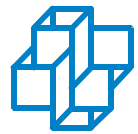
Introdução a Workflows Científicos Paralelos e Distribuídos

Luiz M. R. Gadelha Jr.

CENAPAD-RJ, LNCC

9 de junho de 2014



Laboratório Nacional de Computação Científica

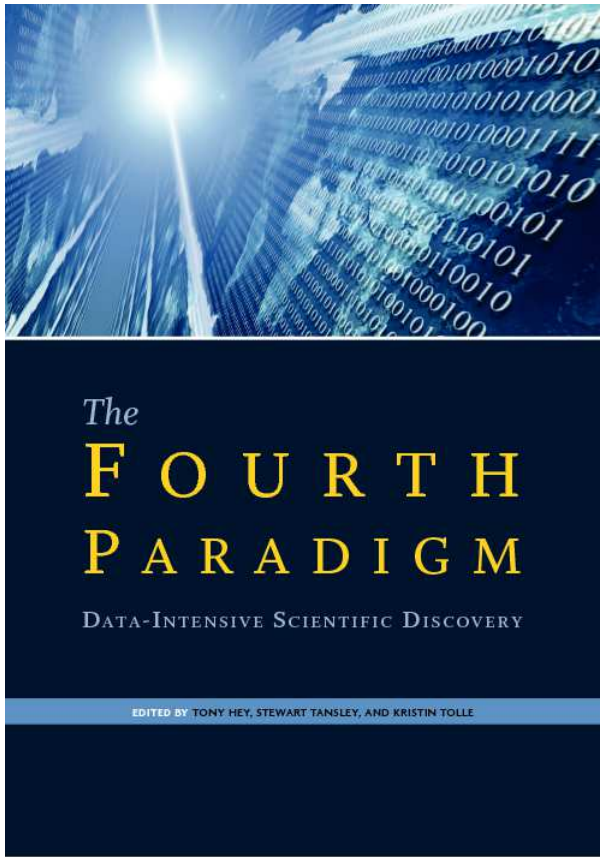
Agenda

Experimentos Científicos Computacionais

- Uma parte crescente da ciência e tecnologia se apoia em experimentos computacionais.
- Características destes experimentos:
  - grande quantidade de tarefas computacionais;
  - grande quantidade de dados;
  - dados armazenados em formatos diversos;
  - utilização de computação paralela e distribuída.

R. Kouzes et al. The Changing Paradigm of Data-Intensive Computing. *IEEE Computer*, 42(1):26-34, 2009.

Quarto Paradigma Científico



- Paradigmas anteriores: empírico, teórico e computacional (simulação).
- Habilitado por instrumentos científicos e simulações que geram grandes massas de dados.
- O quarto paradigma é baseado na análise e exploração destas massas de dados. Também chamado de e-Ciência.

http://research.microsoft.com/en-us/collaboration/fourthparadigm/

Introdução

- Experimentos científicos computacionais (ou *in silico*) atuais atingiram uma escala na qual:
  - é difícil tratá-los com recursos de apenas uma instituição;
  - é difícil gerenciá-los "manualmente" (scripts);
  - é difícil analisá-los em função da quantidade de dados gerada.
- Estes experimentos são intensivos em termos de **processamento** e em termos de **dados**.

Introdução

- A realização destes experimentos requer o compartilhamento de recursos computacionais distribuídos.
- Computação em grade permite ganhos de escala em termos de processamento.
- Gerência de dados em grades é uma área em desenvolvimento.

Introdução

- Problemas em gerência de dados distribuída:
  - armazenamento, acesso e replicação dados: SRB, iRODS;
  - gerência de workflows: Kepler, Swift (Globus), Taverna;
  - gerência de metadados: Chimera;
  - processamento de consultas de alto nível: OGSA DAI/DQP.

A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *Journal of Network and Computer Applications*, 23:187-200, 2001.

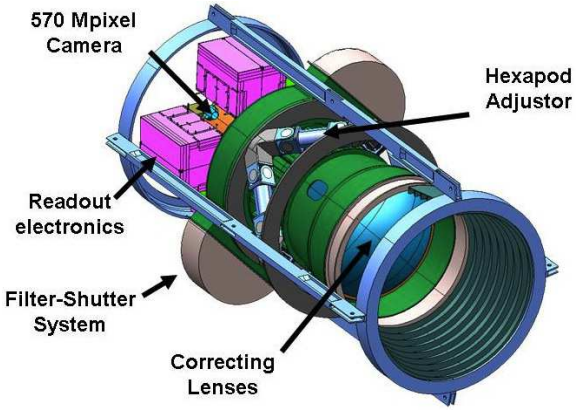
E. Pacitti, P. Valduriez, M. Mattoso. Grid Data Management: Open Problems and New Issues. *Journal of Grid Computing*, 5(3):273-281, 2007.

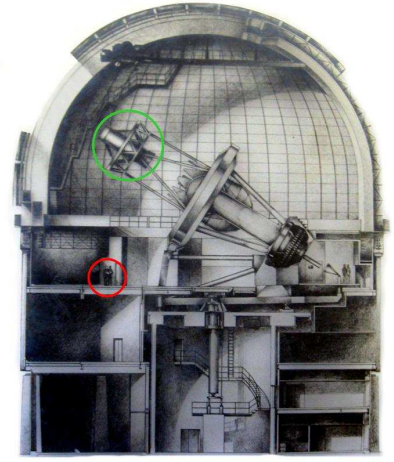
Introdução

- Exemplos:
  - Modelos climáticos mais recentes geram 8TB para uma simulação de 100 anos com resolução de 100km, e 8PB para resolução de 3km.
  - *Large Hadron Collider* gera 10PB de dados brutos por dia 15PB de dados tratados por ano.

R. Kouzes et al. The Changing Paradigm of Data-Intensive Computing. *IEEE Computer*, 42(1):26-34, 2009.

Exemplo: Dark Energy Survey (DES)

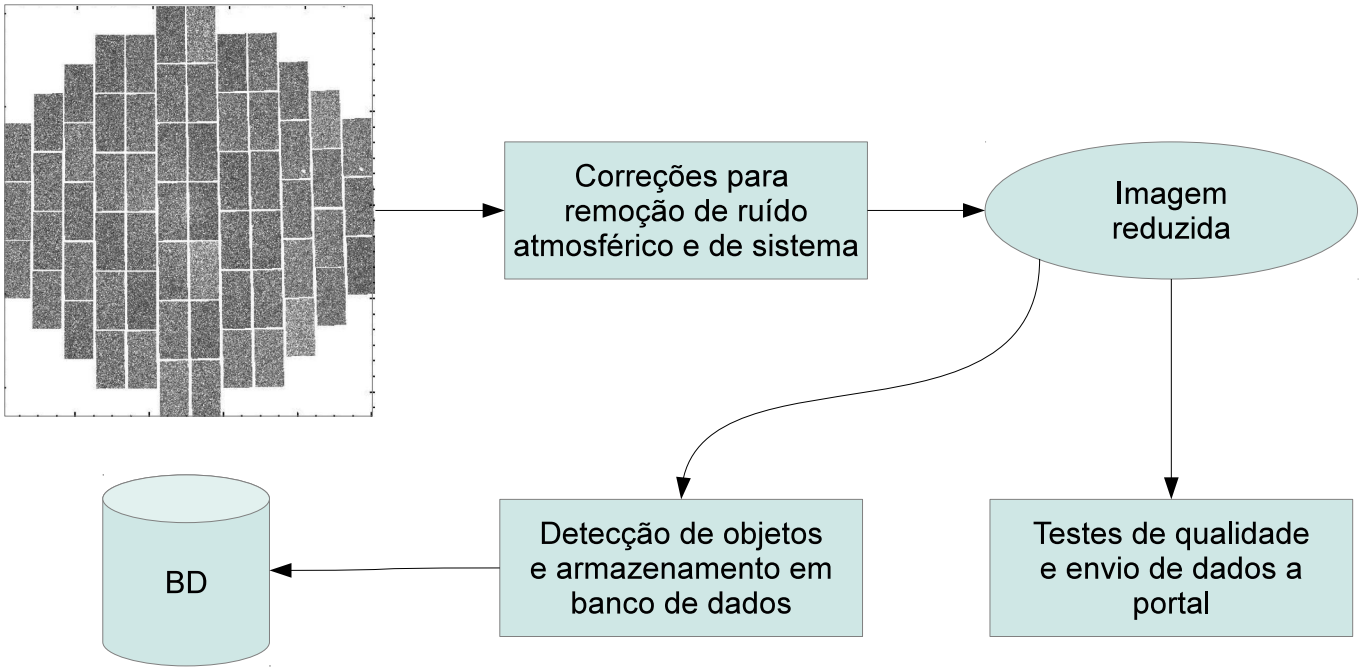




- Câmera de 570 megapixels.
- 400 imagens (6,6TB) por noite.

Fonte: Dark Energy Survey (`http://www.darkenergysurvey.org`)

Exemplo: Workflow do DES

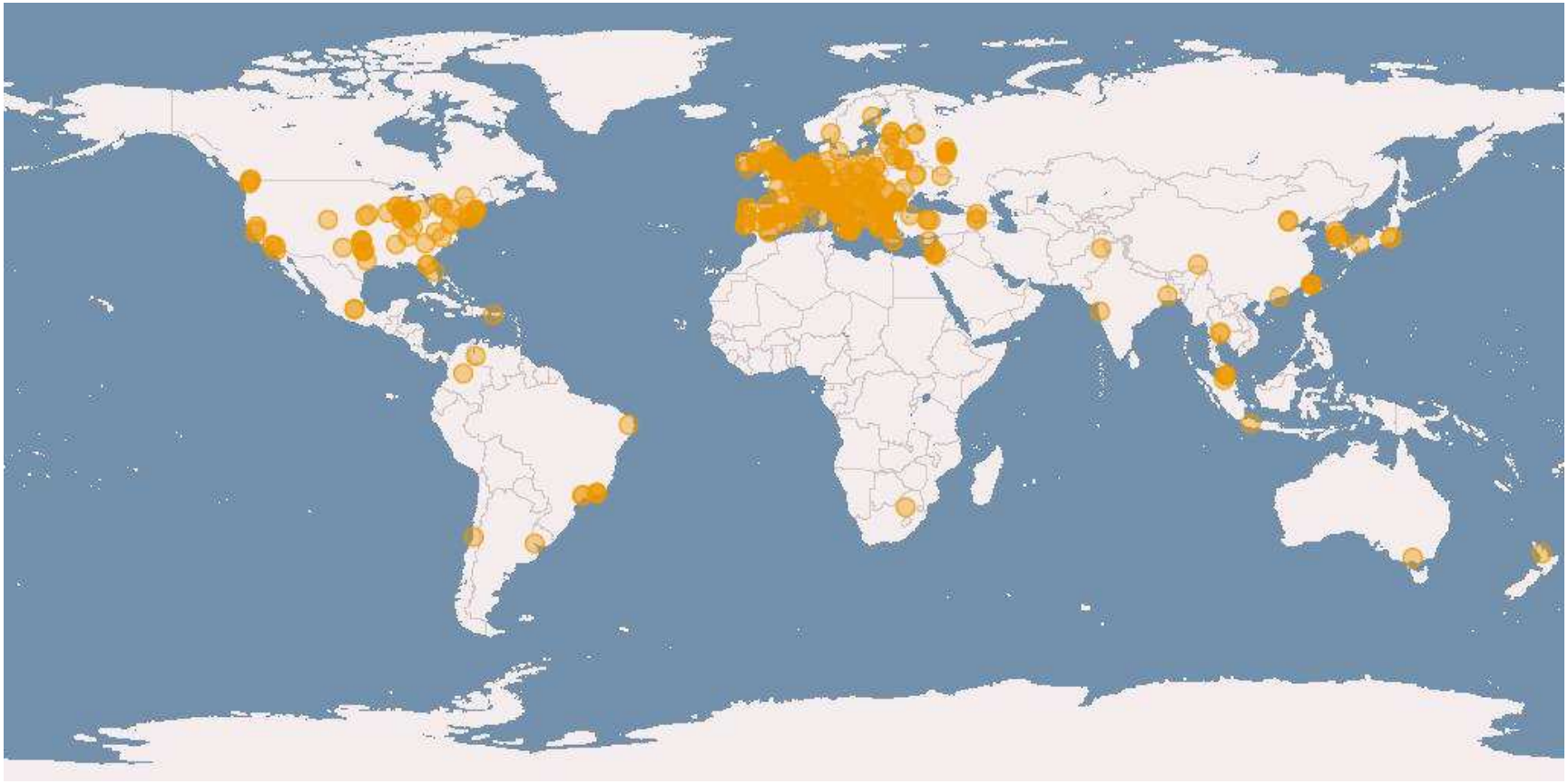


K. Kotwani et al. The Dark Energy Survey Data Management System as a Data Intensive Science Gateway. *Proc. of the 8th International Workshop on Middleware for Grids, Clouds and e-Science* (MGC 2010). ACM, 2010.

Exemplo: Large Hadron Collider (LHC)

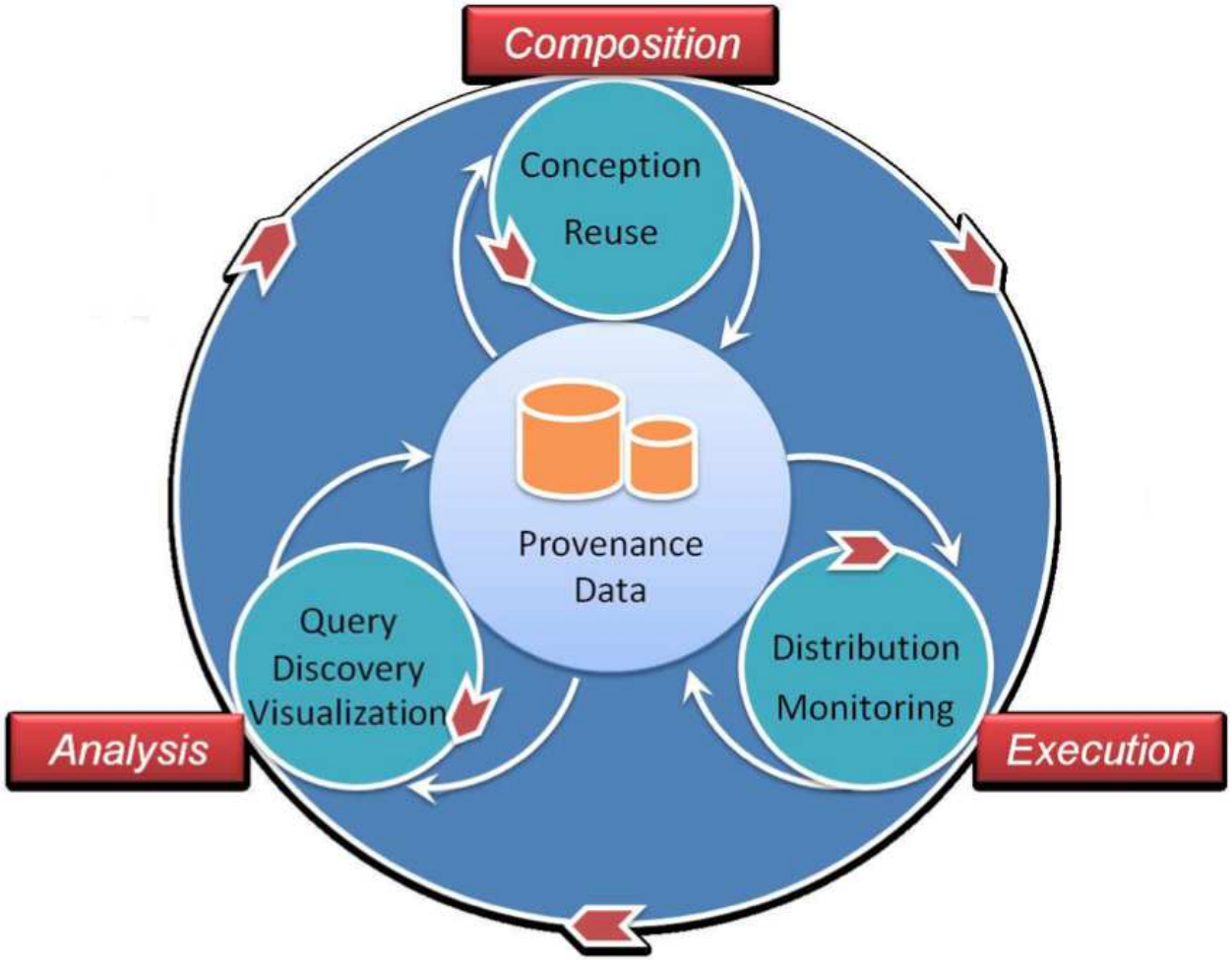
- Colisor de partículas de 27km de circunferência localizado na fronteira da França e Suíça.
- Projeto colaborativo envolvendo cerca de 100 países e 10.000 cientistas.
- Composto por seis detectores que geram cerca de 15PB de dados por ano.
- LHC Computing Grid (LCG):
  - Estrutura de computação distribuída organizada de forma hierárquica, para execução de diferentes partes do workflow do experimento:
    - Tier 0: processamento de dados brutos dos detectores,
    - Tier 1: reconstrução e pré-processamento,
    - Tier 2: simulação e análise.

Exemplo: Mapa do LCG



Fonte: CERN (`http://gstat-prod.cern.ch`)

Ciclo de Vida de Experimentos Científicos



M. Mattoso, C. Werner, G. Travassos, V. Braganholo, E. Ogasawara, D. Oliveira, S. Cruz, W. Martinho, and L. Murta, Towards supporting the life cycle of large scale scientific experiments. *International Journal of Business Process Integration and Management* 5(1):79–92, 2010.

Requisitos de Experimentos Científicos Computacionais

- **Abstração**. Deve ser possível ao cientista especificar o experimento em alto nível, sem precisar detalhar aplicações específicas e onde elas serão executadas.
- **Reprodutibilidade**. O experimento pode ser reproduzido por terceiros de forma independente.

  D. Koop et al. A Provenance-Based Infrastructure for Creating Executable Papers. *Proceedings of the ICCS*, 2011.
- **Reutilização**. Deve ser possível reutilizar especificações de experimentos realizados previamente para concepção de novos experimentos.

  Exemplo: MyExperiment (`http://www.myexperiment.org`).
- **Escalabilidade**. O experimento deve ser resiliente ao aumento do número de tarefas e da quantidade de dados.

Experimentos Científicos Computacionais em Larga Escala

- A execução deve ser escalável, ou seja, resiliente ao aumento do número de tarefas e da quantidade de dados.
- Experimentos em larga escala normalmente requerem o uso de computação paralela e distribuída:
  - **Computação paralela**. Utilização de múltiplos processadores de forma concorrente para reduzir o tempo de execução de atividades.

    I. Foster. Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering. Addison-Wesley, 1994.
  - **Computação em grade**. Compartilhamento de recursos computacionais heterogêneos e distribuídos.

    I. Foster e C. Kesselman, Eds. The Grid 2: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 2003.
  - **Computacão na nuvem**. Utilização de recursos de terceiros.

    D. Oliveira, F. Baião e M. Mattoso. Towards a Taxonomy for Cloud Computing from an e-Science Perspective. Cloud Computing, pp. 47–62. Springer, 2010.

Escalabilidade de Experimentos Científicos Computacionais

- O tempo total de execução de uma computação paralela e distribuída é dado por:

$$t = t_{\text{processando}} + t_{\text{comunicando}} + t_{\text{ocioso}}$$

- Existem diversas ferramentas para análise de performance de aplicações de forma isolada.
- Um desafio é realizar o mesmo tipo de análise para experimentos científicos computacionais, compostos por muitas aplicações.
- A escalabilidade depende da orquestração da execução de tarefas e gerência de dados de forma eficiente.

Workflows Científicos

- Um **workflow científico** consiste da especificação de um conjunto de aplicações científicas a serem executadas e suas dependências mútuas.
- Segue um ciclo de vida análogo ao dos experimentos científicos computacionais:
  - Composição, representação e modelagem de dados.
  - Mapeamento e execução.
  - Coleta de metadados e proveniência.
- Um **sistema de gerência de workflows científicos** (SGWC) permite gerenciar o ciclo de vida de workflows científicos.

E. Deelman et al. Workflows and e-Science: An overview of workflow system features and capabilities. *Future Generation Computer Systems* 25(1):528-540, 2009.

Exemplo: Laboratório de Bioinformática, LNCC

Sequenciador Roche 454 FLX gera 12GB a 15GB por rodada de sequenciamento.
Dados processados por várias aplicações:

- filtragem por qualidade de dados gerados pelo sequenciador.
- formatação de dados.
- comparação das sequências com bases de dados conhecidas.

Exemplo: Laboratório de Bioinformática, LNCC

Laboratório Nacional de Computação Científica

Workflows Científicos

- **Sistemas de gerência de workflows científicos** (SGWC) visam a automação de experimentos científicos computacionais:
  - escalonamento de tarefas baseado em dependências de dados;
  - fluxo de dados entre tarefas;
  - execução paralela de tarefas independentes;
  - escalonamento de tarefas em ambientes de computação de alto desempenho;
  - gerência e consulta de dados de proveniência.

Composição de Workflows Científicos

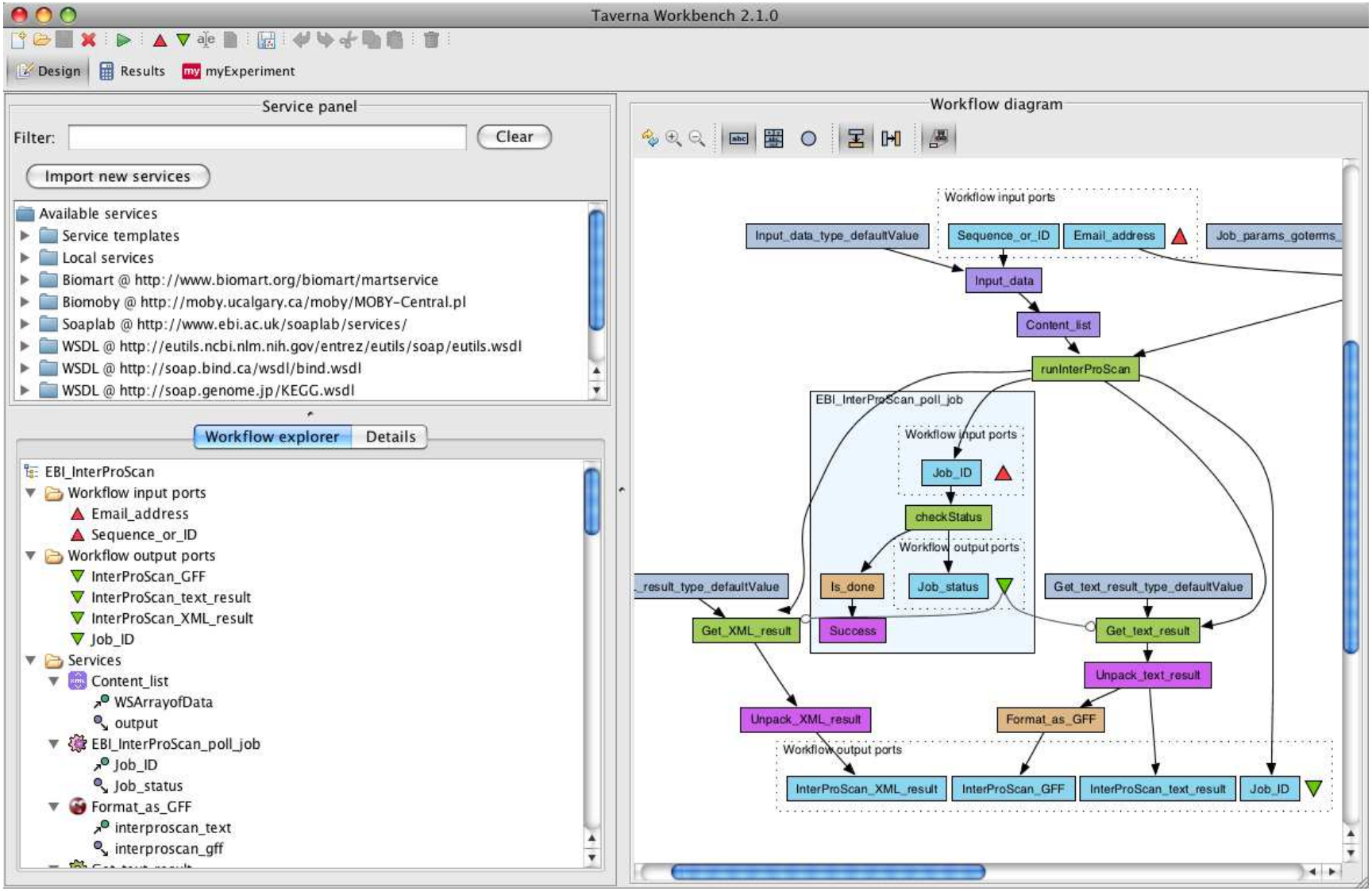
- Etapa em que são definidas as aplicações componentes e as dependências de dados.
- Níveis de especificação:
  - Abstrato: tarefas abstratas, descritas por funcionalidade geral (p. ex., comparação de seqüências).
  - Concreto: aplicações científicas e conjuntos de dados específicos.
- Representação:
  - Textual: linguagem de programação.
  - Gráfica: interface gráfica onde nós são tarefas e arestas são dependências.

Exemplo: Composição Textual

```
type messagefile;
type countfile;

app (countfile t) countwords (messagefile f) {
     wc "-w" @filename(f) stdout=@filename(t);
}

string inputNames = "foreach.1.txt foreach.2.txt foreach.3.txt";

messagefile inputfiles[] <fixed_array_mapper;files=inputNames>;

foreach f in inputfiles {
  countfile c <regexp_mapper; source=@f, match="(.*)txt",
               transform="\\1count">;
  c = countwords(f);
}
```

Workflow especificado em Swift (`http://www.ci.uchicago.edu/swift`)

Exemplo: Composição Gráfica

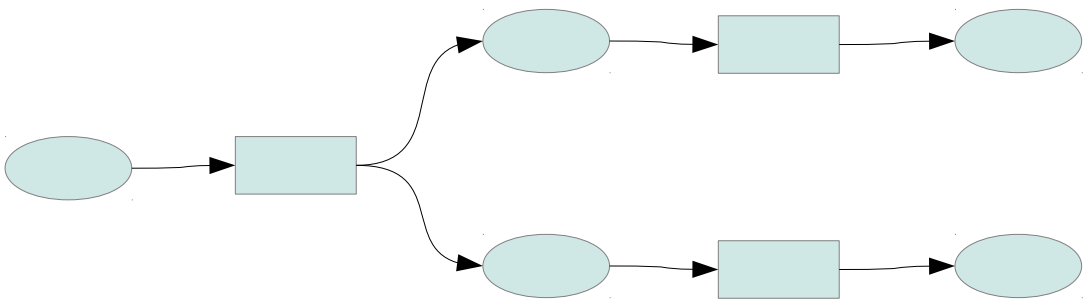


Fonte: Taverna (`http://www.taverna.org.uk`)

Composição de Workflows Científicos: Padrões

- Foram identificados 43 padrões de composição de workflows.
- Exemplos:
  - Seqüência:

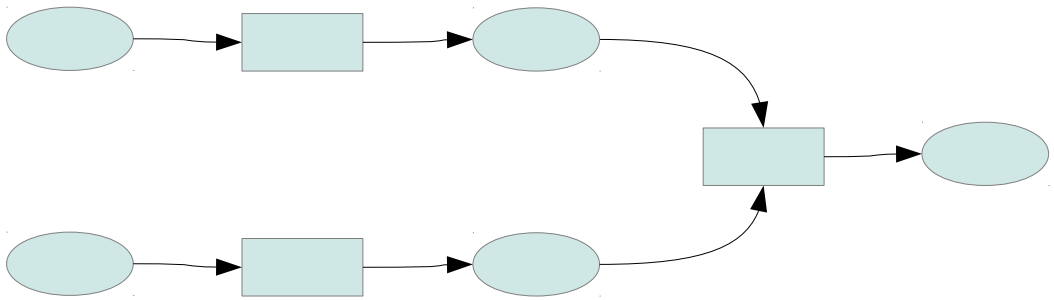
  - Bifurcação paralela:



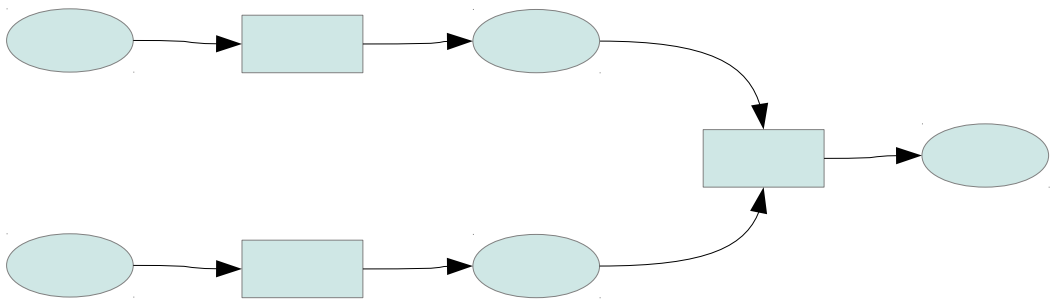
- W. van der Aalst et al. Workflow Patterns. *Distributed and Parallel Databases* 14(1):5-51, 2003.
- N. Russell et al. Workflow Control-Flow Patterns: A Revised View. BPM Center Report BPM-06-22, 2006.
- Workflow Patterns (http://www.workflowpatterns.com).

Composição de Workflows Científicos: Padrões

- Exemplos:
  - Sincronização:

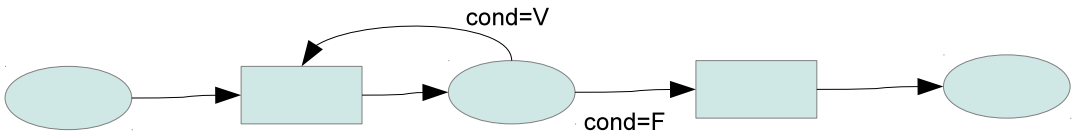


  - Fusão:

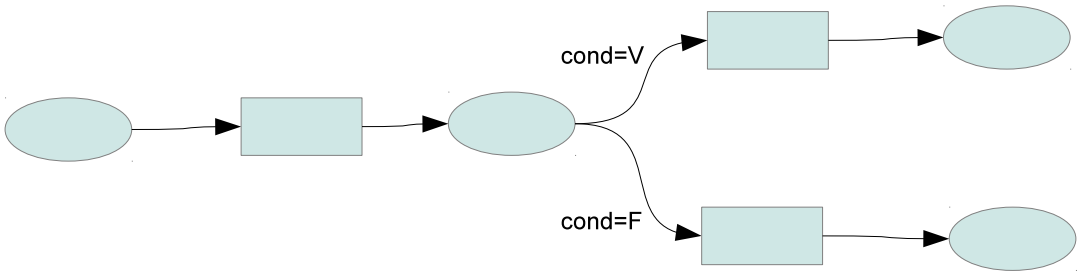


Composição de Workflows Científicos: Padrões

- Exemplos:
  - Laço:



  - Escolha exclusiva:



Mapeamento e Execução de Workflows Científicos

- Etapa em que são definidas as aplicações componentes concretas e os locais de execução das mesmas.
- Escalonamento da execução das aplicações componentes.
  - Interno.
  - Externo.
- Modelo de execução.
  - Execução local.
  - Execução remota.
- Tolerância a falhas.
- Redundância.
- Execução adaptativa.

Coleta de metadados e proveniência

- Coleta de eventos do ciclo de vida do workflow:
  - Mudanças e evolução da especificação.
  - Consumo e produção de dados por aplicações componentes executadas.
- Gerência de metadados relacionados ao domínio científico (semântica do experimento).
- Serviço de consulta a metadados e proveniência.
- Aplicações: reprodutibilidade, verificação, análise.

Exemplos de SGWCs: Taverna

- Aplicações componentes podem ser serviços web.
- Integração com o portal de reutilização de workflows MyExperiment.
- Especificação de workflows através de interface gráfica.
- Suporte a coleta e consultas de proveniência.
- Popular na área de bioinformática.

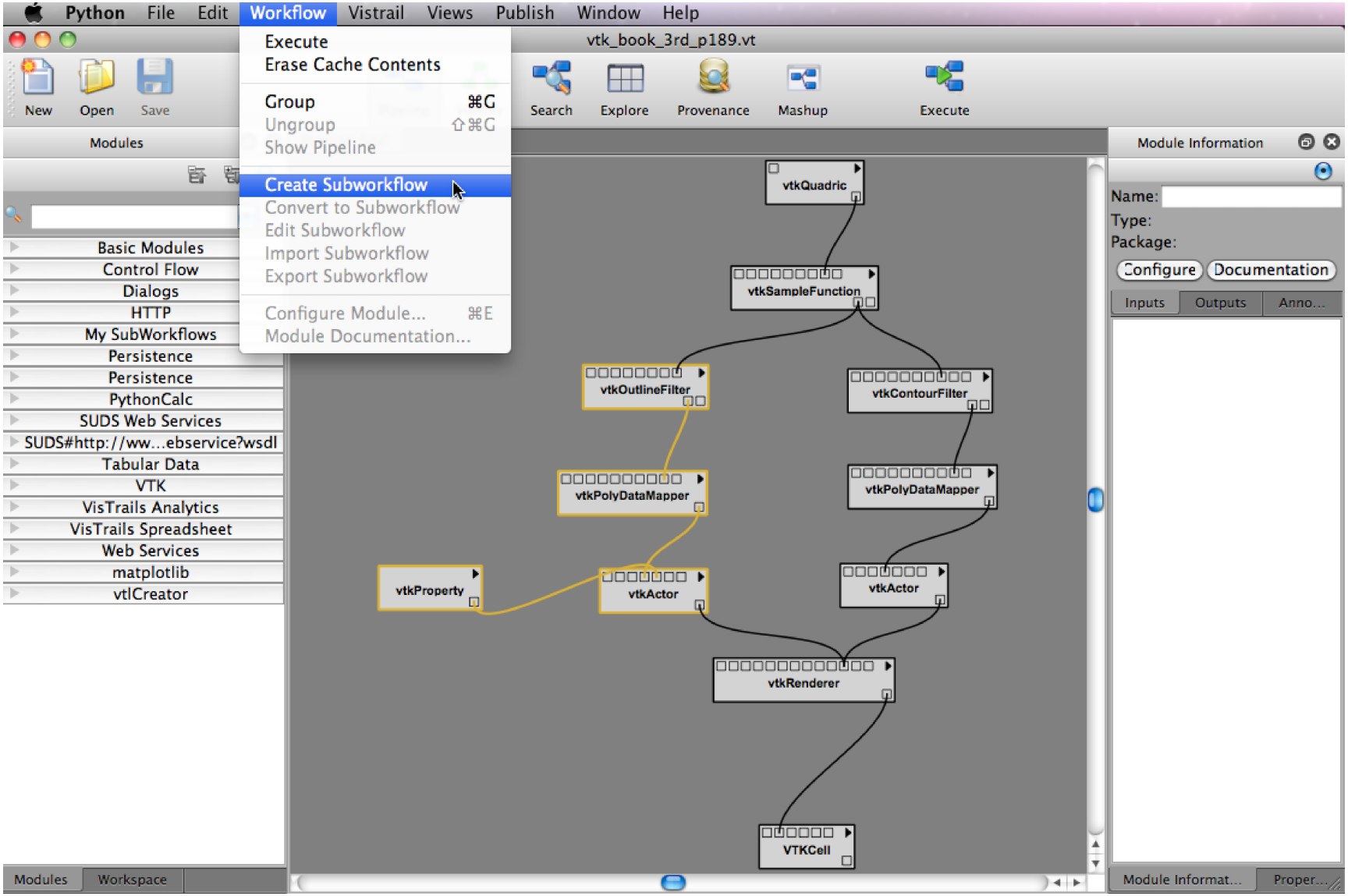
Taverna (`http://www.taverna.org.uk`)

Exemplos de SGWCs: Vistrails

- Suporte a versionamento e gerência da evolução de workflows.
- Especificação de workflows através de interface gráfica.
- Suporte a coleta e consultas de proveniência.
- Popular na área de visualização científica.

Vistrails (`http://www.vistrails.org`)

Exemplos de SGWCs: Vistrails



Fonte: Vistrails (`http://www.vistrails.org`)

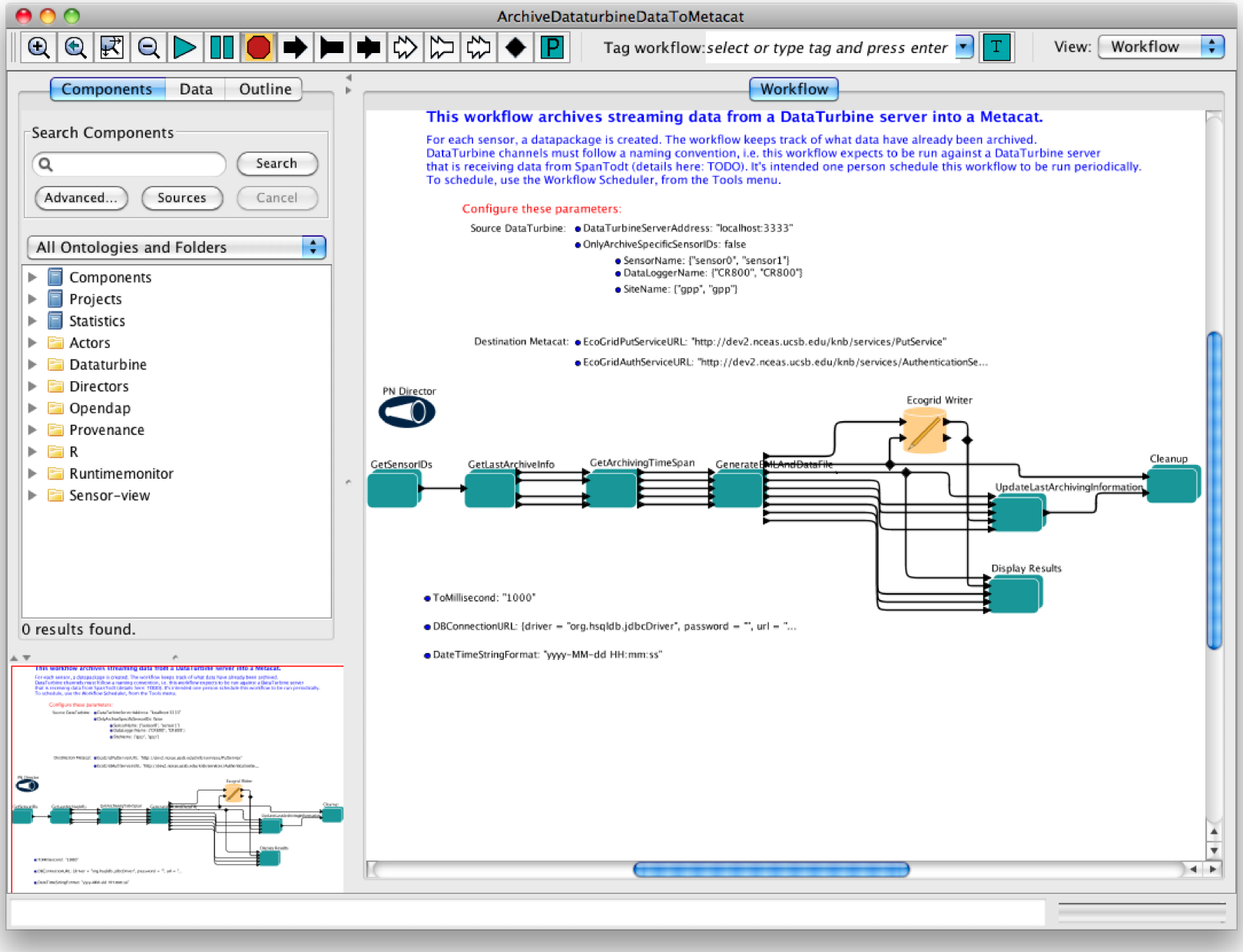
Laboratório Nacional de Computação Científica

Exemplos de SGWCs: Kepler

- Suporte a aplicações disponibilizadas através de serviços web.
- Especificação de workflows através de interface gráfica.
- Suporte a coleta e consultas de proveniência.
- Integração com ferramentas populares, como o R.

Kepler (`https://kepler-project.org`)

Exemplos de SGWCs: Kepler



Proveniência

- Informações de **proveniência** descrevem o histórico de concepção e execução de um experimento.
- Proveniência **prospectiva** captura a especificação e sua evolução.
- Proveniência **retrospectiva** descreve de forma detalhada a geração de um conjunto de dados:
  - processos que o derivaram;
  - outros conjuntos de dados utilizados por estes processos;
  - agentes envolvidos na derivação.

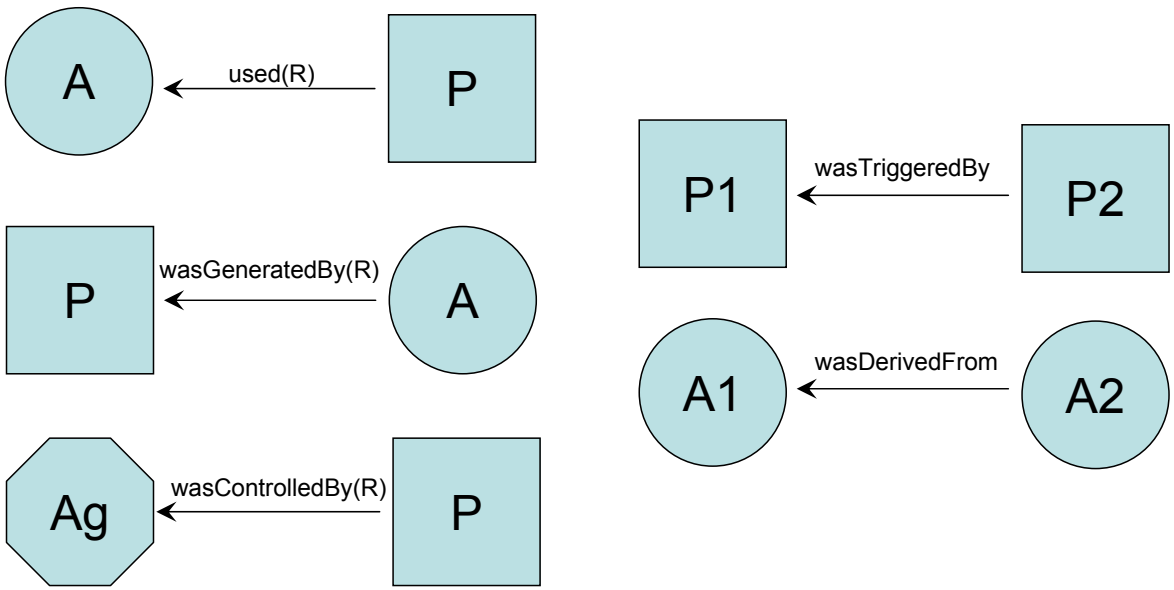
S. B. Davidson and J. Freire, Provenance and scientific workflows: challenges and opportunities. *Proc. of the International Conference on Management of Data* (SIGMOD 2008), pp. 1345–1350. ACM, 2008.

Proveniência de Dados

- Sistemas de proveniência podem ser classificados conforme:
  - aplicação;
  - conteúdo;
  - representação;
  - armazenamento;
  - disseminação.
- Não existe padrão definido para armazenamento e representação de proveniência: XML, BD, arquivos de log.
- Exemplos de sistemas com suporte a proveniência: Kepler, Swift, Taverna.

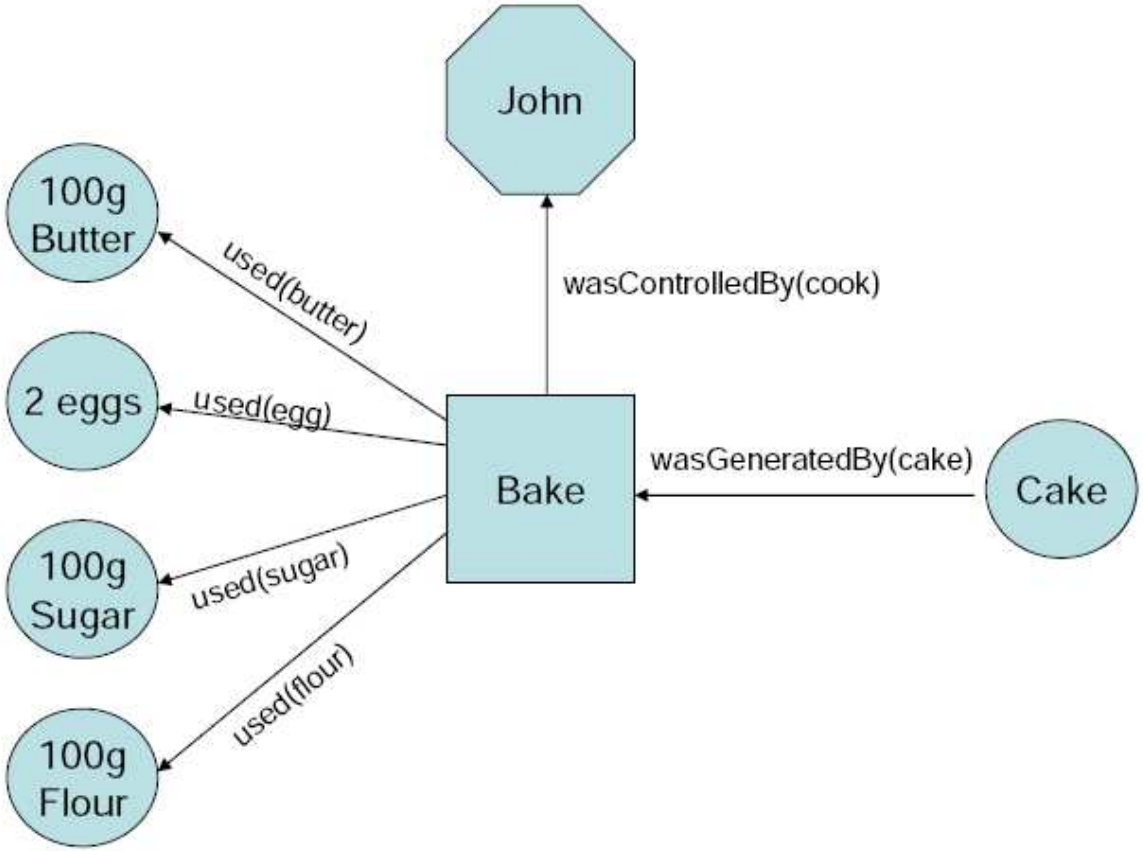
Y. L. Simmhan, B. Plale, D. Gannon. A survey of data provenance in e-science. *ACM SIGMOD Record*, 34(3):31-36, 2005.

Open Provenance Model



L. Moreau et al. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756, 2011.

Open Provenance Model



L. Moreau et al. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756, 2011.

Proveniência de Dados

- Aplicações de dados de proveniência:
  - re-execução inteligente de experimentos;
  - proteção à autoria;

    L. Gadelha, M. Mattoso. Kairos: An Architecture for Securing Authorship and Temporal Information of Provenance Data in Grid-Enabled Workflow Management Systems. *IEEE Fourth International Conference on e-Science* (e-Science 2008), pp. 597-602. 2008.

  - avaliação da qualidade de dados;
  - reproducibilidade de experimentos.

Swift

- Swift permite gerenciar workflows científicos em ambientes paralelos e distribuídos.
- Ambiente ideal para experimentação pois demonstradamente escala para 116.000 CPUs.
- É composto por:
  - Uma linguagem de alto nível para especificação de workflows científicos como scripts.
  - Ambiente de execução com suporte nativo a:
    - execução local,
    - execução paralela (p.ex. PBS, SGE),
    - execução distribuída (p.ex. Condor, Globus).
  - Sistema de gerência de proveniência.

M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford I. Raicu, Parallel Scripting for Applications at the Petascale and Beyond. *IEEE Computer*, 42(11):50-60, 2009.

M. Wilde, M. Hategan, J. Wozniak, B. Clifford, D. Katz, and I. Foster. Swift: A language for distributed parallel scripting. *Parallel Computing*, 37(9):634-652, 2011.

Swift

- Desenvolvimento:

- Colaboração com o LNCC no desenvolvimento do componente de gerência de proveniência.
- Aplicações: OOPS, SciColSim, MODIS, Labinfo, openModeller.

Swift

- Cenário típico: composição de várias tarefas (*many-task computing*).
- As tarefas podem ser:
  - fracamente acopladas (*bag of tasks*),
  - fortemente acopladas:
    - locais (p.ex. paralelismo com memória compartilhada),
    - distribuídas (p.ex. paralelismo com memória distribuída).

Swift

- Paralelismo implícito:
  - Toda expressão de um script é avaliada em paralelo, exceto quando há dependências de dados.
  - $\Rightarrow$ Não é possível assumir uma ordem de execução dessas expressões.
- Independência de localidade:
  - Os scripts não expressam transferências de dados e seleção de sítios de execução.

Swift: Linguagem de Especificação de Workflows

- Aplicações que compõem um workflow são associadas a funções (*app functions*) em um script Swift.
- Variáveis em um script Swift podem ser associadas a:
  - tipos de dados primitivos,
  - dados persistentes armazenados em arquivos,
  - tipos de dados compostos (coleções).
- Suporte a laços e controles condicionais de fluxo.

Swift: Linguagem de Especificação de Workflows

- Sintaxe semelhante ao C, Java.
- Estrutura geral de um script Swift:
  1. Declarações de tipos de dados.
  2. Declarações e atribuições de variáveis.
  3. Declarações e chamadas de funções associadas a aplicações.
  4. Declarações e chamadas de funções compostas.

Swift: Linguagem de Especificação de Workflows

- Semelhanças com paradigma funcional:
  - Computação orientada a avaliação de funções.
  - Chamadas sem *efeitos colaterais*.
  - *Avaliação preguiçosa* de expressões.
  - Variáveis de *atribuição única*.

Swift: Linguagem de Especificação de Workflows

Exemplo:

```
type messagefile;
type countfile;

app (countfile t) countwords (messagefile f) {
     wc "-w" @filename(f) stdout=@filename(t);
}

string inputNames = "foreach.1.txt foreach.2.txt foreach.3.txt";

messagefile inputfiles[] <fixed_array_mapper;files=inputNames>;

foreach f in inputfiles {
  countfile c <regexp_mapper; source=@f, match="(.*)txt",
               transform="\\1count">;
  c = countwords(f);
}
```

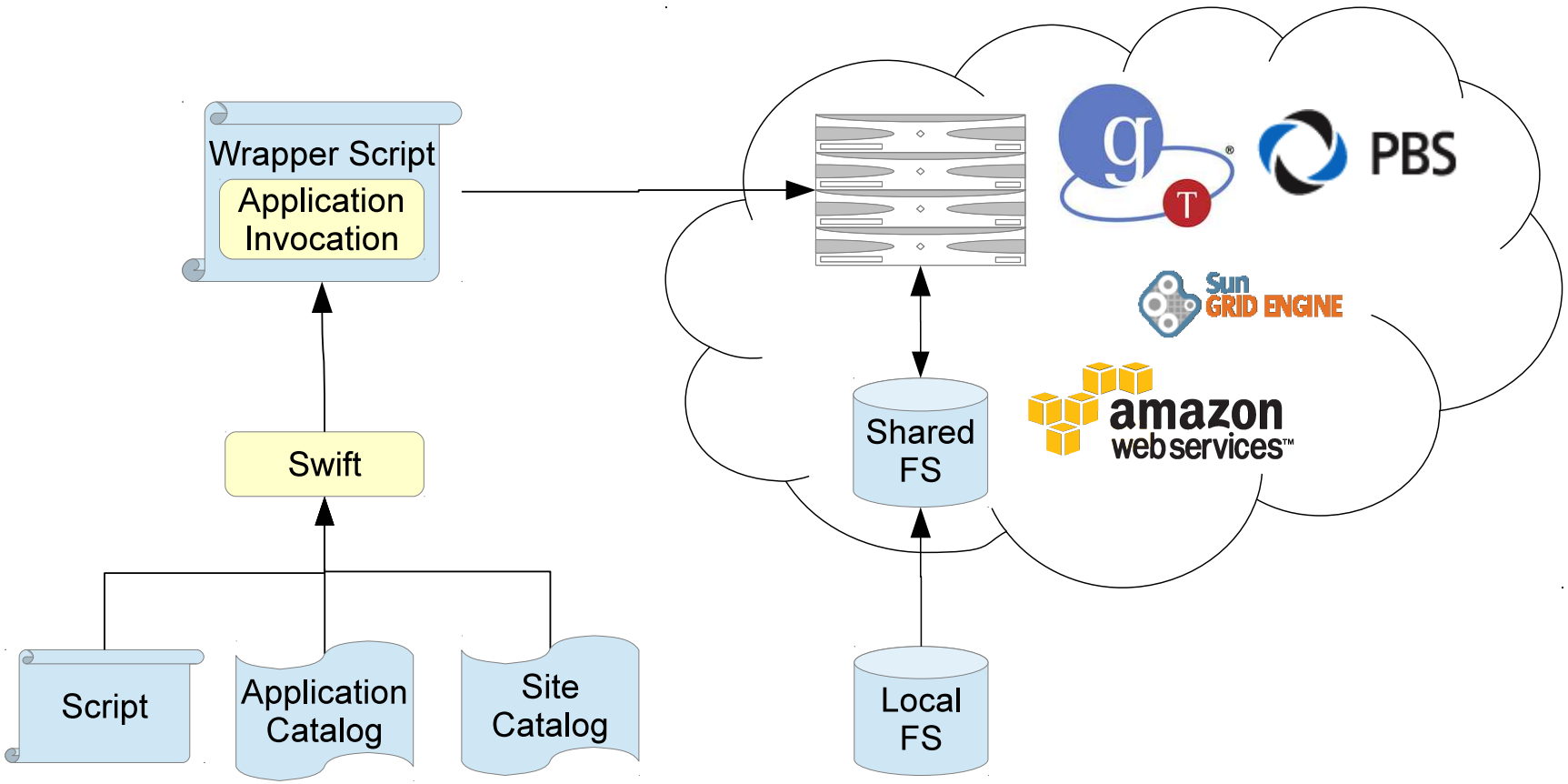
Swift: Ambiente de Execução

- Execução independente de localidade.
- Paralelização automática de execuções de aplicações componentes que não tenham dependências de dados.
- Heurística de balanceamento de carga das aplicações entre os recursos disponíveis.
  - Pontuação de desempenho proporcional à taxa de execução histórica de tarefas.
- Tolerância a falhas:
  - redundância,
  - resubmissão de execuções que falharam,
  - re-execução de script sem repetição de computações.

Swift: Ambiente de Execução

- Responsável submissão de execução de aplicações componentes e transferência de arquivos.
- Provedores de dados e de execução.
- Catálogo de aplicações componentes.
- Catálogo de recursos computacionais.

Swift: Ambiente de Execução



Swift: Ambiente de Execução

Catálogo de recursos computacionais:

```
<config>
  <pool handle="localhost" sysinfo="INTEL32::LINUX">
    <execution provider="local" url="none" />
    <workdirectory>/prj/prjssi/lgadelha/swiftwork</workdirectory>
    <profile namespace="karajan" key="jobThrottle">0.03</profile>
  </pool>

  <pool handle="sge-local">
    <execution provider="sge" url="none" />
    <profile namespace="globus" key="pe">threads</profile>
    <profile key="jobThrottle" namespace="karajan">6.23</profile>
    <profile namespace="karajan" key="initialScore">10000</profile>
    <filesystem provider="local" url="none" />
    <workdirectory>/prj/prjssi/lgadelha/swiftwork</workdirectory>
  </pool>
</config>
```

Swift: Ambiente de Execução

Catálogo de aplicações componentes:

```
localhost   echo     /bin/echo
localhost   cat      /bin/cat
localhost   ls       /bin/ls
localhost   grep     /bin/grep
localhost   sort     /bin/sort
localhost   paste    /bin/paste
localhost   cp       /bin/cp
localhost   touch    /bin/touch
localhost   wc       /usr/bin/wc
localhost   sleep    /bin/sleep
sge-local   cat      /bin/cat
```

MTCProv: Motivação

- Durante o *Third Provenance Challenge* apresentamos um modelo de proveniência retrospectiva para MTC e uma implementação no Swift.
- Problemas detectados:
  - Ausência de informações do domínio científico dificultava a interpretação dos dados de proveniência.
  - Dificuldade de escrita de consultas no modelo relacional: junções e fecho transitivo.

L. Gadelha, B. Clifford, M. Mattoso, M. Wilde, and I. Foster. Provenance Management in Swift. *Future Generation Computer Systems*, 27(6):775-780, 2011.

MTCProv: Motivação

- MTCProv é uma ferramenta de proveniência para workflows paralelos e distribuídos integrada ao Swift.
- Metodologia seguida:
  1. Levantamento de padrões de consulta.
  2. Modelagem de dados.
  3. Implementação de mecanismos de coleta.
  4. Implementação de interface de consulta.

MTCProv: Padrões de Consulta

- Padrões:
  - Atributo de entidade (EA).
  - Relacionamento direto (R).
  - Relacionamento transitivo (R^*).
  - Casamento de grafos (LGM).
  - Resumo de execução (RS).
    - Performance computacional (RRP).
    - Performance científica (RSP).
  - Comparação entre execuções (RCp).
  - Correlações entre execuções (RCr).

L. Gadelha, M. Mattoso, M. Wilde, I. Foster, Provenance Query Patterns for Many-Task Scientific Computations. *Proceedings of the 3rd USENIX Workshop on Theory and Applications of Provenance* (TaPP'11), 2011.

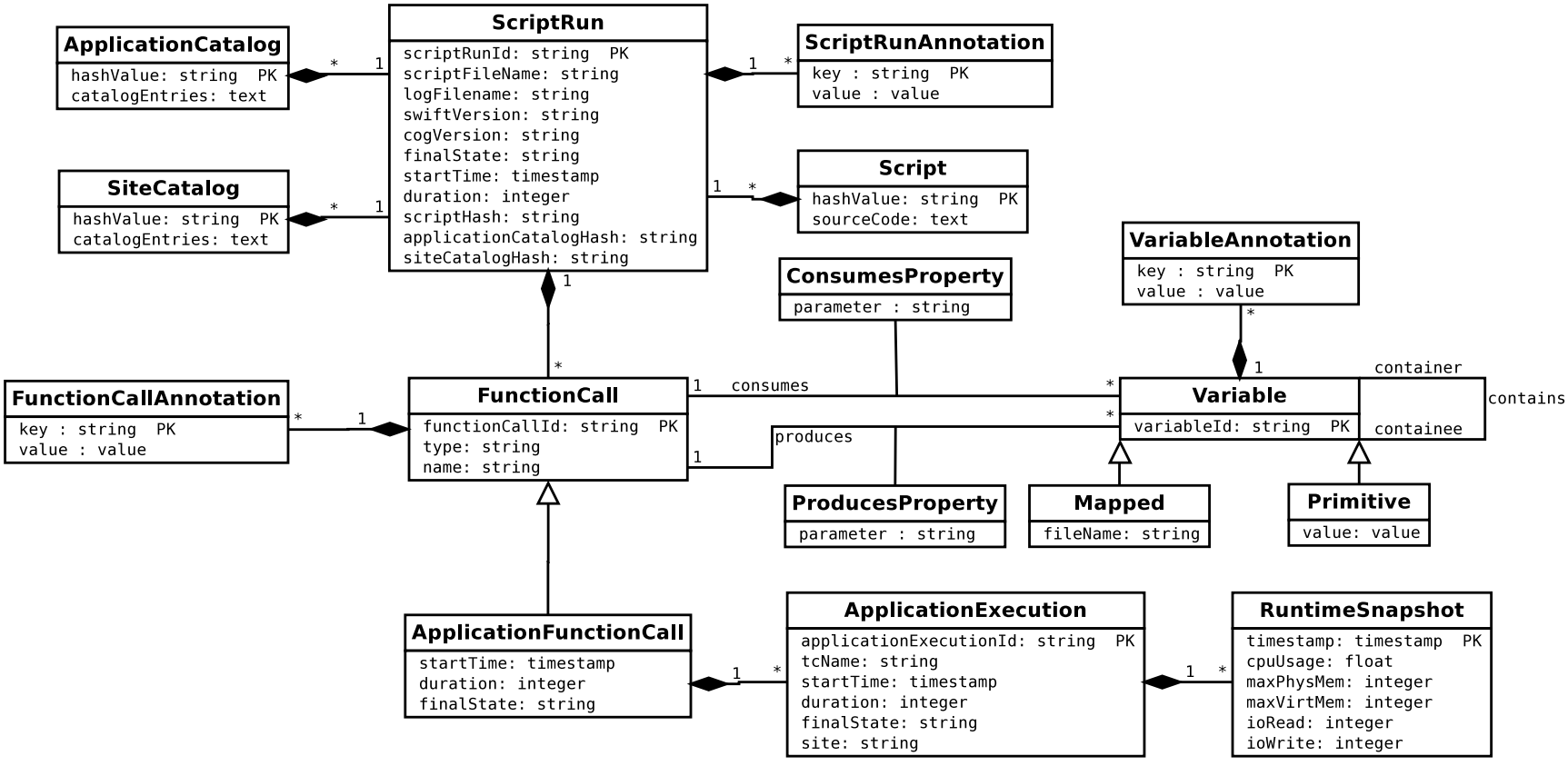
MTCProv: Padrões dos *Provenance Challenges*

Padrão	PC1/PC2									PC3				PC3 (Consultas Opcionais)													
	1	2	3	4	5	6	7	8	9	1	2	3	5	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
EA	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×	×
R	×	×	×		×	×	×	×	×	×	×	×	×		×	×	×	×	×	×	×	×	×	×	×	×	×	×
R*	×	×	×			×	×			×	×	×	×		×		×				×	×			×	×	×	×
LGM							×								×													
RS	×	×	×							×	×	×	×	×		×	×						×	×	×	×	×	
RCp					×	×	×	×	×						×			×	×	×	×	×						×
RCr				×																								

MTCProv: Modelagem de Dados

- Objetivos:
  1. Coletar informações sobre consumo e produção de artefatos e processos.
  2. Coletar informações sobre hierarquia entre artefatos de dados.
  3. Permitir o enriquecimento das informações de proveniência com anotações.
  4. Coletar informações sobre versionamento de workflows científicos e aplicações componentes.
  5. Coletar informações de tempo de execução das aplicações componentes.
  6. Prover uma interface de consulta amigável e útil para as informações de proveniência.

MTCProv: Modelagem de Dados



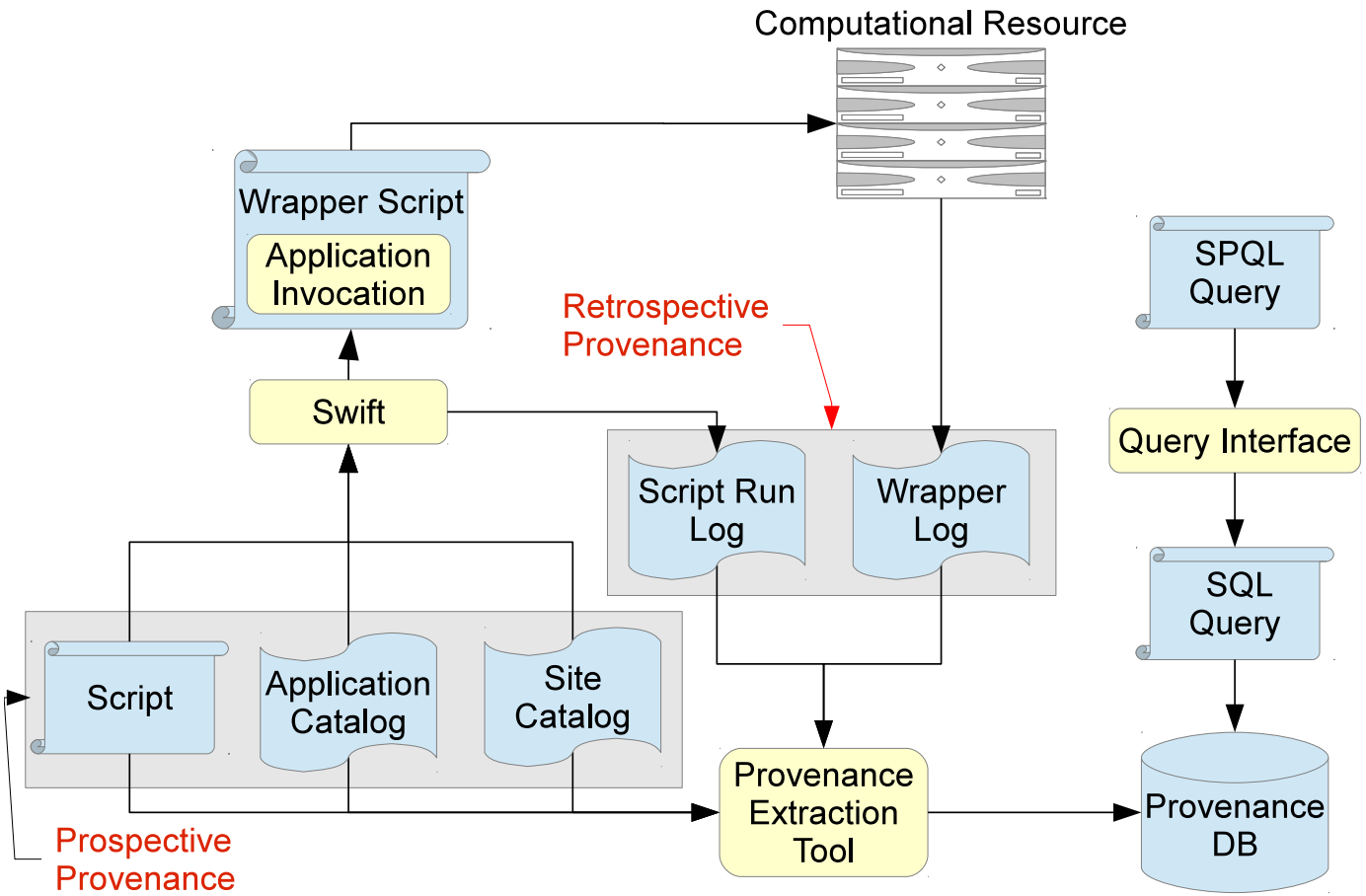
L. Gadelha, M. Wilde, M. Mattoso, and I. Foster. MTCProv: a practical provenance query framework for many-task scientific computing. *Distributed and Parallel Databases*. Springer, 2012.

Laboratório Nacional de Computação Científica

MTCProv: Captura e Armazenamento

- Informações de proveniência são extraídas dos logs gerados Swift a cada execução de script.
- Armazenamento em banco de dados relacional:
  - As restrições de se utilizar um esquema de dados fixo podem ser reduzidas com o uso de anotações.
  - Fecho transitivo pode ser calculado facilmente com funções recursivas nativas a partir do SQL:1999.
- Modelos de dados baseados em grafos requerem travessia frequente para recuperação de atributos em consultas de agregação.

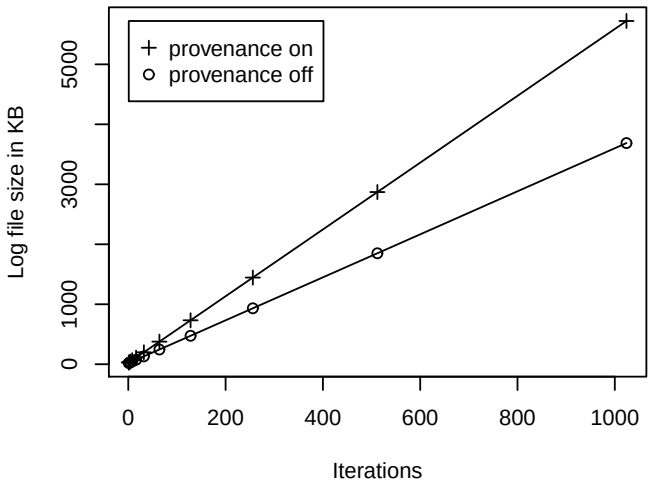
MTCProv: Captura e Armazenamento

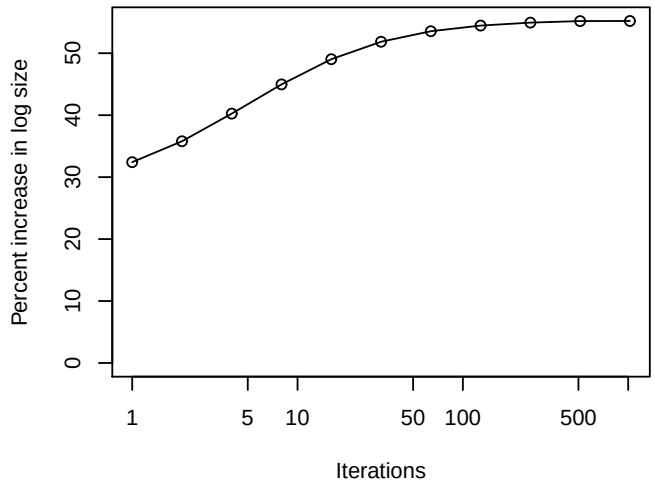


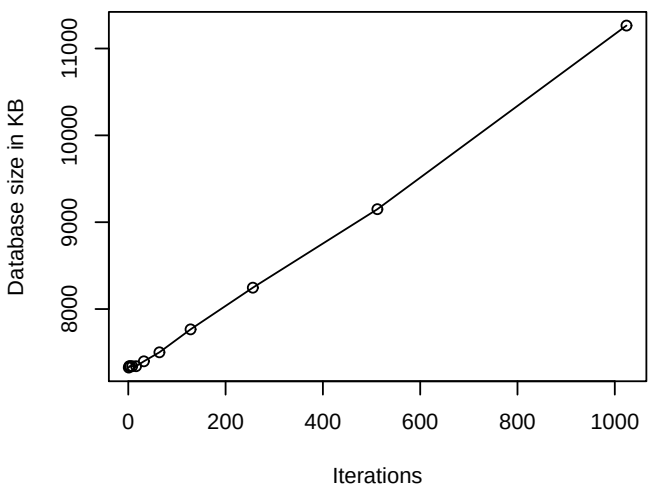
MTCProv: Captura e Armazenamento

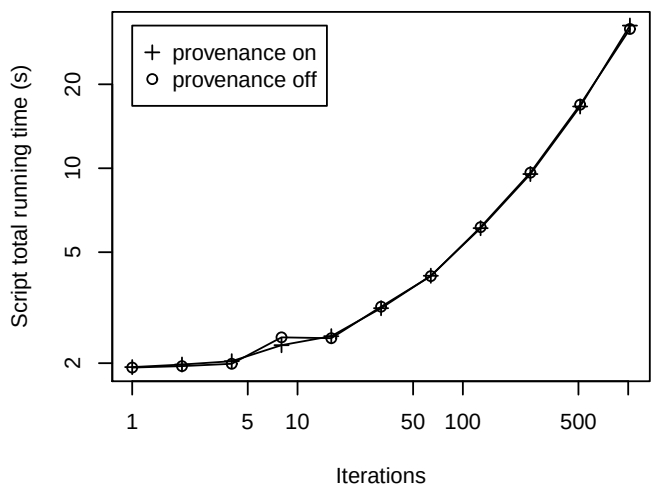
- Anotações podem ser coletadas por:
  - scripts *ad hoc* para um workflow científico,
  - scripts que iniciam a execução remota (*wrapper scripts*).
- Exemplos de anotações:
  - Parâmetros científicos não visíveis ao Swift.
  - Informações sobre segurança.

MTCProv: Impacto da Captura e Armazenamento



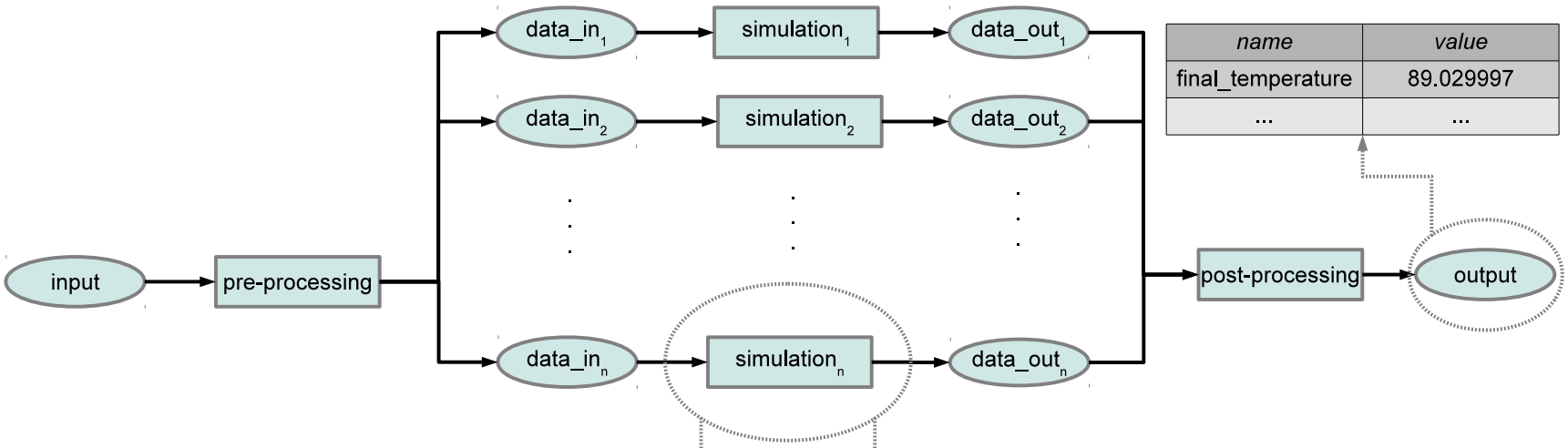






Laboratório Nacional de Computação Científica

MTCProv: Captura e Armazenamento de Anotações



name	value
final_temperature	89.029997
...	...

name	value
execution_site	pads.ci.uchicago.edu
version	1.15
...	...

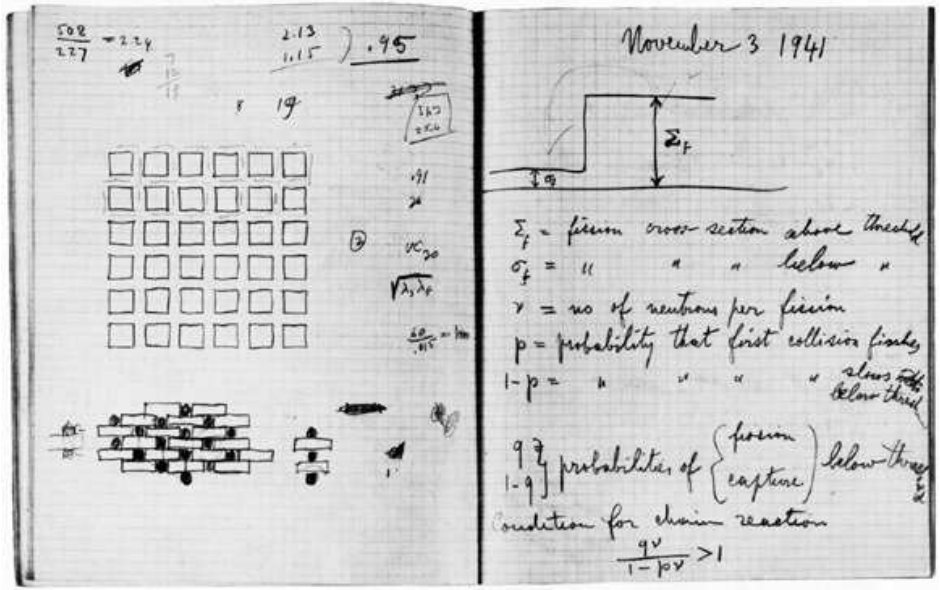
timestamp	mem_usage	cpu_load	io_read	io_write
1314104905	126342	99	230	0
1314104906	155739	100	123	1044
...	...	...	...	...
1314112105	394523	99	0	129

L. Gadelha, M. Wilde, M. Mattoso, and I. Foster. Exploring provenance in high performance scientific computing. Proc. Workshop on High Performance Computing Meets Databases (HPCDC'11), pp. 17–20, 2011.

MTCProv: Interface de Consulta

- Abstração de padrões de consultas comuns através de funções e procedimentos SQL.
- O padrão R^* é implementado com funções recursivas nativas do SQL no procedimento `ancestors`.
- Os padrões RCp e RCr são implementados pelo procedimento:
  - `compare_run`($\langle$ *lista de parâmetros* ou *chaves de anotações*$\rangle$) retorna uma tabela com os valores das anotações e parâmetros por execução de script.
- Foi implementada uma interface de consultas em Java/ANTLR que calcula automaticamente cláusulas `FROM` e expressões para junções.

MTCProv: Segurança



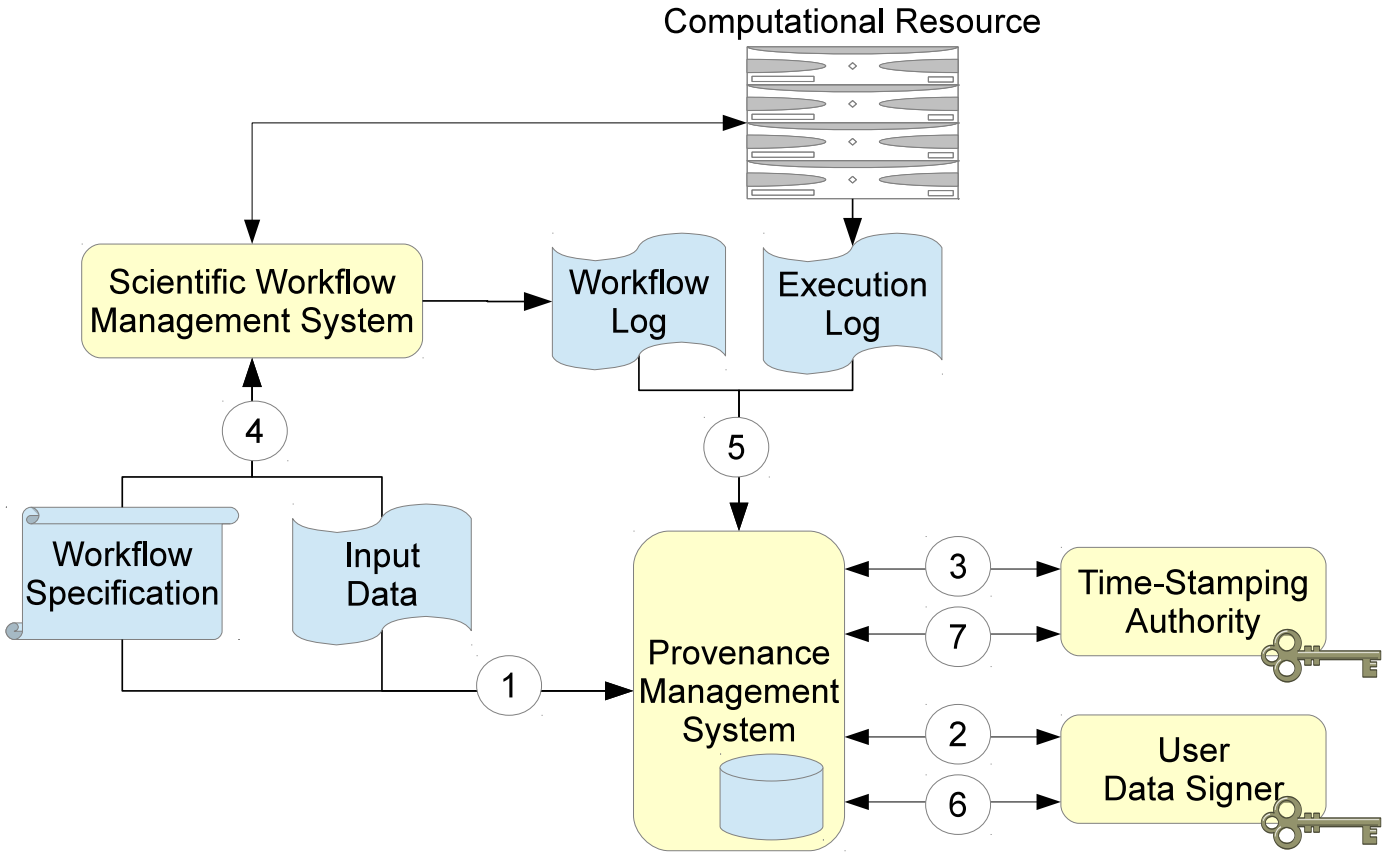
- Os registros de proveniência são análogos aos *cadernos de laboratório* dos experimentos de bancada.
  - plano do experimento;
  - parâmetros iniciais;
  - descrição dos resultados.

MTCProv: Segurança

- Recomendações para proteção de propriedade intelectual contida nos cadernos de laboratório:
  - "O caderno de laboratório é um dos elementos mais importantes do processo de patenteamento."
  - "... é possível perceber a importância de se manter um caderno de laboratório para certificar, e provar em tribunal se necessário, que o seu trabalho para uma invenção foi realizado anteriormente ao de outros."
  - "... as anotações devem ser assinadas e datadas por um terceiro."
  - "... após assinatura e datação, nenhuma modificação pode ser realizada."

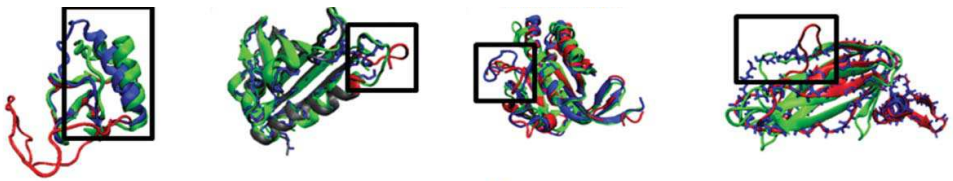
Guidelines for Maintaining a Lab Notebook, Los Alamos National Laboratory, 2012.

MTCProv: Segurança



L. Gadelha, M. Mattoso. Kairos: An Architecture for Securing Authorship and Temporal Information of Provenance Data in Grid-Enabled Workflow Management Systems. *IEEE Fourth International Conference on e-Science* (e-Science 2008), pp. 597-602. 2008.

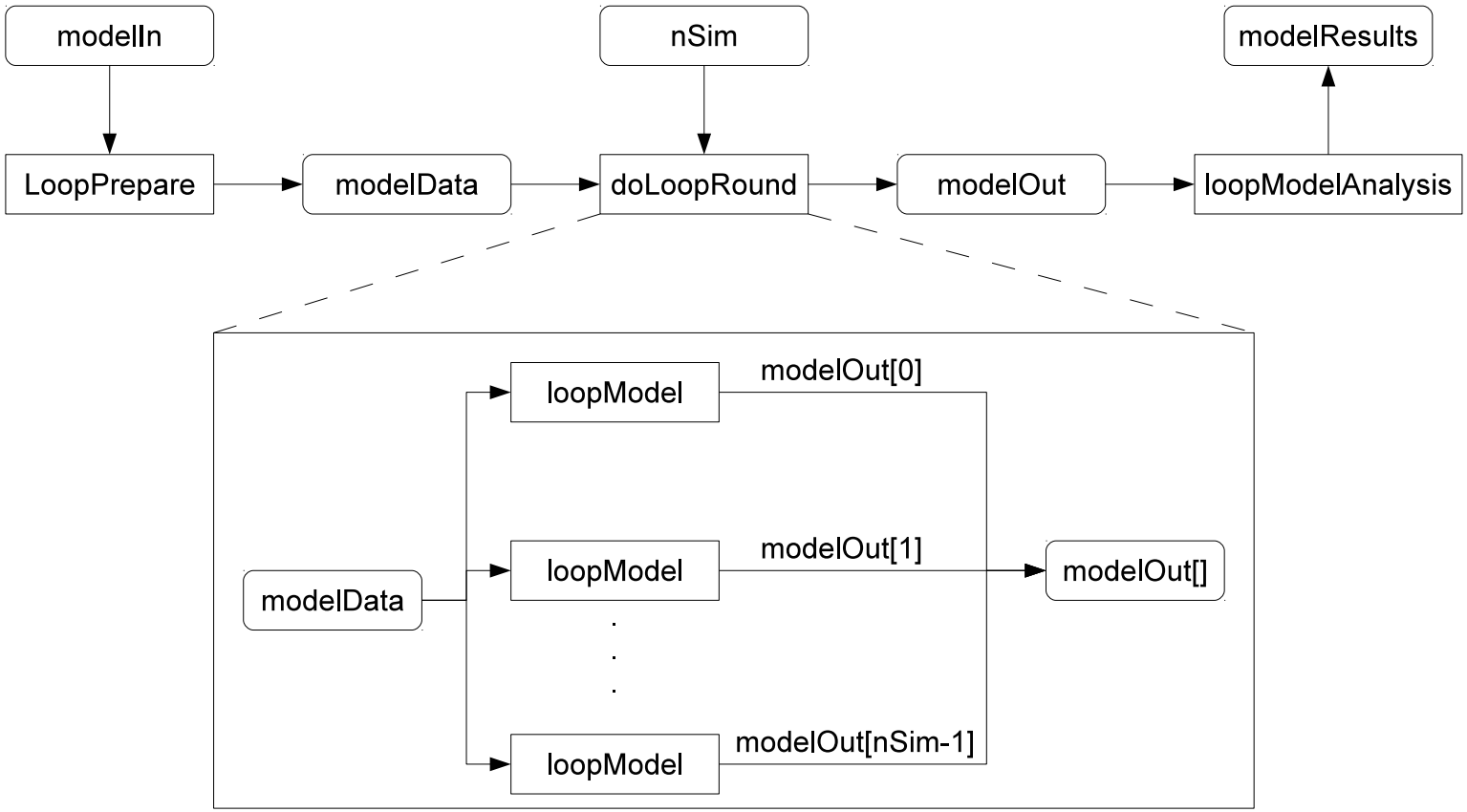
Estudo de Caso: Open Protein Simulator



- Open Protein Simulator (OOPS) é uma aplicação para modelagem de estrutura de proteínas.
- Informação visível ao Swift:
  - Parâmetros científicos (p.ex. identificador da proteína).
  - Informações do tempo de execução (p.ex. duração de execuções).
- Um script coleta informações adicionais:
  - Parâmetros científicos não visíveis para o Swift.
  - Versões SVN das aplicações componentes e do script Swift.

A. Adhikari, J. Peng, M. Wilde, J. Xu, K. Freed, and T. Sosnick, Modeling large regions in proteins: Applications to loops, termini, and folding. *Protein Science* 21(1):107–121, 2012.

Estudo de Caso: Open Protein Simulator



Estudo de Caso: Open Protein Simulator

Listar execuções entre duas datas:

```
select script_run
where  script_run.start_time between '2010-04-04' and '2010-08-08' and
       script_run.filename='psim.loops.swift';

                id                 |         start_time          |   ...
-----------------------------------+-----------------------------+--------
 psim.loops-20100619-0339-b95ull7d | 2010-06-19 03:39:15.18-05   |   ...
 psim.loops-20100618-0402-qhm9ugg4 | 2010-06-18 04:02:21.234-05  |   ...
              ...                  |             ...             |   ...
```

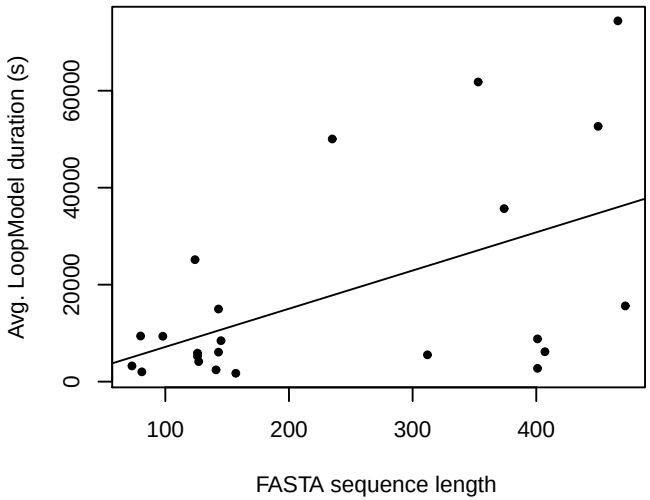
Estudo de Caso: Open Protein Simulator

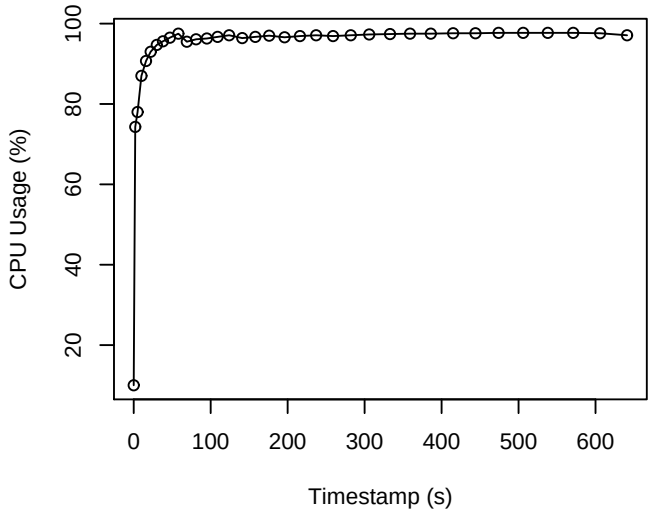
Correlação entre número de simulações e RMSD (performance científica):

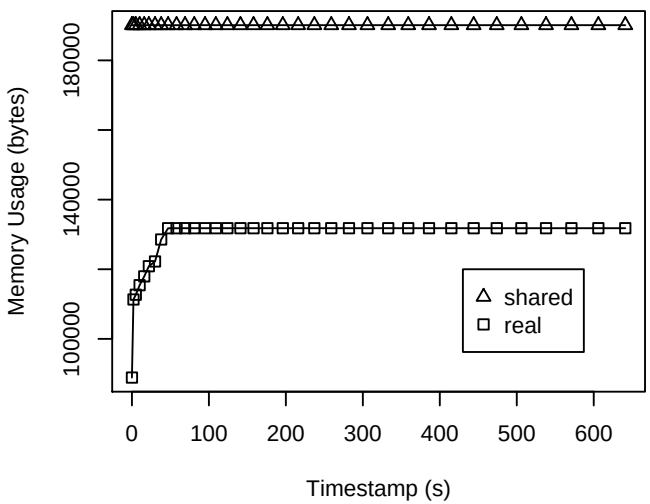
```
SELECT run_id, r.value as nSim, t.value as rmsd
FROM   compare_run_by_param('proteinId') as r
       INNER JOIN
       compare_run_by_param('nSim') as s USING (run_id)
       INNER JOIN
       compare_run_by_annot('rmsd') as t USING (run_id)
WHERE  r.value='TR567' and run.id LIKE 'psim.loops%';
```

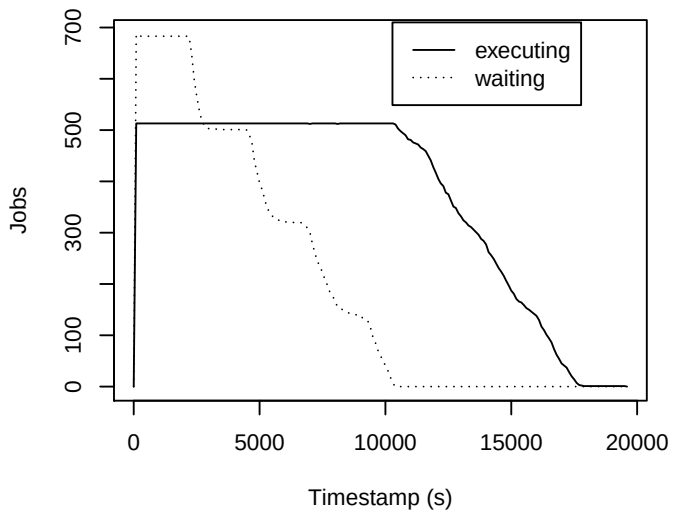
```
             run_id                 | nSim |  rmsd
------------------------------------+------+---------
 psim.loops-20100604-2215-cdifsnb3 |  256 | 3.33123
 psim.loops-20100613-0125-keyyyc35 |  512 | 0.76274
 psim.loops-20100616-1512-h6q4g4ja | 1024 | 0.68426
 ...
```

Estudo de Caso: Open Protein Simulator





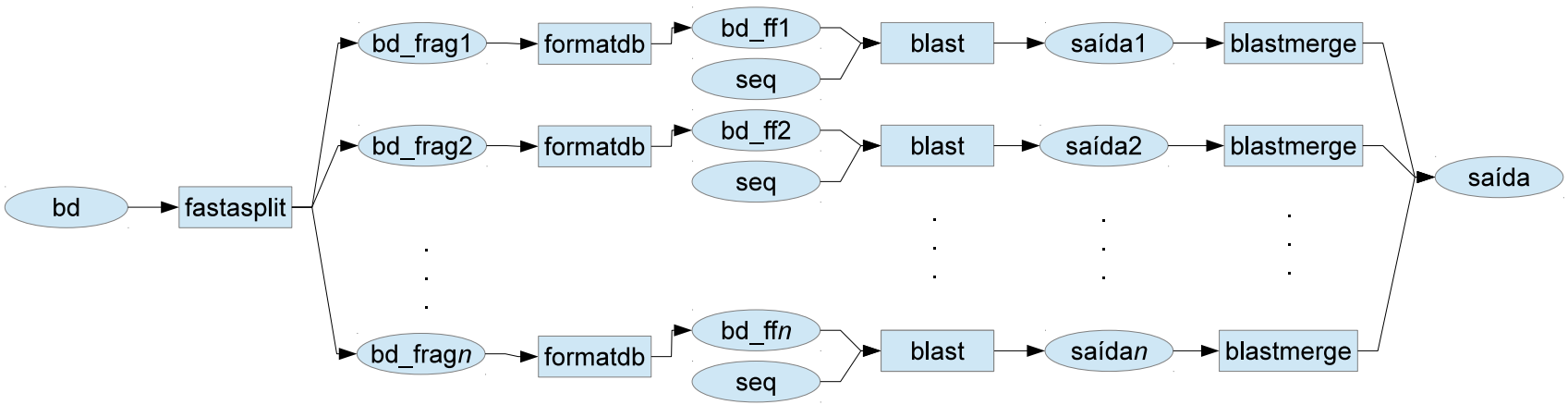




Laboratório Nacional de Computação Científica

Estudo de Caso: BLAST Paralelo

Workflow científico para paralelização do BLAST através de particionamento do banco de dados de entrada:



Estudo de Caso: BLAST Paralelo

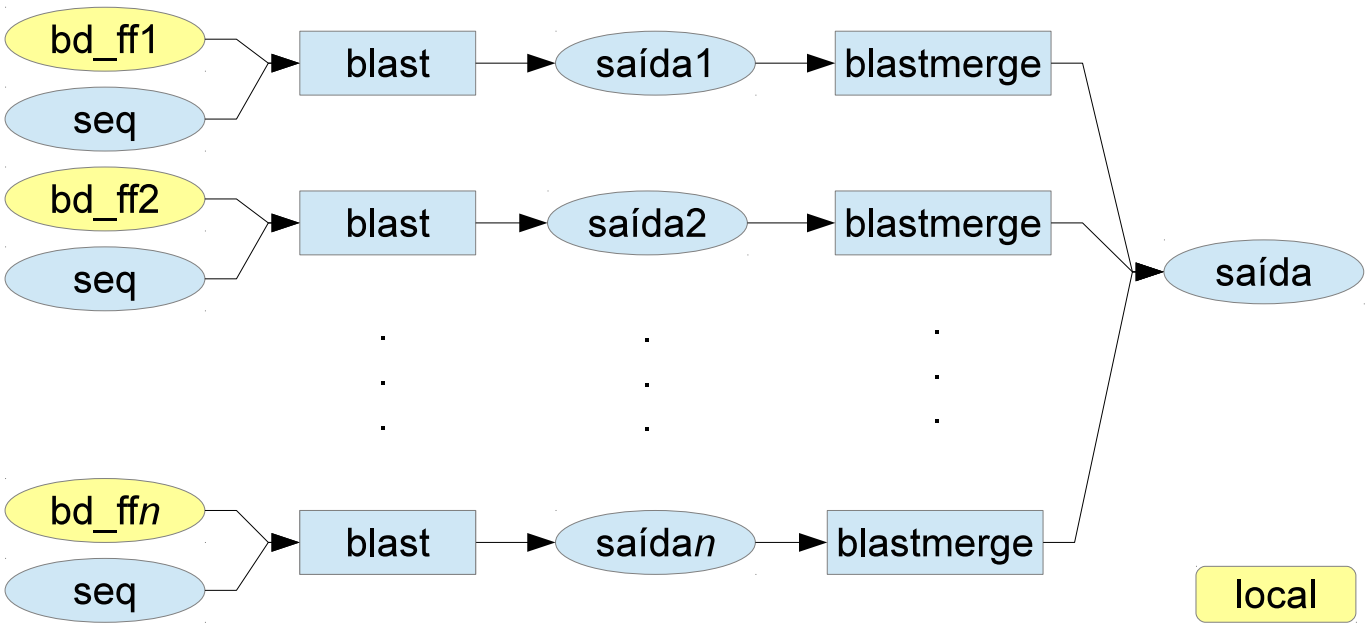
Análise do tempo de espera das aplicações componentes.

```
SELECT app_exec_id, timestamp, wait
FROM   runtime_info;
```

app_exec_id	timestamp	wait
blastall-3i191nsk	1339467548	100.0
blastall-3i191nsk	1339467550	81.7
blastall-3i191nsk	1339467552	50.8
blastall-3i191nsk	1339467554	45.9
blastall-3i191nsk	1339467555	38.4
blastall-3i191nsk	1339467556	31.8
blastall-3i191nsk	1339467558	30.0
blastall-3i191nsk	1339467560	34.7
blastall-3i191nsk	1339467561	33.1

$\Rightarrow$ concorrência de I/O.

Estudo de Caso: BLAST Paralelo



Estudo de Caso: BLAST Paralelo

Análise do tempo de espera das aplicações componentes.

```
SELECT app_exec_id, timestamp, wait
FROM   runtime_info;
```

app_exec_id	timestamp	wait
blastall-leenrisk	1339924431	1.8
blastall-leenrisk	1339924437	2.2
blastall-leenrisk	1339924444	1.6
blastall-leenrisk	1339924453	1.2
blastall-leenrisk	1339924462	0.9
blastall-leenrisk	1339924472	0.7
blastall-leenrisk	1339924483	0.5
blastall-leenrisk	1339924495	0.4
blastall-leenrisk	1339924508	0.3

Referências

- T. Hey, S. Tansley, and K. Tolle, eds. Jim Gray on e-Science: A Transformed Scientific Method. The Fourth Paradigm: Data-Intensive Scientific Discovery. p. xvii-xxxi. Microsoft Research, 2009.
- M. Mattoso et al. Desafios no apoio à composição de experimentos científicos em larga escala. XXXVI Seminário Integrado de Software e Hardware (SEMISH), XXIX Congresso da Sociedade Brasileira de Computação (CSBC), 2009, pp. 307-321.
- A. Ailamaki, V. Kantere, and D. Dash. Managing scientific data. Commun. ACM, vol. 53, no. 6, pp. 68–78, Jun. 2010.
- E. Deelman et al. Workflows and e-Science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25(5): 528-540.
- A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets. *Journal of Network and Computer Applications*, 23:187-200, 2001.
- E. Pacitti, P. Valduriez, M. Mattoso. Grid Data Management: Open Problems and New Issues. *Journal of Grid Computing*, 5(3):273-281, 2007.
- L. Moreau et al. The Open Provenance Model core specification (v1.1). *Future Generation Computer Systems*, 27(6):743–756, 2011.

Referências

- Y. L. Simmhan, B. Plale, D. Gannon. A survey of data provenance in e-science. *ACM SIGMOD Record*, 34(3):31-36, 2005.
- L. Gadelha, M. Mattoso. Kairos: An Architecture for Securing Authorship and Temporal Information of Provenance Data in Grid-Enabled Workflow Management
- M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford I. Raicu, Parallel Scripting for Applications at the Petascale and Beyond. *IEEE Computer*, 42(11):50-60, 2009.
- M. Wilde, M. Hategan, J. Wozniak, B. Clifford, D. Katz, and I. Foster. Swift: A language for distributed parallel scripting. *Parallel Computing*, 37(9):634-652, 2011.
- L. Gadelha, B. Clifford, M. Mattoso, M. Wilde, and I. Foster. Provenance Management in Swift. *Future Generation Computer Systems*, 27(6):775-780, 2011.
- L. Gadelha, M. Mattoso, M. Wilde, I. Foster, Provenance Query Patterns for Many-Task Scientific Computations. *Proceedings of the 3rd USENIX Workshop on Theory and Applications of Provenance* (TaPP'11), 2011.
- L. Gadelha, M. Wilde, M. Mattoso, I. Foster. Exploring provenance in high performance scientific computing. *Proceedings of the First Annual Workshop on High Performance Computing Meets Databases* (HPCDB 2011), pp. 17–20. ACM, 2011.

CENAPAD-RJ

- Recursos Disponíveis:
  - Cluster Sun Blade X6250:
    - 78 nós com 2 processadores quad-core Intel Xeon E5440 e 16GB de memória.
    - Total de 624 núcleos de processamento e 1.24TB de memória.
    - URL: `http://www.lncc.br/sunhpc`.
  - Cluster SGI Altix XE:
    - 30 nós com 2 processadores quad-core Intel Xeon E5520 e 24GB de memória.
    - Total de 240 núcleos de processamento e 720GB de memória.
    - URL: `http://www.lncc.br/altix-xe`.
- Suporte à paralelização (p.ex. MPI, OpenMP), workflows científicos.
- Contato: `cenapad@lncc.br`.
- URL: `http://www.cenapad-rj.lncc.br`.