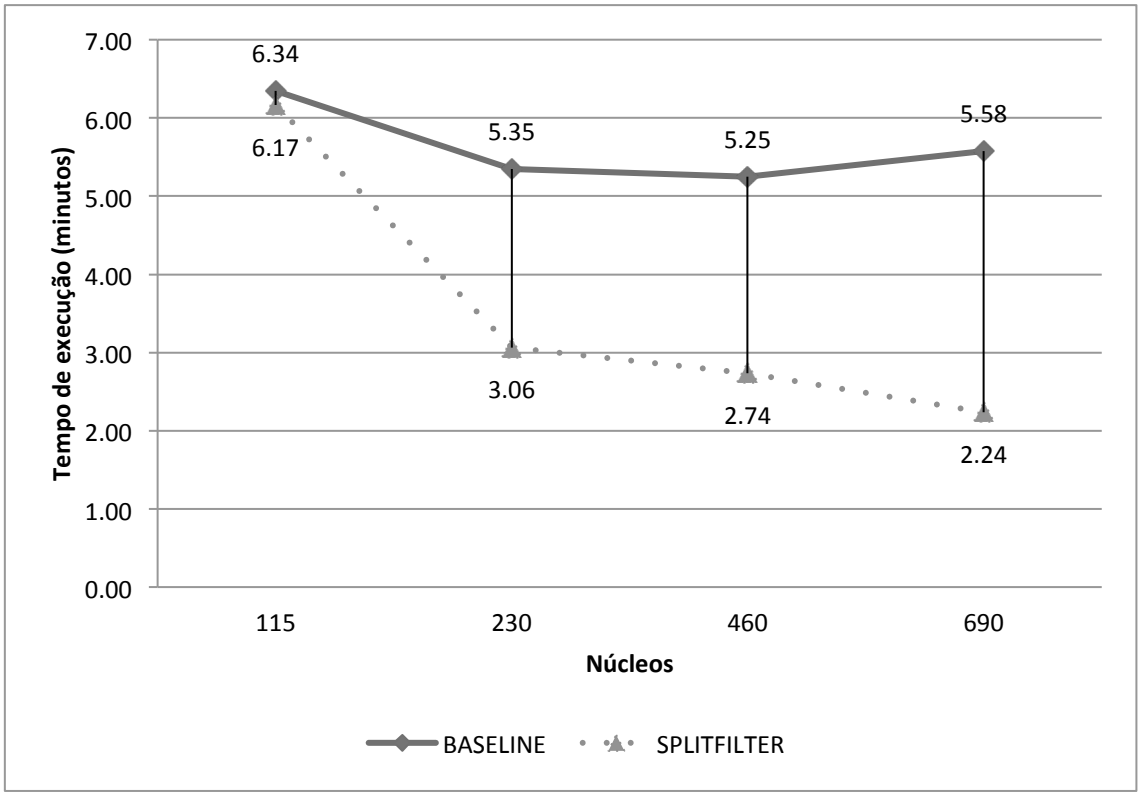


Figura 27 - Derivação *SplitFilter* comparando com a Derivação *Baseline*

5.3.2 Análise de Desempenho dos *Workflows* Derivados da Linha de Experimento SciMG no Cenário 2

Para o segundo cenário, foram utilizados 3 arquivos de entrada na sua configuração original (*i.e.,* não formatados ou reduzidos) com 18.291, 29.404 e 53.310 sequências respectivamente. Quando esta atividade está presente na derivação, o nível de fragmentação adotado é de 200. O tempo de execução sequencial pode chegar em média a 1 hora e 52 minutos. Nesse cenário, as derivações da Linha de Experimento SciMG foram executadas com 230 núcleos no cluster LoboC.

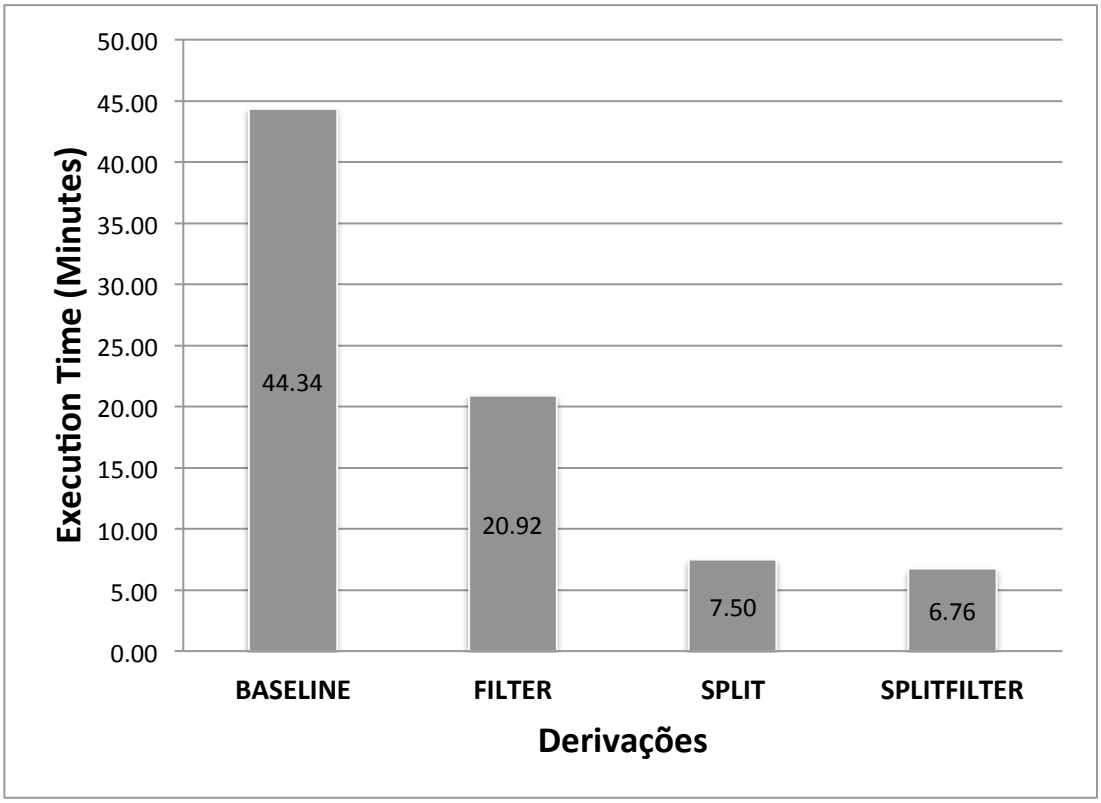


Figura 28 - Tempo médio dos *workflows* derivados da Linha de Experimento SciMG para o Cenário 2

A Figura 28 apresenta o desempenho dos *workflows* derivados para o SciMG do Cenário 2. O *baseline* foi executado em 44,34 minutos e os *workflows filter*, *split* e *splitfilter* foram executados em 20,92 minutos, 7,50 minutos, e 6,76 minutos, respectivamente. Comparando-os com o *workflow baseline*, o *filter*, *split* e *splitfilter* obtiveram uma redução no tempo de execução de 52,8%, 83,1%, e 84,8%, respectivamente.

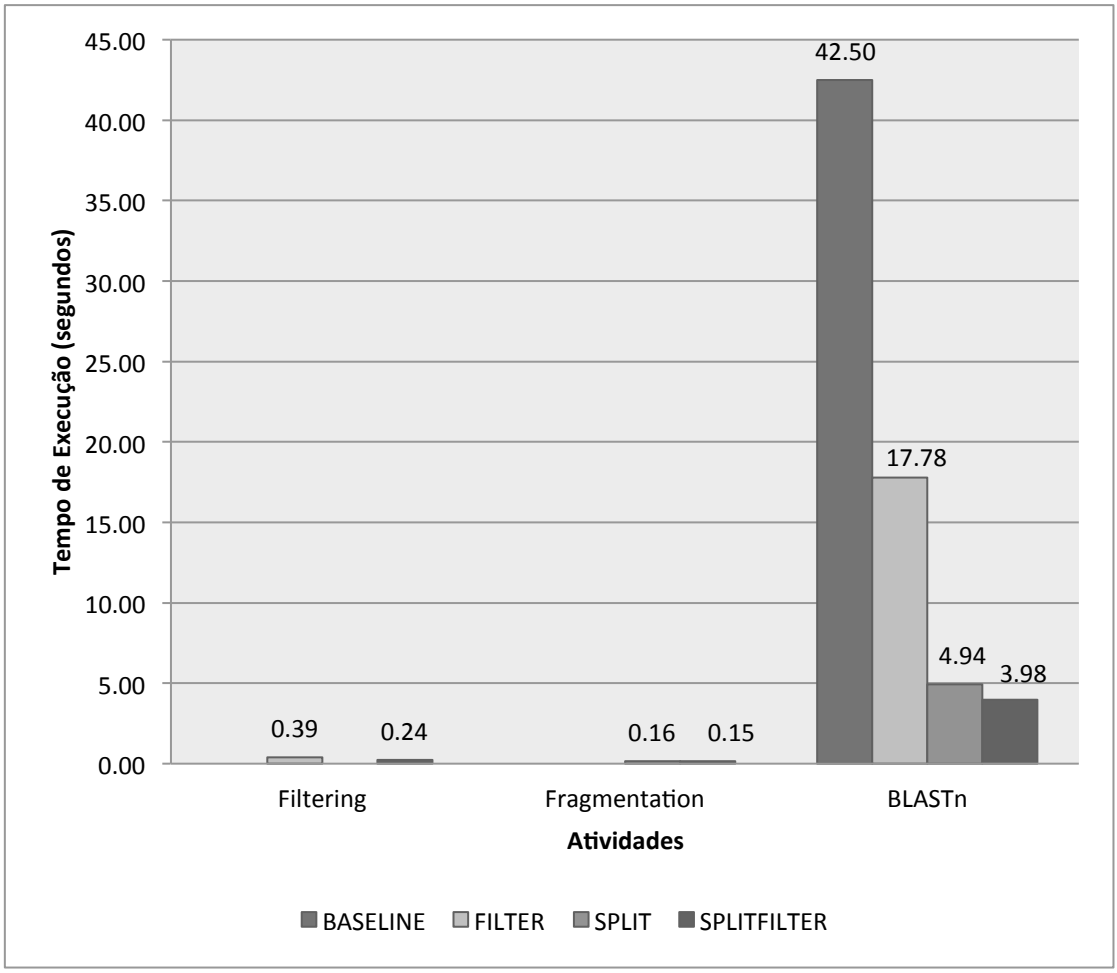


Figura 29 - Impacto em cima da atividade BLASTn no cenário 2

A Figura 29 apresenta o impacto do tempo da atividade BLASTn, considerado o principal gargalo deste experimento. Pode ser observado que as atividades adicionadas nas derivações possuem um impacto muito pequeno, porém elas inserem uma redução considerável no tempo total dos *workflows* derivados com pelo menos uma recomendação. Neste cenário, as recomendações também se mostraram vantajosas, quando utilizados arquivos muito grandes, principalmente com a execução de derivações de fragmentação de dados.

Capítulo 6 - Conclusões

O uso de *workflows* científicos como uma abstração no desenho de experimentos científicos permite definir o fluxo de atividades e dos dados de uma maneira estruturada. As atividades de um *workflow* podem ser reestruturadas através da representação do mesmo como linha de experimento, que levem à derivação de um novo *workflow*, tal que a execução de um experimento científico apresente um melhor desempenho. Esta dissertação apresenta um conjunto de recomendações de forma que os cientistas possam realizar a especificação e/ou alteração de *workflows* científicos de maneira abstrata, focando em obter um melhor desempenho na execução dos seus experimentos. A representação de linhas de experimento permite que as alternativas sejam registradas como pontos de variabilidade e opcionalidade no desenho do *workflow*. Além de ser uma representação natural para as recomendações propostas, a linha de experimento conta com recursos de verificação de correção e proporcionam derivações consistentes com a especificação da linha.

As recomendações propostas se baseiam na inclusão de atividades de redução e paralelização de dados, pois estão diretamente ligadas ao desempenho de *workflows* científicos. As execuções realizadas para esta análise mostraram que há impactos significativos na redução do tempo de execução de um *workflow* científico ao utilizar as recomendações baseadas na especificação de opcionalidade e variabilidade na modelagem de uma linha de experimento.

Esta análise foi realizada com os resultados obtidos após a aplicação das recomendações na linha de experimento de metagenômica SciMG, a qual foi analisada em dois cenários diferentes. O primeiro consome um conjunto de 200 arquivos de entradas com tamanhos homogêneos de dados reais, que foram preparados de maneira a terem quantidades de sequências similares. Já o segundo cenário, consome 3 arquivos na versão original, *i.e.*, sem alterações nem reduções, apresentando tamanhos bem variados e com duas ordens de grandeza maior que no primeiro cenário.

No primeiro cenário foi realizada, inicialmente, uma execução com 230 núcleos e foi observado que a linha de experimento tem uma melhor execução quando na sua derivação é incluída a atividade opcional de filtro, inserida após a aplicação da recomendação. O *workflow,* que foi denominado *filter* derivado da linha de experimento

SciMG, apresenta uma diminuição no tempo de execução de até 54,7% quando comparado com o *workflow baseline,* também executado com 230 núcleos.

Porém, a recomendação de fragmentação de atividades somente mostra seu potencial quando realizada uma análise variando o número de núcleos disponíveis. Com isto foi observado que aumentando o número de recursos disponíveis para a execução dos *workflows* derivados e apresentando a atividade de fragmentação inserida após a recomendação, existe uma diminuição significativa no tempo de execução. A derivação *split* apresenta uma diminuição de até 52,2% e o *splitfilter* de até 59,9% quando comparadas com a derivação *baseline,* executados em 690 núcleos.

Quando reduzidos os núcleos disponíveis, os *workflows* derivados com a atividade de fragmentação de dados possuem um desempenho similar (no caso do *workflow splitfilter*) ou pior (no caso do *workflow split*) que a derivação *baseline*.

Por outro lado, o segundo cenário visou a mostrar o potencial de paralelização de dados em maior escala, ao ser comparado com o *baseline* com os arquivos originais sem modificação no tamanho deles, isto é, mesma quantidade de sequências, porém somente em 3 arquivos. Esta execução fez com que o *baseline* se comportasse quase como um *workflow* sequencial. Neste cenário, a linha tem uma melhor execução quando na sua derivação utiliza a atividade opcional de filtro. A derivação *splitfilter* do SciMG apresenta uma diminuição na execução do *workflow* de até 84,8% comparado com o *workflow baseline*. Esta diminuição foi bastante significativa, pois o experimento se aproveitou da capacidade da máquina paralela aumentando um nível de paralelização com a fragmentação gerando 200 arquivos para cada arquivo de entrada.

Os resultados obtidos indicaram que as recomendações propostas nesta dissertação podem levar à diminuição significativa no desempenho de experimentos científicos. A solução proposta é totalmente aderente aos conceitos de linha de experimento e uma vez modelada, fica à disposição do cientista para que no momento de derivação, possa ser escolhido *workflow* que apresente um menor tempo de execução.

6.1 Trabalhos Futuros

A abordagem proposta nesta dissertação visa assistir ao cientista no desenho de *workflows* científicos tirando proveito de paralelismo e redução de dados. As recomendações são realizadas de maneira abstrata, visando à utilização do conceito de

linha de experimento para serem derivadas no momento do desenho de diferentes *workflows* científicos, sem se preocupar em qual SGWfC o cientista realizará a execução.

Uma oportunidade em aberto é o desenvolvimento de uma ferramenta que permita ao cientista explorar as recomendações a partir de estimativas de custo de execução. Isto é, uma ferramenta que assista ao cientista na especificação, manipulação e gerenciamento dos experimentos como linhas de experimento e por sua vez, verifique a possibilidade de inserção de atividades de filtro e fragmentação de dados automaticamente, ao mesmo tempo em que indica os tempos de execução esperados. Modelos de custos de execução de *workflows* podem ser adotados e alimentados com base em históricos de execuções prévias, encontrados em *logs* e bases de dados de proveniência.

6.2 Heurísticas para Assistir na Utilização das Recomendações

A utilização destas recomendações pode se mostrar bastante útil na diminuição do tempo de execução dos experimentos científicos, porém a escolha dessas recomendações não é uma tarefa trivial de ser realizada. No momento do desenho da linha, o cientista pode se deparar com distintas questões que devem ser resolvidas antes de inserir as atividades recomendadas:

- Identificar as atividades candidatas para as recomendações: que possuem um tempo consideravelmente maior que as outras atividades no experimento como um todo.
- Verificar dependências no fluxo de dados: de modo a verificar a existência daquelas atividades que possuem uma dependência de dados, o que poderia ser útil para a recomendação de inclusão de filtros.
- Verificar possibilidade de fragmentação: nas atividades candidatas que possuem dados de entrada que podem ser fragmentados.

Uma vez obtida a linha de experimento, a nova questão para o cientista é qual derivação é ideal para a configuração atual do experimento:

- Verificar o conjunto de dados: tamanhos de dados de entradas consideravelmente grandes podem se tornar candidatos a utilizar as recomendações de fragmentação e filtro. Porém conjuntos de dados com

tamanhos muito pequenos podem não ter um impacto tão significativo quando utilizada a recomendação de fragmentação.

- Analisar o desempenho dos programas: levando em consideração distintos tamanhos de arquivos, o que poderia ajudar na escolha do nível de fragmentação que possa favorecer ao paralelismo sem prejudicar a execução.
- Analisar a configuração do ambiente de execução disponível: que se refere à disponibilização de ambientes de alto desempenho, dado que o número de recursos disponíveis pode variar, considerando o tipo de ambientes como nuvens ou *clusters* de supercomputadores:
    - Para poucos recursos disponíveis, pode-se optar pela recomendação de filtro.
    - Para uma grande quantidade de recursos disponíveis, pode-se optar pela recomendação de fragmentação.

Capítulo 7 - Referências Bibliográficas

Altintas, I., Berkley, C., Jaeger, E., Jones, M., Ludascher, B., Mock, S., (2004), "Kepler: an extensible system for design and execution of scientific workflows". In: Scientific and Statistical Database Management, p. 423–424, Greece.

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., Lipman, D. J., (1997), "Gapped BLAST and PSI-BLAST: a new generation of protein database search programs", Nucleic acids research, v. 25, n. 17 (set.), p. 3389–3402.

Artem M. Chirkin, Adam S.Z. Belloum, Sergey V. Kovalchuk, Marc X. Makkes,Mikhail A. Melnikb, Alexander A. Visheratin, Denis A. Nasonov. "Execution time estimation for workflow scheduling"

Ball, N. M., Brunner, R. J., (2009), "Data Mining and Machine Learning in Astronomy", arXiv:0906.2173 (jun.)

Bertino, E., Bernstein, P., Agrawal, D., Davidson, S., Dayal, U., Franklin, M., Gehrke, J., Haas, L., Halevy, A., Han, J., Jadadish, H. V., Labrinidis, A., Madden, S., Papokonstantinou, Y., Patel, J., *et al*., (2011), "Challenges and Opportunities with Big Data"

Callahan, S. P., Freire, J., Santos, E., Scheidegger, C. E., Silva, C. T., Vo, H. T., (2006), "VisTrails: visualization meets data management". In: SIGMOD International Conference on Management of Data, p. 745–747, Chicago, Illinois, USA.

Cardoso, L. F., de Souza, J. M., Marques, C., (2002), "A collaborative approach to the reuse of scientific experiments in the Bill of Experiments tool". In: 7th International Conference on Computer Supported Cooperative Work in Design, 2002, p. 296–301

Carpenter, B., Getov, V., Judd, G., Skjellum, A., Fox, G., (2000), "MPJ: MPI-like message passing for Java", Concurrency: Practice and Experience, v. 12, n. 11, p. 1019–1038.

Cavalcanti, M. C., Targino, R., Baião, F., Rössle, S. C., Bisch, P. M., Pires, P. F., Campos, M. L. M., Mattoso, M., (2005), "Managing structural genomic workflows using web services", Data & Knowledge Engineering, v. 53, n. 1, p. 45-74.

Chinchuluun, A., Xanthopoulos, P., Tomaino, V., Pardalos, P. M., (2010), "Data Mining Techniques in Agricultural and Environmental Sciences", International Journal of Agricultural and Environmental Information Systems, v. 1, n. 1 (jan.), p. 26–40.

Dantas, M., (2005), "Clusters Computacionais", Computação Distribuída de Alto Desempenho: Redes, Clusters e Grids Computacionais, 1 edRio de Janeiro: Axcel Books, p. 145–180.

Davidson, S. B., Freire, J., (2008), "Provenance and scientific workflows: challenges and opportunities". In: ACM SIGMOD international conference on Management of data, p. 1345–1350, Vancouver, Canada.

Deelman, E., Chervenak, A., (2008), "Data Management Challenges of Data-Intensive Scientific Workflows". In: CCGRID '08, p. 687–692

Deelman, E., Mehta, G., Singh, G., Su, M.-H., Vahi, K., (2007), "Pegasus: Mapping Large-Scale Workflows to Distributed Resources", Workflows for e-Science, Springer, p. 376–394.

Deelman, E., Gannon, D., Shields, M., Taylor, I., (2009), "Workflows and e-Science: An overview of workflow system features and capabilities", Future Generation Computer Systems, v. 25, n. 5, p. 528–540.

Deshpande, A. and Hellerstein, L. (2012), "Parallel pipelined filter ordering with precedence constraints". ACM Trans. Algor. 8, 4, Article 41 (September 2012), 38 pages

Evsukoff, A. G. Lima, B. S. L. P. de. Ebecken, N. F. F. (2011), "Long-term runoff modeling using rainfall forecasts with application to the Iguaçu River Basin". Water resources management, v. 25, n. 3, p. 963–985.

Freire, J., Koop, D., Santos, E., Silva, C. T., (2008), "Provenance for Computational Tasks: A Survey", Computing in Science and Engineering, v.10, n. 3, p. 11–21.

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J., (2007), "Examining the Challenges of Scientific Workflows", Computer, v. 40, n. 12, p. 24-32.

Greenwood, M., Goble, C., Stevens, R., Zhao, J., Addis, M., Marvin, D., Moreau, L., Oinn, T., (2003), "Provenance of e-Science Experiments - Experience from Bioinformatics", UK OST e-Science second All Hands Meeting, v. 4, p. 223–226.

Guerra, G. Rochinha, F.. (2009), "A Sparse Grid Method Applied to Stochastic Fluid-Structure Interaction". COBEM, 2009, Gramado, Brazil.

Guerra, G. Rochinha, F. Elias, R.; *et al*. (2012), "Uncertainty quantification in computational predictive models for fluid dynamics using a workflow management engine". International Journal for Uncertainty Quantification, v. 2, n. 1

Guerra, G. Rochinha, F. Elias, R.; *et al*. (2009), "Scientific workflow management system applied to uncertainty quantification in large eddy simulation". Congresso Ibero Americano de Métodos Computacionais em Engenharia. . p.1–13, .

Hartman, A. L.; Riddle, S.; McPhillips, T.; Ludascher, B.; Eisen, J. A. (2010), "Introducing WATERS: a workflow for the alignment, taxonomy, and ecology of ribosomal sequences". BMC bioinformatics, v. 11, n. 1, p. 1,.

Hey, T., S. Kurbanoğlu; U. Al; P. L. Erdoğan; Y. Tonta; N. Uçak (2012), "E-Science and Information Management, Communications in Computer and Information Science". The Fourth Paradigm – Data-Intensive Scientific Discovery.p.1–1. Springer Berlin Heidelberg.

Hueske, F. Peters, M. Sax, M. J. Rheinlander, A. Bergmann, R. Krettek, A. Tzoumas, K. (2012), "Opening the Black Boxes in Data Flow Optimization". PVLDB 5

Ho-Sik Seok, Woonyoung Hong, Jaebum Kim (2014), "Estimating the composition of species in metagenomes by clustering of next-generation read sequences"

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M. R., Li, P., Oinn, T., (2006), "Taverna: a tool for building and running workflows of services", Nucleic Acids Research, v. 34, n. 2, p. 729–732.

Jarrard, R. D., (2001), "Scientific Methods". Online book, Url.: http://emotionalcompetency.com/sci/booktoc.html.

Lengauer, T., (2002), "Bioinformatics--from genomes to drugs". Weinheim, Wiley-VCH.

Lesk, A. M., (2008), "Introduction to bioinformatics". Oxford; New York, Oxford University Press.

Lins, E. F.; Elias, R. N.; Guerra, G. M.; Rochinha, F. A.; Coutinho, A. L. (2009),"Edge-based finite element implementation of the residual-based variational multiscale method". International Journal for Numerical Methods in Fluids, v. 61, n. 1, p. 1–22.

Liu, X. (2015), "An ETL Optimization Framework Using Partitioning and Parallelization"

Marinho, A. Oliveira, D. Ogasawara, E. Silva, V. Ocaña, K. Murta, L. Braganholo, V. Mattoso, M (2017) "Deriving scientific workflows from algebraic experiment lines: A practical approach" Future Generation Computer SystemsVolume 68, March 2017, Pages 111–127

Marinos, A., Briscoe, G., (2009), "Community Cloud Computing". In: Proceedings of the 1st International Conference on Cloud Computing, p. 472–484, Beijing, China.

Martinho, W., Ogasawara, E., Oliveira, D., Chirigati, F., Santos, I., Travassos, G. H. T., Mattoso, M., (2009), "A Conception Process for Abstract Workflows: An Example on Deep Water Oil Exploitation Domain". In: 5th IEEE International Conference on e-Science, Oxford, UK.

Mattoso, M., Werner, C., Travassos, G., Braganholo, V., Murta, L., Ogasawara, E., Oliveira, F., Martinho, W., (2009), "Desafios no Apoio à Composição de Experimentos Científicos em Larga Escala". In: SEMISH - CSBC, p. 307–321, Bento Gonçalves, Rio Grande do Sul, Brazil.

Mattoso, M., Coutinho, A., Elias, R., Oliveira, D., Ogasawara, E., (2010a), "Exploring Parallel Parameter Sweep in Scientific Workflows". In: WCCM - World Congress on Computational Mechanics, Australia.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Murta, L., Ogasawara, E., Oliveira, D., Cruz, S. M. S. da, Martinho, W., (2010b), "Towards Supporting the Life Cycle of Large-scale Scientific Experiments", International Journal of Business Process Integration and Management, v. 5, n. 1, p. 79–92

Meiring, T.L., Bauer, R., Scheepers, I., Ohloff, C., Tuffin, M., Cowan, D.A., (2011),“Metagenomics and beyond: current approaches and integration with complementary technologies” Biochemistry, Moscow.

Moreau, L., Freire, J., Futrelle, J., McGrath, R., Myers, J., Paulson, P., (2008), "The Open Provenance Model: An Overview", Provenance and Annotation of Data and Processes, , p. 323–326.

M. Wilde, M. Hategan, J. M. Wozniak, B. Clifford, D. S. Katz, and I. Foster, (2011),“Swift: A language for distributed parallel scripting,” Parallel Computing, no. 37(9), pp. 633–652.

Northrop, L, (2002), “SEI's software product line tenets,” IEEE Software, vol. 19, pp. 32-40.

Ocaña, K. A. C. S., Oliveira, D., Ogasawara, E., Dávila, A. M. R., Lima, A. A. B., Mattoso, M., (2011), "SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes", In: Norberto de Souza, O., Telles, G. P., Palakal, M. [orgs.] (eds), Advances in Bioinformatics and Computational Biology, , chapter 6832, Berlin, Heidelberg: Springer, p. 66–70.

Ocaña, K. A. C. S., Oliveira, D. de, Horta, F., Dias, J., Ogasawara, E., Mattoso, M., (2012a), "Exploring Molecular Evolution Reconstruction Using a Parallel Cloud-based Scientific Workflow", Advances in Bioinformatics and Computational Biology, , chapter 7409, Berlin, Heidelberg: Springer, p. 179–191.

Ocaña, K. A. C. S., Oliveira, D., Dias, J., Ogasawara, E., Mattoso, M., (2012b), "Discovering Drug Targets for Neglected Diseases Using a Pharmacophylogenomic Cloud Workflow". In: Proceedings of the IEEE 8th International Conference on e-Science, USA, Chicago.

Ocaña, K. A. C. S., Oliveira, F., Dias, J., Ogasawara, E., Mattoso, M., (2013), "Designing a parallel cloud based comparative genomics workflow to improve phylogenetic analyses", Future Generation Computer Systems, v. 29, n. 8, p. 2205–2219.

Ogasawara, E., Paulino, C., Murta, L., Werner, C., Mattoso, M., (2009a), "Experiment Line: Software Reuse in Scientific Workflows". In: Scientific and Statistical Database Management, p. 264–272, New Orleans, Louisiana, USA.

Ogasawara, E., Oliveira, D., Chirigati, F., Barbosa, C. E., Elias, R., Braganholo, V., Coutinho, A., Mattoso, M., (2009b), "Exploring many task computing in scientific workflows". In: Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, p. 1-10, Portland, Oregon, USA.

Ogasawara, E, (2011), Uma Abordagem Algébrica para Workflows Científicos com Dados em Larga Escala. Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2011, UFRJ/COPPE

Oliveira, F., Murta, L., Werner, C., Mattoso, M., (2008), "Using Provenance to Improve Workflow Design". In: IPAW, p. 136 – 143, Salt Lake City, UT, USA.

Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M., (2010a), "SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows". In: 3rd International Conference on Cloud Computing, p. 378–385, Washington, DC, USA.

Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M., (2010b), "An Adaptive Approach for Workflow Activity Execution in Clouds". In: International Workshop on Challenges in e-Science - SBAC, p. 9–16, Petrópolis, RJ - Brazil.

Oliveira, D., Ogasawara, E., Seabra, F., Silva, V., Murta, L., Mattoso, M., (2010c), "GExpLine: A Tool for Supporting Experiment Composition", Provenance and Annotation of Data and Processes, Springer Berlin / Heidelberg, p. 251–259.

Oliveira, D., Ocaña, K., Ogasawara, E., Dias, J., Baião, F., Mattoso, M., (2011), "A Performance Evaluation of X-Ray Crystallography Scientific Workflow Using SciCumulus". In: IEEE International Conference on Cloud Computing (CLOUD), p. 708–715, Washington, D.C., USA.

Oliveira, D., (2012), Uma Abordagem de Apoio à Execução Paralela de Workflows Científicos em Nuvens de Computadores. Tese (doutorado) – UFRJ/ COPPE/ Programa de Engenharia de Sistemas e Computação, 2012., UFRJ/COPPE

Oliveira, D., Ogasawara, E., Ocaña, K., Baião, F., Mattoso, M., (2012), "An adaptive parallel execution strategy for cloud-based scientific workflows", Concurrency and Computation: Practice and Experience, v. 24, n. 13 (set.), p. 1531–1550.

Oliveira, D., Ocaña, K. A. C. S., Ogasawara, E., Dias, J., Gonçalves, J., Baião, F., Mattoso, M., (2013), "Performance evaluation of parallel strategies in public clouds: A study with phylogenomic workflows", Future Generation Computer Systems, v. 29, n. 7 (set.), p. 1816–1825.

Pereira, G. C.; Ebecken, N. F. "Combining in situ flow cytometry and artificial neural networks for aquatic systems monitoring". Expert Systems with Applications, v. 38, n. 8, p. 9626–9632, 2011.

Pietri, I. Juve, G. Deelman, E. Sakellariou, R., (2014) "A Performance Model to Estimate Execution Time of Scientific Workflows on the Cloud", The International Conference for High Performance Computing, Networking, Storage, and Analysis

Porto, F.; Moura, A. M.; Silva, F. C.; *et al*. "A metaphoric trajectory data warehouse for Olympic athlete follow-up". Concurrency and Computation: Practice and Experience, v. 24, n. 13, p. 1497–1512, 2012.

Pruitt, K. D., Tatusova, T., Klimke, W., Maglott, D. R., (2009), "NCBI Reference Sequences: current status, policy and new initiatives", Nucleic Acids Research, v. 37, n. Database issue (jan.), p. D32–D36.

Silva, V. Oliveira, D. Mattoso, M (2014), "SciCumulus 2.0: Um Sistema de Gerência de Workflows Científicos para Nuvens Orientado a Fluxo de Dados". 29th SBBD – Demos and Applications Session – ISSN 2316-5170.

Soanes, C., Stevenson, A., (2003), Oxford Dictionary of English. 2nd Revised edition ed. Oxford University Press.

Stevens, R., Zhao, J., Goble, C., (2007), "Using provenance to manage knowledge of In Silico experiments", Brief Bioinform (maio.), p. bbm015.

Taylor, I. J., Deelman, E., Gannon, D. B., Shields, M., (2007a), Workflows for e- Science: Scientific Workflows for Grids. 1 ed. Springer.

Taylor, I., Shields, M., Wang, I., Harrison, A., (2007b), "The Triana Workflow Environment: Architecture and Applications", Workflows for e-Science, Springer, p. 182 320-339.

Thomas T, Gilbert J, Meyer F. (2012) "Metagenomics—a guide from sampling to data analysis". Microb Inform Exp. 2012;2:3. doi: 10.1186/2042-5783-2-3

Tracy L. Meiring, Rolene Bauer, Ilana Scheepers, Colin Ohlhoff, Marla I. Tuffin and Donald A. Cowan. "Metagenomics and beyond: current approaches and integration with complementary technologies"

Travassos, G. H., Barros, M. O., (2003), "Contributions of In Virtuo and In Silico Experiments for the Future of Empirical Studies in Software Engineering". In: 2nd Workshop on Empirical Software Engineering the Future of Empirical Studies in Software Engineering, p. 117–130, Rome, Italy.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., Lindner, M., (2009), "A break in the clouds: towards a cloud definition", SIGCOMM Comput. Commun. Rev., v. 39, n. 1, p. 50–55.

Walker, E., Guiang, C., (2007) "Challenges in executing large parameter sweep studies across widely distributed computing environments". Workshop on Challenges of large applications in distributed environments, 11–18.

Wang, L., Tao, J., Kunze, M., Castellanos, A. C., Kramer, D., Karl, W., (2008), "Scientific Cloud Computing: Early Definition and Experience". In: 10th IEEE HPCC, p. 825–830, Los Alamitos, CA, USA.

Wilde, M., Hategan, M., Wozniak, J. M., Clifford, B., Katz, D. S., Foster, I., (2011), "Swift: A language for distributed parallel scripting", Parallel Computing, n. 37(9), p. 633–652.

WfMC, I., (2009), Binding, WfMC Standards, WFMC-TC-1023, http://www. wfmc. org, 2000.

Zhang, J., Chiodini, R., Badr, A., Zhang, G., (2011), "The impact of next-generation sequencing on genomics", Journal of Genetics and Genomics, v. 38, n. 3 (mar.), p. 95–109.

Zhang, S., Kumar, K., Jiang, X., Wallqvist, A., Reifman, J., (2008), "DOVIS: an implementation for high-throughput virtual screening using AutoDock", BMC bioinformatics, v. 9, p. 126.

Zhao, Y., Hategan, M., Clifford, B., Foster, I., von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T., Wilde, M., (2007), "Swift: Fast, Reliable, Loosely Coupled Parallel Computation". In: 3rd IEEE World Congress on Services, p. 206, 199, Salt Lake City, USA.