ANÁLISE DE DADOS CIENTÍFICOS BASEADA EM ALGORITMOS DE INDEXAÇÃO BITMAP

José Vitor Delgado Leite

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.

Orientadora: Marta Lima de Queirós Mattoso

Rio de Janeiro

Março de 2017

ANÁLISE DE DADOS CIENTÍFICOS BASEADA EM ALGORITMOS DE INDEXAÇÃO BITMAP

José Vitor Delgado Leite

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:

_____

Profª. Marta Lima de Queirós Mattoso, D.Sc.

_____

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

_____

Prof. Fabio Andre Machado Porto, D.Sc.

RIO DE JANEIRO, RJ - BRASIL
MARÇO DE 2017

Leite, José Vitor Delgado

Análise de Dados Científicos Baseada em Algoritmos de Indexação Bitmap / José Vitor Delgado Leite. – Rio de Janeiro: UFRJ/COPPE, 2017.

X, 81 p.: il.; 29,7 cm.

Orientadores: Marta Lima de Queirós Mattoso

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

Referências Bibliográficas: p. 76-81.

1. Análise de Dados Científicos. 2. Indexação. 3. Algoritmo Bitmap. I. Mattoso, Marta Lima de Queirós II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

À minha mãe, Maria Esther.
A melhor pessoa que conheci em toda
minha vida, meu maior exemplo.

AGRADECIMENTOS

À minha mãe, Maria Esther, que pôde participar do começo dessa jornada, como sempre dando muito amor, carinho e apoio. Agradeço a você por toda a motivação para realizar meus sonhos, fazer o meu próprio caminho.

À minha família que sempre esteve presente ao meu lado, me incentivando e me proporcionando toda a base necessária para dar continuidade ao meu trabalho. Em especial ao meu pai, José Luiz, por me dar todas as condições para minha formação, além do amor incondicional, e à minha irmã, Mariana, pelo apoio e carinho mesmo de longe.

À minha namorada, Helena, pelo amor, amizade e companheirismo que foram fundamentais para realizar mais esta etapa da minha vida. Agradeço por você confiar em mim e estar sempre ao meu lado em todas as ocasiões.

À Professora Marta Mattoso, minha orientadora, obrigado pela confiança no meu potencial e no meu trabalho, pelas sugestões que acrescentaram grande valor e pela atenciosa orientação acadêmica proporcionada.

Ao Vítor Silva, um agradecimento especial por ceder tanto tempo de trabalho com preciosas sugestões e compartilhar suas ideias e experiências durante todo o desenvolvimento deste trabalho.

À Silvia Benza e Renan Souza, pela amizade durante este percurso, com a colaboração ao longo das matérias do mestrado e o apoio ao desenvolvimento do projeto. Aos meus demais amigos, obrigado por estarem ao meu lado.

Aos membros da banca, Professor Alexandre Assis e Professor Fabio Porto, por aceitarem o convite para compor a banca da minha dissertação de mestrado.

A todos os professores e funcionários do PESC que fizeram parte dessa jornada, por agregar conhecimento e contribuir para que este trabalho fosse concluído.

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

ANÁLISE DE DADOS CIENTÍFICOS BASEADA EM ALGORITMOS DE INDEXAÇÃO BITMAP

José Vitor Delgado Leite

Março/2017

Orientador: Marta Lima de Queirós Mattoso

Programa: Engenharia de Sistemas e Computação

As simulações computacionais de larga escala usualmente consomem e produzem grandes volumes de arquivos de dados científicos, os quais podem apresentar diferentes formatos. Os usuários, por sua vez, comumente necessitam analisar dados específicos de domínio baseados em elementos de dados relacionados por meio de múltiplos arquivos gerados ao longo da execução de simulações computacionais. Diferentes soluções existentes, como o FastBit e o NoDB, buscam apoiar esta análise por meio da indexação de dados científicos de forma a permitir o acesso direto a elementos específicos de regiões de interesse em arquivos de dados científicos. Entretanto, tais soluções são limitadas a analisar um único arquivo de dados científicos por vez, ao passo que são utilizadas apenas após a execução de simulações computacionais. A arquitetura ARMFUL propõe uma solução capaz de garantir a gerência do fluxo de dados, registrar elementos de dados científicos relacionados em uma base de proveniência e combinar técnicas de análise de arquivos de dados científicos em tempo de execução. A partir de um modelo de dados que apoia a integração de dados de execução da simulação computacional e dados de domínio, a arquitetura permite consultas a elementos de dados relacionados por múltiplos arquivos. Esta dissertação propõe a implementação de instâncias dos componentes de indexação de dados científicos e de processamento de consultas presentes na arquitetura ARMFUL, buscando reduzir o tempo total de ingestão de dados na base de proveniência e apoiar a análise exploratória de dados científicos.

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

RAW DATA ANALYSIS BASED ON BITMAP INDEXING ALGORITHMS

José Vitor Delgado Leite

March/2017

Advisor: Marta Lima de Queirós Mattoso

Department: Systems and Computer Engineering

Computer simulations in large-scale often consume and produce a large volume of raw data files, which can be presented in different formats. Users usually need to analyze domain-specific data based on data elements related through multiple files generated along the computer simulation execution. Different existing solutions, like FastBit and NoDB, intend to support this analysis by indexing raw data in order to allow direct access to specific elements in raw data files regions of interest. However, those solutions are limited to analyze a single raw data file at once, while they are used only after computer simulation execution. The ARMFUL architecture proposes a solution capable of guarantee dataflow management, record related raw data elements in a provenance database and combine techniques of raw data file analysis at runtime. Through a data model that supports integration between computer simulation execution data and domain data, the architecture allows for queries on data elements related by multiple files. This dissertation proposes the implementation of instances of raw data indexing and query processor components presented by ARMFUL architecture, aiming to reduce the elapsed time of data ingestion in the provenance database and support raw data exploratory analysis.

ÍNDICE

LISTAGEM DE FIGURAS

LISTAGEM DE TABELAS

Capítulo 1 Introdução

O uso de simulações computacionais como método analítico tem se tornado uma atividade fundamental para os cientistas (PIDD, 1994), visto que permite o processamento de modelos computacionais cada vez mais complexos. Tais simulações são normalmente baseadas no encadeamento de programas científicos, os quais são caracterizados pela geração de um grande volume de dados. A simulação pode tanto gerar alguns poucos arquivos de tamanho significativo ou inúmeros arquivos de tamanho médio, mas que juntos geram esse grande volume de dados. Devido à larga escala, o resultado final pode levar muito tempo (*i.e.*, dias ou semanas) para ser obtido, fator que pode ser atenuado realizando a execução das simulações de forma paralela. Neste cenário, os ambientes de Processamento de Alto Desempenho (PAD) são fundamentais na aplicação de técnicas de paralelismo em simulações computacionais, atuando diretamente na redução do tempo de execução.

A partir do apoio da modelagem de simulações computacionais e as suas execuções em ambientes de PAD, a geração de dados tem aumentado consideravelmente, sendo capaz de atingir uma vazão de petabytes por dia (CRITCHLOW e VAN DAM, 2013). Por outro lado, uma vez que operações de leitura e escrita em disco são custosas do ponto de vista temporal, as simulações computacionais que envolvem altas taxas de produção de dados apresentam atrasos em sua execução ocasionados pelo custo de acesso às unidades de armazenamento. Este mesmo custo compromete a análise exploratória de conjuntos de dados em larga escala, o que motiva a adoção de novas abordagens voltadas para a redução do tempo de acesso aos dados armazenados em disco (ALAGIANNIS *et al.*, 2012, KARPATHIOTAKIS *et al.*, 2014, ROMOSAN *et al.*, 2013). Esta dissertação está inserida neste contexto motivacional, uma vez que objetiva contribuir para a redução do tempo de acesso a dados de simulações computacionais para a análise exploratória durante a condução da simulação.

A importância da etapa de análise dos dados científicos gerados por modelos computacionais está na informação fornecida ao usuário sobre as próprias condições e características da execução da simulação computacional. Os dados científicos presentes em arquivos são capazes de indicar, do ponto de vista do domínio, possíveis falhas ou anormalidades ao longo da execução de uma simulação. Portanto, considerando a

análise exploratória de dados durante a condução da simulação, esta característica pode ser de grande valia, uma vez que erros podem ser identificados ainda em tempo de execução e, consequentemente, a implementação pode ser interrompida e adaptada para uma nova submissão. Com tal conhecimento, os cientistas também são capazes de ajustar os parâmetros de execução de modo a refinar o processo de simulação, caso necessário. Este fator é de extrema importância considerando simulações computacionais em larga escala que demoram horas ou até mesmo dias, posto que os resultados parciais podem ajudar a identificar divergências na solução muito antes do término da mesma, refletindo em uma economia de tempo e recursos computacionais.

1.1. Caracterização do problema

A possibilidade de realizar consultas que envolvem dados relacionados ao domínio da simulação, antes mesmo do fim do processamento, permite que o usuário verifique e confirme certas características do fenômeno investigado de antemão. No entanto, simulações computacionais são, em sua maioria, formadas por um encadeamento de modelos computacionais compondo um fluxo entre os dados de entrada e saída de cada um deles. Isso significa que, em muitos casos, os dados científicos encontram-se dispersos em diferentes arquivos, os quais podem apresentar formatos distintos após serem gerados por programas científicos diferentes e em etapas distintas (por exemplo, os passos de tempo da simulação), o que dificulta a análise exploratória.

A abordagem tradicional de análise dos dados específicos do domínio ocorre por meio de uma etapa de pós-processamento. Esta etapa consiste na análise dos dados científicos produzidos pelos modelos computacionais somente após o término da execução da simulação. Nesse cenário, o usuário realiza o acesso aos dados científicos formados por elementos que representam características intrínsecas ao domínio observado. A análise no pós-processamento vem se tornando um fator limitante, uma vez que o volume de dados gerados pelos modelos computacionais aumenta mais rápido que a velocidade de acesso aos dados armazenados em discos (CRITCHLOW e VAN DAM, 2013).

O processo de acesso, por sua vez, busca extrair elementos que compõem o dado científico, o qual muitas vezes está contido em diferentes arquivos específicos. Estes arquivos podem apresentar diferentes formatos, desde modelos mais genéricos para

dados científicos, como o HDF5[1] e o NetCDF[2], até característicos para um determinado domínio, como FITS (HANISCH *et al.*, 2001) para astronomia. Portanto, além do custo adicional do armazenamento de dados em simulações computacionais em larga escala, a pluralidade de formatos de arquivos científicos gerados nestes cenários confere aos cientistas a necessidade de desenvolver diferentes mecanismos para o acesso aos dados científicos.

A manipulação de arquivos se torna, então, uma tarefa ainda mais complexa, visto que geralmente é realizada de forma manual e não automatizada, logo, mais propensa a erros. De modo a reverter tais desvantagens, diversos trabalhos buscam aperfeiçoar os meios de descoberta de conhecimento a partir dos dados gerados por modelos computacionais em larga escala. Enquanto algumas abordagens visam à extração dos elementos de dados, *i.e.*, buscam capturar o conteúdo dos arquivos em seu formato original (LOFSTEAD *et al.*, 2008, PARAVIEW, 2011), outras tecnologias se baseiam na indexação de dados científicos, na qual são criadas referências (*i.e.*, ponteiros) para os elementos de dados contidos em arquivos a fim de reduzir o custo, em termos de tempo, para acessos futuros a esses dados (LASLUISA *et al.*, 2015, WU *et al.*, 2009).

Tais abordagens também estão inseridas no contexto de pesquisas que investigam o processamento *in situ* (KIM *et al.*, 2011), no qual a análise exploratória de dados é realizada durante a execução da simulação computacional. Ou seja, os dados científicos são analisados na medida em que são produzidos e não mais apenas na etapa de pós-processamento. O processo *in situ* considera que o acesso e a indexação de dados científicos são favorecidos pelo mesmo local de acesso aos dados, seja na máquina em que os dados estão sendo gerados, seja por estruturas de dados em memória. Dessa forma, ao tirar proveito do mesmo local de acesso aos dados, o tempo total de carga de dados para a análise exploratória que a etapa de pós-processamento apresenta pode ser consequentemente reduzido, visto que o custo das operações de leitura e escrita de dados são minimizados. Além disso, dado o volume de dados gerado, soluções de análise de dados e visualização *in situ* possibilitam a tomada de decisões durante a

[1] http://www.hdfgroup.org/HDF5

[2] http://www.unidata.ucar.edu/software/netcdf

execução, como por exemplo não gravar todos os dados gerados em disco (BAUER *et al.*, 2016).

As soluções existentes para a análise de dados científicos podem ser divididas em quatro categorias. A primeira categoria representa as soluções *ad-hoc*, ou seja, os próprios usuários desenvolvem programas específicos para cada análise desejada. O acesso e a análise de dados científicos expressos em sua forma bruta dentro de arquivos de formatos distintos ainda são largamente realizados por meio do desenvolvimento de programas específicos de acordo com a análise requisitada pelo usuário. Isso significa que os próprios usuários precisam elaborar mecanismos capazes de acessar as regiões de interesse dos arquivos e interpretar esses dados científicos, tarefa que, além de não ser trivial, é propensa a erros.

A segunda categoria também atua diretamente sobre o arquivo científico, porém conta com mecanismos mais genéricos em forma de componentes. Esta categoria provê mecanismos de acesso e consultas sobre o arquivo a ser consultado sem a necessidade de transformações ou carga em outros sistemas de consulta. O FastBit (WU *et al.*, 2009), por exemplo, é uma ferramenta largamente utilizada para indexação de dados científicos, a qual visa tornar sua captura mais eficiente através de índices baseados em mapas de bits gerados sobre os elementos de dados, não sendo específica para um determinado formato de arquivo. A tecnologia também disponibiliza um mecanismo de consulta simples, o qual não exige a carga de dados ao passo que mantém as estruturas de dados originais. Mecanismos de consulta mais eficientes vêm sendo construídos sobre o FastBit, como o FastQuery (CHOU *et al.*, 2011), de modo a acelerar o processo de captura e consulta de dados expressos em diferentes formatos. Nesta categoria também se inclui o ParaView[3] e sua linguagem de consulta. Por ser um sistema de visualização o ParaView é associado com frequência às demais categorias de soluções.

Em uma terceira categoria estão as soluções que proveem consultas a partir de Sistemas de Gerência de Bancos de Dados (SGBD) com recursos específicos para formatos típicos como matrizes. Nessa categoria, os dados científicos precisam sem ingeridos no SGBD para tirar proveito do mecanismo de consulta, como é o caso do SciDB (BROWN, 2010), cuja a meta principal é lidar também com o armazenamento de dados científicos que apresentam estruturas mais complexas (*e.g.*, malhas e matrizes).

[3] http://www.paraview.org/

Tais estruturas simplificam a especificação de consultas, que seriam bastante complexas. No entanto, devido ao tempo de conversão de estruturas de dados entre o arquivo nativo e o SGBD, aliado ao tempo de ingestão de dados e geração de índices, soluções como o SciDB, mesmo com otimizações significativas para essas etapas (LUSTOSA *et al.*, 2016), tendem a ser usadas após a geração de todos os arquivos quando do término da simulação.

A quarta categoria é uma combinação das duas anteriores e alia o acesso direto aos arquivos binários com o poder de consultas dos SGBDs, como é o caso do SDS/Q (DONG *et al.*, 2013), NoDB (ALAGIANNIS *et al.*, 2012) e RAW (KARPATHIOTAKIS *et al.*, 2014). A vantagem dessa categoria de solução é ser adaptativa, no sentido de que os dados científicos são ingeridos no SGBD à medida que eles vão sendo solicitados pelo usuário. Evita-se assim o tempo de transformação e carga da totalidade de dados, já que, na maioria das vezes, apenas uma região de interesse dentre os dados do arquivo científico é inserida no SGBD. Além disso, essa categoria ainda dispõe de técnicas de indexação, de modo a facilitar a consulta a dados científicos (BLANAS *et al.*, 2014, KARPATHIOTAKIS *et al.*, 2014, ROMOSAN *et al.*, 2013). O principal objetivo destes trabalhos é encontrar a melhor maneira de manipular dados científicos de acordo com a estrutura de dados adotada (*e.g.*, grupo de caracteres e malhas), buscando a redução de custos de armazenamento e processamento.

As soluções existentes nas quatro categorias atuam no modo tradicional de consulta pós-processamento, uma vez que são desenvolvidas de forma isolada da geração dos dados, à exceção do ParaView/Catalyst, o qual possui mecanismos para acoplar componentes de visualização e consulta às simulações, provendo a análise *in situ*. No entanto, a capacidade de consulta do ParaView é muito limitada e é ainda restrita a um único arquivo para cada consulta.

A limitação de usar apenas um arquivo da simulação na análise exploratória de dados é encontrada em todas as soluções existentes. Não observamos em nenhuma das categorias, soluções que permitam relacionar elementos de dados existentes em diferentes arquivos em uma consulta. A consulta sobre fluxo de dados da simulação, por sua vez, é fundamental para estabelecer o relacionamento entre os dados ingeridos e produzidos por cada programa científico durante a execução de uma simulação computacional (SILVA *et al.*, 2016). Desse modo, o caminho adotado ao longo da geração do fluxo de dados pode ser consultado pelos usuários de forma a colaborar com

análises de domínio. Portanto, mecanismos capazes de gerenciar os arquivos envolvidos em simulações computacionais são necessários para apoiar o caráter exploratório das análises de dados científicos, conforme discutido no capítulo de trabalhos relacionados.

As soluções para análises sobre dados científicos não devem se limitar apenas a consultas sobre arquivos singulares, sem considerar o fluxo de arquivos estabelecido pelo encadeamento dos programas de simulação (SILVA *et al.*, 2016). Da mesma forma, ter conhecimento somente do relacionamento entre os múltiplos arquivos gerados não é satisfatório para submissão de consultas mais elaboradas que necessitam do conteúdo dos arquivos científicos. Idealmente, o cientista deve ser capaz de acessar os elementos de dados presentes em arquivos de dados científicos, à medida que possui conhecimento sobre o fluxo de arquivos e o fluxo dos próprios elementos de dados.

O presente trabalho argumenta que a análise exploratória de arquivos de dados científicos deve ser apoiada na gerência das simulações computacionais a partir da abstração do fluxo de dados (DIAS *et al.*, 2015), posto que o mesmo representa o encadeamento dos programas de simulação de modo a acompanhar a propagação dos múltiplos arquivos de dados científicos relacionados ao longo de uma execução. A abstração do fluxo de dados representa os dados ingeridos por cada programa de simulação como um conjunto de dados de entrada, enquanto os dados produzidos são definidos como um conjunto de dados de saída. Como definido formalmente no capítulo de referências teóricas, cada um desses conjuntos de dados é formado por conjuntos de elementos de dados, que por sua vez apresentam atributos pré-definidos.

1.2. Abordagem proposta

A partir da argumentação apresentada, uma solução capaz de levar em consideração a abstração do fluxo de dados para gerenciar os programas que compõem a simulação computacional, ao passo que apoia a captura dos dados científicos, provê um grande potencial analítico ao usuário. Em contrapartida, o desempenho computacional da simulação pode ser comprometido visto que tais atividades acrescentam uma sobrecarga em termos de processamento. Portanto, um dos objetivos da abordagem proposta é disponibilizar o acesso eficiente a elementos de dados sobre diferentes arquivos propiciando consultas aos dados *in situ*, buscando minimizar o impacto sobre a execução paralela das simulações e mantendo os dados em seus formatos de origem.

Para atender a essas especificações, o presente trabalho se apoiou na arquitetura ARMFUL (SILVA *et al.*, 2017), desenvolvida como parte da tese, em andamento, do candidato ao doutorado Vítor Silva Sousa pelo Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. A ARMFUL é uma arquitetura baseada em componentes que tem o objetivo de apoiar o acesso e a captura de dados científicos presentes em arquivos gerados por programas de simulação e representá-los em uma base de dados que também agrega dados de execução do fluxo de dados, por meio do modelo de dados PROV-Df (SILVA *et al.*, 2016). Mais especificamente, esta dissertação contribuiu com o desenvolvimento de um componente da arquitetura, o qual realiza a indexação de dados científicos por meio de um algoritmo de indexação baseado em mapas de bits (do termo em inglês, *bitmaps*) (WU *et al.*, 2009). Esta implementação favorece a indexação de dados presentes em arquivos de formatos heterogêneos e que adotam estruturas de dados distintas de acordo com o domínio.

Em relação ao processo de indexação dos dados científicos proposto na dissertação, os dados extraídos são indexados por uma ferramenta de indexação em tempo de execução, tendo como principal meta reduzir o tempo para ingerir os dados científicos em uma base de dados e o espaço de armazenamento requerido, ao mesmo tempo em que preserva o potencial analítico proporcionado apenas pela extração e ingestão dos dados científicos em uma base de dados. Vale ressaltar que, assim como os dados de execução da simulação computacional e dados pontuais de domínio são armazenados diretamente em uma base de dados relacional, as referências para os índices gerados são também armazenadas na mesma base de dados centralizada. Portanto, essa solução apresenta uma menor sobrecarga de armazenamento na base de dados, uma vez que as estruturas de dados não são carregadas, mas sim as referências.

Além de garantir a captura dos dados científicos de interesse e sua subsequente indexação, a abordagem proposta nesta dissertação também proporciona um mecanismo de processamento de consultas baseado em índices que apoia a análise dos múltiplos arquivos contendo dados científicos. Por meio do modelo de dados que a base relacional apresenta (ver Figura 7), os dados de execução podem ser recuperados e relacionados aos arquivos de dados científicos e aos seus respectivos índices. Uma vez que o processador de consultas do SGBD em questão não é capaz de acessar os dados científicos presentes em arquivos, o mecanismo de consultas desenvolvido também faz uso do processador de consultas da ferramenta de indexação responsável pela geração

dos índices. Portanto, a solução apresentada nesta dissertação é capaz de submeter uma especificação de consulta à base de dados, obter os dados de interesse junto às referências para os arquivos de dados científicos e executar uma nova consulta sobre as regiões de interesse desses arquivos com a ajuda dos índices. Os resultados de ambas as consultas (*i.e.*, sobre a base de dados e sobre os arquivos) são coletados e combinados em um resultado final que relaciona dados de execução a dados de domínio.

O gerador de índices e o mecanismo de consultas desenvolvidos com base nos componentes da arquitetura ARMFUL e apresentados nesta dissertação foram avaliados experimentalmente em uma aplicação real, executada em centenas de unidades de processamento, exibindo uma sobrecarga desprezível de apenas 7 minutos a mais no tempo total de execução quando comparada a uma aplicação sem apoio a análise de dados científicos. Além disso, os resultados experimentais mostraram uma aceleração significativa no tempo de ingestão de dados no SGBD para a análise (redução de 67,11% no tempo total de carga de dados). Parte desses resultados foi apresentada em (SILVA *et al.*, 2017).

1.3. Organização da dissertação

Além do presente capítulo de introdução, esta dissertação é composta por mais 6 capítulos. O Capítulo 2 expõe trabalhos relacionados ao cenário da análise exploratória de dados científicos. O Capítulo 3 apresenta as definições teóricas necessárias para melhor compreensão da abordagem proposta, bem como aspectos técnicos das tecnologias utilizadas para apoiar a solução mencionada anteriormente. O Capítulo 4 apresenta a abordagem adotada para a análise exploratória de dados científicos ao longo do fluxo de dados. O Capítulo 5 descreve em detalhes a implementação da solução proposta. O Capítulo 6 apresenta os resultados experimentais obtidos em simulações computacionais em larga escala. O Capítulo 7, por fim, conclui esta dissertação.

Capítulo 2 Trabalhos Relacionados

A partir da motivação e da caracterização do problema, este capítulo tem como objetivo apresentar os principais trabalhos relacionados a esta dissertação de mestrado de modo a estabelecer o contexto da pesquisa. São discutidas algumas das principais abordagens voltadas para a análise exploratória de dados científicos (SILVA *et al.*, 2016), dando ênfase às tecnologias de indexação de dados científicos, bem como os seus principais objetivos, particularidades e limitações.

2.1. Análise de dados científicos a partir de arquivos

Os recentes trabalhos relacionados (ALAGIANNIS *et al.*, 2012, BLANAS *et al.*, 2014, CHOU *et al.*, 2011, KIM *et al.*, 2011, LUSTOSA *et al.*, 2016, MA *et al.*, 2012) focados na análise exploratória de dados científicos compreendem um conjunto de operações importantes para esse processo. De modo geral, as soluções mais abrangentes são capazes de promover o acesso ao dado científico, a extração e a indexação do seu conteúdo e a realização de consultas refinadas sobre os atributos de interesse presentes em arquivos de dados científicos. No entanto, algumas tecnologias não consideram todas essas operações, como é o caso das ferramentas que realizam apenas a extração de elementos de dados, sem o processo de geração de índices ou, até mesmo, sem o suporte a consultas sobre os dados científicos.

O processo de extração dos elementos de dados pode ser realizado de dois modos distintos, a partir da extração total dos elementos ou da extração parcial (SILVA *et al.*, 2016). O primeiro diz respeito à obtenção de todos os valores dos atributos presentes no arquivo de dados científicos, que pode ser realizado pela ferramenta ParaView ao extrair todos os elementos de dados de arquivos no formato XDMF. De forma diferente, o modo de extração parcial obtém apenas parte dos valores dos atributos presentes em um arquivo, em geral, regiões de interesse com dados científicos. A ferramenta HDF5 é um exemplo de tecnologia para o tratamento de dados científicos que possibilita a extração parcial (WANG *et al.*, 2013), assim como ParaView permite que regiões de interesse sejam extraídas (*e.g.*, linhas pelo recurso *Plot Over Line* ou um plano pelo recurso *Slice View*).

Por outro lado, o processo de indexação de dados científicos possui certas características que agregam grande valor para as análises de dados científicos. Posto que

os dados científicos, muitas vezes, apresentam estruturas de dados complexas, o acesso direto aos valores de seus atributos pode ser uma operação complicada, a qual pode ser facilitada por meio da geração de índices. Da mesma forma, o processo de extração dos elementos de dados pode requerer bastante espaço em disco para armazenar estruturas de dados complexas, além de apresentar um alto custo de tempo para carga dos dados em um SGBD considerando simulações em larga escala. A operação de indexação de dados, por sua vez, também é capaz de apoiar essas questões, visto que o custo de espaço e tempo de armazenamento dos índices pode ser menor que dos valores dos atributos, quando se trata de dados científicos em estruturas complexas e em grande volume.

Alguns dos trabalhos existentes sugerem a extração parcial, a indexação e a subsequente carga de dados científicos em um SGBD como solução para conduzir a análise exploratória de dados. Lustosa *et al.* (2016), por exemplo, apresentam uma abordagem que compreende técnicas para mapear atributos de dados científicos expressos em estruturas complexas (*e.g.*, malhas) de forma a prepará-los para serem carregados em um SGBD compatível com tais estruturas, como o SciDB (BROWN, 2010). O mesmo trabalho apresenta uma comparação de desempenho no processamento de consultas entre diferentes tipos de SGBD, em que as tecnologias que são capazes de armazenar estruturas de dados baseadas em matrizes e malhas em larga escala, como o SciDB, proporcionam resultados mais rápidos do que bancos de dados relacionais (*e.g.*, PostgreSQL). A utilização de um SGBD voltado para a análise de dados científicos também aparece como solução para simulações computacionais ligadas a experimentos que buscam comprovar hipóteses científicas (PORTO *et al.*, 2012). Nesse contexto, o trabalho de Gonçalves e Porto (2014) apresenta a importância de bases de dados probabilísticas como forma de apoiar a gerência de dados científicos de natureza incerta.

Em relação ao conjunto de soluções de indexação de dados científicos, diferentes técnicas de indexação são encontradas nesse escopo, sendo que cada uma delas apresenta suas características particulares. Uma das técnicas empregadas em trabalhos recentes é a indexação posicional, a qual faz uso da localização do dado científico no arquivo, ou seja, a sua posição. Basicamente, um índice posicional pode ser representado por dois inteiros, ou outra estrutura de dados básica, em que o primeiro valor identifica a posição inicial do atributo dentro do arquivo e o segundo determina o

comprimento do valor do atributo em bytes. Por se tratar de uma representação que necessita de um ponteiro composto por poucos valores para representar cada valor de atributo do dado, a proposta posicional pode apresentar uma sobrecarga de dados pequena para a gerência de dados científicos. Uma outra característica importante da indexação posicional é a capacidade de referenciar estruturas de dados complexas, como árvores e malhas multidimensionais. Como exemplos de tecnologias que apresentam a solução de indexação posicional para lidar com arquivos de dados científicos podem ser citados o NoDB (ALAGIANNIS *et al.*, 2012) e o RAW (KARPATHIOTAKIS *et al.*, 2014).

A tecnologia do NoDB apresenta um ganho de desempenho expressivo devido ao fato de indexar apenas os dados científicos especificados pelas consultas submetidas. O RAW, por sua vez, foi desenvolvido pelo mesmo grupo de pesquisa responsável pelo NoDB, buscando aprimorar a proposta deste último ao superar uma de suas limitações quanto à sobrecarga que apresenta por realizar a carga dos dados extraídos em um SGBD adaptado, conhecido como PostgresRaw (baseado no PostgresSQL). O RAW realiza consultas adaptativas diretamente sobre os arquivos de dados científicos sem requerer a carga dos dados extraídos em uma base de dados. Tal solução implementa estratégias de acesso aos dados e suas colunas mais relevantes de acordo com a consulta submetida, estabelecendo os caminhos de acesso aos atributos do dado em tempo de execução. Para adiantar ao máximo a operação de seleção no plano de execução da consulta, a tecnologia busca restringir os elementos de dados àqueles definidos pela consulta em questão.

Uma segunda técnica de indexação de dados amplamente aplicada, inclusive por soluções tradicionais de SGBD, é baseada na utilização de algoritmos de indexação *bitmap* (WU *et al.*, 2006). Tal proposta se apoia na criação de mapas de bits a partir da análise do domínio dos atributos presentes nos dados científicos afim de gerar índices booleanos. Para ilustrar, considerando um atributo A do tipo inteiro e que apresenta apenas quatro valores possíveis em todos arquivos de dados científicos, são necessários quatro bits para mapear a presença ou não de um determinado valor, assumindo uma equação de equidade, como mostra a Tabela 1. Em certos casos, como os atributos representados por números em ponto flutuante com um extenso domínio, e de acordo com os tipos de consultas a serem realizadas sobre o dado, se torna interessante o uso de inequações para a geração dos mapas de bits, *i.e.*, menor número de bits visando

representar um intervalo de valores ao invés de apenas um único valor. As principais tecnologias que utilizam a indexação *bitmap* em seu núcleo são o FastBit (WU *et al.*, 2009), o FastQuery (CHOU *et al.*, 2011) e o SDS/Q (DONG *et al.*, 2013).

Tabela 1. Exemplo de mapa de bits para o atributo A.

RID	A	Mapa de bits			
		b_1	b_2	b_3	b_4
		A=0	A=1	A=2	A=3
1	1	0	1	0	0
2	3	0	0	0	1
3	2	0	0	1	0
4	1	0	1	0	0
5	0	1	0	0	0
6	2	0	0	1	0
7	3	0	0	0	1
8	0	1	0	0	0

Uma vez que compõe a solução proposta neste trabalho, a ferramenta FastBit é discutida em mais detalhes em uma seção dedicada no Capítulo 3. No entanto, alguns pontos das soluções FastQuery e SDS/Q valem ser citados, uma vez que ambas utilizam a tecnologia de indexação de dados do FastBit como base. Assim como o RAW, as três ferramentas para a indexação *bitmap* citadas atuam diretamente sobre os arquivos de dados de científicos, eliminando a necessidade de transformação do dado para ser adequado em uma estrutura de dados compatível com um SGBD. A principal contribuição que o FastQuery traz é a extensão da tecnologia de indexação presente no FastBit para ambientes de processamento paralelo, em que tanto a construção dos índices quanto o processamento de consultas fazem uso de recursos alocados de forma distribuída. Desse modo, a solução requer uma quantidade menor de memória para criação dos índices e permite a realização de um alto número de tarefas concorrentes na construção e utilização dos índices. O SDS/Q, por sua vez, apresenta melhorias de desempenho no processamento paralelo de consultas pelo uso de técnicas para paralelizar a execução intra- e inter-nós. As operações de indexação dos dados e de processamento de consultas são realizadas em memória por meio da execução paralela.

Existem outros trabalhos que apresentam metodologias de indexação a partir da combinação de diferentes técnicas, como é o caso do ALACRITY (JENKINS *et al.*, 2013) e o DIRAQ (LAKSHMINARASIMHAN *et al.*, 2013). O primeiro foca sua proposta de indexação apenas em atributos expressos em ponto flutuante, visto que trata-se de uma estrutura de dados muito presente em dados científicos. Basicamente, a

solução divide o processo de geração de índices em duas etapas, uma primeira em que índices *bitmap* são criados apenas sobre os algarismos mais significativos dos valores em ponto flutante e, em seguida, uma segunda fase, na qual índices invertidos são criados para mapear os algarismos menos significativos. Dessa maneira, o ALACRITY garante um melhor desempenho no processamento de consultas, porém, em contrapartida, o custo de armazenamento dos índices se eleva bastante para dados de alta cardinalidade, posto que o tamanho desses índices compostos pode ser maior que o tamanho do próprio dado original. O DIRAQ, por sua vez, compreende a metodologia de indexação do ALACRITY e expande o processo de geração dos índices para ambientes de processamento paralelo, semelhante à relação entre o FastQuery e o FastBit.

As soluções aqui apresentadas contemplam análises sobre arquivos de dados científicos apenas após o término da execução das simulações computacionais, *i.e.*, na etapa de pós-processamento. Em consequência desse fato, tais tecnologias não são capazes de estabelecer o relacionamento entre diferentes arquivos de dados científicos, o que restringe a análise a um processo pontual. Assim, cada arquivo é analisado de forma separada e não há qualquer rastro das transformações às quais os elementos de dados foram submetidos ao longo da execução da simulação computacional. Para que o fluxo de dados fosse representado em bases de dados adaptadas para os dados científicos, os programas de simulação teriam que ser reformulados para facilitar o acesso às estruturas de dados compatíveis com o SGBD, o que não se observa no contexto de desenvolvimento de programas de simulação. Outra opção seria a replicação dos dados em seu formato original nas ferramentas de análise de dados científicos, no entanto, essa proposta não parece adequada à larga escala de dados que os trabalhos de processamento de dados *in situ* vêm apresentando (RUDI *et al.*, 2015).

2.2. Análise de dados científicos ao longo do fluxo de dados

Em contrapartida às propostas descritas na seção anterior, a falta de apoio ao monitoramento de arquivos científicos produzidos ao longo do fluxo de dados de simulações computacionais é solucionada por abordagens que dispõem de mecanismos para gerência do fluxo de arquivos. A seção 3.1 apresentará em mais detalhes o conceito de fluxo de dados, bem como as características intrínsecas à sua gerência. Também serão discutidos em maior amplitude os dois níveis de abstração para lidar com o

gerência do fluxo de dados, o nível físico e nível lógico (SILVA *et al.*, 2016). Ainda assim, uma breve definição de ambas as abordagens será introduzida a seguir para melhor ilustrar os recentes trabalhos desenvolvidos que contemplam cada uma delas.

A gerência do fluxo de dados no nível físico consiste em apoiar a transformação de dados do ponto de vista do sistema de arquivos, ou seja, os arquivos gerados ao longo do fluxo de dados são tratados como caixas pretas, visto que não há possibilidade de gerar índices ou realizar consultas sobre os dados de domínio contidos em tais arquivos. Logo, o usuário se restringe a analisar cada arquivo individualmente ou desenvolver seus próprios programas específicos para extrair e indexar o conteúdo de domínio, atividades essas que exigem grande esforço e são propensas a erros. Como mostra Silva *et al.* (2016), o arcabouço AWARD (ASSUNCAO *et al.*, 2014) propõe uma abordagem baseada no nível físico, a qual captura conjuntos de valores de parâmetros em tempo de execução e gerencia a propagação de dados ao longo dos programas de simulação de acordo com a dependência entre eles. Entretanto, apesar de lidar com estes valores de parâmetros, a tecnologia não os trata como elementos de dados capazes de serem representados em um modelo relacional.

O nível lógico, por outro lado, leva em consideração o rastro dos elementos de dados consumidos e produzidos pelos programas de simulação. Tais elementos podem ser dados científicos contidos em arquivos ou os dados propagados pelos programas de simulação encadeados. Essa forma de abstração do fluxo de dados permite ao usuário estabelecer os relacionamentos entre os elementos de dados, favorecendo as análises sobre os mesmos. Alguns dos principais Sistemas de Gerência de *Workflows* Científicos (SGWfC) são ferramentas de extrema importância para apoiar o nível lógico de abstração, como é o caso do Kepler (BOWERS *et al.*, 2008), do Panda (IKEDA e WIDOM, 2010) e do Chiron (OGASAWARA *et al.*, 2013).

No próximo capítulo é definido formalmente o conceito de *workflows* científicos, os quais se tratam de um conjunto de atividades encadeadas que, por sua vez, relacionam dados de entrada e de saída. Em seguida, também definimos em detalhes os SGWfC. Contudo, vale ressaltar ainda nesta seção algumas das características em particular das tecnologias citadas. O Kepler adota a representação de *workflow* científico para lidar com o fluxo de dados, ou seja, ele concebe o fluxo de dados na forma de atividades encadeadas que consomem e produzem arquivos de dados científicos. Com essa abordagem, a ferramenta é capaz de apoiar a gerência de arquivos

e elementos de dados manipulados diretamente pelo sistema. Também buscando monitorar o fluxo de dados, o Panda apresenta um formalismo para os dados rastreados durante a execução, de forma que seja possível estabelecer o fluxo dos elementos de dados produzidos por um *workflow* composto por operadores de uma álgebra relacional. Porém, ambas ferramentas possuem mecanismos de consulta limitados aos relacionamentos entre os arquivos mapeados ao longo da simulação computacional. O conteúdo dos arquivos de dados científicos não é apoiado e entra em questão apenas nas análises *off-line*. Portanto, as soluções de análises de arquivos científicos apresentadas na seção anterior precisariam ser aplicadas nesta etapa para realizar buscas e acessar os elementos de dados nos arquivos.

O Chiron, por sua vez, contempla o uso de uma álgebra de *workflows* (seção 3.3) para modelar e executar simulações computacionais. Diferente de outras tecnologias, o Chiron permite em tempo de execução a captura e o armazenamento dos dados científicos extraídos de arquivos em uma base de dados relacional. Logo, essa solução possibilita a análise de arquivos de dados científicos por meio de programas externos, os quais acessam e extraem o conteúdo de domínio presente nos arquivos. Após a execução de cada programa de simulação, o dado científico pode ser carregado em um SGBD conectado ao sistema. Tal característica aproxima o Chiron da melhor solução atual para a análise de arquivos de dados científicos dentre os SGWfC existentes, uma vez que ele já apresenta uma abordagem para extração de elementos de dados contidos em arquivos relacionados pelo fluxo de dados (SILVA *et al.*, 2016). Contudo, no contexto de simulações computacionais em larga escala, o Chiron exibe um alto custo de tempo relativo à ingestão dos dados científicos em uma base de dados relacional.

Capítulo 3 Referencial Teórico

Uma vez apresentados os trabalhos relacionados ao contexto de análise de dados científicos, o presente capítulo busca ampliar alguns conceitos teóricos que serviram de base para o desenvolvimento deste trabalho. Assim, são definidos os conceitos de fluxo de dados, proveniência, *workflows* científicos, além da importância dos Sistemas de Gerência de Workflows Científicos (SGWfC) para apoiar a descoberta de conhecimento. A ferramenta FastBit para indexação e consulta sobre dados científicos também encontra-se detalhada neste capítulo, uma vez que é parte integrante da solução proposta nesta dissertação, buscando reduzir o tempo de armazenamento de dados científicos e facilitar o acesso aos mesmos para realização de consultas.

3.1. Fluxo de dados

Os conceitos apresentados a seguir são formalizações extraídas do trabalho de Silva *et al.* (2016). Para definir formalmente o conceito de *fluxo de dados*, é necessário primeiro considerar a unidade mais simples que compõe um dado: o *elemento* de dado. Um *conjunto de dados*, por sua vez, é formado por um conjunto de elementos de dados, o qual pode ser definido por $E=\{e_1, e_2, \ldots, e_X\}$, sendo *x* o número de elementos de dados (*e*). Para cada elemento de dado há também uma sequência de atributos pré-definidos $A=\{a_1, a_2, \ldots, a_Y\}$, em que *y* é o número de atributos, o que significa que cada elemento de dado possui um valor específico para cada um desses atributos. Outro conceito fundamental neste contexto é o de *transformação de dados*, o qual consiste no processamento de dados realizado por um algoritmo ou modelo computacional. Uma transformação de dados T se baseia no consumo de um ou mais conjuntos de dados de entrada I e pela produção de um ou mais conjuntos de dados de saída O. Portanto, uma transformação de dados caracteriza-se pelo processamento de conjuntos de dados de entrada que gera conjuntos de dados de saída, *i.e.*, $O=T(I)$.

A partir dos conceitos apresentados e adaptando o formalismo proposto por Ikeda *et al.* (2013) e Silva *et al.* (2016), um *fluxo de dados* pode ser caracterizado pelo encadeamento de transformações de dados. Considerando duas transformações de dados, T_1 e T_2, a composição $T_1 \circ T_2$ representa uma transformação em que T_1 é aplicada sobre um conjunto de dados de entrada I_1 gerando um conjunto de dados intermediário I_2, este que por seguinte é consumido por T_2 para obter um conjunto de dados de saída

O. Logo, um fluxo de dados D_F formado pela composição linear de *z* transformações de dados pode ser representado por $D_F = T_1 \circ T_2 \circ ... \circ T_Z$. Isso significa que, em termos de conjunto de dados de entrada e saída, o conjunto de saída O desse fluxo de dados pode ser representado por $O = (T_1 \circ T_2 \circ ... \circ T_Z)(I_1)$, ou $O = (D_F)(I_1)$, sendo I_1 o conjunto de dados de entrada.

Posto que um fluxo de dados pode ser integrado por uma extensa composição de transformações de dados e um grande volume de conjuntos de dados, sua gerência torna-se imprescindível para qualquer abordagem que pretenda lidar com simulações computacionais. Um primeiro modo de enxergar a gerência do fluxo de dados se baseia na forma de lidar com as transformações de dados apenas como programas de simulação que consomem e produzem arquivos cujo conteúdo não é considerado. Ou seja, o conteúdo específico de domínio presente nesses arquivos na forma de dados científicos é invisível do ponto de vista de gerência do fluxo de dados. Portanto, essa forma de abstração apresenta uma grande desvantagem, visto que não oferece suporte para o acesso ao conteúdo presente nestes arquivos, nem mesmo a possibilidade de geração de índices para apoio de consultas sobre os dados de domínio. Isso implica na necessidade de analisar cada arquivo individualmente, seja de forma manual ou através do desenvolvimento de programas específicos para acesso ao conteúdo desses arquivos.

Silva *et al.* (2016) definem ainda esta primeira abordagem como um nível físico de abstração da gerência do fluxo de dados, enquanto um nível lógico é apresentado como um refinamento dessa abstração, visto que considera não apenas o fluxo de arquivos, mas também os elementos de dados neles contidos. Ao monitorar os elementos de dados e suas transformações, os relacionamentos entre os conjuntos de dados de entrada e de saída de cada programa de simulação podem ser utilizados para reconstruir o fluxo de dados de maneira mais precisa, conferindo maior potencial analítico a esse nível de gerência. Do ponto de vista prático do usuário, torna-se viável a realização de consultas sobre dados inerentes ao domínio da simulação computacional, sem a necessidade de desenvolver programas específicos para acessar os dados científicos presentes nos arquivos. Logo, o nível lógico de abstração supera a limitação em relação às consultas e minimiza a propensão a erros apresentada pelo nível físico no acesso ao dado. Por outro lado, o nível lógico de abstração consequentemente exibe um custo computacional maior, pois ao invés de considerar apenas ponteiros para arquivos

de dados científicos, os elementos de dados também precisam ser capturados e monitorados em tempo de execução.

Igualmente importante para a gerência do fluxo de dados é o apoio oferecido pela *proveniência de dados*, a qual pode ser definida pelo rastreamento das origens de conjuntos de dados e seu subsequente armazenamento em uma base de dados (BUNEMAN *et al.*, 2001). Com o intuito de padronizar as abordagens de proveniência de dados, grupos como o *World Wide Web Consortium* (W3C) têm proposto modelos de dados para consolidar uma representação de como os dados de proveniência devem ser capturados e armazenados em uma base de dados. Uma das principais iniciativas neste contexto é o modelo PROV-DM (MOREAU e GROTH, 2013), o qual faz parte do W3C PROV e provém uma representação genérica que serve para qualquer tipo de dado de proveniência, independente do domínio ao qual pertencem os dados. A partir das recomendações definidas por essa iniciativa, Silva *et al.* (2016) propõem o modelo de dados PROV-Df para apoiar simulações computacionais, o qual possibilita a captura dos dados de proveniência e dados específicos de domínio e o seu armazenamento em tempo de execução para o desenvolvimento de consultas sobre os mesmos.

Considerando um modelo de dados de proveniência que abrange o nível lógico de abstração do fluxo de dados, os dados de proveniência podem ser capturados em relação às derivações dos elementos de dados a partir das transformações de dados. Aliado a esse modelo, o conceito de proveniência *retrospectiva* também apresenta grandes vantagens para o contexto de simulações computacionais, dado que permite a coleta de informações sobre os dados processados ao longo da execução de cada transformação de dados. Dessa maneira, é possível registrar diferentes informações do ponto de vista da execução, como o tempo de execução de cada transformação, o seu custo computacional, os ponteiros dos arquivos consumidos e produzidos pelo programa de simulação e em qual diretório esses dados estão alocados.

Portanto, o usuário obtém a capacidade de monitorar o desempenho do processo de simulação em tempo de execução e manter o registro dos dados de domínio em uma base de dados. Isso também confere a ele o poder de realizar análises mais objetivas em relação às simulações computacionais executadas, o que significa maior controle durante a verificação e validação dos resultados obtidos. Através dos dados de desempenho e dos dados de domínio, os usuários podem avaliar o andamento de uma

simulação computacional e, caso o fenômeno não apresente um comportamento esperado, ajustes podem ser feitos na configuração do fluxo de dados ou, até mesmo, a execução pode ser abortada para uma nova submissão mais adequada (MATTOSO *et al.*, 2013).

3.2. *Workflows* científicos

Originalmente o termo *workflow* advém de ambientes de negócio, onde o mesmo é utilizado para denominar a automação de processos de criação, armazenamento, compartilhamento e revisão de informações em uma empresa. Este conceito foi estendido para ambientes científicos com o objetivo de suportar a complexidade de simulações computacionais em larga escala (DEELMAN *et al.*, 2009).

Da mesma forma que simulações computacionais são compostas pelo encadeamento de programas de simulação, os *workflows* científicos são formados pelo encadeamento de *atividades*, onde cada par de atividades compõe uma relação entre os dados que são produzidos por uma e consumidos pela seguinte.

Se compararmos esta definição com o conceito de fluxo de dados, podemos ver que cada atividade de um *workflow* científico corresponde a uma transformação de dados, *i.e.*, um programa de simulação. A relação entre os dados de duas atividades, por sua vez, pode ser associada ao fluxo de dados entre duas transformações. Portanto, por meio de *workflows* científicos é possível abstrair e sistematizar o processo experimental (DIAS *et al.*, 2015).

Devido ao paralelo entre *workflows* científicos e fluxos de dados, o termo *dataflow* (WOZNIAK *et al.*, 2012) é comumente utilizado para se referir a *workflows* científicos centrado em dados, ou seja, os quais lidam com um grande volume de dados fluindo entre suas atividades.

Como proposto por Mattoso *et al.* (2010), *workflows* científicos que compreendem dados em larga escala devem ser apoiados em três fases: (1) composição, na qual é definido o encadeamento de atividades, os parâmetros de execução e os conjuntos de dados de entrada; (2) execução, na qual o *workflow* é instanciado em um ambiente de PAD por meio de uma máquina de execução de *workflows*; e (3) análise, etapa de consulta e visualização dos resultados obtidos. Os Sistemas de Gerência de *Workflows* Científicos (SGWfC) são ferramentas desenvolvidas para apoiar *workflows* científicos ao longo dessas três etapas (DEELMAN *et al.*, 2009).

Como descrito anteriormente, a abstração de fluxo de dados para *workflows* científicos permite que os usuários de um SGWfC definam a composição do *workflow* a partir da especificação das transformações de dados e as suas dependências de dados (SILVA *et al.*, 2016). O SGWfC, por sua vez, produz um plano de execução de acordo com um determinado modelo de execução baseado nas especificações do *workflow* científico.

Sob o aspecto dos conjuntos de dados de entrada, seus elementos podem ser particionados em diferentes tarefas, as quais consomem um subconjunto de elementos de dados de entrada e submetem os mesmos à transformação de dados correspondente. Cada tarefa é direcionada para os recursos computacionais que realizarão seu processamento. O presente trabalho dá enfoque nos SGWfC que dão suporte à distribuição de tarefas para recursos computacionais remotos, como é o caso de ambientes de PAD. Como exemplos de SGWfC que oferecem apoio a esta característica, podemos citar o Pegasus (DEELMAN *et al.*, 2015), o Swift/T (ZHAO *et al.*, 2008) e o Chiron (OGASAWARA *et al.*, 2013).

Como discutido na seção 3.1, a gerência do fluxo de dados é de extrema importância quando se trata de transformações sobre grandes volumes de dados. Logo, assumindo a abstração do fluxo de dados, temos que o monitoramento de *workflows* científicos é uma tarefa fundamental para a execução de simulações computacionais. Alguns SGWfC apresentam mecanismos próprios para tal tarefa, desde arquivos de *log*, os quais apresentam certas limitações de acesso e leitura, até a gerência de dados de proveniência. Também discutido na seção 3.1, os dados de proveniência são recursos valiosos para avaliar o desempenho do processo de simulação e, também, realizar consultas sobre os dados de domínio. Existem dois aspectos importantes que devem ser considerados pelos SGWfC que suportam a gerência de dados de proveniência: a representação do modelo de dados de proveniência e a forma como os dados são capturados, *i.e.*, em modo *online* ou *off-line*.

O primeiro diz respeito à forma como os dados de proveniência são representados para garantir que os mesmos sejam capturados e tratados independente do domínio ao qual eles pertençam. Ou seja, inciativas como o PROV-DM, discutidas na seção anterior, têm como principal objetivo prover um modelo de dados de proveniência que possa ser adequado para qualquer domínio da ciência, como a bioinformática (OCAÑA *et al.*, 2011) e a astronomia (JACOB *et al.*, 2009).

O segundo aspecto está relacionado com a forma que o SGWfC realiza o armazenamento dos dados de proveniência, que pode ser *online* (*i.e.*, em tempo de execução) ou *off-line* (*i.e.*, após o término da execução do *workflow* científico). O VisTrails (CALLAHAN *et al.*, 2006) e o Kepler (BOWERS *et al.*, 2008) são exemplos de SGWfC que suportam a gerência dos dados de proveniência apenas em modo *off-line*, enquanto Pegasus e Chiron são exemplos de sistemas capazes de coletar dados de proveniência no modo *online*. A proposta *online* confere ao usuário maior poder analítico sobre a execução do *workflow* científico haja vista que possibilita o processamento de consultas sobre os dados de desempenho e até dados de domínio em tempo de execução. Entretanto, é importante notar que alguns SGWfC que operam em modo *online*, em relação à proveniência de dados, não coletam dados de domínio, apenas dados de execução, como é o caso do Swift/T, o que reduz tal potencial analítico. Em contrapartida, o processo de captura e armazenamento de dados de proveniência em modo *online* apresenta um desempenho inferior ao modo *off-line*, dado que ocorre de forma concorrente ao processamento dos programas de simulação que compõem o *workflow* científico.

Por fim, vale ressaltar que os SGWfC podem também ser integrados à ferramentas externas para superar possíveis limitações do seu mecanismo de análise de dados científicos. Tais ferramentas podem ser programas customizados e específicos para determinado sistema ou até mesmo tecnologias já consolidadas que facilitam a descoberta de conhecimento, como DiNoDB (TIAN *et al.*, 2014), RAW (KARPATHIOTAKIS *et al.*, 2014) e SDS/Q (DONG *et al.*, 2013). No entanto, na maioria dos casos, essa tarefa de integração não é simples, uma vez que requer um conhecimento técnico mais aprofundado do mecanismo de um ou mais sistemas, os quais foram desenvolvidos de forma independente. A abordagem exposta neste trabalho utiliza esse conceito de integração para acoplar o FastBit, uma ferramenta de análise de dados científicos, ao SGWfC Chiron, como é apresentado nos capítulos subsequentes.

3.3. Chiron

O SGWfC Chiron (OGASAWARA *et al.*, 2013) foi desenvolvido para apoiar as etapas de modelagem, submissão, monitoramento e análise de *workflows* científicos em ambientes de PAD, como *clusters* e grades computacionais, oferecendo uma máquina paralela de execução para tais *workflows*. Compreendendo todas essas funções, a

solução garante a gerência de um *workflow* científico e todos os conjuntos de dados consumidos, transformados e produzidos a cada atividade que compõe o mesmo. Complementando ainda suas principais características, o Chiron suporta a gerência de dados de proveniência, contemplando os dados de composição e execução do *workflow* e os dados de domínio em uma mesma base de dados.

Tendo como base a álgebra relacional tradicional para otimização e processamento de consultas em SGBD, o Chiron apresenta uma abordagem algébrica denominada *Scientific Workflow Algebra* (SciWfA) (OGASAWARA *et al.*, 2011). Essa álgebra de *workflows* visa facilitar a definição de *workflows* científicos complexos, a gerência de dados em paralelo presentes em um fluxo de dados e a otimização de *workflows* científicos em tempo de execução. Posto que a abstração do fluxo de dados pode ser adotada para um *workflow* científico, a SciWfA também assume as representações de conjuntos e transformações de dados.

Portanto, consideremos o conjunto $\{T\}$ que contém todas as transformações de dados (*i.e.*, atividades) que compõem um determinado fluxo de dados (*i.e.*, *workflow*). A álgebra de *workflows* propõe que cada transformação de dados $T_i \in \{T\}$ seja regida por um operador algébrico ϕ. Como visto anteriormente na seção 3.1, cada transformação T_i consome um conjunto de dados I_i de elementos de dados de entrada e produz um conjunto de dados O_i de elementos de dados de saída. Da mesma maneira, um conjunto de dados de saída pode ser o conjunto de dados de entrada do operador relacionado a uma próxima transformação de dados. Dependendo do operador algébrico, um operando adicional γ também precisa ser definido. Logo, uma transformação de dados algébrica pode ser representada pela seguinte expressão:

$$O_i \leftarrow \phi(T_i, \gamma, I_i)$$

Dentro da definição da álgebra de *workflows* científicos, temos que um operador ϕ pertence ao conjunto de operadores possíveis apresentados na Tabela 2. É importante notar também que um conjunto de dados pode ser chamado de relação, visto que esse é o nome formal que recebe na álgebra relacional. Outra característica de um operador algébrico ϕ é a razão entre o número de elementos de dados consumidos e produzidos para uma determinada invocação da transformação de dados.

Tabela 2. Operações da SciWfA (OGASAWARA *et al.*, 2011).

Operador	Tipos de atividades	Operandos adicionais	Razão entre os elementos de dados consumidos e produzidos
Map	Aplicação	Relação	1:1
SplitMap	Aplicação	Referência de arquivo ou Relação	1:m
Reduce	Aplicação	Conjunto de atributos de agregação ou Relação	n:1
Filter	Aplicação	Relação	1:(0-1)
SRQuery	Expressão da álgebra relacional	Relação	n:m
MRQuery	Expressão da álgebra relacional	Relação	n:m

Uma particularidade da álgebra definida no Chiron é a presença dos operadores SRQuery e MRQuery, os quais utilizam expressões algébricas relacionais tradicionais para realizar consultas na base de proveniência. Suas funções são basicamente as mesmas, sendo que eles se distinguem apenas pelo número de conjunto de dados de entrada. Enquanto o operador SRQuery lida com apenas um conjunto de dados de entrada, o operador MRQuery é destinado para casos com múltiplos conjuntos de dados.

Sob o ponto de vista de execução de um *workflow* científico, um programa de simulação pode ser invocado múltiplas vezes, dependendo do número de elementos que o conjunto de dados de entrada e o conjunto de dados de saída possuem, sendo que a cada invocação há o consumo de um ou mais elementos de dados e a produção de um ou mais elementos de dados. De acordo também com o operador algébrico que rege a atividade do programa de simulação, uma ou mais invocações podem ocorrer.

Entretanto, de um modo geral, os programas de simulação invocados no Chiron são desenvolvidos para serem executados de modo singular, independentes de mecanismos de execução de *workflows* científicos. Logo, os mesmos não estão cientes da especificação da SciWfA, o que pode ocasionar a produção de elementos de dados de saída desalinhados com a definição do operador algébrico e, consequentemente, inviabilizando o mecanismo de captura de dados de proveniência e a especificação da relação de entrada da próxima transformação de dados. Buscando contornar essa limitação, a SciWfA utiliza o conceito de *ativação*, definida como a menor unidade de

processamento de dados que não pode ser mais fragmentada, *i.e.*, não pode ser mais paralelizada (ÖZSU e VALDURIEZ, 2011).

No contexto do Chiron, uma ativação envolve três procedimentos: *instrumentação*, *invocação* e *extração* (OGASAWARA *et al.*, 2011, SILVA *et al.*, 2014). Na etapa de composição do *workflow* científico, os usuários definem o esquema dos conjuntos de dados de entrada e saída para todas as transformações de dados. Assim, na etapa de execução do *workflow*, temos primeiro o procedimento de instrumentação, no qual os elementos de dados de entrada são obtidos e a invocação do programa de simulação é preparada de acordo o esquema de entrada. Em seguida, ocorre o procedimento de invocação de uma ativação, no qual um programa de simulação é executado em um determinado recurso computacional no ambiente de PAD, consumindo os elementos de dados de entrada definidos na instrumentação e produzindo elementos de dados de saída. Por fim, na extração, os elementos de dados de saída produzidos são coletados.

Em relação à gerência dos dados de proveniência, os mesmos são capturados ao longo de cada um dos procedimentos de uma ativação, sejam eles dados de desempenho ou dados de domínio. Como discutido anteriormente, eles exercem um importante papel para apoiar os cientistas em diversas análises, inclusive no processo de depuração dos próprios programas de simulação. Dessa maneira, consultas analíticas mais complexas podem ser realizadas ainda em tempo de execução, permitindo o monitoramento da execução paralela das simulações computacionais (OLIVEIRA *et al.*, 2014, SOUZA *et al.*, 2015).

Finalmente, o núcleo do Chiron ainda dispõe de duas estratégias para gerenciar a execução paralela do *workflow* científico. Considere o caso de um *workflow* científico composto por apenas duas transformações de dados T_1 e T_2, no qual T_2 depende de T_1. A primeira abordagem, *First Tuple First* (FTF), exibe um comportamento semelhante ao de um *pipeline* em fluxo de dados, ou seja, no exemplo, uma ativação de T_2 espera apenas que os seus elementos de dados de entrada sejam produzidos pela ativação correspondente à T_1, independente do término da execução das outras ativações de T_1. A segunda estratégia, *First Activity First* (FAF), por sua vez, exige que todas as ativações de T_2 aguardem o término da execução das ativações de T_1.

3.4. FastBit

Na seção 2.1 foram apresentadas soluções para análise de dados científicos em arquivos, principalmente tecnologias que empregam diferentes abordagens para a indexação de dados científicos. O conceito de indexação *bitmap* também foi exposto, assim como a solução no estado da arte que o utiliza para a descoberta de conhecimento científico, o FastBit. Nesta seção, abordamos em detalhes o funcionamento dessa tecnologia e seus principais recursos.

O FastBit (WU *et al.*, 2009) é uma biblioteca de código aberto com o objetivo de indexar dados científicos para realizar o processamento eficiente de consultas sobre eles. A tecnologia disponibiliza um conjunto de funções de busca baseadas em índices *bitmap* comprimidos. Assim como outras soluções de gerência de banco de dados não relacionais, como o MonetDB[4] e o Sybase IQ[5], o FastBit lida com o dado de forma orientada a coluna visando acelerar o processo de busca no dado. No entanto, diferente de um SGBD tradicional, os dados do usuário não são carregados em uma base do sistema. Ao invés disso, o mecanismo de acesso para geração de índices e realização de consultas atua diretamente sobre arquivos em disco que armazenam os dados científicos. Essa é uma importante característica da solução, visto que representa uma economia de tempo e recursos computacionais que outros sistemas devem dispor para adequar e carregar dados em diferentes estruturas.

Em relação ao tratamento do dado científico, a solução o enxerga de forma tabular, ou seja, como uma tabela composta por linhas e colunas, similar à maioria dos SGBD. Entretanto, para ser consumido pelo programa, o dado deve ser particionado: cada coluna do dado deve ser alocada em um arquivo binário distinto. Portanto a organização do dado é orientada a coluna, ao que se atribui o nome de *organização vertical*. Este tipo de composição garante que apenas dados homogêneos (*i.e.*, mesmo tipo de dado) sejam agrupados em um mesmo arquivo. Os arquivos binários ficam armazenados em disco, alocados em um mesmo diretório junto a um arquivo texto com metadados, contendo informações como o número de linhas e colunas do dado, o nome e a estrutura de dados de cada coluna, entre outros.

[4] https://www.monetdb.org/

[5] http://www.sybase.com/products/databaseservers/sybaseiq

Uma vez que os arquivos binários e o arquivo de metadados estejam alocados no mesmo diretório, o FastBit pode ser executado para criar os índices sobre o dado e, em seguida, realizar consultas. É importante ressaltar que ele é capaz de lidar apenas com o dado no formato citado anteriormente, ou seja, arquivos binários em organização vertical. Caso o dado não esteja assim organizado, se faz necessária uma conversão. Basicamente, esse processo de conversão deve ler o dado e produzir uma cópia em formato de pequenos dados binários representando as colunas do arquivo de dados científicos original. A solução também conta com uma função embutida que realiza essa conversão para arquivos em formato *Comma-Separated-Value* (CSV), sendo responsável pela criação dos arquivos binários referentes às colunas do dado e ainda pelo arquivo de metadados.

Gerados os índices, estes ficam armazenados em arquivos específicos no mesmo diretório que os arquivos binários do dado. Cada arquivo binário possui um arquivo correspondente com o registro de seus índices *bitmap*, sob a extensão *idx*, como pode ser visto na Figura 1. Os arquivos *a*, *b*, e *c* representam os binários de cada coluna (*i.e.*, atributo) do dado, enquanto os arquivos com extensão *idx* são seus respectivos índices. O arquivo texto da ilustração, denominado '*-part.txt'*, é o arquivo contendo os metadados.

```
-rw-r--r-- 1 kwu Users  402 Aug  3 20:35 -part.txt
-rw-r--r-- 1 kwu Users  400 Aug  3 20:35 a
-rw-r--r-- 1 kwu Users 3520 Aug  4 23:14 a.idx
-rw-r--r-- 1 kwu Users  400 Aug  3 20:35 b
-rw-r--r-- 1 kwu Users 3520 Aug  4 23:14 b.idx
-rw-r--r-- 1 kwu Users  200 Aug  3 20:35 c
-rw-r--r-- 1 kwu Users 3520 Aug  4 23:14 c.idx
```

Figura 1. Estrutura de arquivos do FastBit (adaptado de https://sdm.lbl.gov/fastbit/).

A mesma função utilizada para criação dos índices, *Interactive Bitmap Index Search* (IBIS), deve ser utilizada para o processamento de consultas sobre o dado. É necessário apenas definir o diretório onde estão armazenados os dados e seus índices e especificar a consulta a partir da linguagem proposta pelo FastBit, a qual é muito semelhante à linguagem SQL, utilizando inclusive suas principais palavras-chave (*e.g.*, *select, from, where*, entre outras).

Retomando as discussões sobre indexação *bitmap* da seção 2.1, podemos afirmar que a sua implementação básica é efetiva apenas em variáveis com baixa cardinalidade,

visto que o número de *bitmaps* cresce linearmente com a quantidade de possíveis valores distintos que o atributo pode assumir. Por exemplo, se um atributo A pode assumir *n* valores distintos, um índice *bitmap* formado por *n* vetores de bits se faz necessário para representá-lo. Em muitos casos, especialmente quando se trata de estruturas de dados como ponto-flutuante e sequências de caracteres, o valor de *n* pode ser muito grande, o que transforma o armazenamento e a leitura de índices *bitmap* em processos de alto custo. Portanto, para lidar com o armazenamento de índices de variáveis com alta cardinalidade e reduzir custos de leitura para realização de consultas, o FastBit conta com três técnicas diferentes: *compressão, codificação* e *binning* (WU *et al.*, 2009).

O método de compressão de índices *bitmap* extensamente utilizado por diversas tecnologias é conhecido como *Byte-aligned Bitmap Code* (BBC). O FastBit, por sua vez, amplia o conceito por trás do método BBC para oferecer sua própria técnica de compressão, chamada de *Word-Aligned Hybrid* (WAH), a qual segundo análises de desempenho se mostrou mais que dez vezes mais rápida que a abordagem BBC (WU *et al.*, 2009). O WAH tira proveito do fato de um índice *bitmap* ser formado muitas vezes por grandes sequências de bits iguais (*i.e.*, séries de valores 0 ou 1) para formar as chamadas *fill words*. Essas são capazes de agrupar uma sequência muito grande de bits repetidos em uma representação de apenas 32 bits. Por outro lado, para representar trechos de um índice nos quais não há longas sequências de bits repetidos, a técnica WAH se apoia no uso das *literal words*, as quais representam os bits literalmente. Portanto, os índices comprimidos pelo FastBit são composições de *fill words* e *literal words*.

Vale ressaltar que o método de compressão é automaticamente aplicado no processo de construção de índices pelo FastBit, dado que é imprescindível para a redução do tamanho dos mesmos, consequentemente, poupando custos de armazenamento. Isso significa que o FastBit sempre irá comprimir os índices gerados em sua execução, restando ao usuário apenas a opção de descomprimi-los se assim desejar. No entanto, a operação de descompressão, apesar de possível, não é indicada, pois não acrescenta qualquer benefício em relação aos índices comprimidos.

A técnica de codificação, por sua vez, é disponibilizada pela tecnologia buscando otimizar as consultas baseadas em índices *bitmap* por meio de três métodos distintos: *equality encoding*, *range encoding* e *interval encoding*. A opção de *equality*

encoding representa o mesmo que o índice *bitmap* na sua forma básica. A opção de *range enconding* utiliza o mesmo número de *bitmaps* que a forma básica, entretanto, é projetada para realizar consultas por faixas de valores (*e.g.*, $A < 5$) de modo mais eficiente em troca de maior custo de armazenamento em disco. Já a opção de *interval encoding* também apresenta vantagens no desempenho de consultas por faixas de valores, além de contar com um número menor de *bitmaps* para a composição dos índices. Esta técnica possui um custo menor de armazenamento do que a opção de *range encoding*, todavia é mais custosa que índices na forma básica (*i.e.*, *equality encoding*).

É importante ressaltar que para cada um dos métodos há um compromisso entre eficiência na consulta e custo de espaço em disco para armazenamento dos índices codificados. A figura a seguir ilustra esta relação se consideramos uma consulta que utiliza como filtro de seleção uma faixa de valores específica, como, por exemplo, valores menores que 5 para determinado atributo A ($A < 5$). Se os dados científicos estiverem indexados a partir da técnica de *equality encoding*, teremos um custo menor de armazenamento dos índices em disco, porém, a consulta não será tão eficiente. Por outro lado, a técnica de *range encoding* seria a mais indicada para obter uma consulta mais eficiente, em troca de maior custo de armazenamento dos índices. A técnica de *interval enconding*, por sua vez, apresenta um compromisso intermediário entre o custo de armazenamento e a eficiência na consulta, sendo a mais indicada quando o filtro de seleção das consultas submetidas considera intervalos de valores (*e.g.*, $4 < A < 5$).

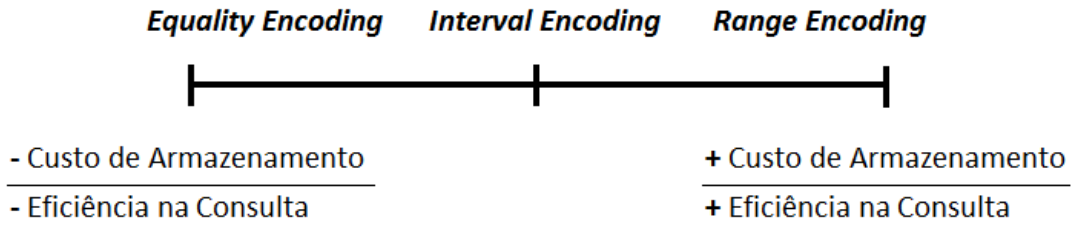


Figura 2. Relação entre custo de armazenamento e eficiência na consulta considerando um filtro de seleção como A < 5.

É possível notar que cada uma das três técnicas de codificação apresenta suas vantagens e desvantagens em relação às outras, havendo sempre uma relação proporcional entre armazenamento e otimização de consultas. Tal fator deve ser levado em consideração quando se trata de atributos com alta cardinalidade, pois em certos

casos os índices gerados podem consumir mais espaço do que o próprio dado científico. O FastBit ainda oferece a possibilidade de utilizar a técnica de codificação em dois níveis, ou seja, combinando até dois métodos possíveis, formando as opções de *equality-equality*, *range-equality* e *interval-equality*. Desse modo, ele busca reunir e equalizar as vantagens apresentadas por cada um dos métodos. Um estudo mais detalhado em relação à comparação de desempenho entre tais técnicas pode ser visto no trabalho de Wu *et al.* (2009).

Por último, a técnica de *binning* busca tirar proveito do caráter exploratório de determinados dados científicos para encontrar formas mais eficientes de armazenar e consultar seus índices. Em muitas ocasiões, certos tipos de dados científicos são expressos em valores com uma precisão muito grande (*e.g.*, ponto-flutuante), os quais mapeados necessitam de uma enorme quantidade de *bitmaps* para serem devidamente representados. Por outro lado, boa parte das consultas sobre esses dados não necessitam de um grau de precisão tão grande. Por exemplo, considere um atributo *A* que apresenta valores distintos com até quatro casas decimais, contudo as consultas realizadas sobre ele buscam por faixas de valores onde apenas uma casa decimal é relevante (*e.g.*, $A > 5{,}5$). Não há necessidade de haver um *bitmap* representando cada valor distinto em até quatro casas decimais se apenas consultas como estas são executadas.

Portanto, a técnica de *binning* permite que *bitmaps* representem grupos de valores (*i.e.*, *bins*) e não apenas valores distintos, de acordo com a precisão definida pelo próprio usuário (WU *et al.*, 2009). A solução admite que o usuário possa especificar o tipo de métrica que ele deseja para criar os agrupamentos. Ou seja, ele pode agrupar valores de diferentes maneiras: especificando a quantidade de dígitos significativos; determinando um número *n* de agrupamentos (*n bins*); ou até mesmo definindo um início e fim de uma faixa de valores.

É importante observar que a técnica de codificação exibe opções que são mais adequadas para dados de baixa cardinalidade, enquanto a técnica de *binning* se aplica bem sobre dados de estruturas específicas e de acordo com as consultas que serão realizadas. Isto significa que ambos os métodos requerem do usuário um grau de conhecimento sobre o dado científico antes mesmo da realização de consultas. Logo, a manipulação de tais técnicas exige cautela quando se trata do processo de análise exploratória de dados científicos, posto que muitas vezes o cientista não possui tanto conhecimento sobre os dados produzidos pelo experimento. Para os usuários que não são capazes de prever a forma que os dados científicos assumirão após as

transformações de dados, a principal recomendação é utilizar a opção padrão do FastBit, a qual emprega apenas o método de compressão e possui um bom desempenho em grande parte dos casos (WU *et al.*, 2009). Caso o usuário possua um grau de conhecimento maior sobre os dados manipulados, especialmente a cardinalidade dos atributos indexados e as estruturas de dados assumidas, as outras técnicas além da compressão podem ser valiosas ferramentas para geração dos índices e o processamento de consultas.

Capítulo 4 Abordagem para análise de dados científicos por meio de algoritmo de indexação de dados

Este capítulo apresenta formalmente a proposta desta dissertação, considerando as discussões dos capítulos anteriores: a abordagem para análise de dados científicos ao longo do fluxo de dados apoiada por um algoritmo de indexação *bitmap*. A solução pode ser dividida em duas partes, sendo a primeira o acesso e a indexação de dados científicos e a segunda o processamento de consultas a partir dos índices gerados.

4.1. Arquitetura para análise de dados científicos em múltiplos arquivos

A abordagem aqui apresentada adota elementos propostos pela arquitetura ARMFUL (SILVA *et al.*, 2017), desenvolvida em colaboração com as pesquisas e trabalhos do M.Sc. Vítor Silva Sousa, atualmente candidato ao doutorado do Programa de Engenharia de Sistemas e Computação da COPPE/UFRJ. O principal objetivo desta arquitetura é viabilizar a análise de dados científicos relacionados por múltiplos arquivos produzidos ao longo do fluxo de dados. Para tal, a solução é composta em parte por características das abordagens avaliadas ao longo do Capítulo 2, buscando realizar consultas sobre os dados científicos em arquivos singulares, bem como ser capaz de analisar o fluxo de arquivos gerados durante a execução de simulações computacionais. Da mesma forma, os conceitos de abstração do fluxo de dados e gerência dos dados de proveniência discutidos no capítulo anterior são fundamentais para alcançar as metas definidas, posto que os dados de proveniência e os dados de domínio formam o conjunto de interesse da análise exploratória de dados científicos.

A primeira etapa que a arquitetura deve garantir é a captura dos dados científicos de forma integrada aos dados de proveniência, ou seja, apoiar o fluxo de arquivos e os relacionamentos entre os elementos de dados rastreados pelo mecanismo de gerência do fluxo de dados no nível lógico. Logo, não apenas os dados associados à composição e execução do fluxo de dados serão capturados, mas também os dados de domínio serão acessados de forma a serem representados em uma base de proveniência. Como mostra o elemento de Captura de Dados de Proveniência na Figura 3, os dados de proveniência

são capturados e armazenados em um repositório específico (*e.g.* uma base de dados relacional) da mesma forma que a gerência de dados de proveniência pode ser implementada por um SGWfC. Tais dados podem representar, por exemplo, as especificações dos programas de simulação encadeados e os tempos de execução de cada uma das transformações de dados conduzidas por esses programas. A arquitetura ARMFUL, por sua vez, tira proveito do mecanismo presente no SGWfC Chiron para solucionar esta etapa.

Os componentes de Extração e Indexação de Dados Científicos, por sua vez, ilustram os procedimentos necessários para a representação dos dados científicos a serem integrados a uma base de proveniência. É importante ressaltar que essas etapas são opcionais, *i.e.*, os usuários não precisam se valer delas durante a execução de uma simulação computacional se assim o quiserem. Do mesmo modo, não há necessidade de realizar as etapas de extração e indexação de dados de forma concorrente, posto que apenas uma delas pode ser suficiente para garantir a captura de dados científicos. Em contrapartida, caso nenhum destes procedimentos seja realizado ao longo de uma execução, os usuários ficam restritos a análises apenas sobre os dados de execução.

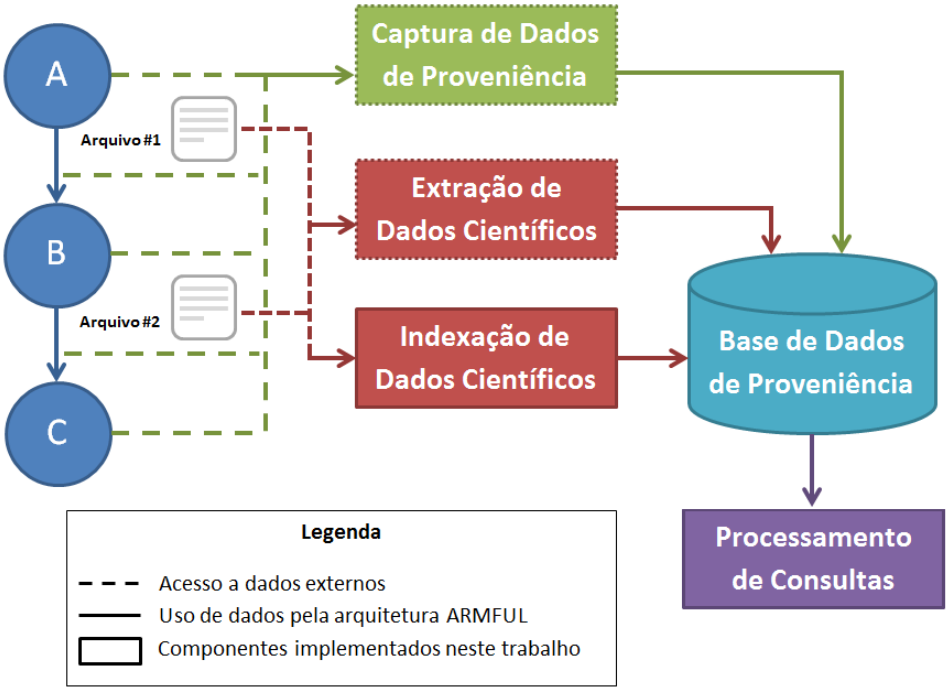


Figura 3. Arquitetura ARMFUL para análise de dados científicos em múltiplos arquivos.

O processo de extração de dados científicos também é contemplado pela arquitetura ARMFUL, no entanto, não faz parte do escopo desta dissertação. Como

visto ao longo dos capítulos anteriores, diferente dos dados de proveniência, os dados científicos apresentam grandes volumes para simulações computacionais em larga escala. Portanto, a carga de dados científicos com o próprio valor assumido para cada atributo em uma base de proveniência é uma estratégia recomendada apenas para situações nas quais poucos atributos serão capturados a cada arquivo e caso os mesmos atributos apresentem estruturas de dados simples, como inteiros e pontos flutuantes.

Em casos nos quais um grande volume de dados deve ser capturado e as estruturas de dados presentes nos arquivos são muito complexas para serem armazenadas em bases relacionais, as técnicas de indexação de dados científicos (CHOU *et al.*, 2011, KIM *et al.*, 2011) podem ser aplicadas de modo a diminuir à sobrecarga de dados na base de dados de proveniência. Isso significa que os índices gerados para referenciar os dados científicos nos arquivos serão as estruturas armazenadas na base de dados de proveniência. Para tal, a arquitetura ARMFUL propõe um componente de indexação de dados científicos capaz de garantir o acesso aos elementos de dados presentes em arquivos, a geração de índices sobre os mesmos e, por fim, a integração com uma base de proveniência. Desse modo, as consultas sobre os arquivos de dados científicos serão feitas por meio desses índices, que produzidos pelo componente e referenciados em uma base de proveniência integrada facilitam o acesso aos valores dos atributos nos arquivos.

O presente trabalho, por sua vez, propõe a implementação do componente de indexação de dados científicos a partir de um algoritmo de geração de índices *bitmap*, agregando vantagens quanto à carga de dados na base de proveniência e eficiência no processamento de consultas, conforme apresentado no capítulo de resultados experimentais. Com o apoio da ferramenta FastBit, os arquivos contendo dados científicos produzidos por programas de simulação podem ser acessados, indexados e, posteriormente, consultados a partir dos índices gerados. Portanto, a solução aqui apresentada contempla a integração entre o mecanismo do SGWfC Chiron e a ferramenta FastBit de forma a realizar a captura e a indexação de dados científicos presentes em múltiplos arquivos produzidos ao longo do fluxo de dados.

Por fim, a arquitetura ARMFUL apresenta o processamento de consultas como última etapa da abordagem, ilustrada pelo último componente da Figura 3. O objetivo desta etapa é possibilitar a especificação de consultas sobre os dados de proveniência e os dados científicos. Para tal, os níveis físico e lógico da abstração do fluxo de dados

devem ser levados em consideração, ou seja, as informações sobre o fluxo de arquivos são fundamentais para as análises que envolvem o relacionamento entre arquivos de diferentes transformações de dados. A escolha por uma base de dados relacional para a gerência dos dados de proveniência e de domínio confere maior poder analítico no processamento de consultas para apoiar a gerência do fluxo de dados. Da mesma forma, os índices gerados para os dados científicos e referenciados em uma base de proveniência integrada viabilizam o desenvolvimento de consultas.

Portanto, esta dissertação também contribui com um mecanismo de processamento de consultas capaz de alinhar os dados presentes na base de proveniência. Parte desses dados consiste nas referências aos índices criados pelo componente de indexação de dados científicos. Assim, o mecanismo de processamento de consultas apoia consultas sobre a base de dados relacional, obtendo dados de proveniência integrados às referências dos índices. Estas, por sua vez, são utilizadas pela solução para acessar os conteúdos (dados específicos do domínio) dos arquivos de dados científicos de forma direta por meio dos índices gerados. Por fim, o mecanismo de processamento de consultas é capaz de retornar resultados que associam dados referentes à execução do fluxo de dados com dados de domínio.

Nas seções a seguir, o procedimento necessário para a captura e a indexação de dados científicos será detalhado, bem como o mecanismo previsto para o processador de consultas, o qual contempla a integração de consultas sobre os dados de proveniência e os índices que referenciam os dados presentes em arquivos.

4.2. Acesso aos dados científicos ao longo do fluxo de dados

Os dados de proveniência relativos à execução de cada um dos programas de simulação são coletados e armazenados em uma base relacional ao longo do fluxo de dados, da mesma forma que é realizada por um SGWfC, como visto anteriormente. No entanto, quando se trata de dados científicos presentes em arquivos, o mecanismo de acesso deve ser capaz de realizar algumas etapas de modo a adequá-los para um armazenamento eficiente, evitando sobrecargas de dados desnecessários para as análises pretendidas. Etapas essas que podem ser divididas em quatro: leitura do conteúdo dos arquivos de dados científicos, análise léxica dos elementos de dados, coleta apenas dos atributos selecionados e, por fim, a análise sintática.

A primeira etapa é responsável pelo acesso aos dados científicos nos arquivos e a subsequente leitura de seu conteúdo. Em seguida, a análise léxica se propõe a investigar se os dados brutos obtidos na etapa de leitura correspondem ao domínio da simulação computacional em questão. Como base para essa verificação, a análise léxica utiliza um catálogo de dados científicos, o qual é composto por metadados relacionados à especificação do formato de cada arquivo científico e seu conteúdo. Uma vez que os dados foram validados, o terceiro passo diz respeito ao filtro aplicado sobre o conteúdo do arquivo, ou seja, de acordo com a especificação definida pelo usuário, quais valores de atributos devem ser selecionados para que sejam indexados e, futuramente, referenciados. Desse modo, este passo garante que não haverá um custo adicional para a geração e o armazenamento de índices sobre elementos de dados indiferentes para as análises de domínio a serem realizadas. Por fim, a análise sintática fornece informações pertinentes para o processo subsequente de indexação dos dados científicos, como a estrutura de dados apresentada por cada atributo selecionado.

4.3. Indexação de dados científicos ao longo do fluxo de dados

O componente de indexação de dados científicos proposto é responsável pela geração de índices sobre os atributos de interesse contidos nos arquivos de dados científicos, os mesmos atributos filtrados e selecionados na etapa de acesso. O principal objetivo é facilitar o acesso direto a regiões específicas do espaço de dados científicos ao passo que os metadados associados ao fluxo de dados são gerenciados. A arquitetura ARMFUL permite diferentes implementações do componente de indexação de dados científicos, ou seja, diferentes algoritmos de indexação encontrados na literatura podem ser aplicados nesta etapa. Como discutido no Capítulo 2, não apenas ferramentas já estabelecidas, como o FastBit, podem ser utilizadas para geração de índices, mas também programas específicos desenvolvidos por usuários.

Para desenvolver uma instância do componente de indexação, é necessário desenvolver três métodos distintos, denominados *index()*, *run()* e *access()*, capazes de realizar de forma completa as operações que o componente demanda. O método *index()* é responsável pela geração de índices sobre todos os atributos de interesse definidos pelo usuário, os quais estão presentes no mesmo arquivo de dados científicos. O Algoritmo 1 a seguir descreve este método, assumindo que o mesmo invoca um algoritmo de indexação (argumento *algorithm*), o qual no caso desta dissertação se trata

de um algoritmo de geração de índices *bitmap* executado pela ferramenta FastBit. O método *run()*, por sua vez, é utilizado para criar índices sobre o conteúdo de diversos arquivos de dados científicos com o mesmo formato, *i.e.*, ele invoca o método *index()* para cada arquivo que possui a mesma especificação.

Algoritmo 1. Geração de índices para acesso a dados científicos em arquivos

Entrada:
algorithm: Algoritmo para acessar dados científicos via índices
path: Caminho para o arquivo de entrada
$file_{name}$: Nome do arquivo de entrada
cts: Lista de conteúdos do dado científico
Saída:
dataSet: Conjunto de dados de saída (índices)

```
1.  function index(algorithm, path, file_name, cts):
2.      if exists file_name in path then:
3.          dataSet ← DataSet(cts)
4.          for each element in file_name do:
5.              values_element ← { }
6.              position ← 0
7.              for each content in cts do:
8.                  ctValue ← generateIndex(algorithm, content, element)
9.                  values_element[position] ← ctValue
10.                 position++
11.             end do
12.             dataSet.addElement(values_element)
13.         end do
14.         return dataSet
15.     end if
16.     return null
17. end function
```

Após a execução do algoritmo de indexação, caminhos mais eficientes de acesso aos dados são gerados para endereçar o conteúdo dos arquivos, o que melhora o tempo total da etapa de processamento de consultas focadas na análise de dados científicos. Portanto, como mencionado anteriormente, o componente de geração de índices sobre dados científicos é altamente recomendado para simulações computacionais que produzem dados científicos em larga escala ou arquivos em formatos científicos que contêm estruturas de dados mais avançadas, como malhas. Por fim, o terceiro método necessário para instanciar este componente, chamado de *access()*, trata-se justamente da implementação do mecanismo capaz de recuperar o conteúdo dos arquivos de dados científicos a partir dos índices gerados pelo método *index()*, como descrito no Algoritmo 2.

Algoritmo 2. Acesso aos dados científicos por meio dos índices

Entrada:
algorithm: Algoritmo para acessar dados científicos via índices
path: Caminho para o arquivo de entrada
file$_{name}$: Nome do arquivo de entrada
dataSet: Conjunto de dados em que cada elemento contém um índice gerado
cts: Conteúdo de interesse do dado científico
Saída:
output: Conjunto de dados de saída

```
1.  function access(algorithm, path, file_name, dataSet, cts):
2.      if exists file_name in path then:
3.          output ← DataSet(cts)
4.          for each element in dataSet do:
5.              values_element ← { }
6.              position ← 0
7.              for each ctIndex in element.values do:
8.                  if dataSet.contents[position] in cts:
9.                      ctValue ← getValueByIndex(method, ctIndex)
10.                     values_element ← values + {ctValue}
11.                 end if
12.                 position++
13.             end do
14.             output.addElement(values_element)
15.         end do
16.         return output
17.     end if
18.     return null
19. end function
```

É importante lembrar que, como discutido ao longo do Capítulo 2, as recentes tecnologias de indexação e processamento de consultas sobre dados científicos, como o NoDB e o ALACRITY, são centradas em análises sobre arquivos isolados, ou seja, os aspectos associados ao fluxo de arquivos e ao fluxo de elementos de dados não são apoiados pelas mesmas. Portanto, para superar essa limitação e apoiar a gerência do fluxo de dados, o componente de indexação proposto deve garantir a representação dos índices gerados em uma base de proveniência integrada, a qual deve seguir um modelo capaz de relacionar os dados de proveniência e os índices que referenciam dados científicos. A estrutura proposta para essa base de proveniência integrada é detalhada no Capítulo 5.

4.4. Processamento de consultas sobre os dados indexados

O último componente da abordagem apresentada engloba as funcionalidades de processamento de consultas, o que permite ao usuário realizar análises sobre os dados

de proveniência e de domínio gerados pela execução das simulações computacionais a ponto de validar ou refutar hipóteses científicas. De acordo com a abordagem apresentada, após a etapa de indexação dos dados científicos, os dados de proveniência e os índices produzidos se encontram armazenados em um SGBD centralizado. É importante notar que, normalmente, a referência para os arquivos de dados científicos não é uma informação presente na base de proveniência. Isso significa que o modelo de dados de proveniência precisa ser capaz de representar também as referências aos arquivos e relacionar os mesmos aos índices dos atributos capturados. Portanto, o componente de processamento de consultas deve ser capaz de obter não apenas os dados de proveniência da base relacional integrada, mas também os índices gerados, estes que, por sua vez, serão utilizados para acessar o conteúdo presente nas regiões de interesse dos arquivos de dados científicos.

A primeira parte deste processo é caracterizada pela consulta executada sobre os dados armazenados na base de proveniência. Usualmente, os SGBD utilizam uma linguagem de alto nível (*e.g.* linguagem SQL) para a estruturação da consulta, esta que é submetida ao processador de consultas inerente ao SGBD. O processador, por sua vez, é responsável pelas análises léxica e sintática, além de realizar a validação da consulta quanto à existência e ao significado semântico dos nomes das relações e dos atributos especificados, buscando respeitar o esquema da base de dados de proveniência. Em seguida, o elemento conhecido como otimizador de consultas se encarrega de gerar o plano de consulta mais adequado para ser executado e obter o resultado final. Toda esta etapa ocorre dentro do mecanismo presente no SGBD centralizado, o qual já possui as ferramentas mais eficientes para a validação dos dados, além da criação e execução do plano de consulta.

A segunda parte, por sua vez, é caracterizada pela invocação de um processador de consultas capaz de acessar o conteúdo dos arquivos de dados científicos por meio dos caminhos referenciados pelos índices. Tal processador de consultas necessita de dois elementos obtidos da base de proveniência como conjunto de entrada: a referência para os arquivos e os índices para os atributos. Este mecanismo pode se tratar de um código fonte específico desenvolvido ou, como no caso do presente trabalho, uma ferramenta consolidada, como o processador de consultas do FastBit. Por fim, os resultados obtidos por cada uma dessas duas consultas devem ser reunidos de acordo

com a especificação da consulta submetida, de forma a compor o resultado final da análise de dados.

Capítulo 5 Implementação de instâncias dos componentes de Indexação de Dados Científicos e de Processamento de Consultas

A partir da abordagem proposta no Capítulo 4, este capítulo apresenta a implementação dessa abordagem para a análise de dados científicos ao longo do fluxo de dados apoiada pela indexação *bitmap*. A solução é composta pela integração entre o SGWfC Chiron e a ferramenta FastBit, sendo o primeiro responsável por apoiar a captura dos dados de proveniência e de domínio ao longo do fluxo de dados gerados pela execução paralela de *workflows* científicos, enquanto o segundo realiza a etapa de indexação dos dados científicos em tempo de execução. Também é brevemente apresentado o modelo de dados desenvolvido por Silva *et al.* (2016), o qual viabiliza a gerência do fluxo de dados nos níveis físico e lógico por meio da base de proveniência integrada. A última seção do capítulo apresenta a arquitetura do processador de consultas desenvolvido para acessar os dados científicos em conjunto com os dados de proveniência, de forma a complementar a abordagem proposta.

5.1. Extensão do método de extração de dados científicos do Chiron

Para introduzir a composição da solução formada pelo arcabouço do Chiron e o método do FastBit, o ponto de partida deve ser o modo de operação desse SGWfC. Apesar de termos apresentado detalhes do SGWfC Chiron na seção 3.3, é importante destacar as camadas que formam sua arquitetura. No nível mais baixo entre elas estão as camadas de Gerência de *Workflow* e Escalonamento de *Workflow*. A primeira tem o objetivo de coordenar a execução de cada uma das atividades ao longo da execução paralela do *workflow* científico de acordo com as dependências entre os dados e a estratégia de paralelismo escolhida (*i.e.* FTF ou FAF, como visto na seção 3.3). A segunda camada, por sua vez, é responsável por particionar as atividades em ativações e distribuir estas entre os recursos computacionais disponíveis. Entende-se por ativação uma unidade de dados autocontida, ou seja, o menor conjunto de dados a ser consumido para o processamento de um programa de simulação especificado pelo usuário.

Em um nível mais acima, a camada de Processamento de Ativação precisa administrar cada procedimento ao qual a ativação será submetida. O primeiro passo é lidar com a instrumentação e a invocação dos programas de simulação de acordo com as especificações da atividade a qual a ativação pertence. Ao longo do processamento de uma ativação, o Chiron também é capaz de extrair os dados gerados pela execução da mesma e armazená-los em uma base de proveniência relacional. Por fim, os dados de proveniência capturados podem ser analisados em tempo de execução por meio de um mecanismo de processamento de consultas presente no SGBD relacional utilizado para a base de proveniência, ao qual o Chiron se mantém conectado.

É possível notar que esse SGWfC contém um mecanismo nativo de extração de dados em tempo de execução. Todavia, o procedimento adotado por tal mecanismo não é capaz de apoiar o processo de extração de dados científicos de forma completa, como definido nas discussões dos capítulos anteriores, em que defendemos o acesso, indexação e carga dos índices que representam os dados científicos. A máquina de extração de dados do Chiron, por sua vez, se limita apenas à captura de dados por meio da invocação de programas externos específicos para a atividade do *workflow*. A tecnologia ainda exige que os programas de extração escrevam os dados capturados em um arquivo de saída chamado de *ERelation.txt*. Tal arquivo possui um formato tabular próximo ao de um arquivo CSV (*Comma-Separated Value*), com a diferença que o delimitador deve ser exclusivamente o ponto e vírgula (‘;’). A primeira linha trata-se de um cabeçalho contendo a nomenclatura dos atributos extraídos, enquanto as linhas seguintes contêm os conjuntos de valores para cada atributo.

Ao lidar com os *workflows* científicos que consomem e produzem dados científicos em larga escala, o componente de extração de dados do Chiron apresenta um alto custo de carga de dados na base de proveniência, posto que são obtidos os valores para todos os atributos especificados em cada relação de saída. Da mesma forma, o acesso aos elementos de dados científicos presentes em arquivos é realizado através de programas externos específicos para o formato dos arquivos e o domínio dos dados. Para superar tais limitações, a abordagem proposta oferece um mecanismo para indexação de dados científicos orientado a análises específicas em dados científicos, enquanto que mantém a carga de dados científicos desenvolvida originalmente no Chiron. Portanto, buscando estender o mecanismo de extração de dados presente na camada de Processamento de Ativação, as seguintes alterações foram realizadas de

forma a contemplar o processo de captura de dados científicos considerando o uso de índices como um todo:

i. Método de especificação da atividade que permita ao usuário definir as estruturas de dados presentes nos arquivos de dados científicos e selecionar quais seriam os atributos de interesse para análises futuras. É importante ressaltar que o conhecimento prévio das estruturas que os dados científicos assumem é fundamental para o processo de captura dos mesmos. Da mesma forma, o usuário deve ser capaz de selecionar apenas atributos de interesse, de modo a reservar recursos de processamento e espaço de armazenamento somente para os dados necessários.

ii. Componente responsável pela invocação de uma ferramenta de indexação de dados científicos (*i.e.* FastBit) de forma a gerar índices em tempo de execução. Uma vez que o acesso ao dado é suportado pela tecnologia, a mesma pode tirar proveito do processo de geração de índices de outras ferramentas específicas para tal. As referências para os índices gerados, por sua vez, serão carregadas na base de proveniência do Chiron.

iii. Modelo de dados para a base de proveniência capaz de integrar os dados de proveniência capturados e os índices gerados que atuam como ponteiros para os atributos presentes em arquivos de dados científicos. Tal solução reduz o tempo gasto com operações de leitura e escrita em disco (*i.e.* arquivos *ERelation.txt*) e evita a carga de um grande volume de dados científicos em estruturas de dados complexas na base de dados relacional.

O tópico (i) já se encontrava parcialmente desenvolvido como parte do mecanismo de composição de *workflows* científicos do Chiron e foi estendido para apoiar novos componentes da arquitetura ARMFUL. O tópico (iii), por sua vez, foi implementado ao longo do trabalho de Silva *et al.* (2016) como base para apoiar a análise de múltiplos arquivos de dados científicos ao longo do fluxo de dados. Um dos focos do presente trabalho, portanto, foi o desenvolvimento do componente explicitado no tópico (ii), o qual se trata de uma implementação do componente de indexação de dados científicos apresentado no capítulo anterior. Para que o componente fosse acoplado ao SGWfC Chiron e respeitasse a arquitetura ARMFUL, o mesmo foi

adequado às especificações definidas pela implementação dos tópicos (i) e (iii), como descrito a seguir.

5.1.1. Especificação dos atributos de interesse

O Chiron consome como produto de entrada um arquivo XML contendo a composição do *workflow* científico, na qual são especificadas todas as atividades e suas respectivas relações de entrada e de saída. Para atender a implementação do tópico (i), este arquivo de entrada do Chiron teve seu conceito estendido de forma a compreender todas as especificações necessárias para cada um dos atributos dos arquivos de saída de uma atividade. Tais especificações englobam a estrutura de dados que o atributo assume e qual o componente responsável pela extração ou indexação dos valores que esse elemento de dado assumirá ao longo da execução da atividade. Lembrando que existem casos em que o usuário não necessita de todos os atributos de um dado científico para realizar suas análises futuras, portanto, o componente de indexação só deve ser especificado para os atributos de interesse. Considere, por exemplo, um dado de saída formado por três atributos (*A*, *B* e *C*), em que *A* e *C* sejam do tipo inteiro enquanto *B* é do tipo ponto-flutuante, no entanto, apenas *A* e *B* são do interesse do usuário para análises subsequentes. Na especificação dos dados de saída da atividade, os três atributos serão especificados com seus tipos respectivos, todavia, o componente de indexação de dados científicos será definido apenas para os atributos *A* e *B*.

```
<activity tag="act1" description="" type="MAP"
activation="./experiment.cmd" template="%=WFDIR%/template_act1">
  <relation reltype="Input" name="IAct1"/>
  <relation reltype="Output" name="OAct1" />
  <extractor name="bitmap" type="INDEXING" cartridge="FASTBIT"
search="*.foo" delimiter=","/>
  <field name="FILE" type="file" input="IAct1" output="OAct1"/>
  <field name="A" type="int" output="OAct1" extractor="bitmap"/>
  <field name="B" type="float" output="OAct1" extractor="bitmap"/>
  <field name="C" type="int" output="OAct1"/>
</activity>
```

Figura 4. Especificação de uma atividade do workflow em que os atributos A e B do conjunto de dados de saída devem ser indexados pelo componente.

5.1.2. Componente de indexação de dados científicos

Como visto anteriormente na seção 3.4, o FastBit consome dados em uma estrutura conhecida como *organização vertical*, na qual o dado científico é visto de

forma tabular, em que cada coluna representa um atributo. Consequentemente, a invocação da ferramenta também requer a especificação de cada coluna (*i.e.* atributo) junto a estrutura de dados que a coluna assume. Portanto, a extensão da especificação dos atributos de dados científicos no mecanismo de composição de *workflows* do Chiron, tornou possível a invocação da ferramenta FastBit. O componente de indexação de dados científicos recebe como parâmetros de entrada os atributos e suas respectivas estruturas de dados, definidos na composição do *workflow*. Da mesma forma, é possível especificar a opção de indexação que o FastBit deve utilizar para a geração de índices, *i.e.*, a composição entre as técnicas de *binning* e *encoding*, também descritas no Capítulo 3.

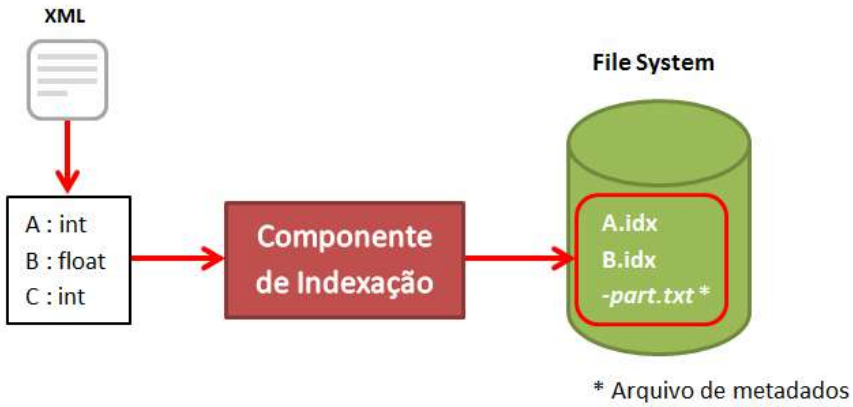


Figura 5. Especificação dos atributos de entrada do componente de indexação de dados científicos e os respectivos arquivos de saída gerados.

A Figura 5 apresenta os parâmetros de entrada que a máquina de execução do Chiron deve receber considerando um determinado dado científico produzido por uma atividade e composto pelos atributos *A*, *B* e *C*. Seguindo o exemplo da seção anterior, no qual apenas *A* e *B* são do interesse para análises futuras, o componente de indexação irá atuar apenas sobre os mesmos. De acordo com a formatação proposta pela ferramenta FastBit, os índices *bitmap* gerados serão escritos, respectivamente, nos arquivos *A.idx* e *B.idx*, além da criação de um arquivo de metadados contendo informações inerentes à forma do dado científico. A opção de indexação é um parâmetro de entrada opcional, uma vez que o componente de indexação utiliza uma configuração padrão do FastBit caso a mesma não seja especificada, como vimos na seção sobre a teoria por trás da ferramenta. Tal opção pode ser utilizada caso o usuário deseje explorar as outras técnicas de manipulação de índices que o FastBit suporta. Vale

lembrar que algumas das opções requerem um conhecimento prévio sobre o dado científico para que sejam empregadas de modo eficiente.

Ao desenvolver o componente de indexação de dados científicos, adotamos o formalismo definido pela arquitetura ARMFUL em relação à sua integração com a álgebra de *workflows* do Chiron. Em Silva *et al.* (2017), o operador algébrico *Raw Data Indexing* (ou apenas *RawI*) foi proposto para indexação de elementos contidos em arquivos de dados científicos ao longo do fluxo de dados, como definido a seguir:

$$I_{i+1} \leftarrow \text{RawI}\ (T_i, RC, RF, I_i)$$

Mais especificamente, a execução do operador aplica a transformação T_i em todos os elementos do conjunto I_i, gerando para cada elemento de entrada de I_i, um elemento de saída correspondente em I_{i+1}. A transformação T_i representa a atividade do *workflow* científico que invoca o programa capaz de acessar e indexar os dados científicos presentes nos arquivos referenciados em I_i. Para que seja possível aplicar T_i como uma operação de indexação em arquivos de dados científicos, o operador *RawI* requer ainda dois parâmetros adicionais: *RF*, o qual contém a referência para os arquivos de dados científicos que compõem o conjunto de entrada I_i, e *RC*, o qual descreve o conjunto de identificadores de cada um dos atributos que devem ser indexados e armazenados como conjunto de saída de I_{i+1}.

Como exemplo, considere uma transformação de dados T_1 regida por um operador do tipo Map (apresentado na seção 3.3 junto a álgebra de *workflows* do Chiron) responsável por invocar um programa de simulação específico que gera a relação de saída *S* a partir de uma relação de entrada *I*. Em seguida, o operador *RawI*, que executa transformação de dados T_2, é aplicado com o intuito de apoiar a análise de dados científicos sobre *S*, *i.e.*, a relação de saída de T_1. As expressões algébricas abaixo descrevem o relacionamento entre os operadores, as transformações de dados e suas respectivas relações de entrada e saída.

$$S \leftarrow \text{Map}\ (T_1, I)$$

$$V \leftarrow \text{RawI}\ (T_2, \{A, B, C\}, FOO, S)$$

A Figura 6 ilustra o fluxo de dados definido por tais expressões algébricas, mais especificamente, a transformação de dados para indexação dos dados científicos, representada por T_2 e regida pelo operador *RawI*. O programa de indexação invocado pelo operador (*i.e.*, ferramenta FastBit) acessa os elementos de dados científicos

relacionados aos atributos especificados pelo argumento *RC*, presentes nos arquivos definidos pelo argumento *RF*, e gera índices para promover o acesso direto aos elementos de dados. Portanto, a medida que o *workflow* científico é executado, o componente de indexação de dados científicos é invocado para criação dos índices sobre cada tupla gerada em *S*, ao passo que produz as tuplas da relação *V*. A partir deste modelo de dados, os usuários têm a possibilidade de submeter consultas sobre os elementos de *S* e *V*, os quais são capazes de referenciar os elementos de dados científicos em arquivos de dados. Da mesma forma, tais relações podem ser associadas a outras tabelas da base de proveniência produzidas por diferentes transformações de dados que compõem o *workflow* científico em questão, garantindo a abstração do fluxo de dados nos níveis físico e lógico.

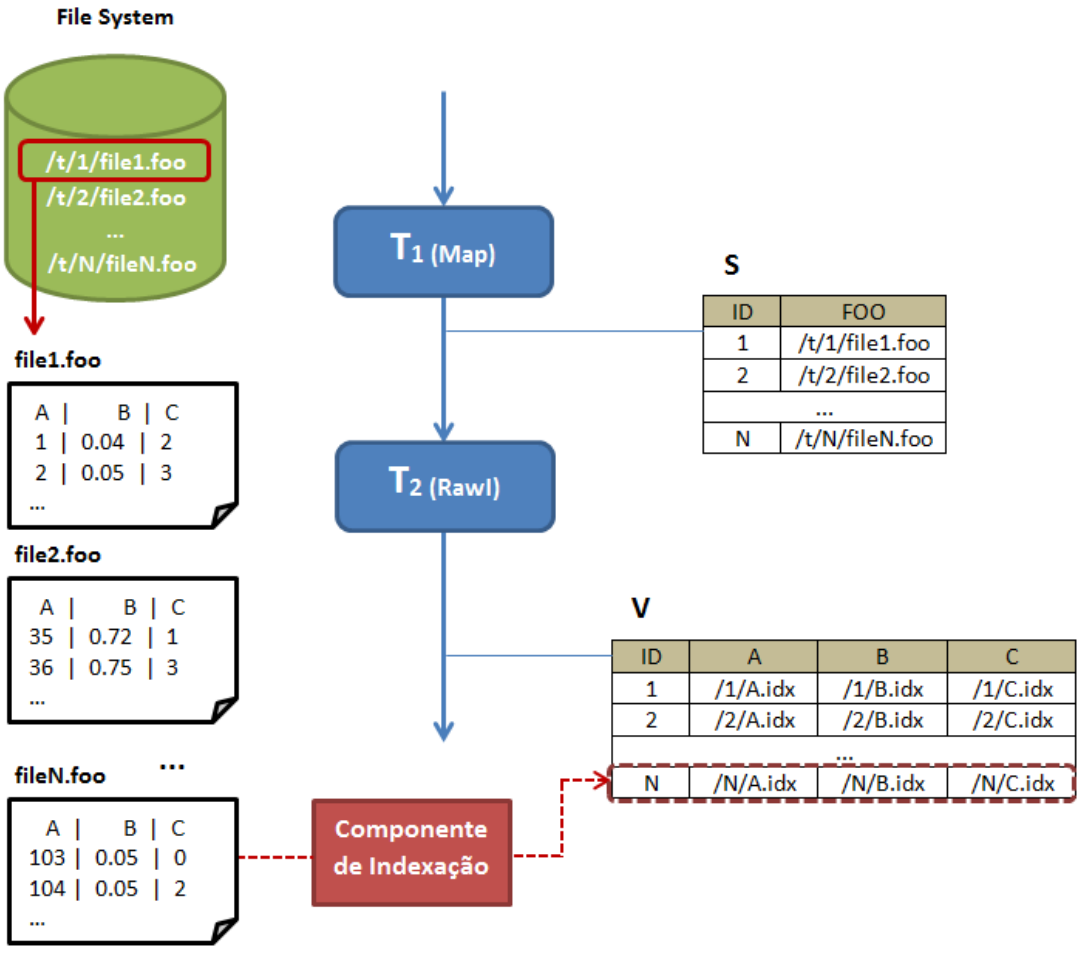


Figura 6. Implementação do operador RawI no mecanismo do Chiron para geração de índices bitmap.

Em relação a geração dos índices, temos que para cada arquivo no formato FOO produzido por T_1, o programa de indexação T_2 gera arquivos contendo índices *bitmap*

(*i.e.*, arquivos *idx*) que referenciam os elementos de cada atributo especificado (no caso, *A*, *B* e *C*). Estes arquivos de índices são armazenados no mesmo diretório que os arquivos de dados científicos, ou seja, no espaço de trabalho da ativação da transformação de dados. Portanto, as tuplas da relação *V* apresentam as referências (*i.e.*, caminhos) para cada um dos arquivos de índices gerado e disposto no sistema de arquivos dentro do diretório da ativação. O armazenamento de referências representa uma redução considerável da carga de dados na base de proveniência quando comparado ao mecanismo original do Chiron, no qual os valores dos atributos são carregados diretamente. Consequentemente, o tempo de execução da simulação computacional também é reduzido.

5.1.3. Modelo de dados de proveniência PROV-Df

Para apoiar ambos os níveis de abstração, o modelo de dados de proveniência do Chiron, conhecido como PROV-Wf, foi estendido por Silva *et al.* (2016) dando origem ao modelo PROV-Df. O novo modelo é capaz de armazenar diferentes tipos de dados na base relacional, como: (a) metadados relacionados aos dados específicos de domínio; (b) arquivos de entrada e de saída de atividades; e (c) elementos de dados extraídos de arquivos de dados científicos, como valores de atributos ou índices. Para cada conjunto de dados consumidos e produzidos em um fluxo de dados, uma tabela física contendo seus respectivos atributos e valores é criada na base de dados centralizada. Logo, as tabelas referentes aos conjuntos de dados permitem o armazenamento de dados específicos do domínio em tempo de execução, tornando possível análises que relacionam dados científicos presentes em múltiplos arquivos.

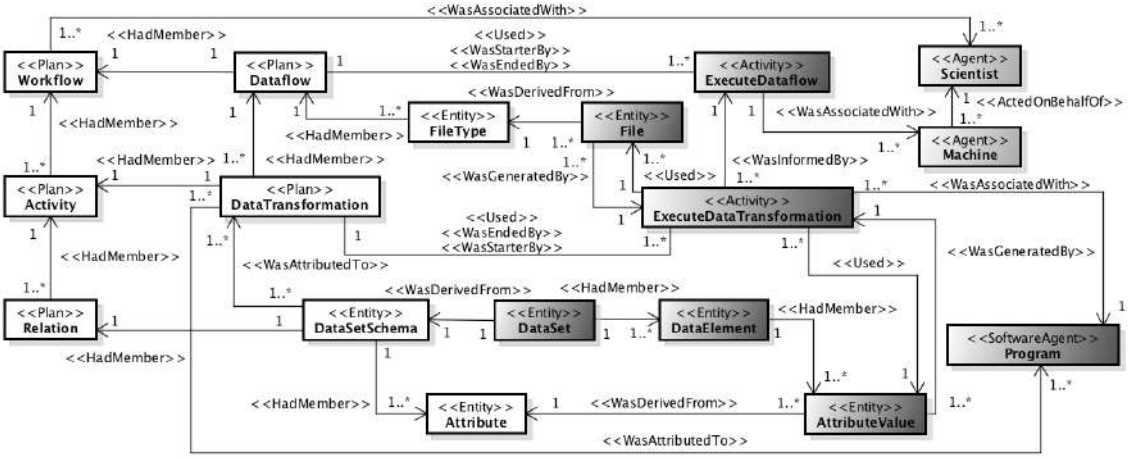


Figura 7. Modelo do PROV-Df (retirado de Silva *et al.* (2016)).

A extensão da base de proveniência do Chiron viabiliza consultas sobre o fluxo de arquivos e o fluxo de elementos de dados, respectivamente, os níveis físico e lógico da abstração de fluxo de dados. Dessa forma, os usuários são capazes de obter todo e qualquer tipo de informação a respeito do fluxo de dados e sua execução, inclusive erros ao longo de uma transformação de dados. O repositório de proveniência garante também o acesso a arquivos de dados científicos gerados por diferentes transformações de dados e armazenados em diretórios distintos no sistema de arquivos. A localidade desses arquivos pode ser guardada na base de proveniência por meio de uma referência (*i.e.*, ponteiro), a qual é recuperada para análises dos relacionamentos entre os elementos de dados contidos em diferentes arquivos. Logo, a partir desse modelo, as informações sobre o fluxo de dados podem ser associadas com os dados específicos de domínio presentes em arquivos de modo a compor análises mais precisas sobre os arquivos manipulados ao longo da execução da simulação computacional.

Com a implementação dos três tópicos abordados para a extensão do mecanismo de extração de dados científicos do Chiron, os usuários são capazes de especificar as dependências de dados entre as transformações de dados, invocar o componente de indexação de dados científicos para criação de índices, acessar o conteúdo de arquivos de dados científicos através dos mesmos índices e realizar análises sobre esses dados por meio da base de proveniência integrada em tempo de execução. De forma a cumprir a última etapa do processo de análise de dados científicos, foi desenvolvido o processador de consultas descrito na seção a seguir.

5.2. Processamento de consultas

No Capítulo 4, defendemos a eficiência dos processadores de consulta presentes nos principais SGBDs, os quais realizam as análises léxica, sintática e semântica da especificação da consulta e ainda possuem técnicas de otimização responsáveis por gerar o plano de consulta mais adequado para obter o resultado final. Uma vez que a extensão do Chiron implementa o modelo de dados do PROV-Df, o processador de consultas nativo ao SGBD utilizado como base de proveniência (*e.g.* PostgresSQL) pode ser, então, utilizado para obter as informações relacionadas ao fluxo de dados, os metadados sobre os dados de domínio e as referências para os índices gerados. Entretanto, os mecanismos dos SGBDs foram projetados para realizar consultas sob dados presentes em tabelas que respeitam o modelo relacional, ou seja, não são capazes

de realizar buscas em arquivos de dados armazenados em disco. Da mesma forma, eles possuem a capacidade de utilizar índices para referenciar dados contidos em relações, porém não podem interpretar índices gerados por outras ferramentas que apontam para regiões de arquivos.

Por outro lado, como detalhado na seção 3.4, a ferramenta FastBit possui um processador de consultas próprio, capaz de interpretar os índices *bitmap* gerados para acessar as regiões de interesse dos arquivos de dados científicos. Utilizando uma linguagem semelhante à linguagem SQL para a especificação de consultas, o mecanismo do FastBit retorna os valores assumidos pelos atributos de dados científicos a partir de buscas eficientes facilitadas pelos índices, como ilustra a figura a seguir. Basicamente, a ferramenta recebe como parâmetros de entrada:

(a) a referência para o diretório que contém os arquivos de dados científicos e seus respectivos arquivos de índices; e

(b) a especificação da consulta, a qual define a região de interesse do arquivo de dados científico (*e.g.* A < 5, para determinado atributo A.).

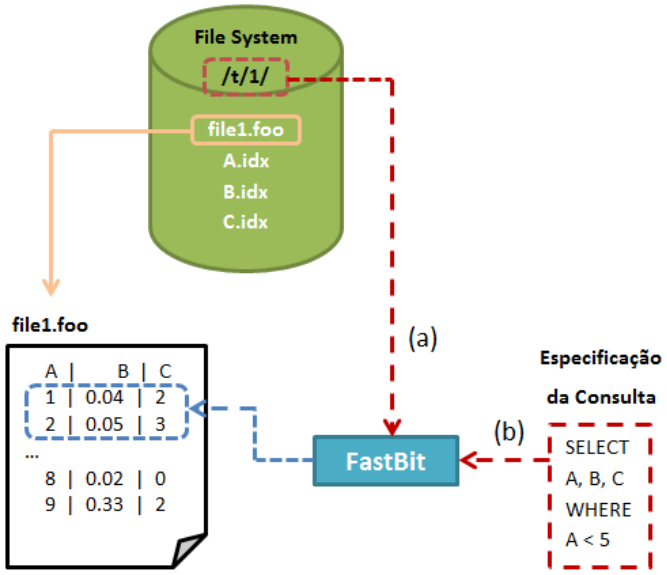


Figura 8. Mecanismo de consulta da ferramenta FastBit.

Portanto, a etapa de processamento de consultas para a análise exploratória de dados científicos baseada na abordagem proposta deve ser fragmentada em dois mecanismos distintos, que integrados retornam o resultado completo para o usuário. O primeiro tira proveito das qualidades do processador de consultas inerente ao SGBD, obtendo tanto os dados sobre o fluxo de arquivos, quanto as referências para a

localidade dos arquivos e seus respectivos índices. A partir de tais referências, o processador de consultas do FastBit pode ser acionado para recuperar os valores específicos de domínio da simulação, presentes nas regiões de interesse dos arquivos científicos em questão. Por conseguinte, a combinação (*i.e.*, junção) dos resultados de ambos processadores de consulta (SGBD e FastBit) produz a solução final, a qual pode ser formada por dados de execução, dados sobre o fluxo de dados e dados específicos de domínio.

A arquitetura do processador de consultas final pode ser vista na Figura 9. A solução requer três elementos de entrada:

i. A especificação da consulta a ser processada pelo SGBD que compõe a base de proveniência (Q_1).

ii. A especificação da consulta a ser processada pela ferramenta FastBit, buscando os elementos de dados presentes em arquivos (Q_2).

iii. Os campos de junção (J), os quais estabelecem o relacionamento entre os campos da consulta Q_1 e os campos da consulta Q_2.

Relembrando o conteúdo da seção 5.1.2, duas relações são geradas na base de proveniência após a execução de uma transformação de dados de modo a representar os dados de domínio produzidos. Enquanto uma relação contém tuplas que apontam para os arquivos de saída de uma transformação de dados, a outra relação compreende as referências dos índices. A consulta Q_1 submetida pelo usuário deve ser capaz de varrer ambas relações presentes na base de dados centralizada, podendo se aproveitar também do modelo de dados de proveniência para obter informações relevantes sobre a execução da simulação e sobre o fluxo de arquivos. O campo de junção (J), por sua vez, determina qual campo da base de proveniência especificado na consulta Q_1 deve ser extraído e utilizado como parâmetro de entrada para a consulta realizada pelo mecanismo do FastBit, operação semelhante ao conceito de *Bind-Join*. No exemplo da Figura 9, podemos ver que o caminho para o arquivo de dados científicos (*e.g.*, /t/1/file1.foo) é utilizado para tal finalidade. Como mencionado há pouco, além da referência para o arquivo de dados científicos e seus respectivos índices, o FastBit deve receber a especificação da consulta responsável por acessar as áreas de interesse do arquivo, definida pela consulta Q_2.

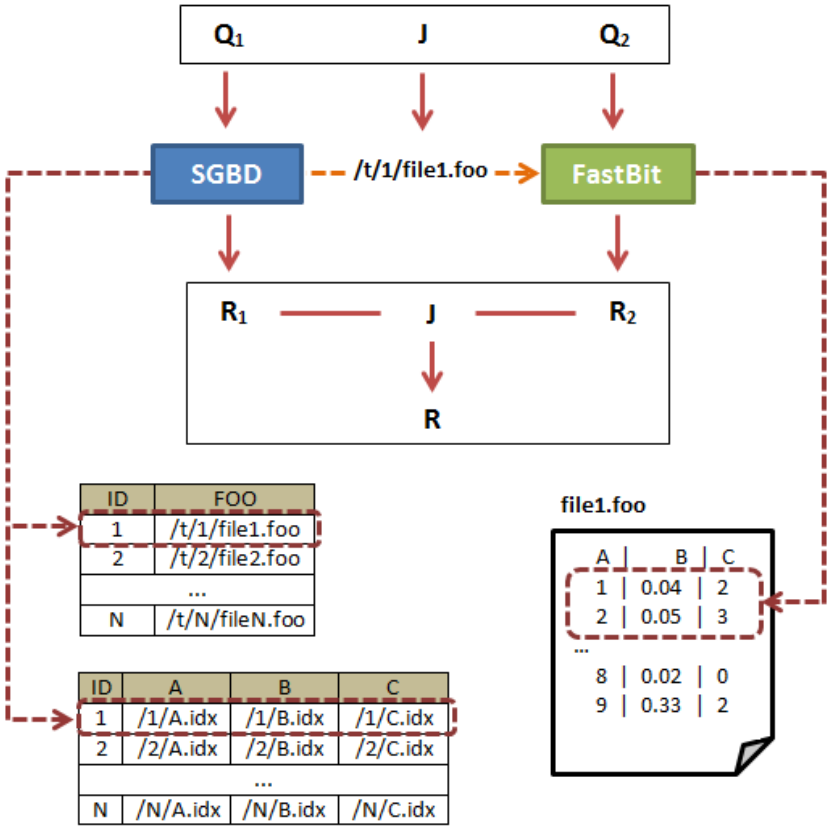


Figura 9. Operação da máquina de processamento de consultas proposta.

É importante ressaltar que a especificação de consulta Q_1 submetida ao SGBD pode acessar todas as relações presentes na base de proveniência do Chiron, de forma a compor a análise de dados com informações relacionadas à execução da simulação computacional. Desse modo, qualquer dado contido na base de proveniência pode ser relacionado a elementos de dados presentes em arquivos de dados científicos, desde que respeite o modelo de dados do PROV-Df. Os campos de junção devem ser obrigatoriamente especificados na forma como foram definidos no modelo de dados de proveniência, *i.e.*, através do nome da coluna que os representa na base de dados. Como no exemplo, se o campo de junção se trata do caminho para o arquivo de dados científicos e este é representado na base pelo campo *FILE* da tabela S, a especificação de J deve conter a referência para *S.FILE*. Em contraste, a consulta Q_2 deve conter os filtros necessários para acessar as regiões de interesse dos arquivos de dados científicos, sendo definida na linguagem própria do FastBit, a qual estende a linguagem SQL.

Por fim, o resultado da consulta processada pelo SGBD (R_1) e o resultado obtido pelo processador de consultas do FastBit (R_2) são combinados em um resultado final R, com o auxílio mais uma vez do campo de junção J. Trançando um paralelo com as

operações de junção realizadas em modelos relacionais, o processador de consultas do SGBD retorna um resultado que pode ser comparado a uma relação *outer*, a qual é utilizada por um processo conhecido como *Bind-Join* para fornecer resultados intermediários que atuam como filtros para o processador do FastBit, que exerce um papel semelhante à uma relação *inner*. Isso significa que para cada tupla que a consulta à base de proveniência retorna, uma nova consulta é realizada pelo FastBit sobre um arquivo de dados científicos afim de obter seus elementos de dados.

Capítulo 6 Avaliação Experimental

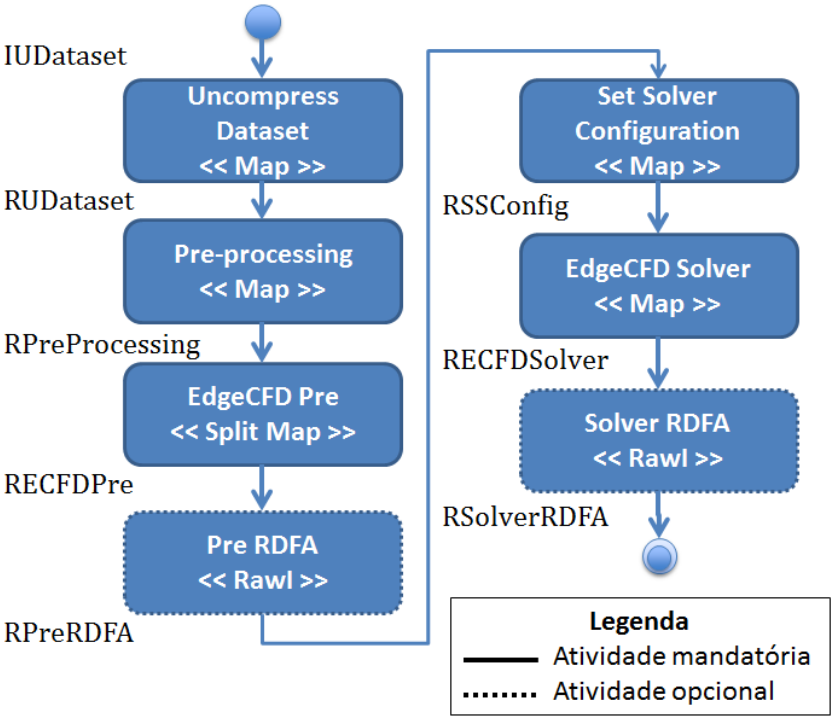
Para avaliar o desempenho da abordagem proposta nesta dissertação em um experimento real, foi utilizado como estudo de caso o *workflow* EdgeCFD, oriundo do contexto de Dinâmica de Fluidos Computacionais (no inglês, *Computational Fluid* Dynamics, e conhecido pela sigla CFD). Neste capítulo são apresentados: a definição e composição do *workflow* científico EdgeCFD, as formas de avaliação do desempenho da abordagem proposta em comparação a outras estratégias aplicadas na execução do *workflow*, o ambiente de execução paralela e os resultados obtidos de cada experimento.

6.1. Estudo de caso: *Workflow* EdgeCFD

O *software* EdgeCFD é uma aplicação bem difundida no domínio de dinâmica dos fluidos cujo objetivo é realizar uma análise tridimensional do fluxo de um fluido dentro de uma cavidade. O *workflow* EdgeCFD, por sua vez, recebe este nome por utilizar a aplicação como base para o seu processamento e a composição de suas atividades. Este é um problema muito empregado também na realização de *benchmarks* no contexto de CFD, especialmente para validação de novos códigos e métodos de solução (ELIAS e COUTINHO, 2007). O *workflow* EdgeCFD foi escolhido como estudo de caso porque o *software* EdgeCFD já foi utilizado em inúmeros experimentos reais com largo uso de visualização via ParaView. Os cenários de aplicações do *software* EdgeCFD manipulam grandes volumes de dados, estes que podem ser armazenados em arquivos de diferentes formatos e assumir estruturas de dados não convencionais para serem ingeridos em uma base de dados relacional. Para apoiar o acesso aos dados científicos em arquivos heterogêneos, a ferramenta ParaView foi utilizada de forma a extrair e adequar os dados para serem consumidos pelo FastBit, responsável pela indexação dos dados científicos antes da sua ingestão na base de dados centralizada.

A Figura 10 apresenta o *workflow* EdgeCFD utilizando a álgebra de *workflows* apresentada na seção 3.3. Os operadores algébricos que regem o comportamento das atividades estão descritos no formato <<*operador*>>. De todas as atividades deste *workflow*, apenas as atividades *EdgeCFD Pre* e *EdgeCFD Solver* utilizam binários do programa EdgeCFD (*i.e.*, *edgecfdPre* e *edgecfdSolver*). As outras atividades, como *uncompressDataset*, *preProcessing* e *setSolverConfiguration*, têm como objetivo

adequar os dados ao longo fluxo do *workflow* científico, lidando com conjuntos de dados comprimidos, arquivos de configuração para a execução dos binários do EdgeCFD e dados científicos armazenados em arquivos de formatos específicos (*e.g.*, arquivos HDF5).



Expressões algébricas considerando a utilização do componente de indexação de dados científicos:

$$\text{RUDataset} \leftarrow \text{Map}(\text{uncompressDataset}, \emptyset, \text{IUDataset})$$

$$\text{RPreProcessing} \leftarrow \text{Map}(\text{preProcessing}, \emptyset, \text{RUDataset})$$

$$\text{RECFDPre} \leftarrow \text{SplitMap}(\text{edgecfdpre}, \emptyset, \text{RPreProcessing})$$

$$\text{RPreRDFA} \leftarrow \text{RawI}\left(\begin{matrix}\text{preRDFA},\\ \{VISC, DENS, ISOLVER, DT\}, \{PREFILE\},\\ \text{RECFDPre}\end{matrix}\right)$$

$$\text{RSSConfig} \leftarrow \text{Map}(\text{setSolverConfiguration}, \emptyset, \text{RPreRDFA})$$

$$\text{RECFDSolver} \leftarrow \text{SplitMap}(\text{edgecfdsolver}, \emptyset, \text{RSSConfig})$$

$$\text{RSolverRDFA} \leftarrow \text{RawI}\left(\begin{matrix}\text{solverRDFA},\\ \left\{\begin{matrix}TIME, TIMESTEP, PRESSURE\\ VELOCITY_0, VELOCITY_1, VELOCITY_2,\\ ITER, RESIDUALS\end{matrix}\right\}, \{MESH\},\\ \text{RECFDSolver}\end{matrix}\right)$$

Figura 10. Especificação algébrica do workflow EdgeCFD com análise de arquivos de dados científicos (adaptado de Silva *et al.* (2017)).

O programa de simulação da atividade *EdgeCFD Pre* consome o conjunto de dados de entrada (*i.e.*, malha) e define as propriedades da malha (*i.e.*, fragmento de dados para o processamento paralelo e a reordenação de nós para melhorar a localidade dos dados) para que esta seja empregada no solver CFD (*i.e.*, entrada da atividade *EdgeCFD Solver*). Dessa maneira, o binário invocado pela atividade *EdgeCFD Solver* consome os arquivos gerados pela atividade *EdgeCFD Pre* e computa o comportamento de um fluido em uma malha específica durante um intervalo de tempo pré-definido. Como resultado, o solver produz diferentes arquivos nos formatos XDMF e HDF5, os quais apresentam informações sobre pressão, velocidade e outras propriedades do fluido naquele intervalo de tempo. Mais especificamente, os arquivos HDF5 contêm os valores dos atributos específicos de domínio (*e.g.*, velocidade do fluido), enquanto os arquivos XDMF apresentam ponteiros para endereçar cada arquivo HDF5 relacionado a um determinado momento.

As atividades *Pre RDFA* e *Solver RDFA* foram contempladas especificamente para apoiar a análise de arquivos de dados científicos, a qual pode ser realizada por meio de qualquer componente de extração ou indexação de dados científicos desenvolvido sob a arquitetura ARMFUL. Portanto, atividades como estas são regidas por operadores algébricos específicos para a análise de arquivos de dados científicos, como o *RawI* apresentado no Capítulo 5. Como mostra a seção 6.2 em detalhes, este *workflow* científico foi adaptado em quatro versões distintas para uma análise comparativa de desempenho entre diferentes estratégias de extração e indexação de dados científicos a partir de componentes da arquitetura ARMFUL. Dentre elas, o componente de indexação de dados científicos proposto por esta dissertação.

Normalmente, os usuários realizam esta simulação computacional com o propósito de analisar o comportamento de um fluido ao longo do tempo, o que requer a análise de dados científicos presentes em arquivos XDMF e HDF5. Entretanto, devido ao grande volume de dados contidos nesses arquivos, o tempo total de armazenamento dos dados científicos na base de proveniência pode ser maior do que o tempo de execução da própria simulação computacional (KARPATHIOTAKIS *et al.*, 2014), se considerarmos a extração de todos os elementos dos dados científicos. Nestas circunstâncias, usuários extraem somente uma região específica de interesse para realizar essas análises (*i.e.*, extração parcial), como uma linha de pontos para a cavidade apresentada no problema, considerando as condições de fronteira e o número de

Reynolds (um número não dimensional que caracteriza o fluxo do fluido). Para tal, são coletados componentes de velocidade relativos a esta região específica, os quais são comparados com as soluções de referência existentes (LO *et al.*, 2005), *i.e.*, a acurácia da solução é controlada para um determinado conjunto de parâmetros.

Como mencionado anteriormente, utilizamos a ferramenta ParaView por meio de sua API Python para acessar os dados científicos de uma região de interesse (*i.e.*, região das malhas geradas) nos arquivos XDMF e HDF5 após a execução da atividade *EdgeCFD Solver*. Como elementos de entrada para essa API, se faz necessário fornecer o caminho para o arquivo de dados científicos e os atributos que devem ser extraídos do mesmo. Esses dados selecionados dentro dos arquivos foram armazenados em novos arquivos de formato tabular (*i.e.*, similar ao formato de arquivo CSV), com intuito de facilitar o consumo dos mesmos pelas soluções de indexação de dados científicos.

Além de variar as especificações deste *workflow* no domínio de CFD, os usuários costumam fazer diferentes tentativas de execução com valores alternativos para os atributos de entrada, um processo conhecido como varredura de parâmetros (RAICU *et al.*, 2008). No cenário de simulações CFD, os usuários geralmente variam elementos de entrada como o tamanho da malha (*i.e.*, granularidade), a densidade do fluido, a viscosidade do fluido e alguns parâmetros do resolvedor, como os termos de força e estratégia de *backtracking*. No caso da simulação computacional em questão, o domínio físico do experimento é discretizado em tetraedros para ser solucionado pelo solver a partir do modelo matemático de elementos finitos. A discretização é necessária dado que, em um cenário real, a malha de entrada pode conter bilhões de vértices e tetraedros. A medida que o tamanho da malha de entrada cresce, o número de vértices e tetraedros também aumenta.

Mais especificamente, cada tetraedro contém quatro números inteiros (conectividade) que mapeiam os seus vértices e mais um inteiro como identificador do mesmo. Uma malha de elementos finitos (representada em arquivos HDF5) contém as coordenadas (*i.e.*, três dimensões) de todos os pontos e as conectividades. Os atributos de saída são a pressão e a velocidade do fluido em três componentes (*i.e.*, coordenadas *x*, *y* e *z*) em dado momento, também armazenados em formato HDF5. Para realizar uma avaliação comparativa, executamos a simulação computacional utilizando três tamanhos diferentes para a malha de entrada, ou seja, em três granularidades distintas, como mostra a Tabela 3.

Tabela 3. Tamanhos da malha de entrada considerados no experimento.

Granularidade	Número de Vértices	Número de Tetraedros
Grossa	32.109	128.336
Média	257.852	1.031.408
Fina	1.145.569	4.583.276

Os experimentos para esse estudo de caso foram executados no cluster *Stampede*, um Dell Linux Cluster composto por 6400+ Dell PowerEdge nós de servidores, sendo cada um deles formado por 2 processadores Intel Xeon E5 (Sandy Bridge) e um coprocessador Intel Xeon Phi (MIC Architecture). O cluster pertence ao Texas Advanced Computing Center (TACC)[2]. Cada nó contém 32GB de memória RAM com 8GB de memória adicional no cartão do coprocessador Xeon Phi. Os nós são interconectados com tecnologia Mellanox FDR InfiniBand em uma topologia *fat-tree* em dois níveis. Foram utilizados para os experimentos um total de 240 *cores* considerando, exclusivamente, os processadores Intel Xeon E5. Nesta arquitetura de cluster, utilizamos o SGBD PostgreSQL como base de proveniência centralizada em um dos nós da arquitetura contendo 16 *cores*. O processo de indexação de dados científicos também foi realizado de forma centralizada a partir do nó principal.

6.2. Versões do *workflow* EdgeCFD para avaliação experimental

Como mencionado anteriormente, para fins de análise e comparação entre a abordagem proposta e outras soluções possíveis de apoio à análise de arquivos de dados científicos ao longo da execução do *workflow* científico, o mesmo experimento foi adaptado para diferentes execuções de acordo com a estratégia em questão. Para tal, apenas a atividade *EdgeCFD Solver* necessitou ser modificada para suportar as diferentes estratégias de extração e indexação, posto que esta atividade manipula a maior parte dos dados gerados pela execução do *workflow* EdgeCFD, os quais encontram-se em estruturas complexas (*i.e.*, malhas) contidas em arquivos no formato HDF5. Em todos os casos, a ferramenta ParaView foi utilizada junto a uma API Python para realizar o acesso ao dado científico nos formatos XDMF e HDF5.

Basicamente, três cenários foram considerados, um para cada solução de apoio à análise de arquivos de dados científicos. A primeira solução é baseada na abordagem

proposta nesta dissertação, na qual a ferramenta FastBit é invocada por meio do componente de indexação de dados para gerar índices *bitmaps* sobre os dados científicos presente nos arquivos. A segunda abordagem também se trata de uma implementação do componente de indexação de dados científicos da arquitetura ARMFUL. Trata-se de um algoritmo para criar índices posicionais sobre os dados científicos, baseado na solução RAW (KARPATHIOTAKIS *et al.*, 2014). Por fim, o último cenário contempla apenas o processo de extração de dados científicos, sem a geração de índices, no qual os dados são acessados e extraídos pela ferramenta ParaView e armazenados diretamente na base de proveniência.

Considerando cada um destes cenários, três versões distintas do *workflow* EdgeCFD foram desenvolvidas buscando avaliar o apoio a extração e indexação de dados científicos. Uma quarta versão do *workflow* foi também desenvolvida para contemplar um cenário sem o apoio à análise de arquivos de dados científicos ao longo do fluxo de dados. Em todos os casos, foram considerados os arquivos de dados científicos de uma região de interesse específica para a simulação CFD, como descrito anteriormente. A Figura 10 (apresentada na seção 6.1) ilustra também as duas atividades do *workflow* científico, *Pre RDFA* e *Solver RDFA*, que foram concebidas especificamente para a análise sobre arquivos de dados científicos. As mesmas são atividades opcionais, dado que elas podem ser removidas da composição de acordo com o tipo de abordagem utilizado. Portanto, quatro versões foram consideradas para o *workflow* EdgeCFD:

- **Base:** Versão do *workflow* sem apoio a análise de arquivos de dados científicos. As atividades *Pre RDFA* e *Solver RDFA* não estão presentes na composição deste *workflow*. Portanto, esta versão não suporta o processamento de consultas sobre elementos de dados científicos;

- **Extração:** Versão do *workflow* que contempla ambas atividades para garantir o processo de extração de dados científicos realizado com a ferramenta ParaView. Esta versão considera uma implementação do componente de extração de dados científicos da arquitetura ARMFUL, uma vez que permite a extração de dados científicos presentes nos arquivos XDMF e HDF5 sem apoio a indexação de dados;

- **Indexação Posicional:** Versão do *workflow* com apoio a indexação de dados científicos a partir de algoritmo de indexação posicional. Esta versão contempla uma implementação do componente de indexação de dados científicos que utiliza um programa de geração de índices posicionais para indexar o conteúdo dos arquivos XDMF e HDF5. A ferramenta ParaView também é utilizada para garantir o acesso aos dados científicos presentes nos arquivos;

- **Indexação Bitmap:** Versão do *workflow* com apoio a indexação de dados científicos a partir de algoritmo de indexação *bitmap*. Esta versão considera a implementação do componente de indexação de dados científicos proposta nesta dissertação para indexar o conteúdo dos arquivos XDMF e HDF5. A ferramenta ParaView também é utilizada para garantir o acesso aos dados científicos presentes nos arquivos.

6.3. Resultados experimentais

De modo a realizar uma análise completa do desempenho da abordagem proposta em comparação às outras estratégias, foram realizadas 4 avaliações experimentais utilizando o *workflow* EdgeCFD. Cada uma delas foi concretizada de acordo com os objetivos a seguir:

- Custo de tempo para a análise de arquivos de dados científicos (SOBRECARGA). Medição do custo de tempo para extrair ou indexar dados científicos (*i.e.*, realizar o processo de análise de arquivos de dados científicos) a partir de cada uma das quatro versões do *workflow* EdgeCFD utilizando a malha em granularidade fina como entrada. Também comparamos este tempo total de processamento com outros custos de tempo associados ao armazenamento de dados na base de proveniência (carga de dados de proveniência e de dados científicos na base de dados) e a execução das atividades do *workflow*;

- Análise da carga de processamento para indexação de arquivos de dados científicos (INDEXAÇÃO). Medição do custo de tempo para geração de índices sobre dados científicos presentes em arquivos com diferentes volumes de dados. Cada carga de processamento analisada considera um

tamanho específico da malha definida no conjunto de dados de entrada. Cada versão do *workflow* foi executada três vezes, de modo a variar o conjunto de dados de entrada, assumindo as três granularidades diferentes para a malha de entrada: grossa, média e fina;

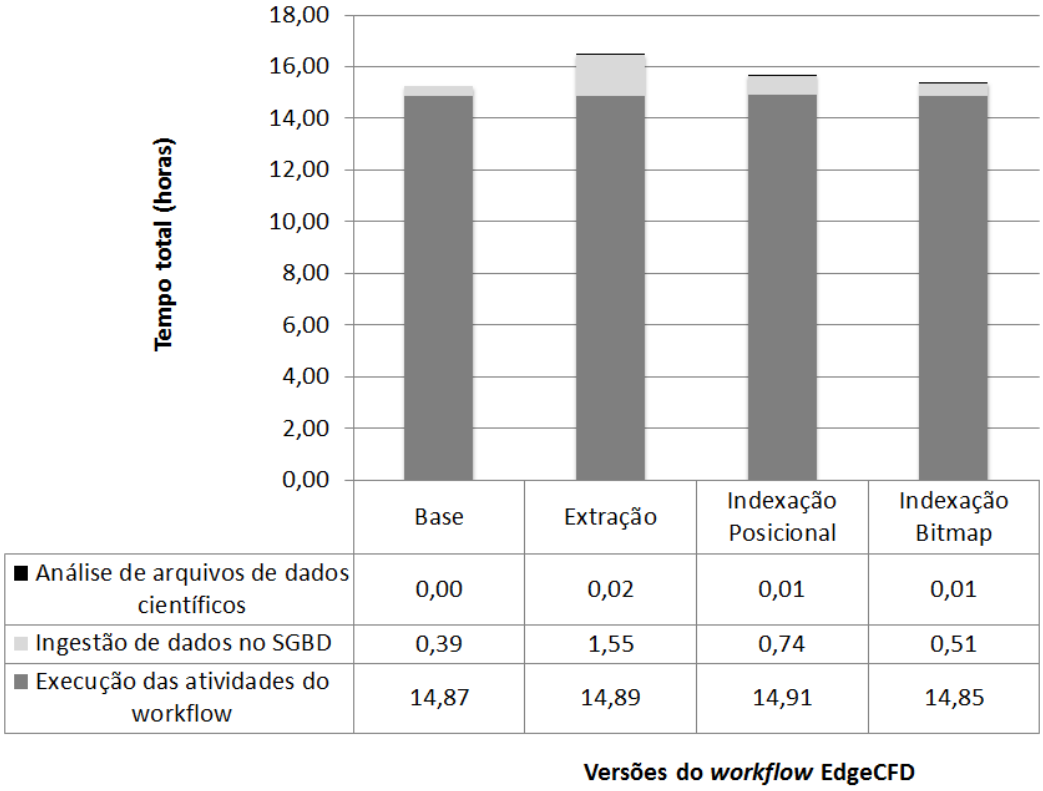
- Análise da carga de processamento para a ingestão de dados na base de proveniência (INGESTÃO). Avaliação do custo de tempo para armazenar dados de proveniência e dados científicos em uma base dados de proveniência de acordo com diferentes volumes de dados, semelhante à avaliação experimental de INDEXAÇÃO.
- Análise das possibilidades e características que o processador de consultas proposto apresenta na especificação de consulta sobre dados científicos (ANÁLISE). Apresentamos algumas consultas definidas para analisar o conteúdo específico de domínio em arquivos produzidos pelo experimento, elementos específicos relacionados por múltiplos arquivos de dados científicos e dados de desempenho da execução.

6.3.1. Análise de arquivos de dados científicos

O primeiro experimento, denominado como SOBRECARGA, foi realizado com o objetivo de medir o desempenho de cada uma das quatro versões do *workflow* EdgeCFD em relação ao tempo total de processamento das seguintes etapas: execução das atividades do *workflow*; análise de arquivos de dados científicos (tempo de extração dos dados científicos e/ou geração dos índices); e armazenamento de dados em um SGBD centralizado. Desse modo, podemos avaliar como a abordagem proposta nesta dissertação para a análise de arquivos de dados científicos se comporta em comparação a outras estratégias, como a extração de dados científicos ou a utilização de outros algoritmos de indexação. Podemos ver também o quanto de sobrecarga no tempo total de processamento a abordagem de indexação *bitmap* acrescenta em relação a versão do *workflow* sem apoio a análise de arquivos de dados científicos. Para cada uma das abordagens aplicadas, executamos o *workflow* científico considerando a malha em granularidade fina no conjunto de dados de entrada, uma vez que este representa um cenário real de uma simulação do domínio de CFD.

A figura a seguir apresenta o tempo total para execução das atividades do *workflow*, análise de arquivos de dados científicos e ingestão de dados em cada versão

do *workflow*. Com tais resultados, podemos notar que a versão do *workflow* científico que utiliza o algoritmo de indexação *bitmap* apresenta uma sobrecarga no tempo total de processamento de apenas 7,00 minutos em relação à versão Base (*i.e.*, sem apoio a análise de dados científicos). A abordagem de extração de dados científicos, por sua vez, aparece como o pior cenário, visto que acrescenta uma sobrecarga de 72,20 minutos, enquanto a abordagem de indexação posicional leva 24,20 minutos a mais.



	Base	Extração	Indexação Posicional	Indexação Bitmap
Análise de arquivos de dados científicos	0,00	0,02	0,01	0,01
Ingestão de dados no SGBD	0,39	1,55	0,74	0,51
Execução das atividades do workflow	14,87	14,89	14,91	14,85

Figura 11. Custo de tempo para a análise de arquivos de dados científicos considerando a malha fina no conjunto de entrada do workflow.

Especificamente em relação a etapa de ingestão de dados no SGBD, a versão do *workflow* com apoio a extração de dados científicos demora 62,37 minutos a mais que a abordagem que aplica a indexação *bitmap*. Portanto, podemos concluir que o armazenamento de dados em um SGBD é um grande gargalo, mesmo considerando a extração parcial de dados científicos. Por outro lado, notamos que o custo para executar o algoritmo de indexação *bitmap* é de aproximadamente 42 segundos, o que representa apenas 0,08% do tempo total de execução do *workflow*. Constatamos, então, que a geração de índices pode reduzir o tempo de armazenamento de dados ao passo que não interfere significativamente no tempo total do *workflow*.

6.3.2. Análise das estratégias de indexação de dados científicos

Neste segundo experimento, denominado como INDEXAÇÃO, avaliamos especificamente a etapa de geração de índices quando variamos o número de vértices da malha definida no conjunto de entrada do *workflow* EdgeCFD, ou seja, considerando as diferentes granularidades da malha. Assim, analisamos o tempo total sequencial do processo de criação de índices de cada um dos algoritmos de indexação empregados a medida que alteramos o volume de dados científicos.

Para a versão do *workflow* que utiliza a extração de dados científicos consideramos como tempo total de indexação o tempo que o SGBD leva para analisar os dados científicos extraídos e gerar índices internamente de modo a ser capaz de correlacionar tais dados aos dados de proveniência. Isto porque a estratégia de extração de dados científicos não gera índices a partir da invocação de programas externos, apenas carrega os dados científicos extraídos na base de dados centralizada. No caso das abordagens que utilizam os algoritmos de indexação para o conteúdo de arquivos de dados científicos (*i.e.*, *bitmap* ou posicional), o custo de tempo para gerar índices internos ao SGBD é somado ao tempo de custo para geração dos índices para acessar o conteúdo dos dados científicos em arquivos.

A partir dos resultados experimentais ilustrados na Figura 12, notamos que a medida que aumentamos o número de vértices da malha de entrada (*i.e.*, granularidade da malha) a geração de índices pela estratégia de extração de dados científicos torna-se mais lenta em relação às outras abordagens com algoritmos de indexação. Considerando a malha de entrada em granularidade fina (*i.e.*, mais próxima ao cenário real), temos que a abordagem proposta nesta dissertação apresenta uma redução de 46,39% no tempo adicional de geração de índices quando comparada a abordagem com extração de dados científicos. A versão Base, por sua vez, não apresenta a extração de dados científicos em arquivos, logo, não gera índices por meio do SGBD, o que leva a um tempo total sequencial para indexação de dados científicos equivalente a zero.

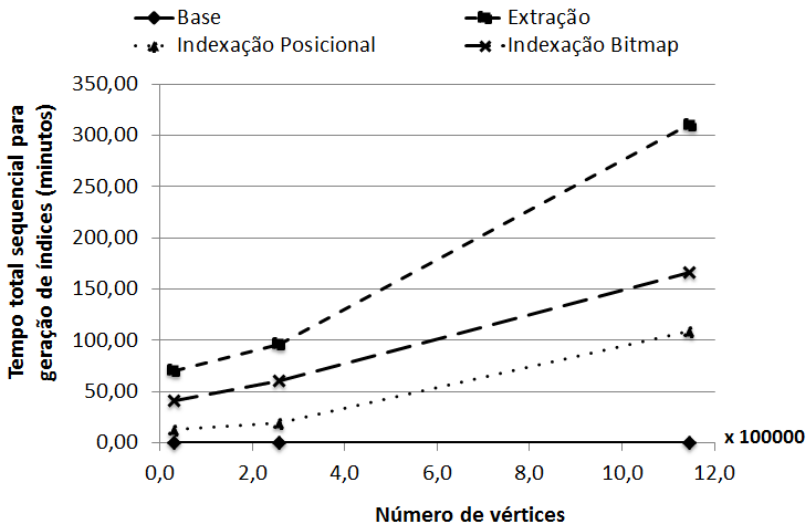


Figura 12. Tempo total sequencial para geração de índices para diferentes cargas de processamento.

A utilização da indexação *bitmap* em dados científicos, no entanto, apresentou um desempenho pior em relação a geração de índices por meio da indexação posicional. Para a malha de entrada em granularidade fina, o algoritmo *bitmap* apresenta uma diferença de 57,60 minutos a mais para o algoritmo de indexação posicional. Isto ocorre devido ao fato de os atributos presentes nos arquivos de dados científicos de saída do resolvedor EdgeCFD apresentarem uma grande variedade de valores distintos. Como detalhado nos capítulos iniciais, os índices *bitmap* tendem a ser maiores à medida que a cardinalidade (*i.e.*, número de valores distintos) do atributo aumenta. Consequentemente, este algoritmo de indexação leva mais tempo para a geração de índices se comparado ao algoritmo posicional, o qual necessita apenas de dois valores inteiros para representação dos atributos de dados científicos.

Embora a ferramenta FastBit permita aos usuários modificarem a opção de indexação aplicada pela ferramenta, como discutido na seção 3.4, este experimento não considerou tal possibilidade, adotando a configuração padrão. Portanto, seria possível ainda reduzir o tempo total de geração de índices utilizando o algoritmo de indexação *bitmap*, uma vez que as opções pelas técnicas de *binning* e *encoding* da ferramenta FastBit poderiam ser exploradas. Para ilustrar a comparação entre estas possibilidades, realizamos algumas execuções do *workflow* EdgeCFD em uma máquina local variando a opção de indexação e considerando a malha em granularidade grossa no conjunto de entrada. Foram utilizados um total de 15 *cores* para cada execução do experimento.

A Tabela 4 apresenta o tempo total para a geração dos índices em cada uma das execuções realizadas utilizando diferentes técnicas para indexação. As versões de indexação que empregaram as técnicas de *binning* e *interval encoding* apresentaram uma redução de, respectivamente, 32,00% e 35,02% no tempo total de indexação em relação a configuração padrão aplicada nos outros experimentos. Por outro lado, a opção de indexação que considera a técnica de *range encoding* mostrou um desempenho consideravelmente pior. Desse modo, podemos concluir que é possível obter melhores tempos para a geração dos índices por meio de diferentes opções de indexação que a ferramenta FastBit proporciona. No entanto, é importante notar que isto depende diretamente da estrutura de dados e do domínio assumido pelos atributos nos arquivos de dados científicos, sendo necessário um conhecimento prévio sobre o conteúdo dos arquivos de dados científicos para que seja possível prever o melhor método a ser aplicado.

Tabela 4. Tempo total para indexação do conteúdo de arquivos de dados científicos produzidos pelo workflow EdgeCFD variando a opção de indexação da ferramenta FastBit.

Opção de Indexação	Tempo Total (minutos)
Padrão	3,88
Binning	2,64
Range	13,38
Interval	2,52

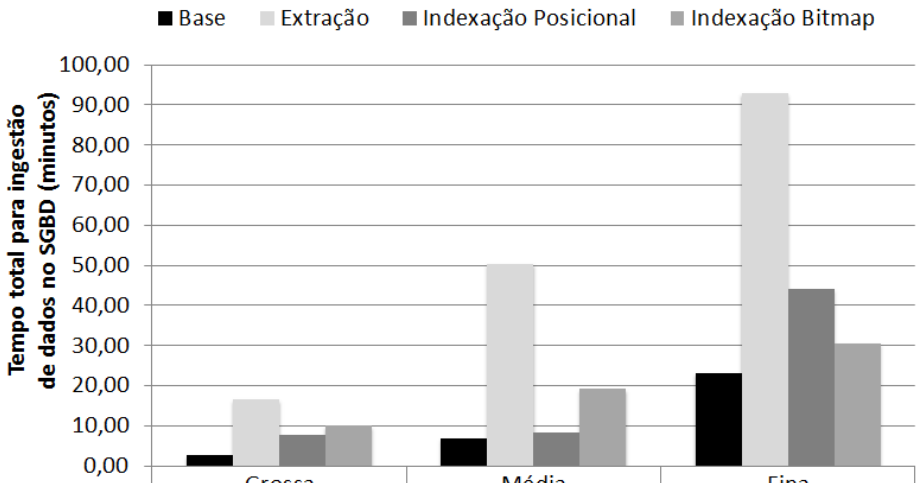
6.3.3.Ingestão de dados científicos

O terceiro experimento, denominado como INGESTÃO, avalia a ingestão de dados em um SGBD quando variamos mais uma vez a granularidade da malha de entrada do *workflow* EdgeCFD. A carga de processamento deste experimento é similar à do experimento descrito na seção anterior. Para cada tamanho de malha, analisamos o tempo total para a ingestão de dados na base de dados de proveniência ao passo que alternamos a estratégia utilizada para o apoio a análise de arquivos de dados científicos.

A Figura 13 apresenta os resultados obtidos para cada uma das quatro versões do *workflow* EdgeCFD considerando os três tipos de granularidade definidos. Podemos notar que o custo de ingestão de dados aumenta de acordo com o volume de dados de entrada, visto que mais dados de proveniência, execução e dados científicos são

armazenados na base de proveniência do SGWfC. A versão do *workflow* baseada na extração de dados científicos apresentou o maior tempo para a ingestão de dados em todos os casos, uma vez que é responsável por acessar o conteúdo dos arquivos de dados científicos, analisar e adequar os mesmo em estruturas de dados específicas e carregar tais dados estruturados na base de proveniência. Diferente desta versão, as abordagens que utilizam algoritmos de indexação de dados científicos apresentam estruturas de dados otimizadas para endereçar dados científicos, como discutido anteriormente.

Mais especificamente, considerando a malha em granularidade fina, percebemos uma redução de 67,11% no tempo para ingestão de dados no SGBD quando utilizamos a abordagem proposta nesta dissertação (30,57 minutos) em comparação com a extração de dados científicos (92,95 minutos). Ao compararmos o desempenho das duas abordagens que utilizam algoritmos de indexação de dados científicos, notamos que, apesar da indexação *bitmap* ser superada pela indexação posicional para a malha em granularidade grossa e média, em um cenário real (*i.e.*, granularidade fina) a indexação *bitmap* apresenta melhor desempenho (redução de 30,68% no tempo de ingestão de dados no SGBD).



	Grossa	Média	Fina
Base	2,58	6,85	23,20
Extração	16,52	50,37	92,95
Indexação Posicional	7,75	8,21	44,11
Indexação Bitmap	9,96	19,16	30,57

Granularidade da malha

Figura 13. Custo de ingestão de dados no SGBD variando o tamanho da malha.

Tabela 5. Espaço de armazenameno na base proveniência integrada para o workflow EdgeCFD variando o tamanho da malha de entrada.

Versão do *workflow*	Espaço de armazenamento em GB		
	Malha grossa	Malha média	Malha fina
Base	2.88	2.98	3.20
Extração	4.80	5.76	6.40
Indexação Posicional	5.44	5.56	5.73
Indexação Bitmap	3.01	3.05	3.20

O algoritmo de indexação *bitmap*, implementado a partir da invocação da ferramenta FastBit, armazena na base de proveniência integrada apenas a referência para os arquivos de índices gerados e, portanto, apresenta um tempo total reduzido na etapa de ingestão de dados. Isto significa que a implementação do componente de dados científicos proposta nesta dissertação requer menos espaço de armazenamento na base de dados de proveniência do que qualquer outro componente para análise de arquivos de dados científicos. Como mostra a Tabela 5, nos experimentos utilizando a malha fina como entrada, identificamos que a base de proveniência requer 3,20 GB de espaço de armazenamento para a versão Base do *workflow*, assim como para a abordagem que utiliza a indexação *bitmap*, enquanto a versão com extração de dados requer 6,40 GB e o algoritmo posicional 5,73 GB.

6.3.4. Processamento de consultas

A última avaliação realizada nesta série de experimentos foca nas versões do *workflow* com apoio a análise de arquivos de dados científicos para analisar o potencial de consultas sobre o fluxo de dados, sendo este experimento denominado como ANÁLISE. Todas as consultas foram realizadas em relações da base de proveniência que contém dados científicos obtidos de arquivos. Executamos 3 consultas para considerar o tempo total de uma atividade do *workflow* e os valores de alguns atributos de forma a analisar dados específicos de domínio presentes em múltiplos arquivos de dados científicos relacionados.

Na primeira consulta descrita na Figura 14, denominada *FLUXO_DE_ELEMENTOS*, identificamos múltiplos valores de dados estruturados em diferentes malhas após a execução do resolvedor EdgeCFD com uma média de velocidade na coordenada *x* (atributo *VELOCITY_0* na relação *RSolverRDFA* ilustrada na Figura 10) entre 0,40 e 0,50 para diferentes valores do parâmetro *ISOLVER*. Também relacionamos estas malhas a propriedades do fluido, como atributos de

viscosidade (*VISC*) e densidade (*DENS*), as quais são mantidas em valores fixos de 1,00 e 0,001, respectivamente. Portanto, esta análise investiga o conteúdo de malhas na simulação CFD considerando o fluxo de elementos de dados entre transformações de dados diferentes (*Pre RDFA*, *Set Solver Configuration*, *EdgeCFD Solver* e *Solver RDFA*).

```
 SELECT rprerdfa.ISOLVER, rsolvrdfa.timestep,
 AVERAGE(rsolvrdfa.velocity_0) as vx
 FROM RPreRDFA rprerdfa, RSSConfig rconf, RECFDSolver rsolv,
RSolverRDFA rsolvrdfa
 WHERE rprerdfa.nextActivationID = rconf.previousActivationID
 AND rconf.nextActivationID = rsolv.previousActivationID
 AND rsolv.nextActivationID = rsolvrdfa.previousActivationID
 AND vx>0.40 AND vx<0.50 AND rprerdfa.DENS=1.00 AND
rprerdfa.VISC=0.001
 GROUP BY rprerdfa.ISOLVER, rsolvrdfa.timestep;
```

Figura 14. FLUXO_DE_ELEMENTOS – Análise da velocidade do fluido na coordenada x variando o parâmetro de entrada do resolvedor.

A partir desta definição, os usuários são capazes de restringir suas análises para malhas que sejam relevantes e compreender a velocidade do fluido na coordenada *x* ao longo do tempo, considerando valores específicos de viscosidade e densidade, ao passo que variam parâmetros de entrada do resolvedor. Mais especificamente, as tabelas *RPreRDFA* e *RSolverRDFA* apresentam conteúdos de dados científicos capturados de arquivos usando os componentes definidos pela arquitetura ARMFUL (*e.g.*, implementação do componente de indexação de dados científicos apresentada nesta dissertação).

A segunda consulta ilustrada na Figura 15, denominada *FLUXO_DE_DADOS*, contempla a seleção de três escalares (atributos *VISC*, *ITER* e *RESIDUALS*) que representam a convergência da execução do resolvedor CFD, considerando valores distintos para a viscosidade do fluido, ao passo que a densidade do mesmo é mantida fixa. Também são considerados um conjunto específico de parâmetros do resolvedor (atributo *ISOLVER*), bem como o tempo de execução do resolvedor CFD (*i.e.*, *time step*).

```
 SELECT rprerdfa.VISC, rsolvrdfa.ITER, rsolvrdfa.RESIDUALS
 FROM RPreRDFA rprerdfa, RSSConfig rconf, RECFDSolver rsolv,
RSolverRDFA rsolvrdfa
 WHERE rprerdfa.nextActivationID = rconf.previousActivationID
 AND rconf.nextActivationID = rsolv.previousActivationID
 AND rsolv.nextActivationID = rsolvrdfa.previousActivationID
 AND rprerdfa.DENS = 1 AND rprerdfa.ISOLVER = 3 AND
rsolvrdfa.TIME=1.5;
```

Figura 15. FLUXO_DE_DADOS – Análise da convergência da simulação CFD considerando valores fixos para a densidade do fluido, o parâmetro de entrada do solver (ISOLVER) e o instante temporal.

Similarmente à primeira consulta, este tipo de análise permite que os usuários relacionem elementos de dados de saída do *workflow* com elementos de dados de entrada que contém propriedades do fluido, ao passo que também representa a convergência do modelo CFD a partir dos atributos *ITER* e *RESIDUALS*. Não obstante, o resultado desta consulta fornece características sobre a execução do resolvedor CFD para um determinado momento no tempo (atributo *TIME*), como ilustra a Figura 16.

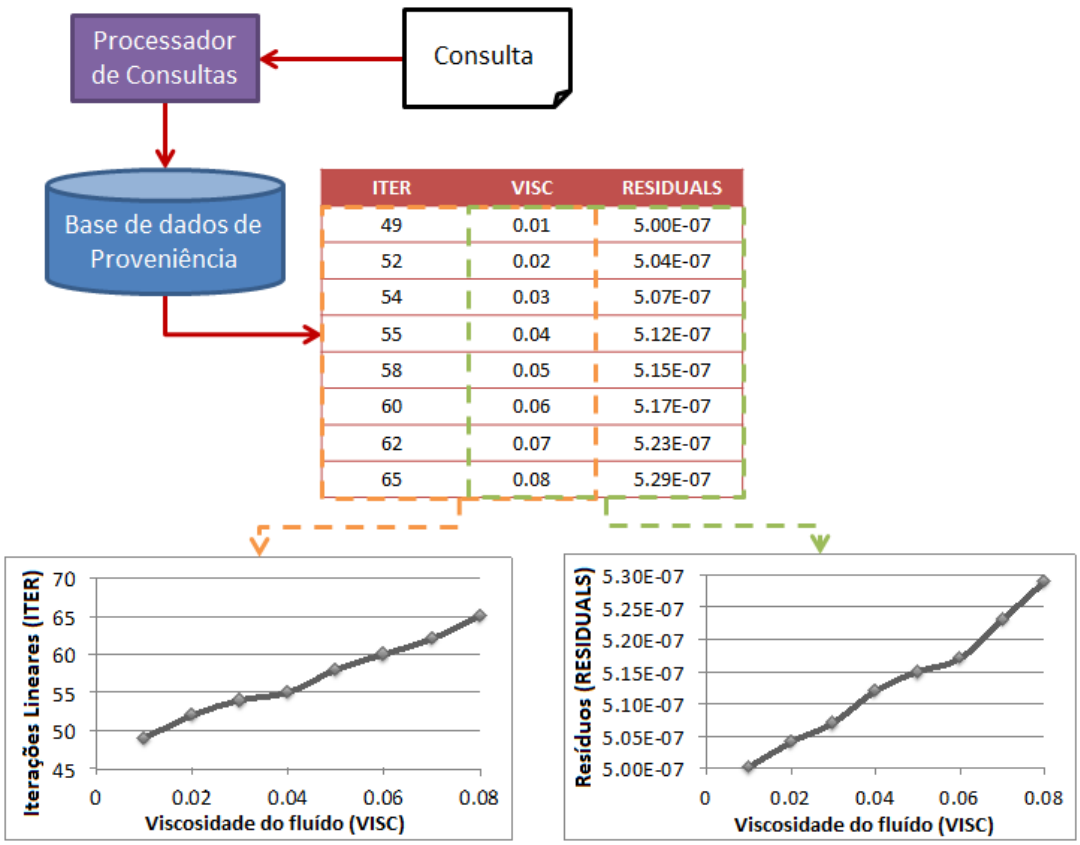


Figura 16. Resultados da consulta para análise de convergência do solver CFD.

Dentre as versões do *workflow* consideradas neste trabalho, apenas a versão baseada na extração de dados científicos é capaz de realizar as consultas descritas utilizando somente o processador de consultas do SGBD, uma vez que todos os conteúdos dos dados científicos se encontram armazenados na base de proveniência. Lembrando que a versão deste *workflow* sem apoio à análise de arquivos de dados científicos (abordagem Base) não suporta estas consultas, uma vez que não apresenta dados científicos armazenados na base de proveniência do Chiron. Isto significa que os usuários não são capazes de executar consultas analíticas ou direcionar o workflow científico em tempo de execução nesta versão.

As versões do *workflow* que contemplam diferentes estratégias de indexação de dados científicos, por sua vez, necessitam do processador de consultas desenvolvido (apresentado nesta dissertação ao longo das seções 4.4 e 5.2) para realizar tais consultas. Como visto anteriormente, o mecanismo de processamento de consultas proposto, uma extensão do componente presente na ARMFUL, é capaz de acessar os dados científicos presentes em arquivos a partir dos índices gerados e referenciados no SGBD, além de reunir estes dados aos resultados obtidos a partir de consultas sobre as relações da própria base de proveniência. No caso específico das versões deste *workflow* CFD que consideram a geração de índices, o SGBD do Chiron fornece os valores para densidade, viscosidade e parâmetros do solver, enquanto as médias de velocidade, presentes em arquivos HDF5, devem ser acessadas nas malhas de saída a partir dos índices criados.

```
SELECT rprerdfa.VISC, a.elapsedTime
FROM RPreRDFA rprerdfa, RSSConfig rconf, RECFDSolver rsolv,
     RSolverRDFA rsolvrdfa, activation a
WHERE rprerdfa.nextActivationID = rconf.previousActivationID
AND rconf.nextActivationID = rsolv.previousActivationID
AND rsolv.nextActivationID = rsolvrdfa.previousActivationID
AND rsolvrdfa.previousActivationID = a.activationID
AND rprerdfa.DENS = 1 AND rprerdfa.ISOLVER = 3;
```

Figura 17. DADOS_DE_DESEMPENHO – Análise de desempenho do solver CFD considerando valores fixos para a densidade do fluido e o parâmetro de entrada do solver (ISOLVER).

A terceira e última consulta, denominada *DADOS_DE_DESEMPENHO*, investiga o tempo de execução do resolvedor em determinadas transformações de dados. Esta consulta apresenta condições semelhantes à consulta *FLUXO_DE_DADOS*, visto que considera valores distintos para viscosidade do fluido enquanto mantém fixos os valores para: densidade, tempo (*time step*) e determinados parâmetros do solver.

Entretanto, esta consulta combina estes dados científicos com o tempo total de execução de uma ativação da atividade *Solver RDFA*, ou seja, tempo de execução do solver. Neste caso, é possível relacionar elementos de dados de múltiplos arquivos de dados científicos com metadados associados à execução do *workflow* (*i.e.*, dados de proveniência), que neste caso trata-se do tempo total de execução de uma ativação específica. Este tipo de consulta é empregado frequentemente pelos usuários para monitorar o desempenho do solver CFD de acordo com propriedades do fluido ou para detecção de erros.

Tabela 6. Tempo total de processamento da consulta.

Versão do *workflow*	Tempo total da consulta (minutos)		
	FLUXO_DE_ELEMENTOS	FLUXO_DE_DADOS	DADOS_DE_DESEMPENHO
Extração	0.28	4.39	3.88
Indexação Bitmap	0.51	41.89	41.04
Indexação Posicional	0.56	45.74	44.81

A Tabela 6 apresenta o tempo total para execução de cada consulta em cada versão do *workflow*, ou seja, o tempo total de acesso aos dados a partir do mecanismo de processamento proposto. De acordo com estes resultados, podemos perceber que extração de dados científicos apresenta neste caso o melhor desempenho em relação às versões do *workflow* que utilizam o componente de indexação de dados científicos. Isto ocorre pelo fato do mecanismo de consultas invocar processadores de consultas externos ao SGBD quando lidamos com as soluções de indexação de dados científicos. As chamadas ao processador de consultas externo ao SGBD são realizadas de forma sequencial, diferente das otimizações e das técnicas de paralelismo que podem ser utilizadas pelo processador de consultas do SGBD. Pela depuração do processador de consultas desenvolvido nesta dissertação, observou-se que a varredura dos arquivos de dados científicos representa aproximadamente 62% do tempo total de processamento da consulta. Ou seja, essa etapa representa um gargalo em termos de tempo de execução da consulta, além de que essa varredura poderia ser analisada e modificada para reduzir esse tempo de execução. A solução de extração de dados científicos, por sua vez, se apoia apenas no processador de consultas inerente ao SGBD, sem precisar percorrer todos os arquivos de dados científicos relevantes.

No entanto, observamos que o custo de tempo adicional para o processamento de consultas utilizando algoritmos de indexação, o qual no pior cenário equivale a 45,74 minutos, é compensado pelo custo de tempo adicional da execução dos programas de

extração de dados científicos em arquivos, geração de índices no SGBD e ingestão de dados na base de proveniência, o qual representa aproximadamente 192,84 minutos como visto nos experimentos anteriores utilizando a malha em granularidade fina. Podemos notar ainda que, entre as abordagens que apresentam implementações do componente de indexação de dados científicos, a implementação do algoritmo de indexação *bitmap* apresenta um desempenho um pouco superior no processamento de consultas.

Este experimento de processamento de consultas considera uma média de tempo de três execuções para cada uma das consultas, enquanto, geralmente, os usuários realizam estas consultas várias vezes. Considerando este cenário, existem ainda algumas oportunidades em relação ao processamento de consultas adaptativo que consideram o armazenamento de metadados ou resultados de consultas com o objetivo de melhorar o desempenhos das consultas, como apresentado por Karpathiotakis *et al.* (2014). Da mesma forma, o mecanismo de varredura de arquivos de dados científicos poderia ser modificado e técnicas de paralelismo poderiam ser empregadas para reduzir o tempo total de processamento de consultas.

Capítulo 7 Conclusão

Simulações computacionais, em sua maioria, lidam com grandes volumes de dados científicos presentes em arquivos, os quais apresentam formatos heterogêneos de acordo com o domínio. Tais dados científicos são essenciais para a análise do comportamento da execução de um *workflow* científico, bem como para os resultados parciais e finais do experimento. Entretanto, a análise de dados científicos dispersos em diversos arquivos não é uma tarefa simples, uma vez que estes dados produzidos devem ser analisados levando em consideração os relacionamentos entre os elementos de dados presentes em múltiplos arquivos de dados científicos.

Para que seja possível realizar consultas sobre estes dados, soluções baseadas na extração de dados científicos propõem o armazenamento dos mesmos em formas de repositório externo, como arquivos ou bases de dados. Todavia, a carga dos dados científicos extraídos representa um custo de tempo muito grande em relação à execução da própria simulação computacional. Devido ao grande tamanho que arquivos de dados científicos podem assumir, há ainda um alto custo adicional para adequar o conteúdo de dados científicos a estruturas de dados. De forma a evitar esta carga adicional, algumas soluções propõem a indexação de dados científicos para garantir o acesso direto ao conteúdo de arquivos de dados científicos (ALAGIANNIS *et al.*, 2012, ROMOSAN *et al.*, 2013, WU *et al.*, 2009) e outras propõem o uso de SGBDs específicos (SciDB, LUSTOSA *et al.*, 2016). Porém, tais soluções não são capazes de apoiar a gerência do fluxo de elementos de dados, logo, os usuários não têm a possibilidade de relacionar elementos de dados presentes em múltiplos arquivos de dados científicos na especificação das consultas. Nesses casos, o próprio usuário precisaria escrever programas para acessar os múltiplos arquivos e realizar os relacionamentos.

Em (SILVA *et al.*, 2016), uma primeira solução foi proposta para o problema de consultas sobre fluxos de dados, baseada no mecanismo do SGWfC Chiron e seu apoio a dados de proveniência, para acesso e consulta a arquivos de dados científicos e o fluxo de elementos de dados. No entanto, esta solução considerava apenas o processo de extração de dados científicos, o qual apresenta um custo significativo de carga de dados e não garante o acesso direto a conteúdos específicos de arquivos de dados científicos.

Baseado no formalismo da arquitetura ARMFUL e sua integração com o SGWfC Chiron, esta dissertação apresenta a implementação de um componente de

indexação de dados científicos para apoiar a análise de arquivos de dados científicos, buscando reduzir o custo da carga de dados e preservar o potencial analítico de consultas sobre os mesmos. A abordagem proposta contempla um algoritmo de indexação de dados científicos em arquivos para a geração de índices *bitmap* por meio da ferramenta FastBit. Não obstante, o componente de indexação de dados científicos implementado foi adaptado para respeitar o modelo de dados de proveniência PROV-Df de modo a garantir que a base de proveniência estivesse integrada aos dados científicos presentes em múltiplos arquivos. Para assegurar o apoio à análise exploratória de arquivos de dados científicos, o presente trabalho também apresenta um mecanismo de processamento de consultas capaz de reunir os resultados de subconsultas isoladas sobre os dados de proveniência armazenados no SGBD e sobre o conteúdo de arquivos de dados científicos.

Os experimentos realizados utilizando um *workflow* científico real de dinâmica dos fluidos com técnicas de simulação numéricas nos permitiram analisar o comportamento da abordagem proposta em comparação a outros cenários com o apoio a análise de arquivos de dados científicos. A partir da avaliação experimental, demonstramos uma redução de 67,11% no custo de ingestão de dados e 46,39% no custo de geração de índices ao utilizarmos a implementação do componente de indexação de dados científicos em relação ao método de extração de dados científicos. Ademais, os resultados experimentais desta dissertação, associados à implementação do componente de indexação *bitmap* de dados científicos presentes em arquivos, foram publicados em (SILVA *et al.*, 2017). Essa publicação evidencia a contribuição científica deste trabalho, assim como a sua aplicação em uma simulação computacional real no domínio de dinâmica de fluidos computacionais.

Além disso, apresentamos consultas em tempo de execução que permitem os usuários monitorar a convergência do resolvedor CFD e analisar dados científicos que assumem estruturas de dados mais complexas, como malhas, a partir do algoritmo de indexação proposto. Exploramos o potencial analítico que a abordagem proporciona em relação à análise de dados científicos relacionados por múltiplos arquivos, ao passo que mostramos melhorias no processamento de consultas utilizando técnicas de indexação *bitmap* e gerência do fluxo de dados. A partir destes resultados, enfatizamos a capacidade de a abordagem proposta ser adequada para simulações computacionais que

possam ser modeladas como um fluxo de dados para promover o apoio à análise de arquivos de dados científicos em tempo de execução.

Uma das limitações apresentadas pela implementação do componente de indexação de dados científicos é o fato de a abordagem considerar que todas as referências para os arquivos de dados científicos são válidas e que seus respectivos conteúdos não serão modificados após a execução do *workflow* científico (a solução permite apenas a adição de mais conteúdo). Outro fator limitante da abordagem proposta é a necessidade de conhecimento sobre o dado científico antes da execução do *workflow*, uma vez que o usuário deve especificar de antemão quais os atributos de domínio que devem ser capturados nos arquivos de dados científicos. Em certas ocasiões, é possível que alguns elementos de dados contidos em arquivos de dados científicos não sejam relevantes para as análises dos usuários, no entanto, eles serão indexados caso tenham sido especificados antes da execução.

Como trabalhos futuros, podemos identificar a necessidade de mecanismos capazes de selecionar dados de domínio de forma dinâmica para a subsequente adaptação da indexação de arquivos de dados científicos. Da mesma forma, é preciso lidar com possíveis modificações dinâmicas no conteúdo de arquivos de dados científicos ao passo que a indexação ainda seja apoiada. Como propõe a arquitetura ARMFUL, outras implementações do componente de indexação de dados científicos podem ser desenvolvidas utilizando diferentes algoritmos de indexação. Como vimos a partir dos resultados experimentais, a indexação *bitmap* apresentou vantagens em relação ao algoritmo posicional quanto ao tempo total de ingestão de dados, no entanto, se mostrou pior quanto ao tempo de indexação de dados científicos. Isto nos mostra que novas implementações podem ser desenvolvidas buscando reduzir custos em ambos os aspectos, seja a partir de outros algoritmos de indexação ou utilizando o algoritmo *bitmap* por meio de outras ferramentas ao invés do FastBit.

Em relação ao processamento de consultas, identificamos que o mecanismo de consulta proposto necessita de melhorias visando à redução do tempo de consulta. Uma possibilidade seria tornar o processo de otimização de consultas em uma solução adaptativa, ou seja, após uma primeira execução mais demorada de uma determinada consulta, as próximas análises baseadas nesta mesma consulta se tornam mais rápidas visto que o processador de consultas já possui um plano de execução otimizado. Dessa forma, o alto custo de tempo que os experimentos de análise apresentaram para

realização de consultas a partir dos índices gerados pode ser consideravelmente reduzido.

Outra melhoria diz respeito à invocação dos índices, que adota um mecanismo externo à execução de consultas do SGBD do Chiron. Tal invocação poderia ser realizada de forma paralela aproveitando os recursos computacionais distribuídos de modo a obter o resultado da consulta em um tempo reduzido. Uma outra possibilidade seria adotar uma solução híbrida por meio de um SGBD que permita a incorporação de mecanismos de indexação para arquivos externos. Assim, o índice seria acessado diretamente pelo SGBD, como proposto pelo PostgresRaw (ALAGIANNIS *et al.*, 2012), no mesmo contexto do processamento de consultas sobre os dados de sua base de dados.

Referências Bibliográficas

ALAGIANNIS, I., BOROVICA, R., BRANCO, M., *et al.*, 2012, "NoDB: efficient query execution on raw data files". In: *Proceedings of the on Management of Data*, pp. 241–252, New York, NY.

ASSUNCAO, L., GONCALVES, C., CUNHA, J. C., 2014, "Autonomic Workflow Activities: The AWARD Framework", *International Journal of Adaptive, Resilient and Autonomic Systems*, v. 5, n. 2, pp. 57–82.

BAUER, A. C., ABBASI, H., AHRENS, J., *et al.*, 2016, "In Situ Methods, Infrastructures, and Applications on High Performance Computing Platforms", *Computer Graphics Forum*

BLANAS, S., WU, K., BYNA, S., *et al.*, 2014, "Parallel data analysis directly on scientific file formats". In: *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, pp. 385–396, Snowbird.

BOWERS, S., MCPHILLIPS, T., RIDDLE, S., *et al.*, 2008, "Kepler/pPOD: Scientific Workflow and Provenance Support for Assembling the Tree of Life". In: *IPAW '08: Second International Provenance and Annotation Workshop*, pp. 70–77, Berlin, Heidelberg.

BROWN, P. G., 2010, "Overview of SciDB: large scale array storage, processing and analysis", pp. 963.

BUNEMAN, P., KHANNA, S., TAN, W. C., 2001, "Why and Where: A Characterization of Data Provenance". In: *Proceedings of the 8th International Conference on Database Theory*, pp. 316–330, London, UK.

CALLAHAN, S. P., FREIRE, J., SANTOS, E., *et al.*, 2006, "VisTrails: Visualization Meets Data Management". In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, pp. 745–747, New York, NY, USA.

CHOU, J., WU, K., PRABHAT, 2011, "FastQuery: A Parallel Indexing System for Scientific Data". In: *2011 IEEE International Conference on Cluster Computing*, pp. 455–464

CRITCHLOW, T., VAN DAM, K. K., 2013, *Data-intensive science*. Boca Raton, CRC Press, Taylor & Francis Group.

DEELMAN, E., GANNON, D., SHIELDS, M., *et al.*, 2009, "Workflows and e-Science: An Overview of Workflow System Features and Capabilities", *Future Generation Computer Systems*, v. 25, n. 5, pp. 528–540.

DEELMAN, E., VAHI, K., JUVE, G., *et al.*, 2015, "Pegasus, a Workflow Management System for Science Automation", *Future Generation Computer Systems*, v. 46, n. C (May.), pp. 17–35.

DIAS, J., GUERRA, G., ROCHINHA, F., *et al.*, 2015, "Data-centric iteration in dynamic workflows", *Future Generation Computer Systems*, v. 46 (May.), pp. 114–126.

DONG, B., BYNA, S., WU, K., 2013, "SDS: A Framework for Scientific Data Services". In: *Proceedings of the 8th Parallel Data Storage Workshop*, pp. 27–32, New York, NY, USA.

ELIAS, R., COUTINHO, A., 2007, "Stabilized edge-based finite element simulation of free-surface flows", *International Journal of Numerical Methods in Fluids*, v. 54, pp. 965–993.

GONÇALVES, B., PORTO, F., 2014, "Gamma-DB: Managing Scientific Hypotheses As Uncertain Data", *Proc. VLDB Endow.*, v. 7, n. 11 (Jul.), pp. 959–962.

HANISCH, R. J., FARRIS, A., GREISEN, E. W., *et al.*, 2001, "Definition of the Flexible Image Transport System (FITS)", *Astronomy and Astrophysics*, v. 376, n. 1 (Sep.), pp. 359–380.

IKEDA, R., DAS SARMA, A., WIDOM, J., 2013, "Logical provenance in data-oriented workflows?". In: *Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 877–888

IKEDA, R., WIDOM, J., 2010, "Panda: A System for Provenance and Data". In: *IEEE Data Engineering Bulletin*, pp. 42–49

JACOB, J. C., KATZ, D. S., BERRIMAN, G. B., *et al.*, 2009, "Montage: a Grid Portal and Software Toolkit for Science-Grade Astronomical Image Mosaicking", *International Journal of Computational Science and Engineering (IJCSE)*, v. 4, n. 2, pp. 73–87.

JENKINS, J., ARKATKAR, I., LAKSHMINARASIMHAN, S., *et al.*, 2013, "ALACRITY: Analytics-Driven Lossless Data Compression for Rapid In-Situ Indexing, Storing, and Querying", In: HAMEURLAIN, A., KÜNG, J., WAGNER, R., *et al.* [eds.] (eds), *Transactions on Large-Scale Data- and Knowledge-Centered Systems X: Special Issue on Database- and Expert-*

Systems Applications, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 95–114.

KARPATHIOTAKIS, M., BRANCO, M., ALAGIANNIS, I., *et al.*, 2014, "Adaptive Query Processing on RAW Data", *Proceedings of the VLDB Endowment*, v. 7, n. 12, pp. 1119–1130.

KIM, J., ABBASI, H., CHACON, L., *et al.*, 2011, "Parallel in situ indexing for data-intensive computing". In: *Proceedings of the 11th on Large Data Analysis and Visualization (LDAV)*, pp. 65–72

LAKSHMINARASIMHAN, S., BOYUKA, D. A., PENDSE, S. V., *et al.*, 2013, "Scalable in Situ Scientific Data Encoding for Analytical Query Processing". In: *Proceedings of the 22Nd International Symposium on High-performance Parallel and Distributed Computing*, pp. 1–12, New York, NY, USA.

LASLUISA, S., ZHANG, F., JIN, T., *et al.*, 2015, "In-situ feature-based objects tracking for data-intensive scientific and enterprise analytics workflows", *Cluster Computing*, v. 18, n. 1 (Mar.), pp. 29–40.

LO, D. C., MURUGESAN, K., YOUNG, D. L., 2005, "Numerical solution of three-dimensional velocity-vorticity Navier-Stokes equations by finite difference method", *International Journal for Numerical Methods in Fluids*, v. 47, pp. 1469–1487.

LOFSTEAD, J. F., KLASKY, S., SCHWAN, K., *et al.*, 2008, "Flexible IO and integration for scientific codes through the adaptable IO system (ADIOS)". In: *Proceedings of the 6th International Workshop on Challenges of Large Applications in Distributed Environments (CLADE '08)*, pp. 15–24, New York, NY, USA.

LUSTOSA, H., PORTO, F., VALDURIEZ, P., *et al.*, 2016, "Database System Support of Simulation Data", *Proc. VLDB Endow.*, v. 9, n. 13 (Sep.), pp. 1329–1340.

MA, B., SHOSHANI, A., SIM, A., *et al.*, 2012, "Efficient Attribute-Based Data Access in Astronomy Analysis". In: *Proceedings of the SC Companion: High Performance Computing, Networking Storage and Analysis (SCC)*, pp. 562–571, Salt Lake City, UT, USA.

MATTOSO, M., OCAÑA, K., HORTA, F., *et al.*, 2013, "User-steering of HPC Workflows: State-of-the-art and Future Directions". In: *Proceedings of the 2nd ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, pp. 4:1–4:6, New York, NY, USA.

MATTOSO, M., WERNER, C., TRAVASSOS, G. H., *et al.*, 2010, "Towards supporting the life cycle of large scale scientific experiments.", *International Journal of Business Process Integration and Management*, v. 5, n. 1, pp. 79–92.

MOREAU, L., GROTH, P. T., 2013, *Provenance: An Introduction to PROV*. Morgan & Claypool Publishers.

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

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

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

OLIVEIRA, D. DE, COSTA, F., SOUSA, V. S., *et al.*, 2014, "Debugging Scientific Workflows with Provenance: Achievements and Lessons Learned". In: *Proceedings of the XXIX Simpósio Brasileiro de Banco de Dados (SBBD 2014)*, pp. 67–76, Curitiba, Paraná, Brasil.

ÖZSU, T. M., VALDURIEZ, P., 2011, *Principles of Distributed Database Systems*. 3rd ed. New York, Springer.

PARAVIEW *ParaView: open-source, multi-platform data analysis and visualization application*., 2011 Disponível em: http://www.paraview.org.

PIDD, M., 1994, "An introduction to computer simulation". In: *Proceedings of the 26th conference on Winter simulation*, pp. 7–14, Orlando, Florida, United States.

PORTO, F., MOURA, A. M. DE C., GONÇALVES, B., *et al.*, 2012, "A scientific hypothesis conceptual model". In: *International Conference on Conceptual Modeling*, pp. 101–110

RAICU, I., FOSTER, I. T., YONG ZHAO, 2008, "Many-task computing for grids and supercomputers". In: *Proceedings of the Workshop on Many-Task Computing on Grids and Supercomputers*, pp. 1–11, Austin, Texas, USA.

ROMOSAN, A., SHOSHANI, A., WU, K., *et al.*, 2013, "Accelerating gene context analysis using bitmaps". In: *Proceedings of the 25th International Conference*

on Scientific and Statistical Database Management (SSDBM), pp. 26, New York, USA.

RUDI, J., MALOSSI, A. C. I., ISAAC, T., *et al.*, 2015, "An Extreme-scale Implicit Solver for Complex PDEs: Highly Heterogeneous Flow in Earth's Mantle". In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 5:1–5:12, New York, NY, USA.

SILVA, V., DE OLIVEIRA, D., VALDURIEZ, P., *et al.*, 2016, "Analyzing related raw data files through dataflows", *Concurrency and Computation: Practice and Experience*, v. 28, n. 8, pp. 2528–2545.

SILVA, V., LEITE, J., CAMATA, J. J., *et al.*, 2017, "Raw data queries during data-intensive parallel workflow execution", *Future Generation Computer Systems*

SILVA, V., OLIVEIRA, D., MATTOSO, M., 2014, "Exploratory analysis of raw data files through dataflows". In: *Proceedings of the Workshop on Parallel and Distributed Computing for Big Data Applications (WPBA 2014)*, pp. 114–119, Paris, France.

SOUZA, R., SILVA, V., NEVES, L., *et al.*, 2015, "Monitoramento de Desempenho usando Dados de Proveniência e de Domínio durante a Execução de Aplicações Científicas". In: *XIV Workshop em Desempenho de Sistemas Computacionais e de Comunicação*, Recife, Brazil.

TIAN, Y., ALAGIANNIS, I., LIAROU, E., *et al.*, 2014, "DiNoDB: Efficient Large-Scale Raw Data Analytics". In: *Proceedings of the First International Workshop on Bringing the Value of "Big Data" to Users (Data4U 2014)*, pp. 1:1–1:6, New York, NY, USA.

WANG, Y., SU, Y., AGRAWAL, G., 2013, "Supporting a Light-Weight Data Management Layer over HDF5". In: *Proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)*, pp. 335–342, Delft, Netherlands.

WOZNIAK, J. M., ARMSTRONG, T. G., MAHESHWARI, K., *et al.*, 2012, "Turbine: A Distributed-memory Dataflow Engine for Extreme-scale Many-task Applications". In: *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies*, pp. 5:1–5:12, New York, NY, USA.

WU, K., AHERN, S., BETHEL, E. W., *et al.*, 2009, "FastBit: interactively searching massive data", *Journal of Physics: Conference Series*, v. 180 (Jul.), pp. 12–53.

WU, K., OTOO, E. J., SHOSHANI, A., 2006, "Optimizing Bitmap Indices with Efficient Compression", *ACM Transactions Database Systems (TODS)*, v. 31, n. 1, pp. 1–38.

ZHAO, Y., RAICU, I., FOSTER, I. T., *et al.*, 2008, "Realizing Fast, Scalable and Reliable Scientific Computations in Grid Environments", *Grid Computing Research Progress*, v. abs/0808.3548, pp. 341.