Using Domain-Specific Data to Enhance Scientific Workflow Steering Queries

João Carlos de A.R. Gonçalves¹, Daniel de Oliveira¹, Kary A.C.S. Ocaña¹,
Eduardo Ogasawara^{1,2}, and Marta Mattoso¹

¹ COPPE, Federal University of Rio de Janeiro, Brazil

² CEFET/RJ, Brazil

{jcg,danielc,kary,ogasawara,marta}@cos.ufrj.br

Abstract. In scientific workflows, provenance data helps scientists in understanding, evaluating and reproducing their results. Provenance data generated at runtime can also support workflow steering mechanisms. Steering facilities for workflows is considered a challenge due to its dynamic demands during execution. To steer, for example, scientists should be able to suspend (or stop) a workflow execution when the approximate solution meets (or deviates) preset criteria. These criteria are commonly evaluated based on provenance data (execution data) and domain-specific data. We claim that the final decision on whether to interfere on the workflow execution may only become feasible when workflows can be steered by scientists using provenance data enriched with domain-specific data. In this paper we propose an approach based on specialized software components, named Data Extractor (DE), to acquire domain-specific data from data files produced during a scientific workflow execution. DE gathers domain-specific data from produced data files and associates it to existing provenance data on the provenance repository. We have evaluated the proposed approach using a real bioinformatics workflow for comparative genomics executed in SciCumulus cloud workflow parallel engine.

1 Introduction

Scientific workflows are used as an abstraction that models and allows for the management of scientific experiments [1]. Many of these workflows are composed by activities that invoke computing and data intensive programs. A workflow may execute for weeks or months, requiring parallel processing in High Performance Computing (HPC) environments. In order to be evaluated and reproduced by third-party scientists or teams, provenance data [2] related to all executions of these workflows has to be captured and organized for further analysis. Capturing provenance in HPC and distributed environments, such as clouds [3], is a very important but also a complicated task.

Provenance, or lineage, of a workflow is related to all metadata associated to the data products generated by a specific workflow execution. Simmhan *et al.* [4] define provenance as the “(...) information that helps determine the derivation history of a data product, starting from its original sources. We use the term data product or

dataset to refer to data in any form, such as files, tables, and virtual collections (...). Intuitively, provenance data is mainly used for a *post* analysis of the experiment. This way, many approaches focus on providing techniques for long-term provenance management [5–7]. Specialized models and technologies are being proposed to guarantee the long-term understandability of the preserved data of a workflow. Semantic Web techniques, such as ontologies and annotations [4,7,8], are used for adding semantics to data products and to allow the construction of complex queries.

Long-term provenance data is fundamental for enabling reproducibility and different kinds of *post* analysis [9–11], but these solutions do not allow for provenance analysis during the course of a workflow execution. Even though the workflow execution *log* could be browsed, this is far from provenance data query. Our previous works in supporting workflow scientists from bioinformatics [12] and numerical methods [13] have led us to develop services to query provenance data during the execution [14]. We define this type of provenance as *runtime provenance data*. It means that all derivation history of a data product and the status of all activities (start time, end time, errors) are available for querying as soon as an activity executes and data products are generated.

This runtime data analysis is the basis for workflow steering [15]. Our preliminary experiments in [11] have shown the potential of user steering features in workflows for tracking, evaluating and adapting the execution of a workflow. Workflow steering remains an open issue that could support iterative methods, *i.e.* scientists analyze the status of the execution and interfere by stopping or changing the space of parameters to be explored [11]. This steering mechanism allows for optimizing workflow execution and has gained importance in the last years since HPC environments, such as clouds, are paid according to the time used. In fact, this is highlighted by Gil *et al.* [15] as “a fundamental technology to fully support e-Science”.

Provenance generated during the execution course of a workflow is a rich resource to support workflow steering. However, runtime provenance data is not sufficient to support steering mechanisms in workflows. In many cases, the execution has to be suspended or stopped when the approximate solution meets a preset user criterion. The final and complex decision on whether to interfere on the workflow or to reduce the space of parameters to explore, increase or even stop the execution has to be taken by scientists using domain-specific data. This kind of data, associated to the workflow provenance data, provides insight for scientists to perform steering actions. These runtime decisions can be taken based on querying provenance enriched with domain-specific data. Although there are some initiatives to propose workflow steering using runtime provenance [16], none of them allows for performing queries using domain-specific data. In this scenario, the goal is how to provide support for developing steering capabilities in parallel scientific workflows using runtime provenance enriched with domain-specific data.

In this paper, we address the problem of querying provenance data enriched with domain-specific data to provide for steering mechanisms. We contribute with a software solution for scientists to extract domain-specific data from input and produced data files (registered in the provenance repository) and use this data to complement runtime queries on provenance data. This provenance enriched

representation is possible due to Data Extractors (DE), which are software components that are deployed in SciCumulus, a cloud workflow engine [17]. Although we used SciCumulus as workflow engine in this paper, the proposed approach can be coupled to other Scientific Workflow Management Systems (SWfMS) especially when it is not possible to modify the underlying workflow engine in order to produce the required, richer provenance metadata for the problem at hand. DE is based on a workflow algebra that uniformly represents workflow data and it allows for extracting domain-specific data and storing it in the provenance repository, *i.e.* a database system, along with information about workflow structure and execution. As a result, DE is independent of application structures, *i.e.* scientists can couple a series of DE to the workflow specification according to their needs.

This paper is organized as follows. In Section 2, we present a motivation workflow for steering while in Section 3, we describe SciCumulus cloud engine and the workflow algebraic approach that provides for the uniform underlying data representation. In Section 4, we detail the proposed approach for enriching provenance with domain-specific data and, in Section 5, we present experimental results. In Section 6, we discuss related work. Next, Section 7 presents final remarks.

2 Motivation: Steering for Comparative Genomics Experiments

This section illustrates a computing-intensive parallel workflow scenario in the comparative genomics bioinformatics domain. We use this example consistently in the rest of the paper. Comparative genomics is one of many bioinformatics fields that aim at computationally comparing hundreds of different genomes [18]. Many types of bioinformatics applications associated to this field, such as multiple sequence alignment (MSA), homologues detection and phylogenetic analysis are increasing in scale and complexity [19]. Managing genomic experiments is far from trivial, since they are computationally intensive and process large amounts of data. One of the main possible usages of comparative genomics workflows is to use profiles hidden Markov models (pHMMs) for improving phylogenetic analyses. One example of this comparative genomics workflow is SciHmm [12]. SciHmm is a parallel workflow based on a cross-validation procedure [12] to decide which MSA method and algorithm offer the best quality in the alignments for a phylogenetic analysis workflow. SciHmm is composed by five main activities: (i) MSA construction, (ii) pHMM build, (iii) pHMM search against a target database, (iv) cross-validation analysis and, (v) generation of Receiver-Operating Characteristic (ROC) curves. A ROC curve is a graphical plot that illustrates the performance of a specific classifier as its threshold is varied. Each ROC curve is generated by plotting the fraction of true positives out of the positives *versus* the fraction of false positives out of the negatives, at various threshold settings.

However, since SciHmm may execute for several hours or even days, scientists may need to analyze if a specific execution is producing the expected results before the workflow finishes. If part of the results is perceived to be under the expectations (*e.g.* sequences that belong to different genes are included in the same input

multi-fasta), scientists may interfere on the execution (e.g. re-executing choosing specific genes to be processed). These actions certainly spare financial resources and scientists' time. One real steering scenario is when scientists are not aware of the entire content of a produced data file; i.e. an aligned multi-fasta file can be formed by a huge volume of biological sequences or, in the worst scenario, sequences belonging to other genes classes. This may lead to incoherent results in ROC curves analysis. Fig. 1 presents a data compilation of a real scenario where the Sensitivity-Specificity (i.e. quality analysis parameters of ROC curves) pair value is under the threshold in a ROC curve and scientists may use this information to interfere (calibrate) in the execution. Fig. 1 presents an excerpt of a data file produced by *hmmsearch* comparison using the pHMM (belongs to the gene 6-phosphogluconate dehydrogenase) against protozoan genomes from the RefSeq database. In Fig. 1 underlined lines represent sequences that belong to genes different from 6-phosphogluconate dehydrogenase, which have to be neglected in this case.

Query: i-201103012E4SG6XC1V [M=483]			
E-value	score	Sequence	Description
-----	-----	-----	-----
<u>3e-251</u>	<u>838.3</u>	<u>167518822</u>	<u>hypothetical protein [Monosiga brevicollis]</u>
3.5e-206	689.6	66816225	6-phosphogluconate dehydrogenase [Dictyostelium discoideum]
1.5e-153	516.1	290997790	6-phosphogluconate dehydrogenase [Naegleria gruberi]
2.5e-141	475.8	194476751	6-phosphogluconate dehydrogenase [Paulinella chromatophora]
2.6e-141	475.7	224000295	6-phosphogluconate dehydrogenase [Thalassiosira pseudonana]
<u>5.9e-140</u>	<u>471.3</u>	<u>219121442</u>	<u>G6PDH/6PGDH fusion protein [Phaeodactylum tricornutum]</u>
8.3e-139	467.5	221059365	6-phosphogluconate dehydrogenase [Plasmodium knowlesi]
2.2e-133	449.6	124809822	6-phosphogluconate dehydrogenase, put. [Plasmodium falciparum]
1.4e-132	446.9	68076479	6-phosphogluconate dehydrogenase [Plasmodium berghei]
4.2e-67	231.0	84999608	6-phosphogluconate dehydrogenase [Theileria annulata]
1.8e-47	166.3	71032157	6-phosphogluconate dehydrogenase G6PDH [Theileria parva]
<u>1.2e-42</u>	<u>150.4</u>	<u>223997774</u>	<u>predicted protein [Thalassiosira pseudonana]</u>
<u>3.5e-41</u>	<u>145.5</u>	<u>70917327</u>	<u>hypothetical protein [Plasmodium chabaudi]</u>
8.4e-37	131.1	71661909	6-phosphogluconate dehydrogenase [Trypanosoma cruzi]

Fig. 1. An excerpt of the output information of SciHmm

In addition, to get the results presented in Fig. 1, scientists have to discover which workflow activities have already finished without errors associated with ROC curves results. For each ROC curve, it is necessary to discover which sequences (*hmmsearch* hits in each multi-fasta file) produced these low quality curves and extract domain-specific data, such as, gene information based on enzyme classification or also species. In the example of Fig. 1, scientists defined that the analysis is based on 6-phosphogluconate dehydrogenase; then, all sequences that do not fulfill this premise have to be excluded (lines underlined). Also, in this particular case, scientists may need to re-execute SciHmm by splitting the input data file in four parts (i.e. one for each human pathogens genus (or taxonomic group): *Plasmodium*, *Trichomonas*, *Giardia*, *Toxoplasma*, and Trypanosomatids. However, this steering mechanism is

only possible if scientists can query runtime provenance data related to domain-specific data. In this example, species and genus information are not part of the provenance data. Following Simmhan *et al.* definition, provenance is related to data products lineage and domain-specific data is the content of those data products.

3 SciCumulus Engine and Algebraic Workflow Representation

Since the proposed approach has to be implemented in a system that manages the parallel execution of workflows while generating runtime provenance data, we have chosen to use, in the case study, SciCumulus [17]. SciCumulus is an engine that is designed to distribute scientific workflow activities (or even entire scientific workflows) dispatched from an SWfMS, such as, VisTrails [21], Taverna [22], into a cloud environment, e.g. Amazon EC2 [23] to be executed in parallel. SciCumulus creates and manages several parallel tasks associated to each activity and orchestrates the execution of these tasks on a distributed set of virtual machines (VMs), thus forming a virtual cluster. SciCumulus promotes the usage of control components distributed over several tiers: client tier, distribution tier, execution tier, and data tier.

The client tier is responsible for starting parallel execution of workflow activities in the cloud. The components of the client tier are deployed in an existing SWfMS. The distribution tier manages the adaptive execution of parallel activities in cloud environments by creating and managing parallel activity executions (named tasks) that contain the program to be executed, its parallel strategy, parameters values and input data to be consumed. SciCumulus addresses cloud elasticity by providing for adaptive VM addition and removal [17]. The execution tier is responsible for invoking executable codes in many VMs in the virtual cluster and to collect provenance at runtime. As a task execution finishes, it immediately records all provenance data in the repository. Finally, the data tier contains all repositories of data used by SciCumulus, including the provenance repository, which is fundamental for the approach proposed in this paper. It encompasses the provenance repository that contains fundamental provenance data collected during the course of the workflow. The data model of this repository is further explained following in this paper (Section 3). SciCumulus generates runtime provenance data, this way; it is a serious candidate to provide steering mechanisms.

SciCumulus represents workflows using an algebraic representation proposed by Ogasawara *et al.* [24]. This algebra defines its operands as a uniform data representation consistently used throughout the workflow execution, where all data (consumed and produced by each activity) is represented as relations (similar to a database). As in relational algebra, relations are defined as sets of tuples of primitive types (*i.e.* integer, float, string, date) and complex types (*e.g.* a pointer to a data file). The parameter values for the activities are represented as values (attributes) in a tuple, whereas the set of tuples composes the relation to be consumed by an activity. This relation-based representation is a uniform way of introducing new attributes (representing information extracted from produced data files) to represent domain-specific data. By using this algebraic representation, SciCumulus can execute workflows in different domains. In addition, this algebraic representation is fundamental for the proposed approach since it allows for extract information and easily add it to the workflow representation.

4 Enriching Provenance with Domain-Specific Data

Following the definition of Simmhan *et al.*, all provenance data is related to lineage metadata about a specific data file. In fact, each data file is considered as a “black-box” by the traditional provenance definition. Commonly represented provenance data is related to execution time of each activity, input and output files (in the case of retrospective provenance [2]), and details about software involved in the execution (in the case of prospective provenance). Although very useful, this provenance data is not sufficient to provide steering capabilities for scientists. In order to facilitate scientists’ work, we present in this section an approach to extract domain-specific data from generated data files and use it to enrich provenance data allowing for queries with more expressivity. First we present Data Extractors (DE) and the adaptations in SciCumulus to incorporate them in the architecture. Then, we present the data model to represent both provenance and domain-specific data for an execution of SciHmm workflow.

SciCumulus Data Extractor

For the domain-specific data acquisition, we designed, developed and incorporated DE into SciCumulus engine. A DE can be implemented as an artificial workflow activity, which invokes an external program (defined) by scientists that analyzes produced data files and extracts domain-specific data from it. These programs encapsulate the domain-specific extraction rules to crawl domain-specific data from generated files, being a fundamental component of the proposed approach. An example of a rule implemented by a program is to verify the format types accepted as input for an activity. In this case, the workflow engine (SciCumulus) generates the OPM *isProducedBy* link (querying its provenance repository) between the activity and data file and the domain contents as depicted in Fig. 2.

Fig. 2. An OPM graph generated based on SciCumulus’ provenance repository

This way, SciCumulus allows for a DE to follow the provenance chain and relate the domain outputs to domain information in the input files. Based on the identification of domain-specific data files, DE can invoke third-party Web services to gather domain-specific data from these files. In the case of bioinformatics workflows, information extracted from data files can be enhanced by querying one of

the several available NCBI databases and services. A DE is placed after each activity execution in SciCumulus and the program to be invoked is defined at the workflow specification file (in XML file similar to the one presented on Fig. 3) created by scientists. Once an activity is executed (and a data file is produced) the DE is invoked to extract associated domain-specific data. All data is stored using a unified provenance schema that comprises provenance data (related to activity execution) associated to domain-specific data. This way, each activity execution in SciCumulus presents three phases: (a) a data production phase, (b) data extraction phase and, (c) provenance repository update as depicted in Fig. 3.

Phase (a) invokes a specific program that is part of the workflow. In the case of SciHmm, HMMER and MAFFT are examples of these programs. Each invocation consumes an input tuple Tp_1 and produces an output tuple Tp_2 that indicates the produced files (using a file pointer in the output tuple). The (b) phase invokes DE and consumes Tp_2 to discover the data files to be processed. It produces a new tuple Tp_3 , which now contains a pointer to data files and additional attributes to represent extracted domain-specific data. The (c) and last phase is also performed by DE and it is focused on registering domain-specific data (Tp_3) in the provenance repository (in specific schema classes that are explained following in this section).

The DE framework is implemented in Java version 6.31 and can invoke source codes of different languages. This is possible since the interface, in this case, is a relation-based file that contains all tuples that were processed or generated. This way, scientists may invoke programs in Python or Perl, without needing to modify the workflow engine. This approach follows the Strategy design pattern [25] where different extraction algorithms can be coupled to SciCumulus as independent cartridges. These cartridges can be dynamically changed without interference on the engine code. In this sense, the DE is a workflow-oriented approach for data manipulation during scientific workflow execution. Other domains can be modeled as new classes into SciCumulus provenance schema. This schema change is restricted to the database which is outside SciCumulus engine.

Fig. 3. Three phase activity execution

Coupling Domain-Specific Data to SciCumulus Provenance Model

Since we use a comparative genomics bioinformatics case study (SciHmm workflow), in which scientists are conducting an experiment to identify protein sequence by similarity, we have to couple domain-specific classes to the SciCumulus provenance model in order to store extracted domain-specific data. These classes have to be modeled and coupled to the provenance model by a computer science expert. This is our research focus, but, currently, this part of the approach still has to be done manually. In the case study presented in this paper, several new classes were added to SciCumulus provenance model. *Sequence* is one example of these classes. Sequences in bioinformatics contain biological information about genes, enzymes or other biological data of interest [12]. This way, sequences, genes, enzymes and organisms (including organism's taxonomy, *i.e.* genus, species, phylum, kingdom, class, family) are fundamental information to be used when a scientist is evaluating a result using steering mechanisms, as they represent domain specific terminology. Sequences are used in genomic comparative analyses to identify homologues sequences of a determined gene that are going to be used in other bioinformatics domains such as phylogeny, evolution, or molecular modeling. This way, during the life cycle of SciHmm we have to associate all these domain-specific data to existing provenance data. Fig. 4 presents the extended provenance repository using a UML class diagram. Original SciCumulus provenance classes are represented in white and domain-specific classes are represented in white inside the dashed line.

In Fig. 4 all information of the original SciCumulus provenance model is captured by the components in the execution tier of SciCumulus that captures the information related to the cloud environment. Domain-specific data is captured by DE. This provenance model is composed by four main parts: (i) elements that represent the processes executed in the cloud; (ii) elements that represent the artifacts consumed and produced by a workflow execution, (iii) elements that represent information about the cloud environment and, (iv) elements that represent domain-specific data.

The elements *VirtualMachine*, *CloudProvider*, *VirtualMachineType*, *Account*, *SecurityPolice*, *OperationSystem*, *Image*, *InstalledSoftware*, and *SoftwareOS* represent all data related to the cloud environment. Some other classes represent the structure of the workflow (*Workflow*) and its activities (*Activity*). Each activity is decomposed in several parallel tasks (*Task*) which consumes parameter values (*Relation*, *Field* and *Value*) and data files (*DataFile*). Since this provenance schema is based on the OPM recommendation [26], the classes *Activity* and *Task* are mapped as *Processes*. The class *DataFile*, *Relation* and *Value* are mapped as *Artifacts*. And class *Account* is mapped to an OPM *Agent*. It is important to highlight that all provenance data in SciCumulus is generated at runtime, *i.e.* during workflow execution.

The remaining classes (inside the dashed line in Fig. 4) represent domain-specific data that was extracted using DE and stored associated to provenance data. *Sequence* represents the smallest grain in a comparative genomic workflow. Each sequence is associated to a specific organism, which is classified using Linnaean taxonomy in

Domain, Kingdom, Phylum, Class, Order, Family, Genus and *Specie*. All this domain-specific data is obtained using NCBI Web services using sequence information extracted from files (Sequence ID). A *Sequence_Group* represents a set of sequences of one gene class present in divergent species. Each *Sequence_Group* is used as input for SciHmm workflow. Once aligned, each *Sequence_Group* produces an *MSA* which is converted to a phylib format thus producing *MSA_Converted*.

Hits represent a specific *MSA_Converted* match with a target database. *MSA*, *MSA_Converted* and *Hits* may be mapped to OPM *Artifact*. By coupling this domain-specific data to SciCumulus provenance data, scientists can perform steering queries using specific terms of the domain. For example, using the proposed data model, scientists are able to discover which organisms or species are considered in a specific ROC curve. If the partial runtime results of the ROC curve are not appropriated, there is no point to continue the execution, especially in paid public clouds. This type of query is important since each ROC curve generation need several hours of processing. If we can reduce total execution time we can spare financial resources in public clouds. Another trial may split a specific sequence group in different new instances of the *Sequence_Group* element and perform the workflow analysis separately. In addition, scientists are still able to *post* validate the execution using domain terminology. All of these features are not possible if domain-specific data is not available. In this case, all queries should be performed based on the name of data files. In the next section, we present steering queries using this data model and an overhead analysis when using DE in a parallel execution.

5 Experimental Results

In this section we present an evaluation of the proposed approach by measuring performance overheads imposed by the execution of DE and present queries for steering mechanisms in scientific workflows using runtime provenance data enriched with domain-specific data. We executed SciHmm workflow in parallel in Amazon EC2 environment using SciCumulus cloud workflow engine.

Environment and Experiment Setup

There are several types of VMs provided by Amazon EC2, such as micro, large, extra-large, high CPU extra-large, and Quadruple Extra Large. In the experiment presented in this paper we have considered just Amazon's micro types (EC2 ID: t1.micro – 613 MB RAM, 30 GB of EBS storage only, 1 core). Each instantiated VM uses Linux Cent OS 5 (64-bit), and it was configured with the necessary software, libraries like MPJ [27], and the bioinformatics applications. All instances are based on the same image (AMI ID ami-7d865614) and it was used to execute SciCumulus. According to Amazon, all VMs were instantiated in the US East - N. Virginia location and follow the pricing rules of that locality.

Fig. 4. SciCumulus provenance model enriched with domain-specific data

To execute SciHmm in parallel, our simulations use as input a dataset of multi-fasta files of protein sequences extracted from RefSeq release 48 [28]. This dataset is formed by 200 multi-fasta files and each multi-fasta file is constituted by an average of 10 biological sequences. All provenance data is persisted using PostgreSQL relational database version 8.4.6 that was configured in a dedicated Amazon EC2 virtual machine.

DE Performance Analysis

In this performance evaluation, we first measured the performance of bioinformatics programs on a single VM to analyze the local optimization before scaling up the number of VMs of the virtual cluster. Some of the existing bioinformatics programs are able to benefit from parallelism in multi-core machines. We measured the scalability using up to 128 micro-size VMs. Two separate executions of SciHmm were performed: (i) it runs SciHmm without inserting DE in the workflow (*i.e.* traditional). This way, no domain specific information is extracted; and (ii) it uses SciHmm with DE, which analyzes the produced set of data and extracts domain-specific information (*i.e.* using DE). The execution times (in hours) are in Fig. 5.

By analyzing the results we can state that the use of DE in SciHmm introduced a small overhead in each one of the executions (varying the number of VMs in the virtual cluster). In average, each DE invocation lasted for 10 seconds. The minimum overhead was of 4.1% (using 4 cores) between the traditional (without DE) and improved version (using DE). The maximum overhead imposed by the insertion of DE was of 7.3% (using 8 cores). Although the execution time increased in all cases (when using DE), as expected, scientists may choose not to use DE, in case of a well-known dataset, or keep DE and benefit from additional information to support their provenance steering queries and avoid useless executions.

Candidate Queries for Workflow Steering in SciHmm

In this subsection, we present a set of queries that are based on provenance and domain-specific data of a comparative genomic experiment and that are candidate queries to be used in steering mechanisms since all this information is generated at runtime in SciCumulus. The first query (Q1) is used for determining which sequences in available hits do not belong to a specific gene of interest (informed as parameter by scientists). Q1 allows for analyzing produced hits to remove noisy data during workflow execution. If some sequence does not belong to the informed gene, it can be removed from the space of data to be processed in a new re-execution. This way, in our relational provenance database, Q1 is modeled as the following SQL statement:

```
SELECT S.NCBI_Ref_Sequence
FROM Task T, Hits H, MSA_CONVERTED M2, MSA M1, SEQUENCE_GROUP SG,
SEQUENCE S, ORGANISM O, SPECIE SP, GENUS G
WHERE T.taskid = H.taskid AND H.msacid = M2.msacid AND M2.msaid =
M1.msaid AND M1.sgid = SG.sgid AND SG.sqid = S.sqid AND S.orgid =
O.orgid AND O.spid = SP.spid and SP.genid = G.genid AND T.exitStatus = 0
/* No error */ AND G.genus = "PLASMIDIUM"
```

Fig. 5. Performance analysis of DE overhead in SciCumulus

The second query (Q2) is used to determine how many sequences from input data are annotated as putative, hypothetical, similar, and more other similar annotations that denote that sequences are not annotated as “true genes”. This value is going to indicate the quality of the ROC curves produced at the end of SciHmm. Query Q2 can be performed after the first alignment activity, *i.e.* when the MSA is produced. However, these sequences annotated as similar, probably can be included in other studies to determine relationships between the “true annotated genes”. Scientists are able to filter undesired data to avoid unnecessary processing by Q2. It is important to highlight that in both Q1 and Q2 we have to consider only tasks that have not presented execution errors (based on provenance data). This way, Q2, in our relational provenance database, is modeled as the following SQL statement:

```
SELECT COUNT(*)
FROM Task T, Hits H, MSA_CONVERTED M2, MSA M1, SEQUENCE_GROUP SG,
SEQUENCE S
WHERE T.taskid = H.taskid AND H.msacid = M2.msacid AND M2.msaid =
M1.msaid and M1.sgid = SG.sgid AND SG.sqid = S.sqid AND T.status =
“FINISHED” AND T.exitStatus = 0 /* No error */ AND (S.NCBI_Ref_Sequence
LIKE “%PUTATIVE%” OR S.NCBI_Ref_Sequence LIKE “%HYPOTHETICAL%” OR
S.NCBI_Ref_Sequence LIKE “%SIMILAR%”)
```

6 Related Work

There are several provenance management frameworks proposed in the literature. To the best of authors’ knowledge, none of them offers support for steering queries based on runtime provenance enriched with domain-specific data. Runtime queries are particularly required in big scientific data, being processed in parallel, in HPC

environments. The existing approaches can be grouped in two categories: the first one deals with provenance data representation enriched with domain specific data and the second one provides complex and useful provenance queries for scientists.

In the first category, the most similar approach to the one proposed in this paper is Karma [29], a framework for collecting provenance from heterogeneous workflow environments. It is based on Web services and there are previous work using Karma for extracting domain-specific data. Another approach is Provenir [7], which is an upper-level ontology designed as a model for representing provenance, based on the ontology's capability of performing inference for provenance queries. There is also Janus [30], an extension of the Provenir ontology for modeling domain-specific provenance data. It is used in Taverna [22], focusing on the semantics of a specific domain on a provenance graph.

In the second category, Anand *et al.* [31] propose the effective use of provenance information represented as fine-grained relationships over nested collections of data. Due to that, the authors present a provenance model that supports multiple invocations of the same process. This model allows for multiple processes operating on the same nested data collection. This feature can be useful in domain-specific query and thus can be considered as a first step to acquire domain specific information and associate it to produced provenance data. Gadelha *et al.* [32] introduce a set of query patterns that can be identified in provenance queries of Swift's parallel scripting system [33]. Their provenance management system is focused on large-scale many-task scientific computations. They developed a data model following the OPM specification, with extensions that enrich core structural provenance data, represented as consumption and production relationships between applications and data sets. Domain-specific data is modeled through free text user annotation. They contribute by proposing a query interface to leverage SQL syntax.

Although all of these approaches represent a step forward and many of them can be seen as complementary to our work, none of them provide for provenance data query at runtime. Thus, they can be helpful in planning the next workflow execution, but they cannot be used to provide steering mechanisms based on runtime structured queries. The approach proposed in this paper focus on a different perspective, extracting runtime domain-specific data from data files and representing them related to workflow provenance data.

7 Conclusions

Scientific workflows are used to represent the steps of a scientific experiment. SWfMS manage workflow's programs and data for validating or refuting a scientific hypothesis. Provenance data is fundamental to allow for validation and reproducibility of workflows. Although provenance data is mainly used for a *post* validation and analysis of the experiment, there are other possible usages such as workflow execution steering.

In order to provide workflow steering capabilities, it is necessary that provenance be made available during workflow execution. By steering, scientists may interfere in

the execution course interrupting or adjusting parameters based on a runtime analysis. In previous work, we started to provide for provenance data query at runtime [17]. However, scientists cannot decide how to interfere on the execution using provenance data (activity start time, errors, data files produced) disconnected from domain-specific data. In many cases, to suspend or stop an execution, scientists have to analyze if the current solution meets a preset criterion, which is usually based on domain-specific data. This type of data is not available for runtime querying in related work.

This paper proposes an approach for extracting domain-specific data from produced data files and storing them along with provenance data. This allows for scientists to create queries that can be used as a basis for steering mechanisms, thus allowing for verifications based on provenance and domain-specific data at runtime. One of the advantages of our solution is modeling domain-specific data in the same formalism of the workflow algebra [17] underneath the execution engine. This algebra allows for a formal control of the execution, providing for the consistent sequence of the workflow execution, after the scientist's interference. The proposed approach is promising, since it presents small runtime overhead and allows for building complex steering mechanisms based on enriched provenance data with domain-specific data.

Acknowledgments. This work was partially sponsored by CNPq, FAPERJ and CAPES.

References

- [1] Taylor, I.J., Deelman, E., Gannon, D.B., Shields, M.: *Workflows for e-Science: Scientific Workflows for Grids*, 1st edn. Springer (2007)
- [2] Freire, J., Koop, D., Santos, E., Silva, C.T.: Provenance for Computational Tasks: A Survey. *Computing in Science and Engineering* 10(3), 11–21 (2008)
- [3] Vaquero, L.M., Roderio-Merino, L., Caceres, J., Lindner, M.: A break in the clouds: towards a cloud definition. *SIGCOMM Comput. Commun. Rev.* 39(1), 50–55 (2009)
- [4] Simmhan, Y.L., Plale, B., Gannon, D.: A survey of data provenance in e-science. *ACM SIGMOD Record* 34(3), 31–36 (2005)
- [5] Factor, M., Henis, E., Naor, D., Rabinovici-Cohen, S., Reshef, P., Ronen, S., Michetti, G., Guercio, M.: Authenticity and provenance in long term digital preservation: modeling and implementation in preservation aware storage. In: *First Workshop on Theory and Practice of Provenance*, Berkeley, CA, USA, pp. 6:1–6:10 (2009)
- [6] Groth, P., Deelman, E., Juve, G., Mehta, G., Berriman, B.: Pipeline-centric provenance model. In: *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, Portland, Oregon, pp. 1–8 (2009)
- [7] Sahoo, S., Sheth, A.: Provenir ontology: Towards a Framework for eScience Provenance Management. In: *Microsoft eScience Workshop*, Pittsburgh, PA, pp. 15–17 (2009)
- [8] Wolstencroft, K., Alper, P., Hull, D., Wroe, C., Lord, P.W., Stevens, R.D., Goble, C.A.: The myGrid ontology: bioinformatics service discovery. *Int. J. Bioinformatics Res. Appl.* 3(3), 303–325 (2007)

- [9] Crawl, D., Altintas, I.: A Provenance-Based Fault Tolerance Mechanism for Scientific Workflows. In: Freire, J., Koop, D., Moreau, L. (eds.) IPAW 2008. LNCS, vol. 5272, pp. 152–159. Springer, Heidelberg (2008)
- [10] de Oliveira, D., Ogasawara, E., Seabra, F., Silva, V., Murta, L., Mattoso, M.: GExpLine: A Tool for Supporting Experiment Composition. In: McGuinness, D.L., Michaelis, J.R., Moreau, L. (eds.) IPAW 2010. LNCS, vol. 6378, pp. 251–259. Springer, Heidelberg (2010)
- [11] Missier, P.: Incremental workflow improvement through analysis of its data provenance. In: 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP 2011), Heraklion, Crete, Greece (2011)
- [12] Ocaña, K.A.C.S., Oliveira, D., Dias, J., Ogasawara, E., Mattoso, M.: Optimizing Phylogenetic Analysis Using SciHm Cloud-based Scientific Workflow. In: 2011 IEEE Seventh International Conference on e-Science (e-Science) IEEE e-Science 2011, Stockholm, Sweden, pp. 190–197 (2011)
- [13] Guerra, G., Rochinha, F., Elias, R., Oliveira, D., Ogasawara, E., Dias, J., Mattoso, M., Coutinho, A.L.G.A.: Uncertainty Quantification in Computational Predictive Models for Fluid Dynamics Using Workflow Management Engine. *International Journal for Uncertainty Quantification* 2(1), 53–71 (2012)
- [14] Ogasawara, E., Oliveira, D., Chirigati, F., Barbosa, C.E., Elias, R., Braganholo, V., Coutinho, A., Mattoso, M.: Exploring many task computing in scientific workflows. In: *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, MTAGS 2009*, Portland, Oregon, USA, pp. 1–10 (2009)
- [15] Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., et al.: Examining the Challenges of Scientific Workflows. *Computer* 40(12), 24–32 (2007)
- [16] Dias, J., Ogasawara, E., Oliveira, D., Porto, F., Coutinho, A., Mattoso, M.: Supporting Dynamic Parameter Sweep in Adaptive and User-Steered Workflow. In: *6th Workshop on Workflows in Support of Large-Scale Science WORKS 2011*, Seattle, WA, USA, pp. 31–36 (2011)
- [17] Oliveira, D., Ogasawara, E., Ocaña, K., Baiao, F., Mattoso, M.: An Adaptive Parallel Execution Strategy for Cloud-based Scientific Workflows. *Concurrency and Computation: Practice and Experience* (2011) (online)
- [18] Miller, W., Makova, K.D., Nekrutenko, A., Hardison, R.C.: Comparative Genomics. *Annual Review of Genomics and Human Genetics* 5(1), 15–56 (2004)
- [19] Clark, A.G.: Genomics of the evolutionary process. *Trends in Ecology & Evolution* 21(6), 316–321 (2006)
- [20] Baldi, P., Brunak, S., Chauvin, Y., Andersen, C.A.F., Nielsen, H.: Assessing the accuracy of prediction algorithms for classification: an overview. *Bioinformatics* 16(5), 412–424 (2000)
- [21] Callahan, S.P., Freire, J., Santos, E., Scheidegger, C.E., Silva, C.T., Vo, H.T.: VisTrails: visualization meets data management. In: *SIGMOD International Conference on Management of Data*, Chicago, Illinois, USA, pp. 745–747 (2006)
- [22] Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., Oinn, T.: Taverna: a tool for building and running workflows of services. *Nucleic Acids Research* 34(2), 729–732 (2006)
- [23] Amazon EC2, Amazon Elastic Compute Cloud (Amazon EC2) (2010), <http://aws.amazon.com/ec2/>

- [24] Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., Mattoso, M.: An Algebraic Approach for Data-Centric Scientific Workflows. *Proc. of VLDB Endowment* 4(12), 1328–1339 (2011)
- [25] Gamma, E., Helm, R., Johnson, R., Vlissides, J.M.: *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional (1994)
- [26] Moreau, L., Freire, J., Futrelle, J., McGrath, R.E., Myers, J., Paulson, P.: The Open Provenance Model: An Overview. In: Freire, J., Koop, D., Moreau, L. (eds.) *IPAW 2008*. LNCS, vol. 5272, pp. 323–326. Springer, Heidelberg (2008)
- [27] Carpenter, B., Getov, V., Judd, G., Skjellum, A., Fox, G.: MPJ: MPI-like message passing for Java. *Concurrency: Practice and Experience* 12(11), 1019–1038 (2000)
- [28] Pruitt, K.D., Tatusova, T., Klimke, W., Maglott, D.R.: NCBI Reference Sequences: current status, policy and new initiatives. *Nucleic Acids Research* 37(Database issue), D32–D36 (2009)
- [29] Simmhan, Y.L., Plale, B., Gannon, D.: A Framework for Collecting Provenance in Data-Centric Scientific Workflows. In: *ICWS*, pp. 427–436 (2006)
- [30] Missier, P., Sahoo, S.S., Zhao, J., Goble, C., Sheth, A.: *Janus*: From Workflows to Semantic Provenance and Linked Open Data. In: McGuinness, D.L., Michaelis, J.R., Moreau, L. (eds.) *IPAW 2010*. LNCS, vol. 6378, pp. 129–141. Springer, Heidelberg (2010)
- [31] Anand, M.K., Bowers, S., McPhillips, T., Ludäscher, B.: Exploring Scientific Workflow Provenance Using Hybrid Queries over Nested Data and Lineage Graphs. In: Winslett, M. (ed.) *SSDBM 2009*. LNCS, vol. 5566, pp. 237–254. Springer, Heidelberg (2009)
- [32] Gadelha, L., Mattoso, M., Wilde, M., Foster, I.: Provenance Query Patterns for Many-Task Scientific Computing. In: *USENIX Workshop on the Theory and Practice of Provenance (TaPP)*, Heraklion, Crete, Greece (2011)
- [33] Zhao, Y., Hategan, M., Clifford, B., Foster, I., von Laszewski, G., Nefedova, V., Raicu, I., Stef-Praun, T., Wilde, M.: Swift: Fast, Reliable, Loosely Coupled Parallel Computation. In: *3rd IEEE World Congress on Services*, Salt Lake City, USA, pp. 199–206 (2007)