$PhenoManager$: um Sistema de Gerência de Hipóteses de Fenômenos Científicos*

Leonardo Ramos[1], Kary Ocaña[2], Douglas de Oliveira[2], Fabio Porto[2], Daniel de Oliveira[1]

[1]Instituto de Computação - Universidade Federal Fluminense (IC/UFF) - Brasil

leoslramos@id.uff.br, danielcmo@ic.uff.br

[2]Laboratório Nacional de Computação Científica (LNCC) - Brasil

{karyann,ericsonmarc,fporto}@lncc.br

Resumo. *Experimentos científicos baseados em simulações computacionais envolvem a gerência de grande volume de dados e metadados que são produzidos durante o ciclo de vida de um experimento científico. Tal ciclo se inicia com a identificação de um fenômeno, a formulação de uma hipótese, a modelagem e execução da simulação associada ao fenômeno até sua avaliação. A avaliação de uma hipótese pode requerer a execução de diversos experimentos diferentes, que por sua vez, podem demandar execuções de diferentes Workflows científicos ou scripts complexos, o que torna a tarefa bastante árdua, pois os mesmos podem ser executados em diferentes Sistemas de Gerência de Workflows (SGWf) e ambientes distribuídos. Atualmente, cada SGWf ou script gerencia uma simulação de forma isolada, não permitindo analisar resultados dessas simulações associadas ao mesmo experimento de forma integrada. Este artigo apresenta uma abordagem chamada $PhenoManager$ que tem como objetivo auxiliar os cientistas a gerenciar os fenômenos observados e as hipóteses definidas em conjunto com os resultados das simulações computacionais (que podem executar em múltiplos sistemas e ambientes). O $PhenoManager$ é capaz de auxiliar o cientista na estruturação, validação e reprodução de hipóteses de um fenômeno, por meio de modelos computacionais configuráveis, além de prover uma API de consulta e exportação de metadados por meio de Research Objects.*

1. Introdução

Nos últimos anos, houve um crescimento na utilização de simulações computacionais em experimentos científicos [Hey et al. 2012]. De acordo com [Hey et al. 2012], os experimentos científicos de hoje se baseiam fortemente na análise de dados gerados a partir de complexas simulações computacionais. Existem diversas abordagens existentes para modelar, gerenciar, monitorar e depurar experimentos baseados em simulações. Muitos usuários implementam seus próprios *scripts* e programas, enquanto que outros modelam seus experimentos utilizando Sistemas de Gerência de *Workflows* (SGWf) [de Oliveira et al. 2019], *Gateways* científicos [Ocaña et al.] ou *frameworks* MapReduce como o Apache Spark ou o Hadoop [Karau et al. 2015].

Entretanto, nenhuma dessas abordagens é capaz de representar todos os conceitos envolvidos no método científico [Mattoso et al. 2008]. Atualmente, as abordagens existentes focam somente em representar as simulações que são executadas no contexto de um determinado experimento [Deelman et al. 2009]. Porém, o ponto de partida de uma investigação científica é a descrição de um fenômeno, seja ele natural ou não. O fenômeno estudado ocorre em algum espaco-tempo, em que se observam quantidades físicas selecionadas [Porto et al. 2015]. As hipóteses científicas interpretam conceitualmente o fenômeno estudado por meio de sua representação e através de modelos matemáticos. O teste de hipóteses *in silico* envolve a execução de experimentos, representando os modelos matemáticos e confrontando dados gerados a partir de simulações computacionais complexas [Porto et al. 2015]. Ou seja, de forma a confirmar ou refutar uma hipótese,

*$PhenoManager$ video: https://www.facebook.com/uffescience/

experimentos devem ser definidos, e esses experimentos podem demandar a execução de diversas simulações implementadas de diferentes formas, conforme apresentado na Figura 1 (um *workflow* no SGWf Pegasus, um *script* Python e uma aplicação MapReduce implementada no Apache Spark).

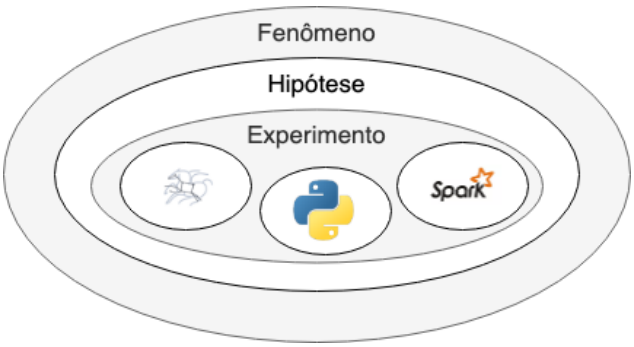

Figura 1. Relação entre os conceitos do método científico

Dessa forma, como não há uma associação entre a execução das simulações, os fenômenos observados e as hipóteses definidas, fica a cargo do cientista gerenciar todo esse conhecimento de forma manual e *ad-hoc* (propensa à erros). Ou seja, se o cientista necessitar descobrir quais execuções de um determinado *script* ou *workflow* foram essenciais para refutar a hipótese α, ele terá que registrar essas informações por conta própria. Em um cenário ainda mais complexo, a validação de uma hipótese científica pode demandar a execução de diversos *workflows*, *scripts* ou aplicações MapReduce distintas que podem executar em ambientes distribuídos e de alto desempenho, como as nuvens de computadores e supecomputadores.

Assim, seria interessante que os cientistas tivessem acesso à uma abordagem que auxiliasse na gerência do projeto científico como um todo e no apoio ao método científico, ajudando na documentação, compartilhamento dos dados obtidos e na facilitação da reprodução dos experimentos realizados, o que representa um desafio em aberto. Sendo assim, este artigo apresenta uma abordagem chamada $PhenoManager$ que visa apoiar a gerência e validação de hipóteses científicas de forma integrada à execução dos experimentos, seja via *workflows*, *scripts* ou aplicações MapReduce. O $PhenoManager$ aborda desde a etapa de concepção, de configuração do modelo de execução, até a validação e reprodução dos experimentos, por meio dos dados de proveniência [Freire et al. 2008]. Além disso, o $PhenoManager$ permite que o cientista execute experimentos em ambientes de computação de alto desempenho (como o supercomputador Santos Dumont[1] do Laboratório Nacional de Computação Científica - LNCC) e se integra com o sistema *SciManager* [Ramos et al. 2016], que gerencia tarefas de equipes em projetos científicos, em um mesmo ecossistema de *software*.

O restante do artigo está organizado em três seções além da introdução. A Seção 2 apresenta uma visão geral do $PhenoManager$. A Seção 3 discute como será realizada a demonstração do $PhenoManager$ e, finalmente, a Seção 4 conclui o presente artigo.

2. Arquitetura do $PhenoManager$

A Figura 2 apresenta a arquitetura do $PhenoManager$ e seus componentes principais. O $PhenoManager$ pode ser dividido em seis camadas funcionais: (i) Camada de Autenticação, (ii) Camada de Gerência do Ambiente, (iii) Camada de Execução, (iv) Camada de Dados, (v) Camada de Consulta, e (vi) Portal *Web*. A seguir discutimos em detalhes cada uma dessas camadas.

A Camada de Autenticação é a responsável por gerenciar as credenciais de acesso ao $PhenoManager$. Essa camada é fundamental, uma vez que dados de pesquisas não publicados

[1] `https://sdumont.lncc.br/`

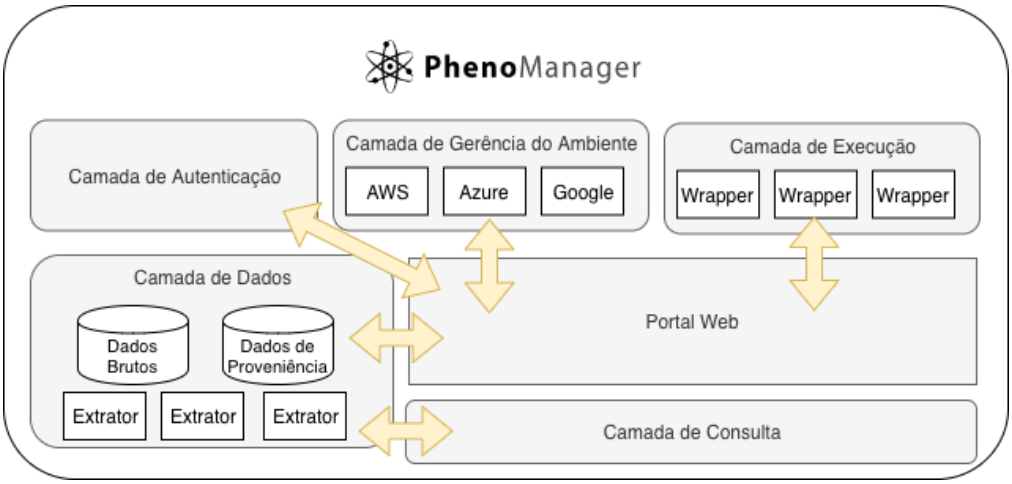

Figura 2. Arquitetura do $PhenoManager$

serão manipulados pelo sistema. O controle de credenciais é realizado em dois níveis. No nível do "Perfil Pessoal", cada usuário da ferramenta configura/preenche suas informações pessoais e carrega credenciais para acesso aos diversos ambientes (*e.g.*, Amazon AWS). No nível de "Grupos de Usuários", o(s) usuário(s) administrador(es) (com privilégios para criar grupos) procuram, selecionam e agrupam perfis de usuários existentes, e, a partir da criação do grupo, compartilham os mesmos privilégios no ambiente. Cada usuário ou grupo poderá ter acesso aos dados e funcionalidades providos pelo sistema de acordo com os seguintes privilégios: (i) Permissão para leitura ("*READ*"): o usuário e/ou grupo pode apenas visualizar os dados, porém não pode editar e/ou cadastrar qualquer dado; (ii) Permissão para escrita ("*WRITE*"): o usuário e/ou grupo pode ler e cadastrar informações dentro do contexto especificado, tornando-o membro ativo do sistema; (iii) Permissão de administrador ("*ADMIN*"): além das permissões anteriormente citadas, o usuário e/ou grupo pode cadastrar permissões para outros usuários e grupos;

A Camada de Gerência do Ambiente é a responsável por configurar ambientes distribuídos para a execução das simulações. Para cada ambiente diferente, um componente deve ser desenvolvido com as chamadas para a API específica do mesmo. O $PhenoManager$ já provê nativamente a integração com três tipos de ambientes diferentes: *Cluster*, *Cloud (Amazon AWS)* e *SSH*. Além disso, é possível configurar uma conexão *VPN* para estes ambientes, podendo selecionar entre *Cisco VPN* e *VPN default*. Para o ambiente *Cloud*, é possível configurar no detalhe, os tipos e imagens das máquinas virtuais que serão construídas no ambiente da *Amazon AWS*.

A Camada de Execução é a responsável por invocar SGWfs, *scripts* ou aplicações externas nos ambientes de alto desempenho. Para cada sistema diferente ou aplicação a ser invocada, um *wrapper* específico deve ser provido (já que o $PhenoManager$ necessita conhecer o processo de invocação da aplicação externa). A chamada aos *wrappers* é assíncrona, logo o serviço dessa camada pode ser escalado em mais instâncias, aumentando, dessa forma, o *throughput* de execuções paralelas de simulações científicos para diferentes usuários. Sendo assim, por meio desse artifício, procuramos garantir o paralelismo e a alta disponibilidade do $PhenoManager$.

A Camada de Dados contém o banco de dados de proveniência (com todos os metadados registrados no $PhenoManager$) e os dados brutos produzidos pelas simulações computacionais. Além disso, essa camada contém uma série de extratores responsáveis por acessar a base de proveniência ou o *log* da aplicação externa e carregar as informações no banco de dados de proveniência do $PhenoManager$. Em sua versão atual, o banco de dados de proveniência se encontra modelado no PostgreSQL e os dados brutos são carregados no Google Drive.

A Camada de Consulta é a responsável por permitir que o cientista possa submeter con-

sultas aos dados gerenciados pelo $PhenoManager$. Essa camada provê uma API que abstrai as consultas aos dados de modo a facilitar sua manipulação por cientistas e outras aplicações consumidoras deste serviço. A API permite que o cientista submeta consultas contendo filtros, ordenações, funções de agregação e projeções de campos em todas as entidades expostas no modelo de dados do $PhenoManager$. Outro ponto importante é que a *API* só responde com sucesso se as credenciais corretas forem passadas no cabeçalho da solicitação. Além disso, a Camada de Consulta é responsável por exportar pacotes chamados *Research Objects* (RO) [Holl et al. 2013], que contém tanto os metadados consultados quanto os dados brutos produzidos pela simulação. Por meio dos ROs, os cientistas são capazes de reproduzir uma determinada simulação, um experimento ou verificar se uma hipótese foi efetivamente validada.

Finalmente, o Portal *Web* é o responsável por toda a interface com o cientista e a integração com as demais camadas. Nele o cientista registra os fenômenos observados, as hipóteses associadas, seus experimentos e os modelos que executam as simulações de cada experimento (*e.g., workflow*, *script* ou aplicação). Além disso, por meio do Portal *Web*, o cientista é capaz de executar efetivamente suas simulações e consultar os dados de proveniência coletados de forma integrada, *i.e.,* se um mesmo experimento for composto de diversos *workflows* e aplicações, as consultas à base de proveniência consideram todas as simulações como parte do mesmo experimento, o que não ocorre nas ferramentas existentes que gerenciam os modelos computacionais de maneira isolada [Deelman et al. 2009].

Em termos de implementação, o $PhenoManager$ foi desenvolvido na linguagem Java e segue o padrão arquitetural de APIs como microserviços, ou seja, cada componente é um serviço *Web* autônomo e pequeno que disponibiliza apenas uma funcionalidade [Newman 2015]. Todos os microserviços foram construídos por meio do *framework Spring Boot*, que já oferece apoio para desenvolvimento de aplicações nesse padrão de uma maneira rápida e pouco verbosa. Para segurança de dados e autenticação entre os componentes, foi utilizado o arcabouço *Spring Security*. O desenvolvimento das interfaces, templates e telas do Portal *Web* foi realizado com *AngularJs*. Como cada serviço que compõe a arquitetura do $PhenoManager$ é completamente isolado dos demais, a escalabilidade se torna um dos pontos chave deste ecossistema. Para garantir a invocação de aplicações externas de forma assíncrona, foi escolhido o *message Broker* de código aberto RabbitMQ.

3. Demonstração do $PhenoManager$

Propõe-se aos participantes do SBBD um cenário de uso para que os mesmos tenham uma visão completa do $PhenoManager$ (Figura 3). A demonstração se inicia com o *dashboard* do $PhenoManager$ (Figura 3(a)) que apresenta todas as execuções de simulações em andamento, finalizadas, *etc* para controle do cientista. Após, é necessário o cadastro do fenômeno e hipóteses no $PhenoManager$. Basicamente, o usuário deve informar um nome e uma descrição tanto para o fenômeno quanto para a hipótese. Após, os usuários devem registrar os seus *scripts* ou *workflows* no $PhenoManager$ (Figura 3(b)). É importante ressaltar que os modelos que executam as simulações são chamados de *Executors* no $PhenoManager$. Além dos modelos, o cientista deve configurar o ambiente de execução (Figura 3(c)). Uma vez que tanto os modelos quanto os ambientes estão configurados, o usuário deve carregar os dados no $PhenoManager$ por meio da interface *Web*. Com os dados carregados, os modelos podem ser executados e o usuário pode monitorar as execuções dos mesmos (Figura 3(d)). Serão fornecidos dados de exemplo, mas os usuários são encorajados a realizar a carga de seus próprios dados e *scripts*.

Uma vez que as execuções se iniciem ou já tenham terminado, o usuário é capaz de consultar a base de proveniência do $PhenoManager$. Um exemplo de consulta que pode ser submetida é ”Quais os nomes e versões dos modelos utilizados na validação de uma hipótese com um nome específico (“sciphy”)?”. Essa consulta é invocada por meio da chamada à API de consulta do $PhenoManager$ apresentada na Figura 4. Após o processamento da consulta é gerado um *Re-*

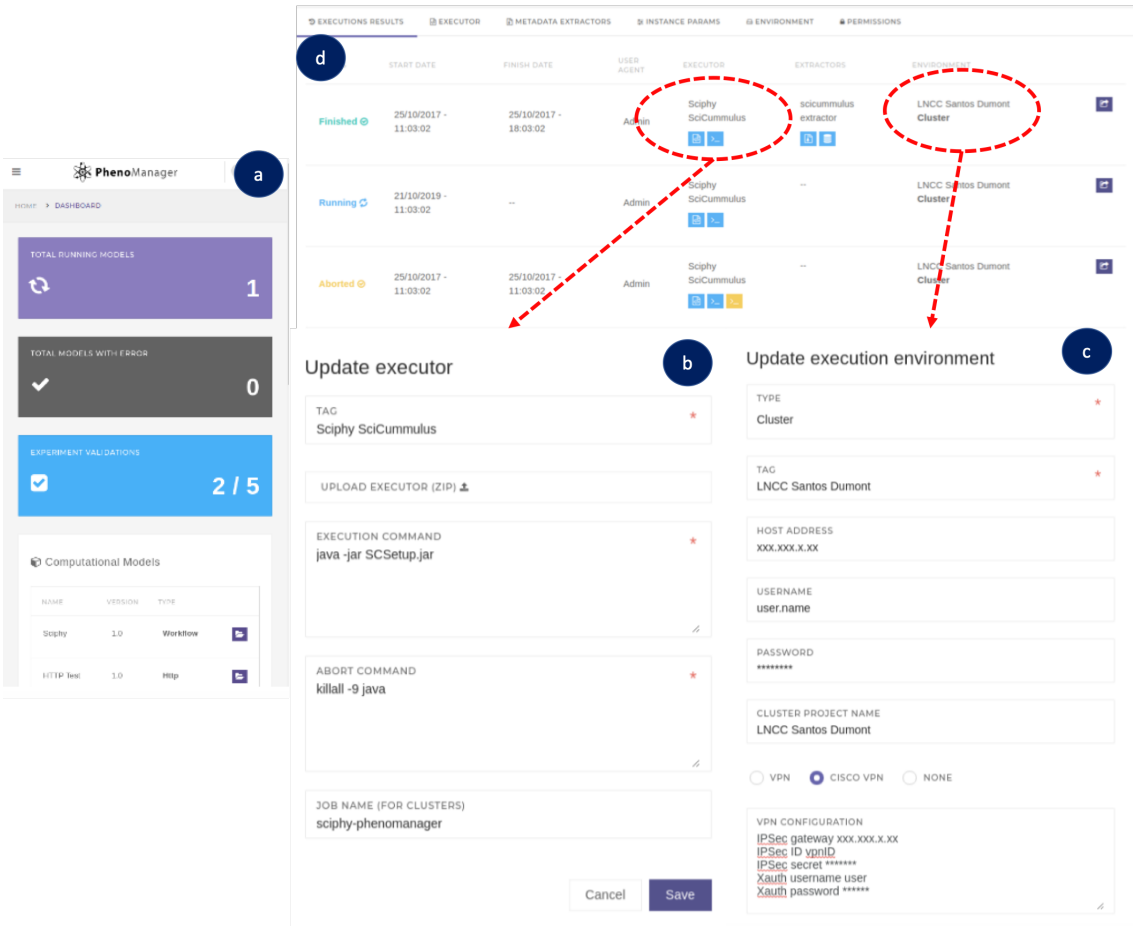

Figura 3. Interface do $PhenoManager$

search Object contendo a resposta da consulta e os dados associados para *download*. A Figura 5 apresenta um fragmento do *Research Object* gerado.

```
http://phenomanager.ic.uff.br/PhenoManagerApi/v1/computational_models?projection=[name,currentVersion]&filter=[experiment.hypothesis.name=like=sciphy;experiment.hypothesis.state=VALIDATED]
```

Figura 4. Exemplo de Consulta na API do $PhenoManager$

4. Conclusões

O presente artigo apresenta o $PhenoManager$, um Sistema de Gerência de Hipóteses de Fenômenos científicos que é capaz de gerenciar fenômenos e hipóteses em conjunto com a execução dos seus experimentos e simulações computacionais associadas. O $PhenoManager$ se baseia em uma arquitetura de microserviços, o que facilita a sua extensão e escalabilidade. Dessa forma, o $PhenoManager$ fornece um valioso ponto de partida para a análise integrada de dados de proveniência de múltiplos sistemas e programas. Como trabalho futuro, pretendemos fornecer análises adicionais como por exemplo, análises de desempenho das simulações executadas. Além disso, pretendemos implantar um mecanismo automático de validação das hipóteses a partir dos dados de proveniência coletados. O $PhenoManager$ pode ser obtido em `https://github.com/UFFeScience/Phenomanager`.

Agradecimentos

A pesquisa apresentada neste artigo foi parcialmente financiada por CNPq, CAPES e FAPERJ.

```
{
    "@context":{
        "schema":"http://schema.org/",
        .
        .
        .
    },
    "@graph":[{
            "@type":[
                "ro:ResearchObject",
                "ore:Aggregation"
            ],
            "@id":"4B471432FCD146018593817458D6E21D"
        },{
            "schema:name":"Sciphy"
        },{
            "dc:creator":"QWE123987POEIWQPEWQ12687EWQEWQEF"
        },{
            "dc:abstract":"Sciphy GPU"
        },{
            "dc:contributor":["QWE123987POEIWQPEWQ12687EWQEWQEFP"]
        },{
            "dc:title":"Sciphy"
        },{
            "Ore:aggregates":[{
                    "@type":"ro:Resource",
"@id":"http://localhost:9500/PhenoManagerApi/v1/computational_models/4B471432FCD146018593817
458D6E21D/instance_params/3EE92B89774345BD9A8CA4DF77FB148A/value_file"
                },{
                    "@type":"ro:Resource",
"@id":"http://localhost:9500/PhenoManagerApi/v1/computational_models/4B471432FCD146018593817
458D6E21D/instance_params/E4A3F7035B0D45D29ABA0754E89FE8F5/value_file"
                },{
```


Figura 5. Fragmento do *Research Object* gerado pelo $PhenoManager$

Referências

de Oliveira, D. C. M., Liu, J., and Pacitti, E. (2019). *Data-Intensive Workflow Management: For Clouds and Data-Intensive and Scalable Computing Environments*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers.

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

Freire, J., Koop, D., Santos, E., and Silva, C. T. (2008). Provenance for computational tasks: A survey. *Computing in Science & Engineering*, 10(3).

Hey, T., Gannon, D., and Pinkelman, J. (2012). The future of data-intensive science. *IEEE Computer*, 45(5):81–82.

Holl, S., Garijo, D., Belhajjame, K., Zimmermann, O., Giovanni, R. D., Obst, M., and Goble, C. A. (2013). On specifying and sharing scientific workflow optimization results using research objects. In *WORKS 2013, Denver, CO, USA, November 17, 2013*, pages 28–37.

Karau, H., Konwinski, A., Wendell, P., and Zaharia, M. (2015). *Learning spark: lightning-fast big data analysis*. "O'Reilly Media, Inc.".

Mattoso, M., Werner, C., Travassos, G., Braganholo, V., and Murta, L. (2008). Gerenciando experimentos científicos em larga escala. *SBC-SEMISH*, 8:121–135.

Newman, S. (2015). *Building microservices: designing fine-grained systems*. "O'Reilly Media, Inc.".

Ocaña, K., Galheigo, M., Osthoff, C., Jr., L. M. R. G., Gomes, A. T. A., de Oliveira, D., Porto, F., and de Vasconcelos, A. T. R. Towards a science gateway for bioinformatics: Experiences in the brazilian system of high performance computing. In *CCGRID*.

Porto, F., Costa, R. G., de Carvalho Moura, A. M., and Gonçalves, B. (2015). Modeling and implementing scientific hypothesis. *J. Database Manag.*, 26(2):1–13.

Ramos, L. S., Ocaña, K. A., and de Oliveira, D. (2016). Um sistema de informação para gerência de projetos científicos baseados em simulações computacionais. In *Anais do XII Simpósio Brasileiro de Sistemas de Informação*, pages 216–223. SBC.