Universidade Federal
do Rio de Janeiro
Escola Politécnica

PROV-VIS: VISUALIZAÇÃO DE DADOS DE EXPERIMENTOS EM LARGA ESCALA POR MEIO DE PROVENIÊNCIA

Felipe Figueira Horta

Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro.

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

Rio de Janeiro
Março de 2013

PROV-VIS: VISUALIZAÇÃO DE DADOS DE EXPERIMENTOS EM LARGA ESCALA POR MEIO DE PROVENIÊNCIA

Felipe Figueira Horta

PROJETO DE GRADUAÇÃO SUBMETIDO AO CORPO DOCENTE DO CURSO DE ENGENHARIA DE COMPUTAÇÃO E INFORMAÇÃO DA ESCOLA POLITÉCNICA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO DE COMPUTAÇÃO E INFORMAÇÃO.

Examinada por:

_____

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

_____

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

_____

Prof. Alvaro Luiz Gayoso de Azeredo Coutinho, D. Sc.

_____

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

RIO DE JANEIRO, RJ – BRASIL

MARÇO de 2013

Horta, Felipe Figueira

Prov-Vis: Visualização de Dados de Experimentos em Larga Escala Por Meio de Proveniência / Felipe Figueira Horta. – Rio de Janeiro: UFRJ/ Escola Politécnica, 2013.

VIII, 60 p.: il.; 29,7 cm.

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

Projeto de Graduação – UFRJ/ Escola Politécnica/ Curso de Engenharia de Computação e Informação, 2013.

Referências Bibliográficas: p. 49-51.

1.Arquitetura 2. *Workflows* Científicos 3. Linhas de Experimento 4. Visualização Científica I. Mattoso, Marta Lima de Queirós *et al.* II. Universidade Federal do Rio de Janeiro, Escola Politécnica, Curso de Engenharia de Computação e Informação. III. Título.

À minha mãe Elza Cristina, ao meu pai Roberto Horta,
ao meu irmão Rafael e aos meus amigos,
por tudo que representam na minha vida;

AGRADECIMENTOS

À professora Marta Mattoso, minha orientadora, pelo acompanhamento acadêmico e pela grande oportunidade de trabalhar em sua equipe;

Ao professor Daniel de Oliveira, meu orientador, pelas valiosas críticas, incontáveis revisões, e acima de tudo pela paciência;

À professora Kary Ocanã, pela atenção e grande apoio na área de bioinformática, e suporte com seus casos de estudos;

Aos alunos Jonas Dias e Vitor Sousa, pela amizade, atenção e colaboração, fundamentais para a realização deste projeto;

Ao professor Alvaro, pela motivação contagiante, e à toda equipe do NACAD – Mara, Sandra e todos os outros funcionários e alunos – por garantirem o apoio necessário para conclusão deste trabalho;

À Isabel, pelo carinho, paciência e atenção nas horas mais importantes;

Agradeço.

Resumo do Projeto de Graduação apresentação à Escola Politécnica/ UFRJ como parte dos requisitos necessários para a obtenção do grau de Engenheiro de Computação e Informação.

Prov-Vis: Visualização de Dados de Experimentos em Larga Escala
Por meio de Proveniência

Felipe Figueira Horta

Março/2013

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

Curso: Engenharia de Computação e Informação

Experimentos científicos em larga escala são muitas vezes organizados como uma composição de diversas tarefas computacionais ligadas por meio de fluxo de atividades. A esse fluxo de atividades damos o nome de *workflow* científico. Os dados que fluem ao longo do *workflow* muitas vezes são transferidos de um computador *desktop* para um ambiente de alto desempenho, como um *cluster,* e em seguida para um ambiente de visualização. Manter o controle do fluxo de dados é um desafio para o apoio à proveniência em Sistemas de Gerenciamento de *workflows* Científicos (SGWfC) de alto desempenho. Após a conclusão de um experimento científico, muitas vezes um cientista deve selecionar manualmente e analisar seus dados, por exemplo, verificando as entradas e saídas ao longo de diversas atividades computacionais que fazem parte do seu experimento. Neste projeto, o objetivo é propor um sistema de gerência dos dados de proveniência que descreva as relações de produção e consumo entre artefatos, tais como arquivos, e as tarefas computacionais que compõem o experimento. O projeto propõe uma interface de consulta que permita ao cientista procurar dados de proveniência em um ambiente de alto desempenho e selecionar a saída que deseja visualizar usando seu próprio navegador ou um ambiente de visualização remota.

Abstract of Undergraduate Project presented to POLI/UFRJ as a partial fulfillment of the requirements for the degree of Computer and Information Engineer.

Prov-Vis: Data Visualization of Large-Scale Experiments Through Provenance

Felipe Figueira Horta

Março/2013

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

Major: Computer and Information Engineering

Large-scale scientific computations are often organized as a composition of many computational tasks linked through data flow. The data that flows along this many- task computing often moves from a desktop to a high- performance environment and to a visualization environment. Keeping track of this data flow is a challenge to provenance support in high-performance Scientific workflow Management Systems. After the completion of a computational scientific experiment, a scientist has to manually select and analyze its staged-out data, for instance, by checking inputs and outputs along computational tasks that were part of the experiment. In this project, we present a provenance management system that describes the production and consumption relationships between data artifacts, such as files, and the computational tasks that compose the experiment. We propose a query interface that allows for scientists to browse provenance data and select the output they want to visualize using browsers or a high-resolution tiled wall display.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE SIGLAS

AJAX – Asynchronous Javascript And XML

API – Application Programming Interface

HTML – HyperText Markup Language

NACAD – Núcleo Avançado em Computação de Alto Desempenho

NFS – Network File System

PAD – Processamento de Alto Desempenho

PC – Personal Computer

SOAP – Simple Object Access Protocol

SSH – Secure Shell

SGWfC - Sistema de Gerenciamento de workflows Científicos

UDDI – Universal Description, Discovery and Integration

UFRJ – Universidade Federal do Rio de Janeiro

UML – Unified Modeling Language

WSDL – Web Services Description Language

XML – eXtensible Markup Language

ZUML – ZK User Interface Markup Language

1. Introdução

1.1. Tema

Este documento descreve a concepção e a implementação do projeto Prov-Vis, uma ferramenta de apoio à visualização remota de dados de experimentos científicos em larga escala. Com o intuito de trazer praticidade e usabilidade à cenários arquiteturais complexos na gerência e manutenção do ciclo de vida destes experimentos que envolvem um grande número de máquinas e ambientes heterogêneos, o projeto é construído sob uma arquitetura baseada em serviços, utilizando, como interface, um aplicativo *Web* dinâmico que pode ser acoplado de forma transparente em um ambiente de visualização. O projeto também visa a integração de outras ferramentas de código aberto bem difundidas na comunidade, buscando maior aceitação e familiaridade com ambientes já conhecidos.

1.2. Delimitação

O projeto é direcionado a centros de pesquisa universitários e empresariais que demandem o gerenciamento e visualização de experimentos científicos de larga escala em ambientes de alto desempenho. Estes experimentos podem servir para análises científicas de qualquer natureza, tais como, bioinformática (Ocaña *et al*., 2011a, 2012), dinâmica dos fluidos computacional (CFD) (Guerra *et al*., 2009), análises térmicas, volumétricas e barométricas, previsões meteorológicas e financeiras, dentre outras possíveis.

Neste primeiro momento, todo o esforço de desenvolvimento está direcionado ao Núcleo Avançado em Computação de Alto Desempenho (NACAD), laboratório da COPPE/UFRJ, devido à demanda por visualização de experimentos em larga escala

realizados no laboratório. Além disso, o NACAD já possui os equipamentos voltados à computação de auto desempenho e o equipamento de visualização científica.

Por ser um projeto de código aberto hospedado em espaço público na Internet[1], sua extensão para implantação em outros cenários é desejável e incentivada, proporcionando maior divulgação e portabilidade para a aplicação.

1.3. Justificativa

Experimentos científicos de larga escala geralmente demandam Processamento de Alto Desempenho (PAD) (Deelman *et al.*, 2009). A gerência destes experimentos pode ser muito complexa por envolver o uso de diferentes programas de computadores e diferentes tipos de *scripts*, sempre lidando com dados muito heterogêneos e alta granularidade. Visualizar o resultado final também não é uma tarefa trivial (Vo *et al.*, 2011), ao passo que a maioria destes experimentos gera uma grande quantidade de dados e muitas informações destinadas à proveniência (Freire *et al.*, 2008, Mattoso *et al.*, 2010). Estes dados podem estar distribuídos em diferentes computadores, em diferentes arquiteturas. Por outro lado, parte dos resultados destes experimentos também podem não ser úteis. Entretanto, para analisar os resultados, os cientistas precisam visualizar estes dados e horas exaustivas de transferência podem ser gastas desnecessariamente.

Como caso de uso para este projeto foi decidido explorar experimentos da área de bioinformática, *e.g.* o *workflow* de filogenética *SciPhy* (Ocaña *et al*., 2011b). Com o SciPhy é possível otimizar o gerenciamento do experimento, provendo uniformidade para os dados relacionados à área de filogenética, enquanto se constrói uma base de dados de proveniência. A proveniência é a chave para análise dos dados de *workflows*

[1] https://bitbucket.org/fhorta/prov-vis

científicos (Seção 2.2). Por meio da proveniência, é possível realizar o rastreamento dos dados do experimento em execução, além de realizar consultas de alto nível em domínios específicos, auxiliando cientistas a encontrarem respostas como: Quais sequências levam às melhores árvores filogenéticas? Quais resultados apresentam *e-value* superior à 1e-10? Quais parâmetros de entrada geram um determinado resultado? Estes tipos de consultas são possíveis graças ao armazenamento de informações de determinado domínio dentro da base de dados da proveniência. Os metadados dos experimentos aliados com resultados experimentais estratégicos podem ser uma poderosa associação. O que nos leva às seguintes perguntas: Hoje em dia, é possível visualizar os resultados de um experimento de larga-escala enriquecido pelas informações da proveniência? Quais são as barreiras presentes na visualização de um experimento? Como aproveitar este tipo de resultados sem a perda de informações?

Atualmente existem diferentes tipos de Sistemas de Gerência de *workflows* Científicos (SGWfC). Alguns deles oferecem apoio a visualização. O VisTrails (Callahan *et al.*, 2006a) é capaz de se integrar à bibliotecas ou programas dedicados à visualização, como Paraview (ParaView, 2011) ou Maya[2], permitindo a definição de atividades de visualização do *workflow* exiba dados em ambientes de visualização. No entanto, infelizmente, não há apoio para PAD integrado para visualizar dados diretamente de *clusters*, grades ou nuvens.

Entretanto, existem outros SGWfC que oferecem apoio a PAD, como Swift / Turbine (Wozniak *et al.*, 2012) ou o Pegasus (Deelman *et al.*, 2007). Infelizmente, estes SGWfC apresentam uma fraca integração entre os dados da proveniência e os resultados do experimento. Até onde foi pesquisado, não há uma solução que integre apoio de execução ao PAD com as necessidades de visualização enriquecendo os resultados com

[2] http://www.vistrails.com/maya.html

dados de proveniência. Sendo assim, o objetivo do Prov-Vis é explorar esta característica, propondo uma nova abordagem de visualização para experimentos de larga escala.

1.4. Objetivo

O objetivo deste projeto é fundamentar o desenvolvimento de um sistema de apoio a visualização remota de experimentos científicos de larga escala, em tempo real, proposto em (Horta *et al.*, 2012), com bons atributos de usabilidade facilitando o processo de consulta à base de proveniência e acoplamento dos ambientes de execução, armazenamento e visualização. Busca-se construir a ferramenta partindo das funcionalidades básicas, para, adiante, manter o projeto de código aberto evoluindo de acordo com as necessidades dos cientistas.

Neste primeiro momento, o foco do projeto é a modelagem e documentação partindo para a elaboração de uma arquitetura robusta sob a qual seja possível estender a ferramenta futuramente.

1.5. Metodologia

O projeto teve como referência, e parte de sua implementação, trabalhos e ferramentas já realizados, publicados, e amplamente difundidos na comunidade científica. Diante disso, e após a leitura e aprofundamento do material didático encontrado e recomendado por especialistas, iniciou-se uma fase de análise de requisitos. Estes seriam levantados a fim de modelar a primeira versão da arquitetura do projeto.

Sendo assim, desde a sua concepção, o projeto foi dividido em dois elementos fundamentais: a interface *Web* dinâmica e a interface de acesso aos serviços do *cluster*

de visualização. Tendo esta compreensão, foi necessário pesquisar, separadamente, qual solução melhor atenderia cada um desses aspectos. A partir deste ponto, protótipos foram implementados à fim de testar tecnologias candidatas. Ao final de cada iteração foram realizadas baterias de testes e validações com os requisitos levantados.

Ao final da implementação da solução proposta, o passo seguinte é testar seu desempenho e avaliar suas funcionalidades frente aos benefícios propostos.

1.6. Organização do Texto

No capítulo seguinte iremos discutir conceitos fundamentais envolvidos nesse projeto, elencando os principais componentes, com base científica, presentes na concepção do Prov-Vis. Faremos uma abordagem sobre as principais características de um experimento científico, sua proveniência, os ambientes de processamento de alto desempenho (PAD), e os motores de execução de *workflows* científicos.

O capítulo 3 abordará o tema Visualização Científica, seus trabalhos já existentes na comunidade científica e os potenciais requisitos que podemos levantar – gerados à partir necessidades identificadas no processo de recuperação e análise dos resultados.

O projeto como um todo é detalhado no capítulo 4, onde é explicado os requisitos não funcionais e funcionais de cada módulo até os diagramas realizados para modelar o sistema desenvolvido. Ainda neste capítulo, discutimos as tecnologias envolvidas no projeto, explicando como foram feitas as escolhas de cada uma delas e os motivos envolvidos nestas. Descreve-se o ambiente onde foi realizado o desenvolvimento do projeto e a solução utilizada para integração com o painel de visualização (*tiled-wall)*. Também são explicadas a tecnologia AJAX e Serviços *Web*, descrevendo o *framework* para o desenvolvimento *Web* e os servidores utilizados para

implantação do projeto. Veremos as arquiteturas possíveis para se utilizar o projeto. São discutidas desde a forma mais simples de implantação do projeto até o modo mais complexo, permitindo a administração de inúmeros *displays*, em um único ambiente de visualização, através de uma única interface *Web*.

Por fim, no capítulo 5, são apresentados os resultados alcançados e é feita a conclusão a cerca do projeto desenvolvido, do que foi aprendido e solucionado. Em seguida, abordamos a perspectiva de novas funcionalidades idealizadas no decorrer do desenvolvimento, e a possibilidade de integração com trabalhos correlatos.

2. Fundamentação Teórica

Neste capítulo, será feita a introdução dos principais conceitos envolvidos na discussão do tema proposto neste projeto. Para isso, vamos definir alguns conceitos importantes vinculados à um experimento científico; como o seu ciclo de vida; *workflows* científicos e o seu papel no ciclo de vida; e o apoio e a definição de Sistemas de Gerência de *workflows* Científicos (SGWfC). Em seguida, abordaremos o tema Proveniência de Dados, palavra-chave em *workflows* científicos. Finalizando a fundamentação, apresentaremos o principal motor de execução de *workflows*, presente neste projeto.

2.1. Experimento Científico

Uma experiência científica consiste na montagem de uma estratégia concreta a partir da qual se organizam diversas ações observáveis, de forma direta ou indireta, com o intuito de provar a plausibilidade ou falsidade de uma dada hipótese ou de forma a estabelecer relações de causa/efeito entre fenómenos. A experiência científica é uma das abordagens fundamentais necessárias, empirista à ampliação do conhecimento humano ("Wikipédia"). Devido a importância deste conceito, nesta seção iremos aborda-lo, definindo sub-conceitos importantes aplicados neste projeto.

2.1.1. Ciclo de Vida de um Experimento Científico

Um conceito de um possível ciclo de vida de um experimento científico foi proposto em (Mattoso *et al.*, 2009) devido a necessidade de organizar, processar, controlar e analisar os experimentos para atender aos modelos propostos pelos cientistas. A partir deste modelo é possível definir os experimentos, as própria

execução, e assim avaliar se os resultados obtidos estão de acordo com suas próprias definições. Em caso negativo, os experimentos podem ser alterados e o ciclo de vida é realimentado. Tais etapas são conhecidas como ciclo de vida de um experimento científico, onde este é dividido em três fases: composição, execução e análise.

Como se pode ver no modelo do ciclo de vida de um experimento apresentado na Figura 1, a fase de concepção consiste na composição e configuração dos programas, que visam atender ao experimento científico. Nessa fase, podem-se criar novas versões desses programas ou reutilizar outros programas presentes em experimentos anteriores.

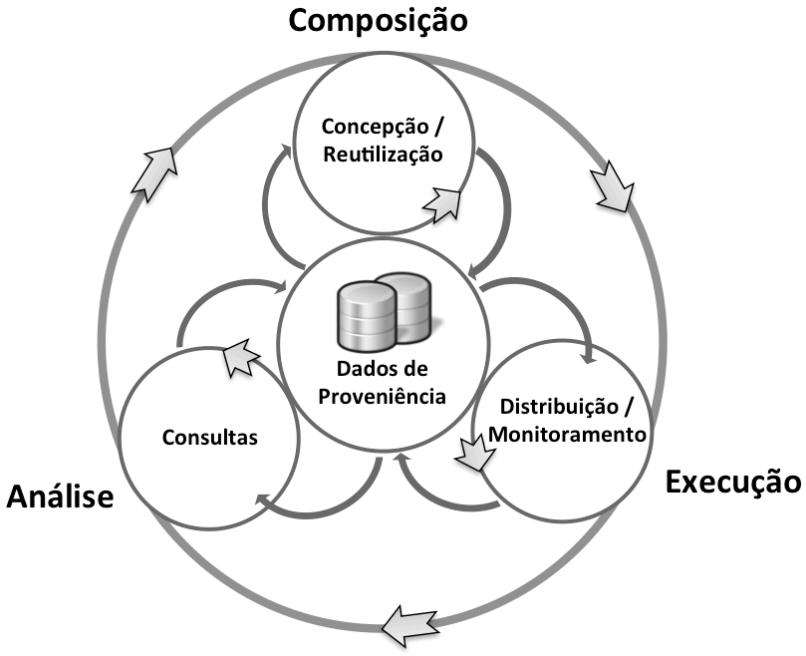

Figura 1 - Ciclo de vida do experimento científico - Adaptado de (Mattoso *et al.*, 2009).

A fase de execução é definida pelo processamento e acompanhamento dos *workflows* desenvolvidos na etapa anterior. Caso haja disponibilidade, a proposta de soluções que utilizem alto desempenho computacional e computação paralela é uma boa alternativa para a otimização do tempo de execução do experimento, uma vez que estes necessitam de alto recurso computacional e manipulação de grande volume de dados.

Completando o ciclo de vida do experimento, a fase de análise é responsável pela avaliação dos resultados obtidos pela execução dos *workflows* científicos. Desta forma, com o objetivo de permitir uma avaliação total do experimento, que pode conter informações tanto da fase de composição quanto da fase de execução, se faz uso de um mecanismo para fazer esta análise, nomeado proveniência de dados. A seção 2.2 descreve com mais detalhes os mecanismos e conceitos relacionados à proveniência de dados.

2.1.2. *Workflow* Científico

Geralmente, os experimentos científicos necessitam da execução de uma grande quantidade de simulações computacionais com a finalidade de avaliar seus modelos ou proposições. Os *workflows* científicos são qualificados por fornecer a abstração que permite a especificação desses experimentos de maneira estruturada, por meio de um fluxo de programas, serviços e dados para produzir um resultado final (Deelman *et al.*, 2009). Este resultado tem o objetivo de atender as necessidades do experimento em questão.

Podemos traçar uma relação no fluxo de programas, seus serviços e dados, com os componentes que fazem parte de um *workflow* científico. Sendo os seguintes: atividade, porta, relacionamento e *sub-workflow*. Nesta relação, a atividade é responsável por consumir e produzir dados, sendo assim caracterizada pela execução de um programa ou serviço; as portas especificam quais dados devem ser consumidos e produzidos em cada atividade. Além disso, cada definição de porta contém a estrutura de dados utilizada, como por exemplo, uma porta do tipo inteiro ou uma sequência de caracteres.

Outra característica importante na especificação das portas de uma atividade consiste em avaliar se uma determinada porta comporta-se apenas para o consumo de dados (porta de entrada), apenas para a produção de dados (porta de saída) ou tanto para o consumo, como para a produção de dados (porta de entrada e saída). Para as portas de entrada e de saída, o dado de entrada será o mesmo que o dado produzido pela atividade em questão (Sousa, 2011).

Um relacionamento é o componente que permite o encadeamento lógico das atividades, definindo o fluxo de dados ou de controle entre estas. Desta forma, ao especificar um relacionamento de um *workflow* científico, se faz necessário estabelecer o fluxo dos dados. Em seguida, definem-se as portas de origem e de destino do relacionamento; ao mesmo tempo em que as portas são estabelecidas, as atividades também são, apresentando qual atividade produz um dado e, qual atividade o consumirá. Por fim, a porta de origem é atribuída à porta de saída de uma atividade, enquanto que a porta de destino é atribuída a uma porta de entrada de outra atividade.

A Figura 2 expõe um esquema geral de um *workflow* científico, em que é possível observar a existência de duas atividades (atividades *A* e *B*). Podemos reparar que a atividade *A* apresenta três portas de entrada (portas *X*, *Y* e *Z*) e duas portas de saídas (portas *a* e *b*). Enquanto isso, a atividade *B* apresenta duas portas de entrada (portas *W* e *d*) e duas portas de saída (portas *c* e *d*). É importante ressaltar que o nome igual para as duas portas (porta *d*), tanto para a entrada como para saída, enfatiza a presença de uma porta do tipo entrada e saída. Além das atividades e das portas, definimos os relacionamentos presentes entre as atividades *A* e *B* (relacionamentos *rel1* e *rel2*). Para cada relacionamento existe a especificação da atividade de origem, porta de origem, atividade de destino e porta de destino. No caso do relacionamento *rel1*, a porta *a* é a porta de origem e a porta *W* é a porta de destino. Dessa forma, para o

relacionamento *rel1*, as atividades de origem e de destino são, respectivamente, *A* e *B*. Assim como ocorre para o relacionamento *rel1*, o relacionamento *rel2* apresenta as mesmas atividades de origem e de destino. Contudo as portas de origem e de destino são diferentes, sendo, respectivamente, as portas *b e d*. (Sousa, 2011)

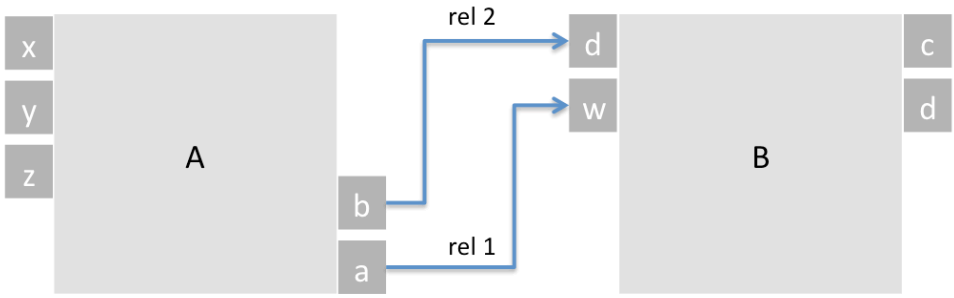

Figura 2 - Esquema geral de um *workflow* científico. Adaptado de (Sousa, 2011).

2.1.3. Sistemas de Gerência de *Workflows* Científicos

A fim de permitir a composição, execução e monitoramento de *workflows* científicos, foram desenvolvidos os Sistemas de Gerência de *workflows* Científicos (SGWfC). A partir destes sistemas é possível configurar a execução dos programas e serviços, o condicionamento dos dados existentes no *workflow*, de acordo com o experimento que se deseja representar.

Hoje em dia, existem diversos SGWfC diferentes, como o Kepler (Altintas *et al.*, 2004), o VisTrails (Callahan *et al.*, 2006b), o Taverna (Oinn *et al.*, 2004), o Swift / Turbine (Wozniak *et al.*, 2012) ou o Pegasus (Deelman *et al.*, 2007). Cada uma destes apresenta suas próprias características e comportamentos (seção 3.1).

No capítulo 3 será feita uma análise mais detalhada, alinhada ao tema do projeto Prov-Vis, sobre do comportamento de trabalhos correlatos quanto à funcionalidade de visualização do experimento. Desta forma, será possível ver com mais detalhes como

estes sistemas se comportam, suas vantagens e desvantagens, e as necessidades que ainda não cumprem, atingindo as propostas e requisitos deste projeto.

2.2. Proveniência de Dados

O termo "proveniência" pode ser definido de diversas formas. No dicionário Aurélio (Buarque de Holanda, 2004) o termo proveniência é definido como "*s.f.* Origem, procedência: mercadorias de proveniência estrangeira; uma pessoa de proveniência ignorada" (Freire *et al*., 2008, Oliveira, 2012) .

De forma geral, o termo, para a aplicação em questão, se designa à origem ou procedência de algum evento. Desta forma, a funcionalidade do mecanismo de proveniência de dados, usado para registrar a origem e o histórico de informações de um experimento científico, fica mais clara.

Com o passar do tempo, este tema foi sendo amplamente estudado, devido a sua importância na rastreabilidade da informação de experimentos em gerais. Quando, em (Freire *et al.*, 2008), foram propostas duas formas de se caracterizar a proveniência, sendo estas: a proveniência prospectiva, interessada em capturar e armazenar os dados relativos à estrutura do processo que levou à geração de um determinado produto. Ou seja, no contexto de *workflows* científicos, a proveniência prospectiva está preocupada em capturar os dados relativos à estrutura do *workflow* bem como as configurações de ambiente utilizadas para executá-lo; e a proveniência retrospectiva, com foco em capturar os dados e seus descritores produzidos a partir da execução de um determinado processo, no nosso caso, de um determinado *workflow*. Dados de proveniência retrospectiva englobam tempos de início e fim de execução, arquivos produzidos, erros que ocorreram, informações de desempenho de atividades, entre outros (Oliveira, 2012).

Os dados de proveniência podem ser capturados de diferentes formas, a partir de diversas ferramentas. Para garantir a interoperabilidade de aplicações que consultem proveniências construídas por diferentes agentes, é de preocupação da comunidade científica a padronização do modelo de dados destes descritores de proveniência oriundos de ambientes heterogêneos, independentemente da tecnologia e dos SGWfC utilizados. Modelos de dados bem difundidos e recomendados na comunidade são propostos, abordando diferentes aspectos (não abordados neste projeto), para suprir estas necessidades, projetos como o *Open Provenance Model* [3](OPM) e PROV[4].

2.3. Ambientes para Processamento de Alto Desempenho

A fim de serem executados em um tempo hábil, os *workflows* científicos necessitam de ambientes de Processamento de Alto Desempenho (PAD). Contudo, sabemos que existem diversos tipos de ambiente de PAD que podem ser utilizados para esta finalidade, cada qual com vantagens e desvantagens associadas. Dentre estes ambientes, podemos listar: os aglomerados de computadores (*clusters*); as grades de computadores (*grids*); ambientes de computação voluntárias; e recentemente o conceito de computação em nuvem (do inglês *cloud computing*).

Como fundamento para a abordagem do projeto Prov-Vis, mais atenção será dada para ambientes como *clusters* e nuvens. Ambos ambientes possuem ferramentas bem difundidas na comunidade e fazem parte da engrenagem principal deste projeto.

Os *clusters*, ou aglomerado de computadores, são formados por um conjunto de computadores que utilizam um tipo especial de sistema operacional classificado como um sistema distribuído (Figura 3). Muitas vezes, estes aglomerados são construídos a

[3] http://openprovenance.org/
[4] http://www.w3.org/TR/prov-primer/

partir de computadores convencionais, como PCs (*personal computers*), os quais são ligados em rede e comunicam-se através do sistema, enlaçados por um *switch*, trabalhando como se fossem uma única máquina de grande porte. Sendo assim, estas máquinas possuem um alto poder de processamento e armazenamento.

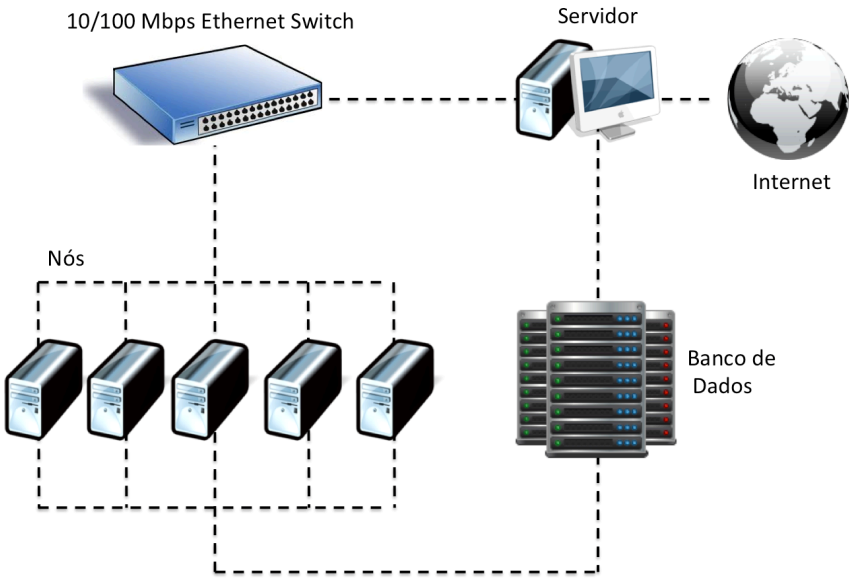

Figura 3 - Um tipo famoso é o *cluster* da classe *Beowulf*, constituído por diversos nós escravos gerenciados por um só computador.

Uma característica importante dos *clusters* é a homogeneidade de sua estrutura e a utilização de redes de alto desempenho com baixa latência (*e.g. infiniband*). As execuções de um *workflow* em ambientes de *cluster* requerem a utilização de um escalonador (Bayucan *et al.*, 2000, Staples, 2006, Thain *et al.*, 2002) para que se consiga tirar alguma vantagem da estabilidade e das características de homogeneidade dos *clusters* (Oliveira, 2012).

Já os ambientes de computação em nuvem (Figura 4), são caracterizados pela diversidade dos recursos e pelo acesso através de uma camada de virtualização (*i.e.* máquinas virtuais). Sendo assim, refere-se à utilização da memória e das capacidades de armazenamento e cálculo de computadores e servidores compartilhados e interligados

por meio da Internet, seguindo o princípio da computação em grade. Neste caso, sua grande vantagem é que o usuário (no contexto deste projeto, o cientista) tem um ambiente de alta capacidade de processamento. Porém, diferentemente dos *clusters* e *grids* onde é necessário um investimento inicial grande, seja monetário ou de configuração e manutenção, desta vez o cientista possui o controle absoluto das ações, elasticidade dos recursos e, além disso, só necessita pagar pelo que é efetivamente usado.

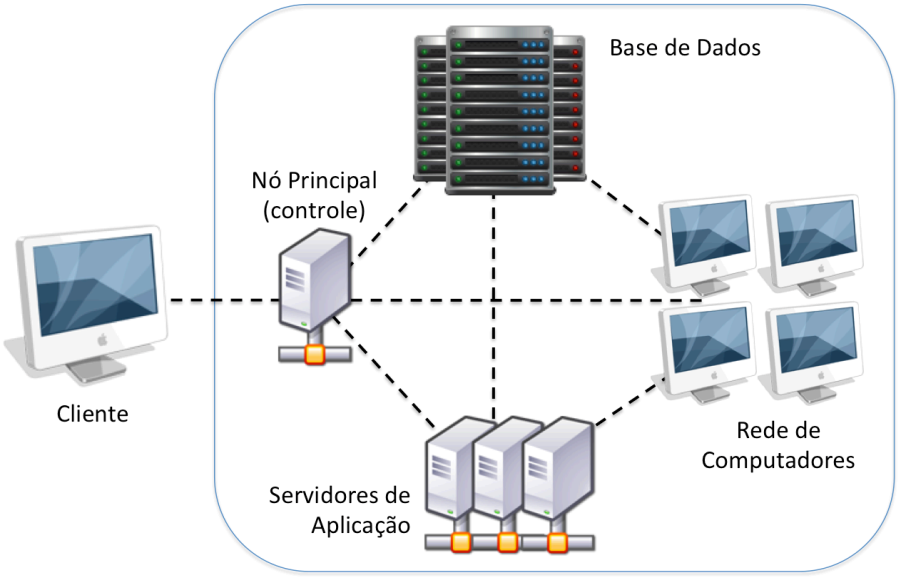

Figura 4 – Esquema geral do funcionamento da computação em nuvem.

2.4. Chiron

O Chiron (Ogasawara *et al.*, 2011a) é um motor de execução de *workflows* que tem como objetivo executar *workflows* em paralelo em ambientes de PAD. Executar um *workflow* nestes ambientes pode ser problemático, apresentando algumas dificuldades de gerenciamento de execuções de atividades e distribuição das informações coletadas para proveniência. Considerando este cenário, o Chiron é integrado ao sistema de banco relacional PostgreSQL a fim de prover um gerenciamento da execução do *workflow* de forma confiável. Para prover o paralelismo, o Chiron cria uma instância de

cada conjunto de parâmetros necessários para executar uma atividade. Esta instancia é chamada de ativação de uma atividade. Desta forma, qualquer instância do Chiron é capaz de executar ativações de um *workflow*, porém, apenas o nó principal tem permissão para acessar a base de dados para ler e escrever dados de controle e proveniência.

```
<?xml version="1.0" encoding="UTF-8"?>
<Chiron>
 <database name="chiron-opt" server="146.164.31.200"
   port="5432" username="postgres" password="postgres"/>
 <Workflow tag="exp" description="exp" exectag="case1_OPT_8"
   expdir="/scratch/ogasawara/optimization/exp1/case1/1/exp">
 <Activity tag="F" description="F" type="SPLIT_MAP"
  activation="java -jar /optimization/SWB.jar
   -directory=%=DIREXP% -type=F -K=%=K% -F=%=XF%">
  <Relation reltype="Input" name="I_F" filename="I_F.txt"/>
  <Relation reltype="Output" name="O_F" filename="O_F.txt"/>
  <Field name="K" type="float" pk="true"
   input="I_F" output="O_F"/>
  <Field name="TG" type="float" input="I_F" output="O_F"/>
  <Field name="XF" type="file" input="I_F" operation="COPY"/>
  <Field name="I" type="float" output="O_F"/>
  <Field name="XFF" type="file" output="O_F"/>
 </Activity>
```

Figura 5 - Especificação do *workflow* no Chiron que executa duas atividades (Ogasawara, 2011)

Além disso, as execuções paralelas das ativações do Chiron usam uma abordagem de álgebra relacional e a representação do *workflow* é feita por uma descrição em um arquivo XML. Esta decisão é motivada pelo fato do XML saber interpretar a álgebra pelo motor do *workflow* e com isso representar as informações necessárias para a execução do *workflow*. Os arquivos de relação das atividades são referenciados no arquivo XML, usando os elementos *Relation* e *Field* (Figura 5).

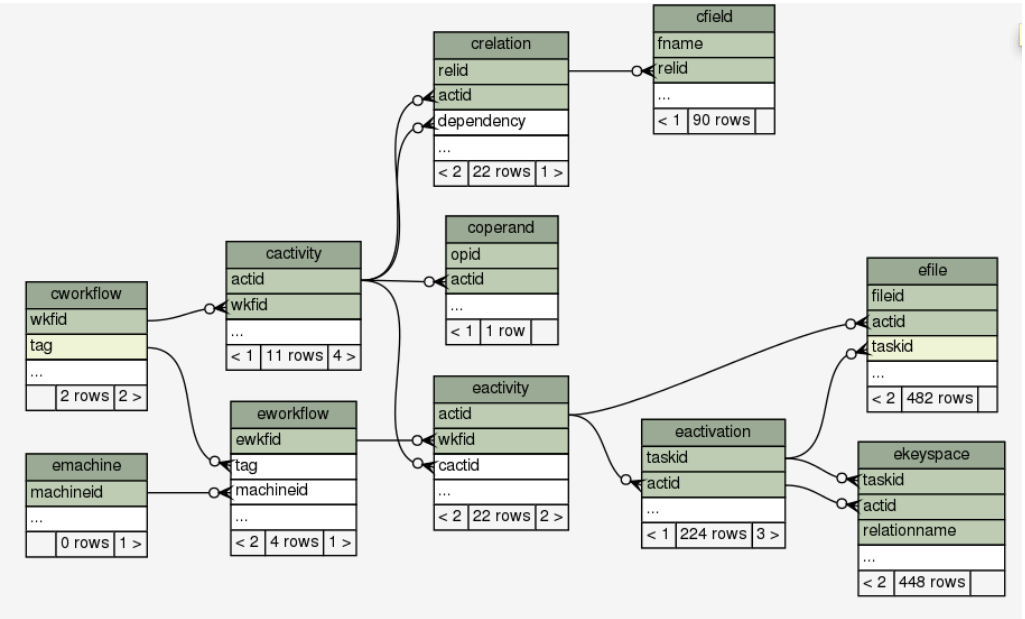

Figura 6 – Modelo de dados da proveniência prospectiva gerada pelo motor do Chiron (Horta *et al*., 2013).

O motor de execução Chiron foi primeiramente criado na COPPE, instituto da Universidade Federal do Rio de Janeiro, no laboratório NACAD com o grupo de pesquisa liderado pela Profa. Marta Mattoso em março de 2011.

3. Análise

3.1. A Visualização Científica

Em experimentos científicos de larga escala, cientistas precisam visualizar uma grande quantidade de resultados. Em meio a tantos resultados, ainda é necessário navegar pelos dados do experimento e da proveniência cruzando informações de atividades realizadas e seus artefatos. Neste capítulo, veremos um pouco sobre a importância da visualização, e como esta é abordada em trabalhos correlatos mais difundidos na comunidade científica.

3.2. Trabalhos Correlatos

Hoje em dia existem diferentes tipos de Sistemas de Gerência de *workflows* Científicos (SGWfC). Alguns deles oferecem suporte à visualização. O VisTrails (Callahan *et al.*, 2006a) é integrado com bibliotecas de visualização e permite a definição de atividades de visualização do *workflow* para exibir seus dados em paredes de monitores (*tiled-wall*). Sendo um de seus diferenciais o uso de uma área exclusiva de visualização (chamada *spreadsheet*), onde é possível visualizar e comparar os resultados das execuções de diferentes versões de um mesmo *workflow* como se fosse uma planilha. Em cada célula dessa "planilha" é apresentada o resultado obtido por uma instância da execução do *workflow*, conforme apresentado pela Figura 7. Contudo, infelizmente, não há suporte para PAD integrado para visualizar dados diretamente de *clusters* ou nuvens, ou a funcionalidade de acompanhamento, local ou remoto, da execução do *workflow* em tempo real.

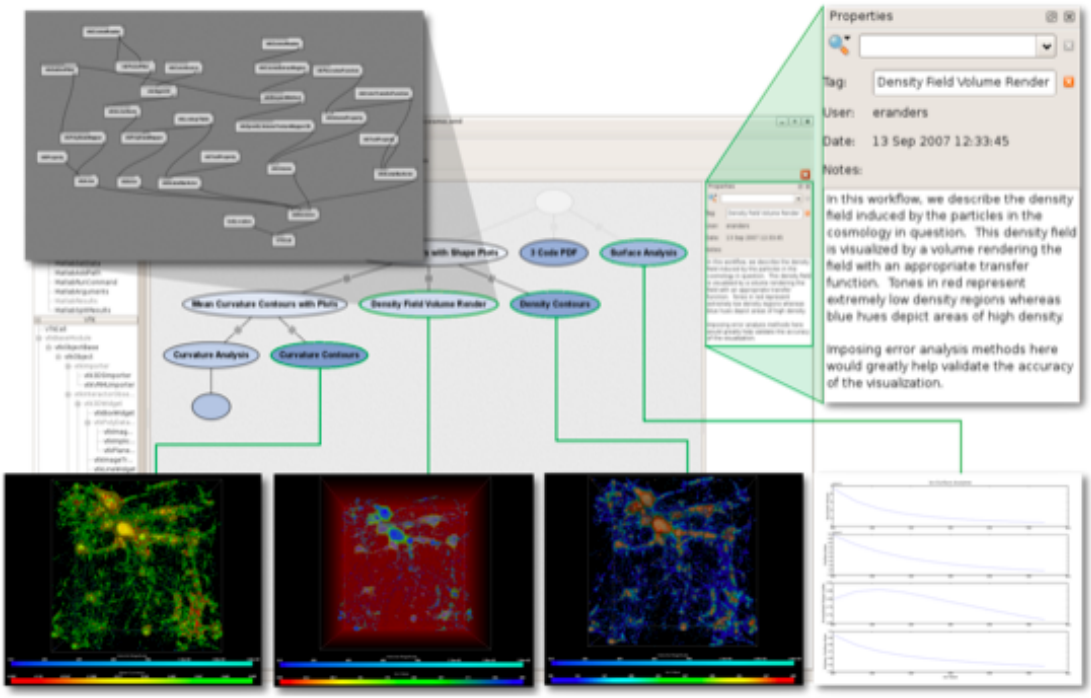

Figura 7 - Aplicação de visualização científica no VisTrails[5].

Por outro lado, existem outros SGWfC que fornecem apoio a PAD, como o Swift / Turbine (Wozniak *et al.*, 2012) ou o Pegasus (Deelman *et al.*, 2007). Estes por sua vez apresentam uma fraca integração entre os dados da proveniência e os resultados do experimento.

3.3. Desafios em Aberto

Como podemos ver, na seção 3.2, até onde foi pesquisado, não há uma solução que integre apoio de execução à PAD com as necessidades de visualização que enriqueça os resultados com dados de proveniência fazendo uso de um ambiente dedicado.

Desta forma, a fim de melhorar esta experiência, a abordagem proposta neste projeto visa prover um mecanismo para filtrar os resultados usando informações da proveniência, para que desta forma os cientistas possam recuperar apenas os dados que

[5] http://www.vistrails.org/index.php/

precisam para a análise, e num segundo passo façam uso destes em um ambiente de visualização. Seguindo esta linha, ressaltamos as contribuições da abordagem proposta.

3.3.1. Recuperação de Dados

Em primeiro lugar, sobre a recuperação de dados. A transferência de dados pode ser bem custosa, considerando o uso de *clusters* remotos e ambientes instanciados na nuvem como plataforma de execução dos experimentos. No caso de ambientes em nuvem, o caso pode ser ainda pior se levarmos em consideração os custos financeiros de transferências de dados através de serviços privados com a Amazon EC2[6]. Em ambos os casos, ao transferir resultados em larga escala, como vídeos de simulações ou imagens de alta resolução, ou até mesmo um número grande de pequenos arquivos, deve-se tomar cuidado com os custos envolvidos na transação sejam eles custos em dinheiro ou tempo.

3.3.2. Ambientes Dedicados para Visualização

Um segundo ponto importante diz respeito ao uso de ambientes dedicados para visualização. Estes devem explorar ao máximo as informações dos resultados. Uma análise com um equipamento especializado (*e.g.* uma parede de monitores de alta resolução), pode auxiliar a visualização de múltiplos resultados ao mesmo tempo, sendo possível estabelecer comparações entre eles, tendo uma visão maior da situação. De forma parecida, pode ser útil para visualizar resultados de experimentos com dimensão ou resolução que extrapole as configurações de dispositivos de vídeos padrão. Seguindo esta linha, podemos tirar vantagens de diferentes tecnologias disponíveis para

[6] http://aws.amazon.com/ec2/

visualização em larga escala, tais como SAGE[7], CGLX[8], TACC DisplayCluster[9] ou Paraview (ParaView-3.10.1, 2011) .

3.3.3. Caráter Colaborativo

Por último, vale ressaltar o caráter colaborativo desta abordagem. Ao passo que aumentamos o perfil de distribuição de nossas ferramentas, processamento e a forma que armazenamos nossos dados, analogamente devemos sustentar a distribuição dos recursos humanos. Analisando a gerência de *workflows* científicos, quase sempre, independente da localidade dos dados ou do processamento, dependemos do fator humano para a avaliação de resultados. Por sua vez, certos resultados necessitam de uma avaliação criteriosa, e muitas vezes dependem de equipamentos dedicados para sua visualização, por conseguinte estes não podem ser distribuídos, por serem bens físicos e muitas vezes de difícil mobilidade. Enfim, chegamos ao problema da distribuição do fator humano, que por depender do conhecimento individual, na maioria dos casos intransferível entre pessoas, muitas vezes não possui a disponibilidade de estar no mesmo local do recurso físico, neste caso, o ambiente de visualização. Sendo assim, inviabilizando a apresentação de certos resultados, devido à ausência do apresentador.

Desta forma, na abordagem proposta nesse projeto propomos uma solução flexível ao ponto quebrar as barreiras físicas. Onde um cientista seja capaz de apresentar seus resultados sem estar necessariamente ao lado dos dados, e muito menos do ambiente de visualização. Em uma conferência, congresso, ou reunião seria possível assistir uma apresentação onde os dados apresentados poderiam ser manipulados por um ou mais usuários, não necessariamente presentes fisicamente.

---

[7] http://www.sagecommons.org/
[8] http://vis.ucsd.edu/~cglx/
[9] http://www.tacc.utexas.edu/tacc-software/displaycluster

3.4. Requisitos Levantados

Portanto, para enfrentar as necessidades da visualização, com o caráter colaborativo, de experimentos em larga escala utilizando dados de proveniência, este projeto decidiu apoiar as seguintes características:

i. **Visualização dos dados do experimento por meio do *workflow***: ser capaz de apresentar os dados de uma determinada base de proveniência em uma solução *Web* com interface dinâmica e de fácil uso. Permitindo a navegação através de *workflows*, atividades ou relações a partir de conexão com a proveniência informada.

ii. **Seleção e filtro dos resultados**: permitir que o cientista realize consultas em uma proveniência através da interface para filtrar e selecionar os resultados desejados para visualização.

iii. **Rastreamento dos parâmetros responsáveis pela produção de determinado resultado**: permitir o rastreamento do processo de produção dos resultados à partir da proveniência.

iv. **Apresentação dos resultados na máquina local ou no ambiente de visualização**: permitir que resultados selecionados sejam exibidos em ambiente local ou remoto através de acoplamento com ambientes dedicados para visualização.

4. A Abordagem Prov-Vis

Neste capítulo serão detalhados todos os passos do desenvolvimento do projeto Prov-Vis a partir de sua fase de análise. Baseados nos requisitos levantados no capítulo 3, os próximos passos a serem estudados foram: a escolha da arquitetura; a elaboração dos módulos principais, suas tecnologias e interfaces; seu desenvolvimento, dificuldades e soluções; e por fim, a fase de experimentação.

4.1. Arquitetura

De acordo nos requisitos I, II, III e IV identificado na seção 3.4, a arquitetura da aplicação proposta foi dividida em dois módulos independentes. Estes módulos são: (A) módulo da **Aplicação *Web*,** conectada à base de dados da proveniência e ao ambiente de execução; (B) módulo da **Aplicação Remota**, responsável pela comunicação com o ambiente de Visualização.

4.1.1. Aplicação *Web*

Neste módulo, cientistas utilizam uma interface *Web* interativa, onde é possível navegar através de atividades do *workflow* e seus artefatos de entrada e saída. Sendo possível customizar opções de filtro para selecionar apenas os resultados desejados.

O módulo *Web,* além de responsável pela interface com o usuário, deve: (i) comunicar-se diretamente com a base da proveniência escolhida pelo usuário, e desta forma (ii) acessar os ambientes de execução escolhidos pelo motores responsáveis pela execução dos experimentos. Por exemplo, como ilustrado na Figura 8, o *workflow* de filogenética SciPhy, pode ter sido executado em ambiente remoto; a base de proveniência descrita pelo motor de execução, contém os dados de proveniência da execução de diversos experimentos deste *workflow*; a aplicação *Web* acessa os dados de proveniência, coletando estes informações.

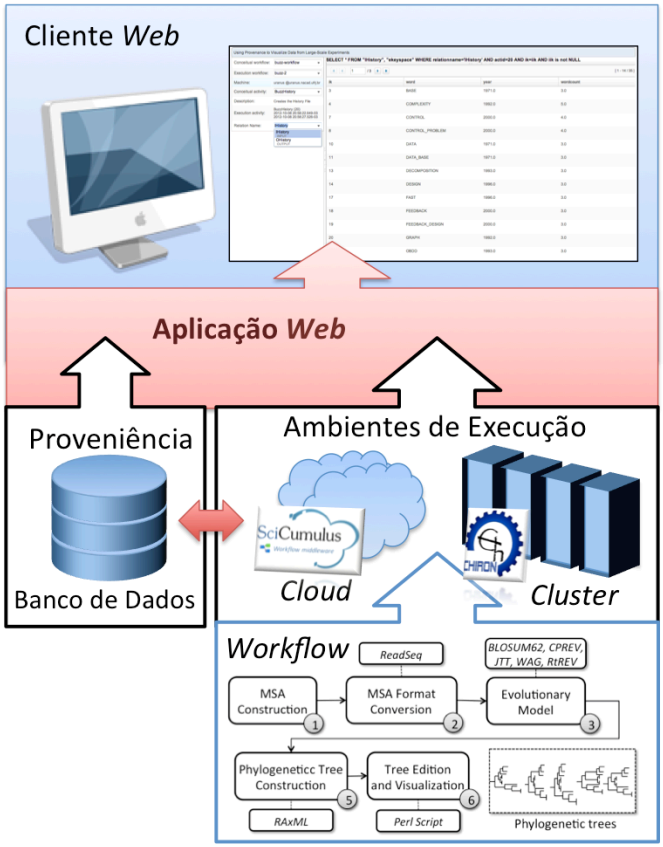

Figura 8 – Esquema geral da arquitetura do módulo *Web*.

4.1.2. Aplicação Remota

Aproveitando-se de uma implementação de serviço *Web*, como veremos na seção 4.4, foi possível criar uma aplicação remota que, hospedada no ambiente de visualização, possui a responsabilidade de gerenciar as requisições de acesso ao equipamento de visualização. Este aplicativo remoto fornece uma interface que é acessada por nosso módulo *Web* em tempo de execução.

Este módulo é capaz de tratar diversas solicitações do usuário, sendo possível apresentar mídias no ambiente de exibição, ou salvar sessões de mídias. Ele é projetado para ser conectável (do inglês *pluggable*) e por isso pode apoiar diferentes *middleware*

de ambientes de visualização, como por exemplo SAGE[10], CGLX[11], TACC DisplayCluster[12] ou Paraview. Este módulo e sua comunicação por serviços *Web* completam a arquitetura proposta neste projeto, como ilustrado na Figura 9.

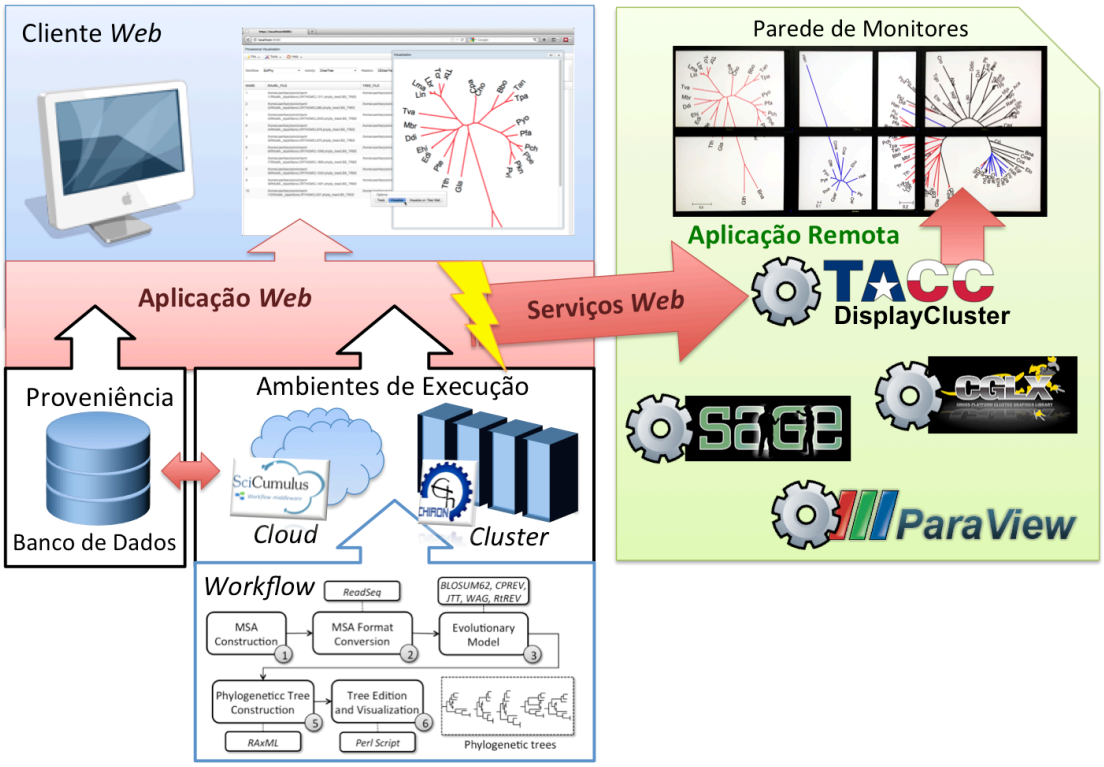

Figura 9 – Esquema geral da arquitetura proposta para o Prov-Vis.

4.2. Metodologia de Desenvolvimento

Como proposto na seção 4.1, o projeto foi dividido em dois elementos fundamentais: a interface *Web* dinâmica e a interface de acesso aos serviços do cluster de visualização. Tendo esta compreensão, foi necessário pesquisar, separadamente, quais tecnologias melhor atenderiam cada um desses aspectos.

Com a análise destes requisitos não funcionais, foi escolhida a tecnologia AJAX (Cheng and Cheng, 2007) e, em seguida, um *framework* que a implementasse de uma

[10] http://www.sagecommons.org/
[11] http://vis.ucsd.edu/~cglx/
[12] http://www.tacc.utexas.edu/tacc-software/displaycluster

forma simplificada e transparente. Por tal motivo, estudou-se o ZK[13], um *framework* de aplicação *Web* em AJAX, e de código-aberto.

Para o desenvolvimento da interface com o ambiente de visualização, foi escolhido a abordagem de serviços *Web*. A interface orientada à serviços oferece uma abstração da tecnologia aplicada em ambientes heterogêneos, oferecendo maior facilidade de acoplamento. Outras interfaces tornariam o sistema muito dependente do tipo de arquitetura a qual o ambiente foi construído. Na seção 4.4, são apresentados mais detalhes sobre este tema.

Como um último desafio, foi necessário discutir possíveis formas de integrar o servidor dos serviços *Web* com a parede de monitores. Neste passo, abordado no item 4.5, foi escolhido uma tecnologia para a visualização remota.

Em seguida, com a escolha das tecnologias, foram realizados testes e alguns protótipos foram construídos para estudar a viabilidade de utilizá-las de forma integrada. Desta forma, com a descrição do cenário de trabalho, as ferramentas a serem utilizadas e tendo legitimado a possibilidade de utilização das tecnologias escolhidas, foi possível observar as dificuldades encontradas nos protótipos e outras particularidades interessantes presentes em outros projetos similares. Desta forma, foi dado início ao estudo de modelagem do projeto buscando adequar um ambiente já funcional, mas que fosse base para possíveis extensões e adaptações.

4.3. O *Framework* ZK

Como mencionado na seção 4.2, foi escolhida a tecnologia AJAX (Cheng and Cheng, 2007) e, em seguida, um *framework* que a implementasse de uma forma

[13] http://www.zkoss.org/

simplificada e transparente. Desta forma, estudou-se o ZK[14], um *framework* de aplicação *Web* em AJAX, e de código-aberto.

O ZK é um *framework* para construção de aplicativos *Web* em AJAX de código fonte aberto, escrito em Java. O *framework* permite a criação de interfaces gráficas ricas para aplicativos *Web* sem que ao menos seja necessário desenvolver em *JavaScript*[15].

Uma vantagem desta ferramenta se dá pela manutenabilidade e uniformidade do código que faz a integração entre operações executadas no lado do cliente e no servidor da aplicação. Logo, este *framework* não obriga que o desenvolvedor tenha grande conhecimento em *JavaScript*, diferentemente da maioria dos outros *frameworks* para AJAX. Porém, o fato de não ser necessário aprender esta tecnologia, não implica que o *framework* não a utilize. A questão é que o motor do ZK transforma os scripts produzidos em outra linguagem em código *JavaScript*, sendo este interpretado pelo navegador de forma transparente para o usuário.

Para fazer uso do framework o desenvolvedor pode utilizar diversas linguagens de programação. Por padrão, usa-se o Java, porém outras linguagens como Ruby, Groovy, Python e o próprio *JavaScript* também são linguagens de desenvolvimento disponíveis para interpretação.

A fim de permitir a elaboração de componentes e páginas ZK de maneira intuitiva, através da declaração de *tags*, em um formato similar ao HTML, o *framework* define uma linguagem de marcação própria, o ZUML (ZK *User Interface Markup Language*). Sendo assim, a ferramenta simplifica a criação de componentes, de forma que seja apenas necessário incluir na página a *tag* deste com seus atributos. Um exemplo pode ser visto na Figura 10.

---

[14] http://www.zkoss.org/
[15] http://en.wikipedia.org/wiki/JavaScript

```
<tabbox id="tb" width="100%" height="100%">
    <tabs>
        <tab label="Query and Result" />
        <tab label="Result Files" />
        <tab label="Tiled-Wall Viewer" />
    </tabs>
    <tabpanels>
        <tabpanel>
            <label id="tableTitle" multiline="true" style="font-size:13px;"/>
            <grid id="dbresults" width="100%" height="100%"/>
        </tabpanel>
        <tabpanel>
            <grid id="dbfiles" width="100%" height="100%"/>
        </tabpanel>
        <tabpanel>
            <a style="diplay:none;" href="#" id="send_xml" class="send_xml" >Refresh!</a>
            <zscript>
                send_xml.setWidgetListener(Events.ON_CLICK, "sendXmlToServer();");
            </zscript>
            <include src="viewer.html">
                <custom-attributes org.zkoss.zul.include.html.defer="true"/>
            </include>
        </tabpanel>
    </tabpanels>
</tabbox>
```

Figura 10 – Exemplo de chamada de *tags* na chamada de componentes.

Como se pode observar, as páginas ZUML modelam a interface que será visualizada no navegador, ou também conhecido como *front-end*. Contudo, para que seja possível dar funcionalidades ao sistema aplicando assim as regras de negócios, ainda é necessária a definição dos scripts no lado do servidor, ou *server-side*. Diante disso, nosso *framework* permite a inserção de *scripts* diretamente na página ZUML através da *tag* <zscript>. No interior desta *tag*, pode-se inserir código na linguagem de programação disponível e escolhida anteriormente pelo desenvolvedor.

De outra forma, também é permitido e desejável, criar *scripts* em arquivos separados para melhor manutenção e compreensão do código. Logo, seguindo essa segunda metodologia, a página ZUML efetuará uma inclusão do arquivo contendo os *scripts* desejados. A Figura 11 mostra a arquitetura de desenvolvimento do ZK:

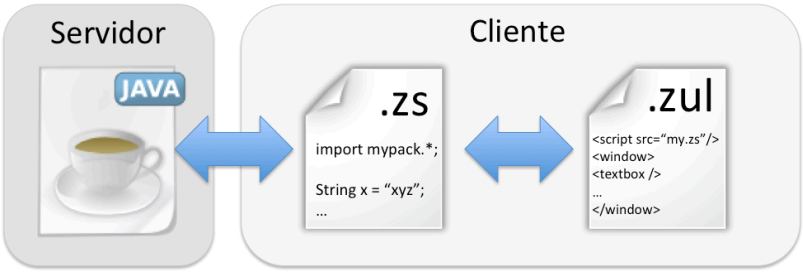

Figura 11 - Estrutura de desenvolvimento do ZK

É necessário que o servidor de aplicação escolhido como *container* desta aplicação *Web*, possua as bibliotecas do ZK para poder carregar e exibir os componentes definidos nas páginas ZUML e para transformar o modelo de desenvolvimento (Figura 11) no modelo AJAX tradicional, com HTML e *JavaScript*. Com isso, podemos relacionar o que foi desenvolvido com o que foi gerado pelo *framework* e utilizado pelo servidor em produção, conforme ilustrado na Figura 12.

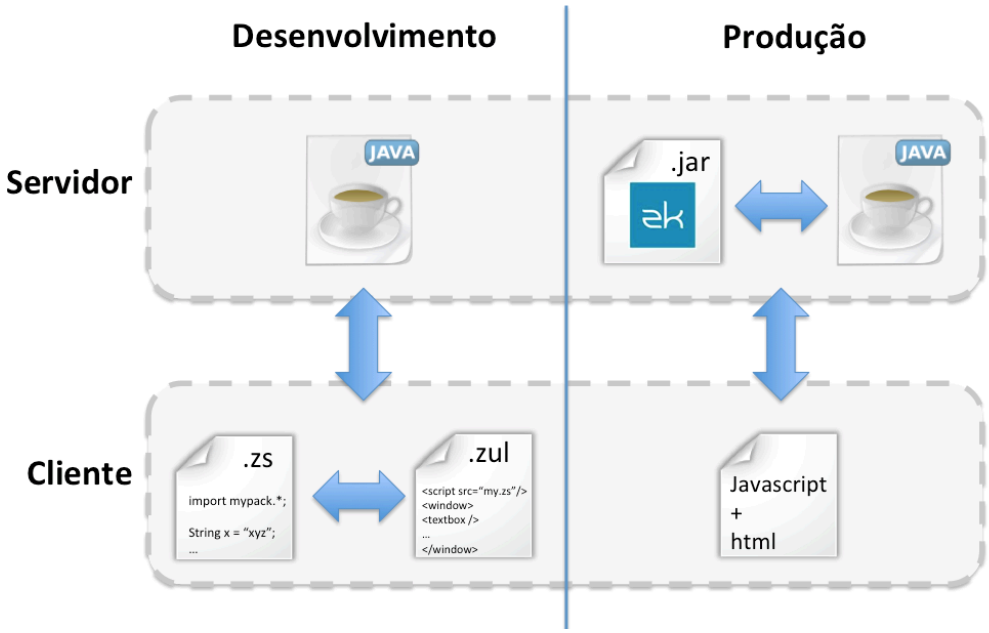

Figura 12 - Resultado produzido pelo *framework* ZK.

O *framework* escolhido mostrou-se bem eficiente na criação dos protótipos de interface e o resultado alcançado foi muito satisfatório. A integração com outras tecnologias, os seus componentes disponíveis, documentação e a facilidade no aprendizado foram os principais fatores que apoiaram sua adoção no projeto.

4.4. Serviços *Web*

Em engenharia de software, a Arquitetura Orientada a Serviços (SOA) é um estilo de arquitetura para projetar e desenvolver *softwares* na forma de serviços, provendo interoperabilidade. Sendo assim, neste modelo, é necessário que os serviços possuam duas características essenciais: a autodescrição e a facilidade de descoberta .

A autodescrição consiste em fornecer, junto à publicação do serviço, sua documentação. Esta documentação deve fornecer a descrição da interface pública do serviço de forma inteligível para que os demais desenvolvedores possam facilmente integrar o serviço. Idealmente, a interface pública pode ser escrita numa gramática XML comum que identifique todos os métodos públicos, argumentos e valores de retorno do serviço. A facilidade de descoberta requer um mecanismo simples de publicação e localização do serviço e de sua interface pública. Um esquema geral de uma arquitetura SOA é ilustrado na Figura 13. Nesta imagem podemos reparar o intermediador (do inglês *service-broker*), porém sua presença é opcional dependendo da definição de SOA aplicado.

Figura 13 - Esquema geral do comportamento de uma arquitetura SOA.

Para aplicar a arquitetura proposta, foram definidos os Serviços *Web*, onde os serviços são disponibilizados através de requisições HTTP. Sendo assim, das implementações de Serviços *Web* pesquisadas, a suíte SOAP[16] se mostrou a mais completa. O SOAP (*Simple Object Access Protocol*) providencia as características de autodescrição e facilidade de descoberta, respectivamente através dos padrões WSDL[17] (*Web Services Description Language)* e UDDI[18] (*Universal Description, Discovery and Integration*).

Para representar as requisições e respostas dos serviços, o WSDL implementa uma gramática comum. Portanto, a linguagem é capaz de informar, essencialmente, as interfaces de todas as funções públicas dos serviços definidos, o tipo dos valores de retorno de cada resposta e dos argumentos da requisição, e informações relativas ao endereçamento do serviço. Em resumo, a linguagem representa um contrato entre requisitante e fornecedor, da mesma forma que a interface Java intermedia o código cliente da classe propriamente dita. O UDDI, por sua vez, apresenta-se como uma técnica para catalogar, especificar e apontar a localização dos serviços.

A Figura 14 mostra o ciclo de vida de um serviço *Web* no qual, após o desenvolvimento ele é publicado, tendo seu endereço e interfaces descritas em WSDL. Então, via uma busca UDDI, uma aplicação cliente obtém a descrição do serviço e seu endereço e passa a utilizá-lo com mensagens SOAP.

[16] http://www.w3.org/TR/soap12-part1
[17] http://www.w3.org/TR/wsdl
[18] http://uddi.xml.org/

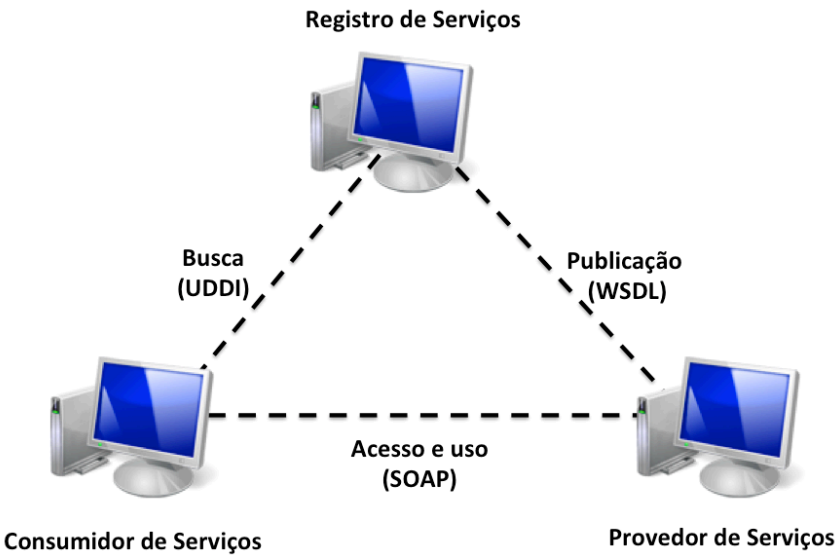

Figura 14 - Ciclo de vida de um serviço *Web*. Adaptado de (Dias, 2009).

4.5. Integração com a Proveniência de Dados

Como proposto na fase de análise, seção 3.4, os principais requisitos funcionais do servidor de aplicação *Web* identificadas seriam*:* (1) se conectar à proveniência de dados informada pelo usuário, segundo requisitos I, II e III; e (2) a recuperação de resultados, localizados no ambiente de execução específico à cada *workflow*, contemplando parte do requisito IV.

Para a integração da aplicação Prov-Vis com uma base de dados, informada pelo usuário, existem duas delimitações: primeiramente, a proveniência a ser conectada à aplicação deveria ser gerada por um descritor específico, no caso o motor de execução de *workflows* Chiron (Ogasawara *et al.*, 2011b); em segundo lugar, a base de dados indicada tem o compromisso de prover as informações para a conexão à ambientes de armazenamento remoto.

Desta forma, foi implementada uma interface de conexão (Figura 15), responsável por coletar informações sobre a localização e autenticação à uma base de

proveniência. Consequentemente, esta fase precederia qualquer funcionalidade do sistema proposto.

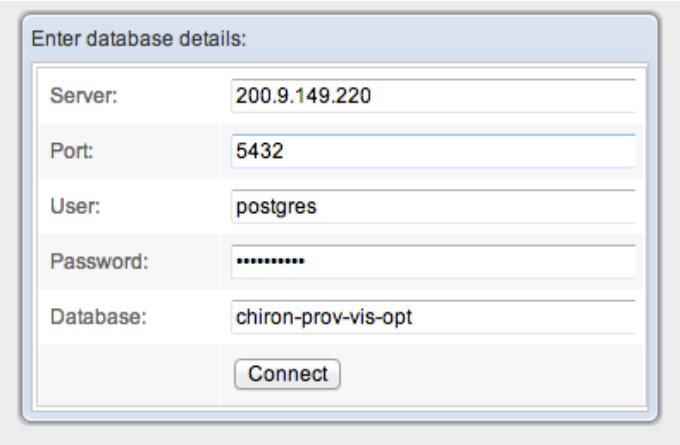

Figura 15 - Janela para aquisição de dados para a conexão.

Após a conexão com a base de dados da proveniência, ao usuário é apresentada uma interface composta em duas seções: seção de consulta (a), apenas para filtro e seleção, onde é possível navegar por *workflows*, suas atividades e relações (Figura 16); e a segunda, seção de visualização (b), responsável pela apresentação dos resultados filtrados e a gerência destes no ambiente remoto de visualização.

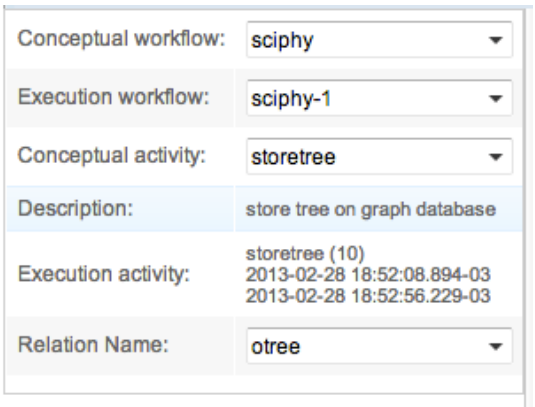

Figura 16 – Filtros para a consulta à base de dados da proveniência.

Esta segunda seção, presente na tela principal do Prov-Vis pode ser divida em até três outras, organizadas em abas, dependendo dos resultados da consulta aos dados de proveniência. A primeira aba desta seção apresenta a sintaxe da consulta realizada na

base de dados da proveniência, apresentando os resultados da relação escolhida, independente da presença de arquivos (Figura 17). A segunda aba, no caso de existirem arquivos na relação de saída consultada, apresenta suas informações com detalhes, disponibilizando as opções de *download* do arquivo ou apenas extração para visualização em um ambiente dedicado (Figura 18); a terceira aba, responsável pela gerência da visualização destes arquivos com o ambiente de visualização conectado à aplicação, disponibiliza uma interface de simulação da composição visual dos resultados e disposição das telas presente na parede de monitores alvo. (Figura 19).

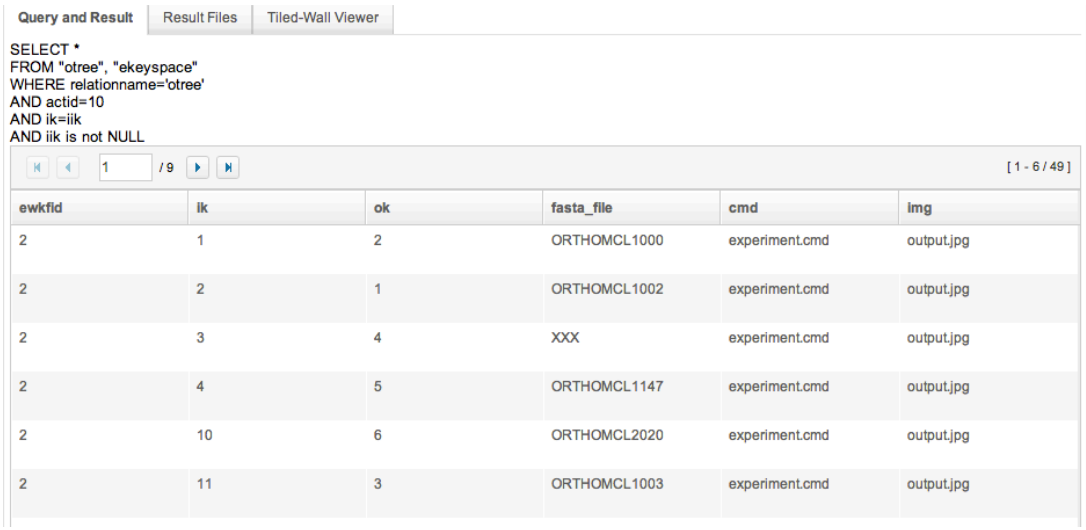

Figura 17 - Apresentação da sintaxe da consulta realizada e seus resultados.

Figura 18 – Apresentação dos arquivos encontrados na relação consultada.

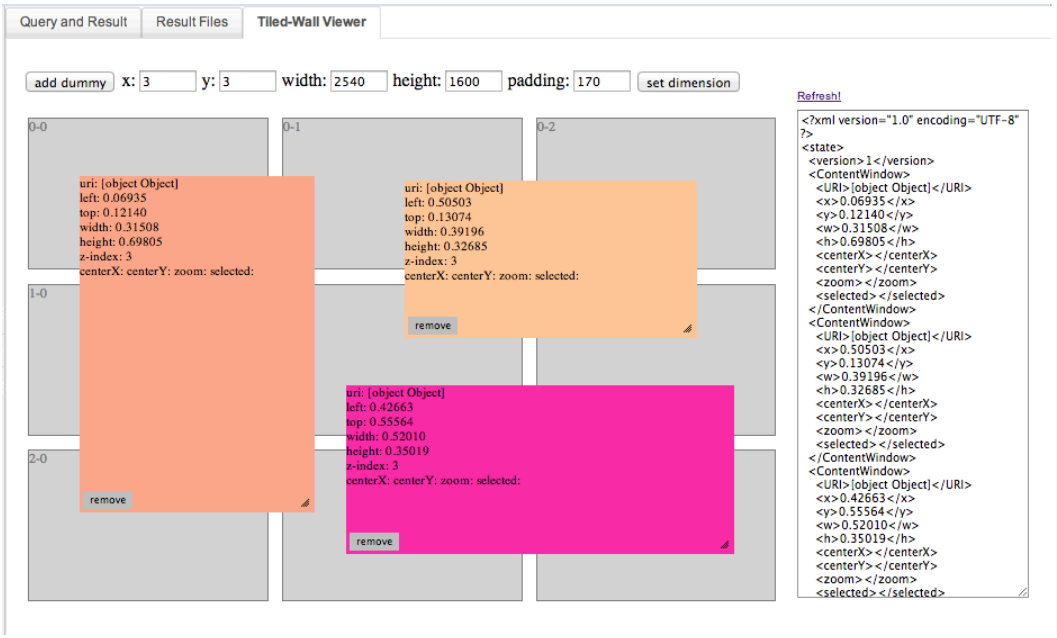

Figura 19 – Gerência da disposição dos resultados escolhidos para a visualização.

Para a conexão com o ambiente de execução específico ao *workflow* executado, o qual desejamos extrair resultados, se faz necessário que as informações de conexão estejam armazenadas na proveniência de dados. Em posse destas informações, ao escolher um resultado para visualizar, o arquivo selecionado para a extração pode ser extraído do ambiente de execução (processo conhecido como *data stage-out*). Esta atividade é realizada através da interface *Web* (Figura 16), utilizando um módulo de transferência de arquivos chamado SCP (*Secure Copy*), baseado no protocolo SSH[19] (*Secure Shell*). Sendo feita esta integração diretamente entre o servidor de aplicação *Web* e o repositório do arquivo à partir do seguinte comando:

```
scp usuario@servidorRemoto:diretório/ArquivoOriginal ArquivoAlvo
```

[19] http://nixdoc.net/man-pages/FreeBSD/scp.1.html#HISTORY

4.6. Integração com o Ambiente de Visualização

Com a aquisição dos resultados selecionados à serem visualizados, a última etapa deste processo é a projeção destes no ambiente de visualização conectado ao servidor da aplicação. Esta etapa depende da configuração de acesso ao ambiente remoto, que por sua vez possui algumas delimitações detalhadas a seguir.

Foi decidido, dentre as possíveis formas de integrar o servidor dos serviços *Web* com o ambiente de visualização, que entre as tecnologias existentes para visualização, seria utilizada a melhor que se adaptasse ao protótipo proposto, segundo as especificações dos equipamentos disponíveis (seção 4.7.1). Esta decisão apesar de unilateral e oposta à adaptação da ferramenta à qualquer ambiente, serviu como base experimental para o desenvolvimento da ferramenta Prov-Vis, e é até então o estado da arte deste projeto.

A melhor solução para esta integração seria em nosso cenário de teste seria a ferramenta DisplayCluster[20], desenvolvida no TACC[21] (*Texas Advanced Computer Center*). Esta ferramenta, pode ser chamada como um ambiente de software implementado para conduzir de forma interativa parede monitores em larga escala. O software permite aos usuários visualizar interativamente mídias, tais como imagens de alta resolução ou vídeos.

Além disso, uma interface de *script* Python[22] é fornecida pela ferramenta DisplayCluster, para automatizar interação entre a mesma e outras aplicações. Sendo assim, esta foi a interface utilizada na integração entre servidor de aplicação *Web* e o ambiente de visualização, por meio de um provedor de serviços *Web* (seção 4.4).

---

[20] http://www.tacc.utexas.edu/tacc-software/displaycluster
[21] http://www.tacc.utexas.edu/
[22] http://www.python.org/

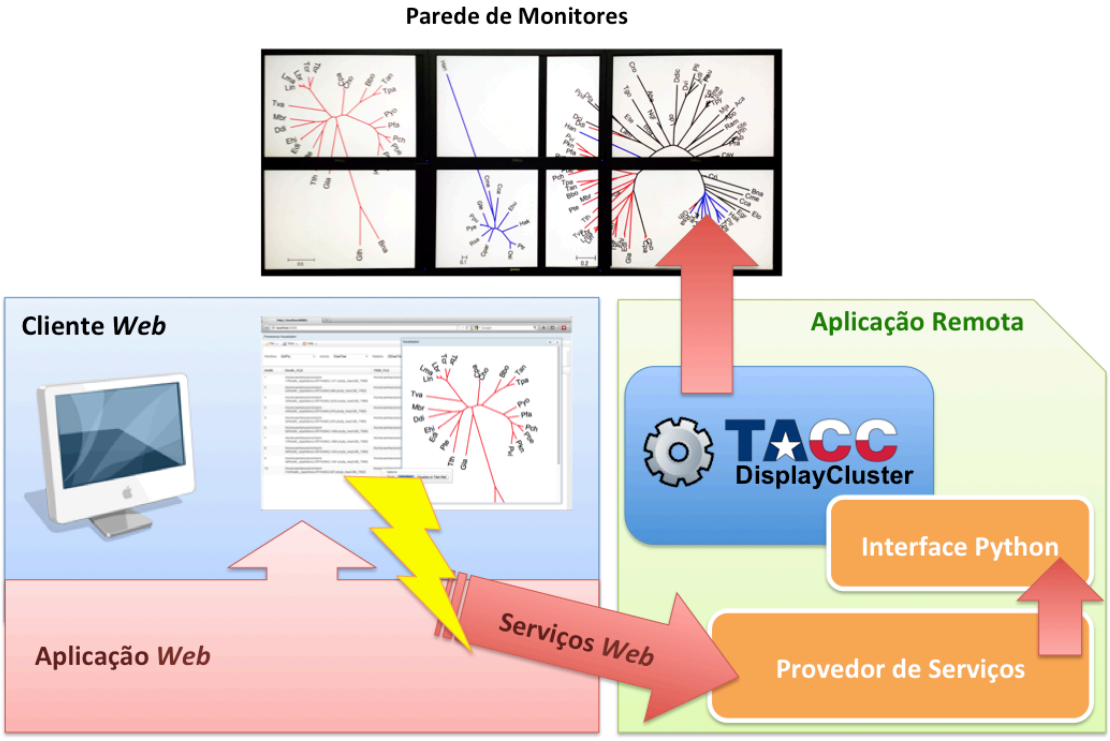

Figura 20 – Esquema geral de comunicação do cliente à visualização.

4.7. Estudo de Caso

Como caso de uso para este projeto, foi escolhido explorar o caso bioinformática representado pelo *workflow* de filogenética SciPhy (Ocaña *et al.*, 2011b). Neste *workflow*, em linhas gerais, é executada uma série de atividades de seleção, normalização e construção de árvores filogenéticas.

De forma a obter imagens ao final da execução deste *workflow*, foi necessário adicionar uma nova atividade, chamada *Tree Storage*, responsável por armazenar as árvores filogenéticas e exporta-las no formato de imagens em sua relação de saída (Figura 21). Desta forma o resultado principal produzido ao final de suas execuções são arquivos, estruturados em texto e imagens, que representam árvores filogenéticas.

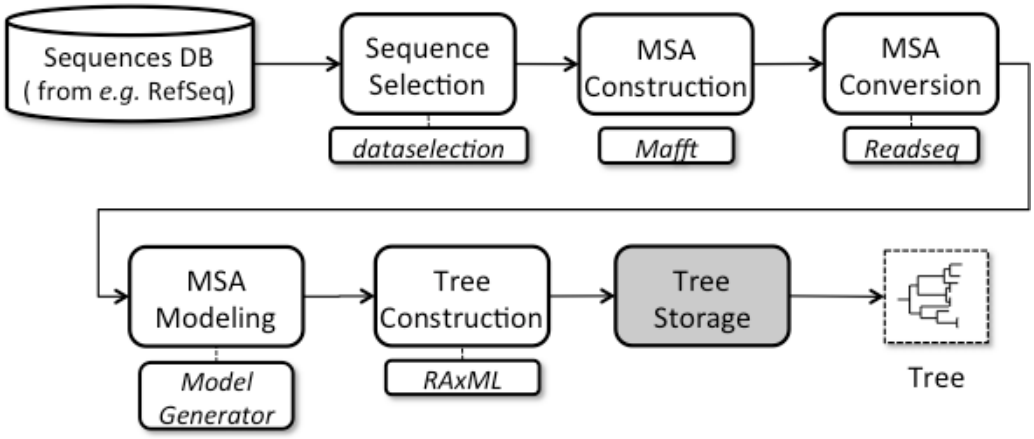

Figura 21 – *Workflow SciPhy* adaptado para armazenar árvores filogenéticas.

Portanto, o objetivo da fase de experimentação do projeto Prov-Vis foi aplicar todas as suas funcionalidades implementadas com este *workflow*, visando recuperar os principais resultados de imagens gerados em sua última atividade *Tree Storage*. Sendo estas: configuração do ambiente de visualização (seção 4.7.1); acesso a aplicação *Web,* instanciada no próprio ambiente de visualização (Figura 22); a conexão a base de dados da proveniência (Figura 23); filtro e seleção dos resultados (Figura 24, 25 e 26); composição e organização dos resultados para visualização (Figura 27); análise dos resultados na parede de monitores (Figura 28).

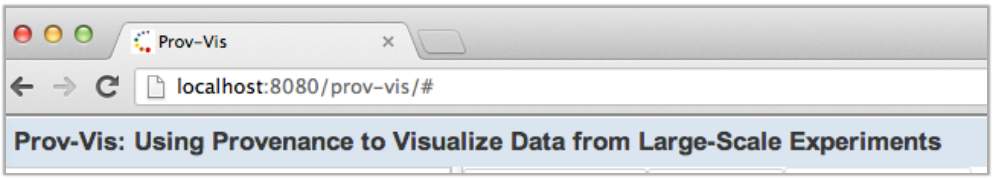

Figura 22 - Acesso a aplicação *Web.*

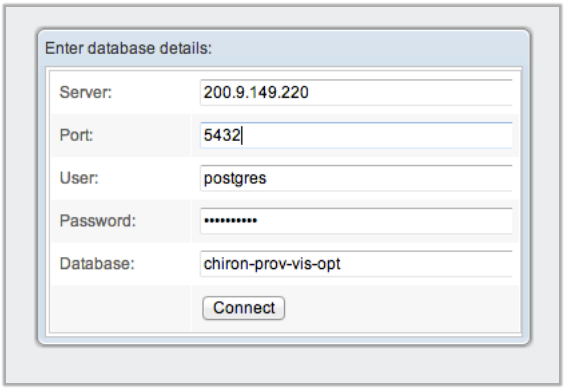

Figura 23 - Conexão a base de dados de proveniência.

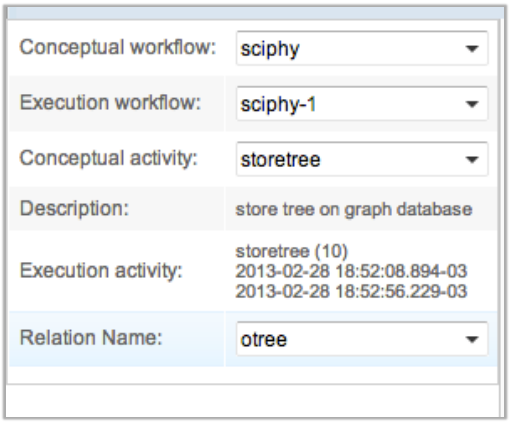

Figura 24 - Filtro e seleção de resultados.

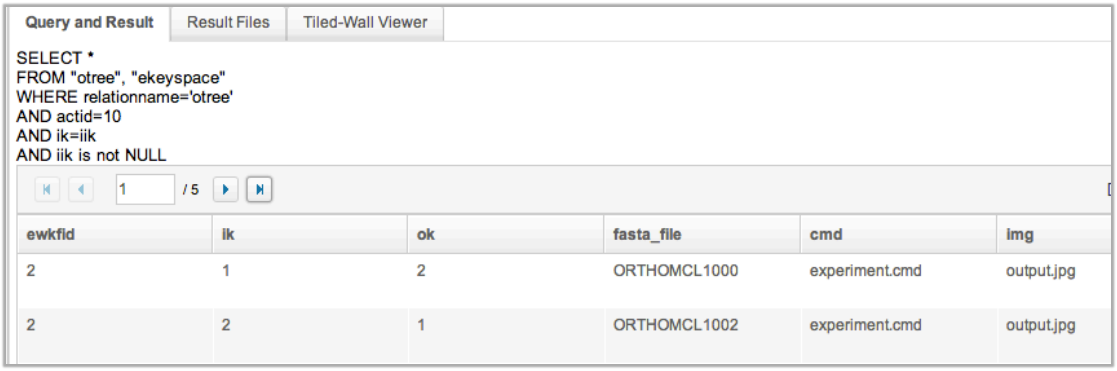

Figura 25 – Análise dos resultados selecionados pelo filtro.

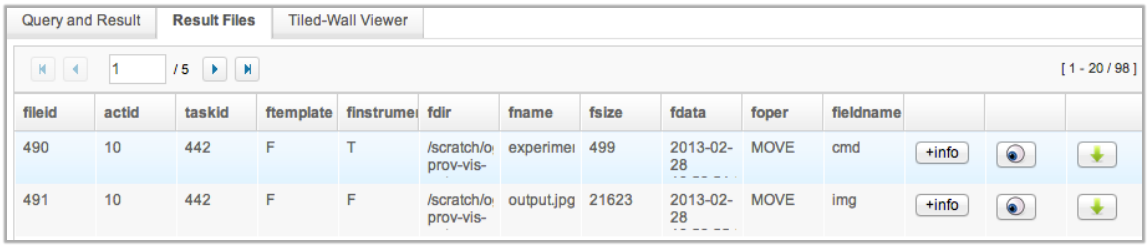

Figura 26 – Seleção de mídias para a visualização.

Figura 27 – Composição e organização dos resultados para visualização.

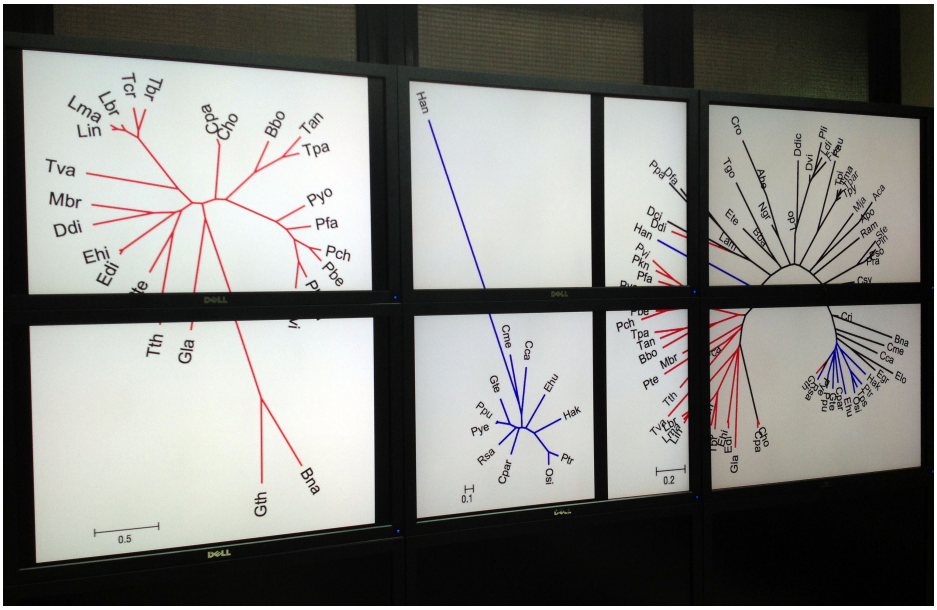

Figura 28 – Resultados exportados para o ambiente de visualização do NACAD.

4.7.1. Ambiente de Visualização

Todo o ambiente de visualização foi disponibilizado pelo NACAD. O ambiente, instalado dentro do laboratório, é composto por dez telas e três servidores, compondo um *cluster* de visualização, nomeado como Mercury. Sua configuração física é a seguinte:

- Três *workstations* DELL
  - 24 núcleos de processamento
    - Intel Xeon E5506 @ 2.13GHz
  - 36 GB de memória RAM
  - 5 placas gráficas NVIDIA Quadro 6000
  - 2,5 TB de disco rígido
- Dez monitores DELL
  - 31” com resolução máxima de 2560x1600
  - 9 monitores no painel de visualização
  - 1 para administração

Os dois nós do *cluster* (N1 e N2) possuem cada um 8 núcleos de processamento e 12 GB de memória volátil. Cada um dos nós possui duas placas de vídeo NVIDIA Quadro 6000 ligadas nos dois barramento PCI Express x16 de cada máquina. As placas gráficas são configuradas em *dual-head* para funcionarem em conjunto e estão conectadas a quatro monitores (onde cada placa controla dois monitores). Cada um dos nós possui 500GB de disco rígido para armazenamento local.

O servidor do *cluster* Mercury (ADM, acrônimo para administrador) é semelhante aos nós e possui o mesmo número de processadores e memória volátil. Entretanto ele só possui um placa gráfica NVIDIA Quadro 6000 e controla apenas um monitor do painel de visualização. Além disso, o servidor ADM também controla o monitor de administração do cluster. A estrutura de disco do servidor também é diferente, uma vez que possui dois discos de 500GB ligados em RAID1 para armazenar

os diretórios dos usuários e outros dois discos de 250GB ligados também em RAID1 para armazenamento local. No máquina ADM, o diretório dos usuários (/home) e o diretório /sw são exportados através do sistema de arquivos de rede (NFS, *Network File System*), de forma que todos os nós podem ter acesso aos arquivos destes diretórios. Além disso, o SSH entre os nós foi configurado para funcionar sem senha, para facilitar o disparo das aplicações de visualização que iniciam processos diversos remotamente nos nós N1 e N2.

Como uma mesma máquina física pode controlar até quatro monitores, a configuração do painel de visualização é composta como mostra a Figura 29. Ou seja, a máquina ADM controla um servidor do painel e o monitor de controle e administração. Os outros nós controlam, cada um, quatro monitores do painel.

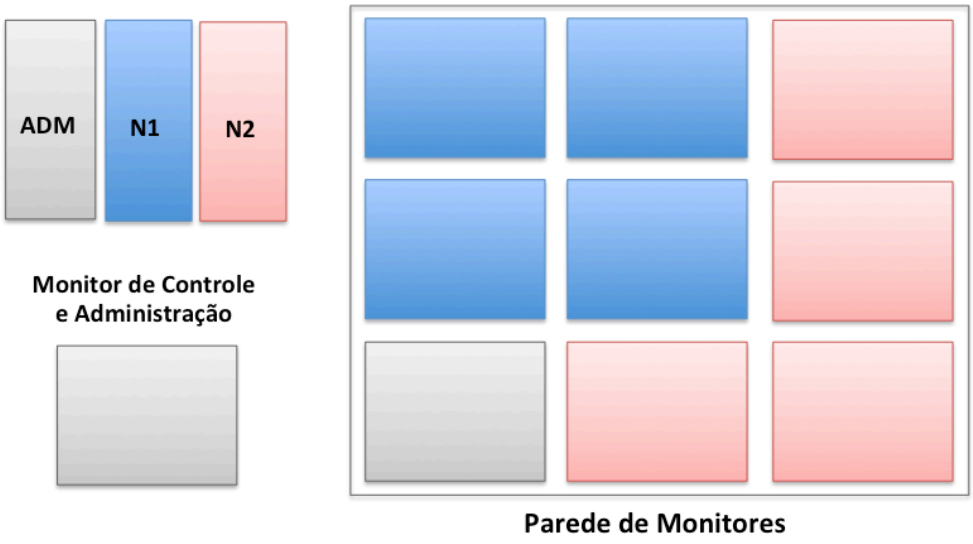

Figura 29 - Configuração do ambiente de visualização científica do NACAD.

A configuração de *software* do *cluster* de visualização é uma composição das ferramentas de instalação de *clusters* do NACAD em conjunto com a ferramenta de visualização científica do *Texas Advanced Computing Center* (TACC[23]). Como sistema

[23] http://www.tacc.utexas.edu/

operacional, optou-se pelo Fedora 17, Kernel 3.6.11. O driver de vídeo utilizado é proprietário da NVIDIA 310.32. O *software* de visualização instalado até o momento é o TACC DisplayCluster[24], como abordado na seção 4.6.

O TACC é um centro de excelência em visualização científica e apresentou sua nova ferramenta DisplayCluster em 2012, na conferência *International Conference for High Performance Computing, Networking, Storage and Analysis* (*Supercomputing* 2012, ou SC12). A configuração apresentada pelo TACC nesta conferência foi exatamente a configuração utilizada no painel de visualização do NACAD.

[24] http://www.tacc.utexas.edu/tacc-software/displaycluster

5. Conclusão e Trabalhos Futuros

Hoje em dia existem diferentes tipos de Sistemas de Gerência de *workflows* Científicos (SGWfC). Alguns deles oferecem apoio a visualização. O VisTrails (Callahan *et al.*, 2006a) é integrado com as bibliotecas de visualização e permite a definição de atividades de visualização do *workflow* exiba dados em ambientes de visualização. Infelizmente, não há suporte a PAD integrado para visualizar dados diretamente de *clusters* ou nuvens.

Entretanto, existem outros SGWfC que oferecem apoio a PAD, como Swift / Turbine (Wozniak *et al.*, 2012) ou Pegasus (Deelman *et al.*, 2007). Infelizmente, estes apresentam uma fraca integração entre os dados da proveniência e os resultados do experimento. Até onde foi pesquisado, não há uma solução que integre apoio a PAD com as necessidades de visualização enriquecendo os resultados com dados de proveniência.

Experimentos em larga escala normalmente produzem grande quantidade de dados a ser visualizado. Cientistas precisam de ambientes de visualização para melhorar a análise de seus experimentos. Em meio à complexas comparações, é de extrema necessidade aproveitar ao máximo da visualização do resultado gerado no *workflow*. Sendo assim, neste projeto foi proposto e desenvolvido uma aplicação *Web* para visualizar os resultados experimentais enriquecidos com os dados de proveniência.

Como experimentado, o aplicativo permite que cientistas naveguem através dos dados produzidos por seu *workflow*, sendo possível selecionar os resultados desejados e visualizá-los em seu navegador ou em um ambiente de visualização disponível, local ou remotamente. Sendo também possível controlar quais os dados produzidos a cada resultado e fazer consultas gerais no banco de dados de proveniência e assim usufruir de uma maior visão do experimento.

Com a evolução do desenvolvimento deste projeto, fatores importantes foram levantados referentes a novas funcionalidades que poderiam ser agregadas para trazer maior facilidade e usabilidade à tarefas de gerenciamento de *workflows* em experimentos de larga escala. Exemplos de novas funcionalidades podem surgir ao explorar o conceito de análise em tempo de execução. Com esta abordagem, é possível poupar o trabalho de se chegar a resultados não esperados devido à erros causados por falhas em pequenos ajustes. Falhas as quais poderiam ter sido reparadas antes do término da execução do *workflow*.

6. Referências Bibliográficas

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

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

CHENG, H., CHENG, R., 2007, *ZK: Ajax Without the Javascript Framework*. Apress.

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

DIAS, J., 2009, *LEMMing: Sistema de Gerenciamento de Clusters através de Serviços Web e uma Interface Web Dinâmica*. Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Engenheiro., Universidade Federal do Rio de Janeiro

DIAS, J., OGASAWARA, E., OLIVEIRA, D., *et al*., 2011, "Supporting Dynamic Parameter Sweep in Adaptive and User-Steered Workflow". In: *6th Workshop on Workflows in Support of Large-Scale ScienceWORKS '11*, pp. 31–36, Seattle, WA, USA.

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

HORTA, F. ; DIAS, J. ; OCAÑA, K. ; OLIVEIRA, D. ; OGASAWARA, E. ; MATTOSO, MARTA . Poster*: Using Provenance to Visualize Data from Large-Scale Experiments. In: Research Poster, 2012, Salt Lake City. In: Super Computing* SC12, 2012.

HORTA, F., SILVA, V., COSTA, F., *et al*., 2013, "Provenance Traces from Chiron Parallel Workflow Engine". In: *Proceedings of the International Workshop on Managing and Querying Provenance Data at ScaleInternational Workshop on Managing and Querying Provenance Data at Scale*, Genova, Italy.

MATTOSO, M., WERNER, C., TRAVASSOS, G., *et al*., 2009, "Desafios no Apoio à Composição de Experimentos Científicos em Larga Escala". In: *SEMISH - CSBC*, pp. 307–321, Bento Gonçalves, Rio Grande do Sul, Brazil.

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.

OCAÑA, K. A. C. S., OLIVEIRA, D. DE, HORTA, F., *et al*., 2012, "Exploring Molecular Evolution Reconstruction Using a Parallel Cloud-based Scientific Workflow", *Advances in Bioinformatics and Computational Biology*, , chapter 7409, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 179–191.

OCAÑA, K. A. C. S., OLIVEIRA, D., OGASAWARA, E., *et al*., 2011, "SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes", In: Norberto de Souza, O., Telles, G. P., Palakal, M. [eds.] (eds), *Advances in Bioinformatics and Computational Biology*, , chapter 6832, Berlin, Heidelberg: Springer Berlin, pp. 66–70.

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

OGASAWARA, E., DIAS, J., OLIVEIRA, D., *ET AL*., 2011, "An Algebraic Approach for Data-Centric Scientific Workflows", *Proc. of VLDB Endowment*, v. 4, n. 12, pp. 1328–1339.

OINN, T., ADDIS, M., FERRIS, J., *et al*., 2004, "Taverna: a tool for the composition and enactment of bioinformatics workflows", *Bioinformatics*, v. 20, pp. 3045–3054.

OLIVEIRA, D., 2012, *Uma Abordagem De Apoio À Execução Paralela De Workflows Científicos Em Nuvens De Computadores*. Tese de Doutorado 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 Doutor em Engenharia de Sistemas e Computação, UFRJ/COPPE

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

SILVA, V., CHIRIGATI, F., MAIA, K., *et al*., 2010, "SimiFlow: Uma Arquitetura para Agrupamento de Workflows por Similaridade". In: *IV e-Science*, Belo Horizonte, Minas Gerais, Brazil.

SOUSA, V., 2011, *SimiFlow: Uma Arquitetura para Agrupamento de Workflows por Similaridade*. Projeto de Graduação apresentado ao Curso de Engenharia de Computação e Informação da Escola Politécnica, Universidade Federal do Rio de Janeiro, Universidade Federal do Rio de Janeiro

VO, H. T., BRONSON, J., SUMMA, B., *et al*., 2011, "Parallel Visualization on Large Clusters using MapReduce". *2011 IEEE Symposium on Large Data Analysis and Visualization (2011)*, pp. 81–88

WOZNIAK, J., ARMSTRONG, T., MAHESHWARI, K., *et al*., 2012, "Turbine: A distributed-memory dataflow engine for extreme-scale many-task applications". In: *Proceeding of 1st International workshop on Scalable Workflow Enactment Engines and TechnologiesSIGMOD*, Scottsdale, AZ, EUA.

Paraview, 2009, *Paraview*, http://www.paraview.org.

ParaView-3.10.1, 2011, *ParaView open-source, multi-platform data analysis and visualization application*.

. Amazon Elastic Compute Cloud., 2012 Disponível em: http://aws.amazon.com/ec2/. Acesso em: 26 Feb 2013.

. VisTrailsWiki., 2012 Disponível em: http://www.vistrails.org/index.php/Main_Page. Acesso em: 25 Feb 2013.

. Python Programming Language – Official Website. Disponível em: http://www.python.org/. Acesso em: 3 Mar 2013.

. scp - FreeBSD. Disponível em: http://nixdoc.net/man-pages/FreeBSD/scp.1.html#HISTORY. Acesso em: 3 Mar 2013.

. UDDI | Online community for the Universal Description, Discovery, and Integration. Disponível em: http://uddi.xml.org/. Acesso em: 3 Mar 2013.

. Web Service Definition Language (WSDL). Disponível em: http://www.w3.org/TR/wsdl. Acesso em: 3 Mar 2013.

. Texas Advanced Computing Center - Home. Disponível em: http://www.tacc.utexas.edu/. Acesso em: 1 Mar 2013.

. SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). Disponível em: http://www.w3.org/TR/soap12-part1/#intro. Acesso em: 1 Mar 2013.

. SOAP - Wikipedia, the free encyclopedia. Disponível em: http://en.wikipedia.org/wiki/SOAP. Acesso em: 1 Mar 2013.

. ZK. Disponível em: http://www.zkoss.org/. Acesso em: 25 Feb 2013.

. VisTrails, Inc. - Maya. Disponível em: http://www.vistrails.com/maya.html. Acesso em: 27 Feb 2013.

. CGLX. Disponível em: http://vis.ucsd.edu/~cglx/. Acesso em: 27 Feb 2013.

. Texas Advanced Computing Center - DisplayCluster. Disponível em: http://www.tacc.utexas.edu/tacc-software/displaycluster. Acesso em: 27 Feb 2013.

. PROV Model Primer. Disponível em: http://www.w3.org/TR/prov-primer/. Acesso em: 27 Feb 2013.

. The Open Provenance Model. Disponível em: http://openprovenance.org/. Acesso em: 27 Feb 2013.

. Wikipédia, a enciclopédia livre. Disponível em: http://pt.wikipedia.org/. Acesso em: 27 Feb 2013.

. SAGE. Disponível em: http://www.sagecommons.org/. Acesso em: 26 Feb 2013.

. Using gRAVI Services in Taverna - Globus. Disponível em: http://dev.globus.org/wiki/Using_gRAVI_Services_in_Taverna. Acesso em: 25 Feb 2013.

. Sample Workflows — Kepler. Disponível em: https://kepler-project.org/users/sample-workflows. Acesso em: 25 Feb 2013.

. fhorta / prov-vis — Bitbucket. Disponível em: https://bitbucket.org/fhorta/prov-vis. Acesso em: 25 Feb 2013.