Performance Analysis of Data Filtering in Scientific Workflows

João Gonçalves[1],Daniel de Oliveira[2],Kary Ocaña[1], Eduardo Ogasawara[1,3], Jonas Dias[1], Marta Mattoso[1]

[1] Federal University of Rio de Janeiro (COPPE/UFRJ), Brazil
[2] Fluminense Federal University (UFF), Brazil
[3] Federal Center of Technological Education (CEFET/RJ), Brazil
{jcg, kary, jonasdias, marta}@cos.ufrj.br, danielcmo@ic.uff.br, eogasawara@cefet-rj.br

Abstract. A major issue during scientific workflow execution is how to manage the large volume of data to be processed. This is more complex in clouds where all resources are configurable in a pay per use model. A possible solution is to take advantage of the exploratory nature of the experiment and adopt filters to reduce data flow between activities. During a data exploration evaluation, scientists may discard superfluous data (which is producing results that do not comply with a given quality criteria) produced during the workflow execution, avoiding unnecessary computations in the future. We claim that the final decision on whether to discard superfluous data may become feasible only when workflows can be steered by scientists at runtime using provenance data enriched with domain-specific data. In this article, we introduce Provenance Analyzer (PA), which is an approach that allows for examining the quality of data during the workflow execution by querying provenance. PA removes superfluous data, improving execution time that typically lasts for days or weeks. This is possible as PA relies on data centric workflow algebra. In this context, PA plays the role of filter operator in the algebra. Scientists are able to change filter criteria during workflow execution according to the behavior of the execution. Our experiments use a real phylogenetic analysis on top of SciCumulus cloud workflow engine. Results show data reduction of 23%, which led to performance improvements of up to 36.2% when compared to a workflow without PA.

Categories and Subject Descriptors: H. Information Systems [**H.2.8 Database Applications**]: Scientific databases

Keywords: cloud computing, scientific experiments, scientific workflows

1. INTRODUCTION

In the current scientific scenario, many experiments are based on the use of complex scientific apparatus and computational simulations that consume and produce large volumes of data. To aid the management of this type of experiments, scientific workflows [Taylor et al. 2007][Mattoso et al. 2010] play a fundamental role in many application domains [Hey et al. 2009]. Scientific Workflow Management Systems (SWfMS) [Taylor et al. 2007] are complex systems that provide ways for modeling and executing workflows. There are many SWfMS available such as VisTrails [Callahan et al. 2006] and Swift [Wilde et al. 2009]. Each of them is focused on different aspects such as semantics, provenance and performance. Due to the growth of data production, many of these scientific workflows require data parallelism and High Performance Computing (HPC) environments, such as clusters, grids or clouds. Cloud computing [Vaquero et al. 2009] is a paradigm that offers ways for running scientific workflows in parallel based on a pay per use mode. Scientific workflows benefit from the elasticity and availability of virtual computing resources.

In a scenario where workflows produce large volumes of data flow, their execution may last for days or weeks even in HPC environments. Due to that, it is fundamental to improve the performance

The authors thank CNPq, CAPES and FAPERJ for partially sponsoring this research.

of parallel workflow execution as well as reducing the total execution time. In clouds, there is also the need of reducing the financial cost. Since scientists pay in accordance to the usage of resources when they run workflows in clouds, a long-term execution can become financially unviable. One way of improving workflow execution time is to reduce data to be processed during the course of the workflow execution. Filtering is widely used in databases and data mining and it is the transformation of a large amount of data into a corrected, ordered, and simplified form. The basic concept is the reduction of large amounts of data down to the meaningful parts.

Currently, scientists manage the design of the data-flow manually, defining quality or filtering criteria to be used as part of pre-selection of the data-flow at each phase of the workflow. However, due to the vast amount of data involved in data-intensive experiments, this filtering becomes difficult to be planned *a priori*, since it is defined based on runtime analysis of partial results. During a workflow execution, some produced data, which do not comply with quality criteria, may be discarded, avoiding unnecessary computations in the future. However, to predict when data is produced in the execution course of the workflow is a complex task. There are several parameters that have to be examined, and this choice depends on the behavior of the workflow execution. This data filtering may be obtained by scientists if they can analyze data generated during workflow execution, also known as workflow steering [Dias et al. 2011]. Scientists have to be able to verify data quality of partial results, filter this data and avoid further activities to consume low quality data. We claim that final decision on whether to filter and reduce data volume to be explored has to be taken by scientists using domain-specific data based on the execution behavior. This would reduce the time that is spent on processing low quality (or even irrelevant) data and minimize total execution time. Provenance information [Freire et al. 2008] has valuable data to allow for this quality analysis. However, in most SWfMS scientists only have access to provenance data, *i.e.* the behavior of the workflow, after the complete execution of the workflow. Consequently, changing the workflow execution before it finishes can only be done based on data file browsing, since provenance data is not available to be queried at runtime. Scientists need a mechanism for interfering with the workflow execution, not only to abort but also do adjust parameters, such as quality criteria. In this scenario, the goal is to establish events that can determine if produced data is valid or not to be consumed by the next activity in the workflow. We call this event as "reducing the set of data to be processed". Let us illustrate the need of allowing runtime filtering with a data-intensive parallel workflow in the phylogenetic bioinformatics domain. We are going to use this example consistently in the rest of the article.

Phylogenetic experiments aim at producing phylogenetic trees to represent existing evolutionary relationships. SciPhy [Ocana et al. 2011] represents a phylogenetic analysis as a scientific workflow. SciPhy executes in parallel in Amazon EC2 using a specific cloud workflow engine named SciCumulus [Oliveira et al. 2010], which manages parallel workflow execution in a set of virtual machines (VMs) that form a virtual cluster in the cloud. A phylogenetic analysis requires a complex workflow composed by several data-intensive activities that may take considerable time, *e.g.* weeks, to produce the desired result. A typical phylogenetic analysis workflow (Figure 1) consumes several input data files containing a set of DNA, RNA, or amino acid sequences, and produces phylogenetic trees to be further analyzed by scientists. In Figure 1, each rectangle indicates an activity, solid lines represent input and output parameters that are transferred between activities, and dashed lines represent input and output parameters shared through data files. For a single exploratory phylogenetic analysis, SciPhy may consume 2,000 input multi-fasta files and may produce more than 14,000 (about 8.5GB) phylogenetic trees. Since SciPhy is a typical exploratory workflow, all of its activities, are repeatedly executed for many different input multi-fasta files, and depending on the amount of biological sequences, their length and the number of hits (*i.e.* similarity degree) found in each one of the multi-fasta files, each single activity execution may take hours to produce results, even when input data present low quality. As workflows scale to 10,000 and sometimes 100,000 or more parallel executions of a specific activity, if we do not avoid processing irrelevant data, the performance is affected since we spend more time and money than actually necessary. Scientists could improve workflow execution if they could

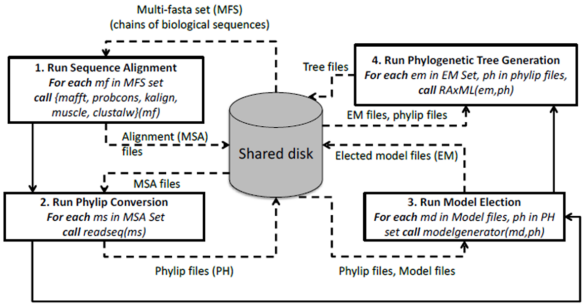

Fig. 1. SciPhy scientific workflow

analyze provenance data at runtime and change quality criteria (filter) during workflow execution. For example, let us consider that an execution of SciPhy is consuming more time than expected. In this case, scientists could change filtering criteria at runtime, as for example, the e-value. The e-value is a biological criterion typically used to assess the quality of phylogenetic trees. Some standard e-values are suggested in the literature but this value is highly dependent on the input data. In case where many similarities are found, the criterion may be more restrictive, and conversely, when similarities are rare, the criterion can be more flexible. By changing the e-value during execution, a higher degree of quality is presented on the output results of an activity. In Figure 1, this action would prevent Activity 4 from processing irrelevant data.

In this article, we address the problem of improving the performance of parallel execution of scientific workflows by reducing the set of data to be processed. We propose a way for scientists to eliminate intermediate data that do not comply with quality criteria by analyzing provenance data at runtime. These performance improvements are possible using Provenance Analyzer (PA), which is a component that is deployed in SWfMS. PA allows for querying provenance data and intermediate files (based on a criteria defined and submitted at runtime) and filter these data based on specific informed criteria at runtime. This approach is possible since SciCumulus uses data-centric workflow algebra [Ogasawara et al. 2011], and provenance is based on the relational model. PA plays the role of a filter operator in the workflow algebra. As a result, PA is independent of the application domain data structures, *i.e.* scientists can configure PA at runtime according to their needs (using domain specific filters). The main contributions of this article are: (i) the Provenance Analyzer component, which allows provenance queries and the automatic reduction of the set of data to be processed and (ii) a thorough experimental evaluation based on the implementation of PA in SciCumulus workflow engine. We executed the phylogenetic analysis workflow on a 128-core virtual cluster on Amazon EC2 and obtained a performance improvement of up to 36% while using PA in SciCumulus.

This article is organized as follows. In Section 2, we discuss related work. We describe in Section 3 the workflow algebra and the SciCumulus parallel workflow engine used in the experiments. In Section 4, we present the proposed approach while in Section 5 we discuss experimental results. Next, Section 6 presents final remarks.

2. RELATED WORK

There are few approaches in the literature that improve parallel workflow execution performance by avoiding the unnecessary use of resources through provenance analysis. [Missier 2011] proposes an adaptive control of a scientific workflow execution, using provenance information, demonstrating that provenance can be used for workflow control. Two other proposals are closely related to data quality issues, emphasizing the quality of data used during the workflow execution and the filtering using threshold (defined specifically by the scientists). The first one is related to a Quality of Data (QoD) measurement framework for scientific workflows [Reiter et al. 2011], focusing on the validation of

data in two phases: (i) the Analysis phase, where the desired characteristics of data is computed and (ii) the Evaluation phase that uses the pre-computed characteristics to evaluate the quality of data. The second proposal [Na'im et al. 2010] presents the Data Quality Monitor for the Kepler SWfMS, enabling users to inform the quality threshold value before workflow execution, as well as, providing visualization support for the quality evaluation of the results. [Dias et al. 2011] highlighted the importance of provenance queries at runtime to improve workflow execution by reducing slices of parameter space in parameter sweep workflows. [Dias et al. 2011] propose control structures as user-steering points and adjustment knobs for running workflow on large clusters. However, [Dias et al. 2011] do not take into account data quality issues in their paper and their approach is not connected to the workflow algebra used in this article. The works [Missier 2011], [Reiter et al. 2011], [Na'im et al. 2010] do not provide for provenance query at runtime, which makes monitoring data quality very complex.

This article is a step forward as it uses PA to address data reduction in cloud environments by allowing scientists to inform domain-specific filtering criteria at runtime. Our approach complements these related proposals by proposing a new type of execution control and data quality analysis, comprising: (i) the possibility of reducing the set of intermediate data to be processed during parallel workflow execution; (ii) the usage of a workflow algebra and its operators, which enable a uniform internal provenance data representation and analysis and; (iii) the application of software engineering methods, using design patterns [Gamma et al. 1994], thus enabling scientists to encapsulate existing third party code to the PA.

3. SCICUMULUS CLOUD WORKFLOW ENGINE

SciCumulus is a cloud workflow engine that aims at scheduling, monitoring, and load balancing the parallel execution of scientific workflow activities in clouds [Oliveira et al. 2010]. SciCumulus orchestrates workflow activities execution on a virtual cluster that is composed by a set of VMs that exchange communication messages (Figure 2(A)). All virtual cluster configurations are also performed by SciCumulus accessing the cloud provider API. SciCumulus is based on four tiers that are also distributed. The client tier starts the parallel execution of activities by staging data in and out the cloud and then dispatches the parallel execution of the activity. This tier is deployed in SWfMS, such as VisTrails [Callahan et al. 2006]. The distribution tier manages the parallel execution of the activities in computing clouds by creating cloud activities (*i.e.* activity execution) that contain the program to be executed, parameter values and input data to be consumed. The execution tier invokes executable codes in several VMs of the virtual cluster. The execution tier is also responsible for capturing provenance data and to send it to a provenance repository (fundamental for the approach proposed in this paper). This provenance repository is also located in the cloud. The data tier is responsible for storing input data and provenance data. This data tier has information about the environment characteristics collected by an autonomous agent.

The SciCumulus execution model follows the workflow algebra proposed by [Ogasawara et al. 2011] that encapsulates workflow activities and provide for basic operators to be manipulated by the workflow execution engine. This algebra is inspired on the concepts of relational algebra of databases. Algebraic operators rule workflow activities, according to how activities consume and produce data. This workflow algebra defines its operands as a uniform data representation consistently used throughout the workflow execution, where all data is represented as relations. 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 file). These operators also define the ratio between input data consumption and output data production. The parameter values for the activities are represented as attributes in a tuple, whereas the set of tuples composes the relation to be consumed by an activity. A scientific workflow definition is then mapped to a set of algebraic expressions placed in a coherent flow. The approach proposed by Ogasawara et al. includes six operators (explained following) as presented in

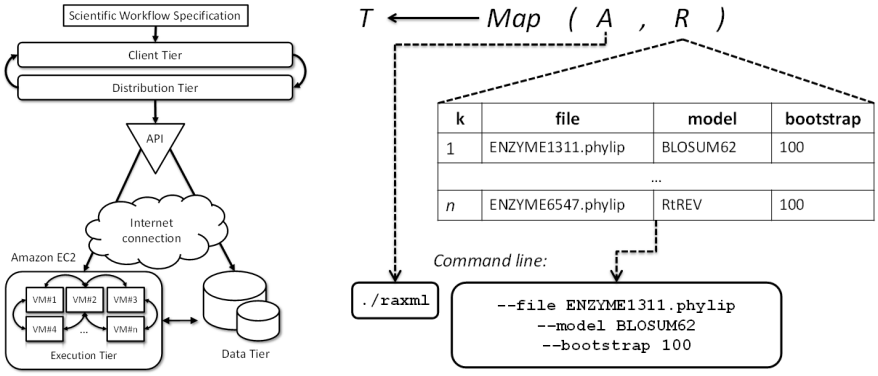

Fig. 2. (A) SciCumulus conceptual architecture and (B) Activity RAxML ruled by Map operator

Table I.

Table I. Algebra Operations (Ogasawara et al., 2011)

Operator	Type of activity operated	Result	Ratio of consumed and produced tuple
Map	Program	Relation	1:1
SplitMap	Program	Relation	1:m
Reduce	Program	Relation	n:1
Filter	Program	Relation	1:(0-1)
SRQuery	Relation algebra expression	Relation	n:m
MRQuery	Relation algebra expression	Relation	n:m

The operator Map is the simplest operator. It is related to the direct execution of a program that consumes one single input tuple and produces one single output tuple. The activity ruled by the SplitMap operator fragments and decomposes a single input tuple into several output tuples. In this case, the input tuple must have a pointer to a file, which is analyzed and processed to extract information to generate the new tuples. Note that the extraction rules are defined by scientists for each different problem. The Reduce operator receives several input tuples and aggregates all of them into a single output tuple. It is similar to the Reduce function in the MapReduce model [Dean and Ghemawat 2010] and it requires a grouping attribute to work on. The Filter operator receives a single tuple as input, and according to a specific criterion, it produces (or not) a single output tuple. The SRQuery and MRQuery operators are used to execute traditional relational algebra expressions such as selections, projections, and joins on workflow relations. While SRQuery consumes a single relation to produce another single relation, MRQuery consumes a set of relations. Figure 2(B) presents one example of the activity RAxML of SciPhy ruled by a Map operator. Note that all parameters of this activity (presented in the command line) are mapped to relation attributes. Each tuple in this case can be processed in parallel in different VM.

4. USING PROVENANCE ANALYZERS FOR DATA FILTERING

Aiming at providing a solution for the problem of reducing the set of data to be processed, and, consequently, reducing workflow execution time, we propose the effective use of PA for analysis and validation of the data that flows along the workflow. In addition to data filtering, this analysis can help improving data quality in the workflow since scientists are able to analyze corresponding provenance information at runtime. A PA is an artificial activity, which encapsulates a data extractor (DE - that can be coded by scientists) and a Filter operation. Note that the PA cannot be implemented using other query operators such as SRQuery and MRQuery because both SRQuery and MRQuery produce

a tuple as output. On the other hand, a filter operation only informs if an input tuple should be copied to output or not, which is more appropriate. PA can be included at any position of the workflow, placed at the beginning of the flow of activities, or its end or even placed between two activities of the workflow according to scientists' analysis need. A PA is used to extract important data from the data produced by the incoming activity. This data is commonly domain-specific data, which can enrich provenance data. The provenance data enriched with domain-specific data is used to evaluate if data quality is acceptable for assessing. If so, data can be transferred to the outcoming activity. PA is a supervised approach that requires scientists to define the criteria that is expressed according to provenance information to perform filtering. Although this type of filtering process is not new in data and text mining domains [Feldman and Sanger 2007], the main advantage here is to couple the filtering process with the domain-specific provenance extraction.

The execution of a PA follows a 2-phase procedure: the provenance extraction and the filtering. The extraction phase is already provided by the workflow algebra execution model during the activation process [Ogasawara et al. 2011]. It focuses on analyzing data produced by the workflow activities (including data files used as an activity's input or output data) and extracting domain-specific data from these files [Gonçalves et al. 2012]. Although the OPM/PROV [Moreau and Missier 2011] recommendations consider as "provenance" only the sources of information, we extend this view by considering the relationship of these provenance data to intermediate data files and their content. This domain information, extracted from the contents of files, is used for analyzing quality of data. Each PA invokes a third party DE (external program) that is responsible for reading an input tuple (which contains a pointer to a file), and opening this file for analysis. Based on the result it inserts new attributes in the tuple for representing domain data to be further used. In the second phase, *i.e.* after invoking the DE, an algebraic Filter operation is performed [Ogasawara et al. 2011]. This Filter is based on a criteria informed at runtime by scientists. It copies the input tuple to the output of the PA if and only if scientists' quality criteria are met. Moreover, each filter follows specific quality criteria defined by scientists which are based on the previously extracted domain-specific data. One example of the extraction process is presented in Figure 3(A) where the execution of this PA is placed between the Data Conversion and Model Election activities of SciPhy. In this case, the ENZYME1311.fasta file is produced by Data Conversion activity and is given as input for the PA, which extracts information from the file pointed in the input relation, and validates the data file according to scientists' criteria. For each extracted data, a new attribute value is added to the output relation. At the example, the scientist's criteria was defined as the specific file format for the simulation (IS_VALID), the origin of the data (FASTA_ORIGIN_ID) and the correct format of the protein sequence (FASTA_SEQ), all of them part of the data to be validated before its usage as input for an activity. The attributes in this example may be generated in different forms, depending on the filter criteria informed by scientists at runtime.

The PA approach is implemented as a component, called SciCumulusPA (Figure 3(B)), for SciCumulus engine. An instance of SciCumulusPA can be inserted at any position of the workflow, making the approach pervasive and user steered. The current version of the PA component is implemented using Python version 2.7, and the dynamic choice of DE and filter criteria is based on the Strategy design pattern [Gamma et al. 1994]. The Strategy design pattern enables the selection of a specific DE and filter criteria at runtime. Each DE code is also classified by its context, according to information about specific activities in which the algorithm can be applied (*e.g.* compatible formats of input data). The component also performs necessary operations, such as provenance database access, separating these operations from the specific DE code, created by the scientists. The SciCumulus workflow XML definition file has to be instrumented to insert the necessary PAs along the flow. Scientists initially specify the workflow (XML file) with activities and dependencies. In the beginning of the workflow execution, SciCumulus instruments the workflow, including all PA in the workflow to analyze and validate the data products generated by activities reducing the set of data to be processed (if possible). In its current version, SciCumulusPA accesses a table in the provenance database to acquire

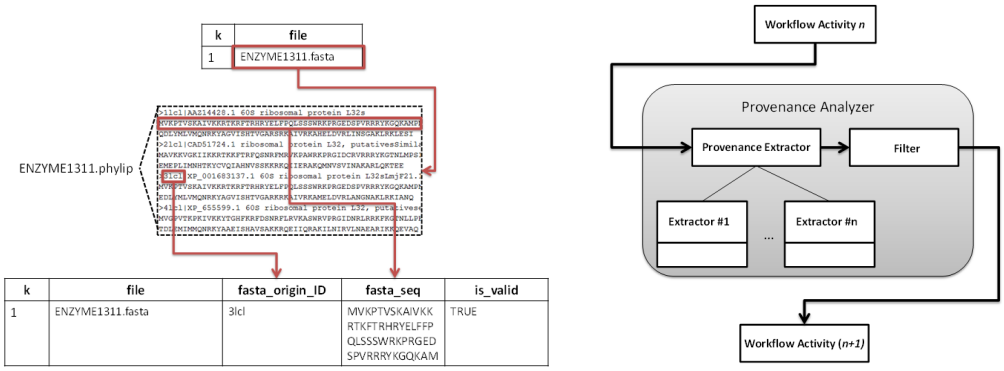

Fig. 3. (A) Provenance Analyzer execution and (B) SciCumulusPA implementation

the quality criteria. In this table we represent specific attributes, relational operators such as $<$, $>$, $=$ and the value associated. This table also has an USER_ID to associate each quality criterion to a specific scientist, thus only affecting his/her workflows. The provenance query is dynamically created and executed. An example of generated query is presented following, based on the previous example. In this case, scientists can eliminate invalid files (according to IS_VALID attribute) or reduce the number of organisms for a specific genus and biological family (domain-specific information). In the context of this article these provenance queries are represented as SQL statements and RELATION is the name of the relation produced by activities. In Listing 1, ORGANISM, SPECIES, GENUS and FAMILY are tables stored with domain information downloaded from NCBI [Gonçalves et al. 2012].

Listing 1. SQL statement for domain-specific filtering in SciCumulusPA

```
SELECT * FROM RELATION r, ORGANISM o, SPECIES s, GENUS g, FAMILY f
WHERE IS_VALID = True
AND r.FASTA_ORIGIN_ID = o.ID
AND o.SL_ID = s.ID
AND s.GN_ID = g.ID
AND g.FM_ID = f.ID
AND g.NAME IN ('Trypanosoma')
AND f.NAME IN ('Trypanosomatidae')
AND r.E_VALUE > 1e-5
```

5. EVALUATING PROVENANCE ANALYZERS USING SCIPHY WORKFLOW

In this section we present an evaluation of the proposed approach by measuring performance improvements obtained by reducing the set of data to be processed during workflow execution. We executed SciPhy workflow in parallel in Amazon EC2 environment using SciCumulus workflow engine. The main idea is to measure and analyze the performance gains with PA. In the experiments presented in this article we have instantiated Amazon's micro VMs (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, and the bioinformatics applications. All instances are based on the same image (ami-7d865614) and it was used to execute SciCumulus. To execute SciPhy in parallel, our simulations use as input a dataset of multi-fasta files of protein sequences extracted from RefSeq release 48 [Gonçalves et al. 2012]. This dataset is formed by 200 multi-fasta files and each multi-fasta file is constituted by an average of 10 biological sequences. To perform phylogenetic analysis, once downloaded, each input multi-fasta file is processed using the following versions of the programs: ClustalW 2.1, Kalign 1.04, MAFFT 6.857, Muscle 3.8.31, ProbCons 1.12, ModelGenerator 0.85, and RAxML-7.2.8-alpha. Specific filtering requirements for each PA in SciPhy are defined by scientists that have experience with the workflow and also with the program involved. For each ac-

tivity, specialists analyze provenance data and the validation code. In this experiment a provenance query named Q1 (presented in Listing 1) is used for determining whether homologue sequences comply with a specific e-value. This query Q1 was encapsulated into a PA in the beginning of the workflow and before phylogenetic tree generation activity. The following criterion was considered in our study to process input homologues sequences. It was considered an e-value of 1e-05 as cut-off, thus the sequences that reached this e-value were considered as homologues and included as input for the next activity. If we do not filter input data, several problems may occur. For example, we observed that some sequences were erroneously included in input dataset (generated by SciHmm workflow [Ocana et al. 2011]), *i.e.* sequences that were annotated as other enzymes, hypothetical, partial, probable conserved protein, or sequences that could represent distant homologues or paralogues, which can introduce error in phylogenetic analyses as well as inducing to increment the total execution time in the execution of this type experiments. This data represents thousands of files, parameters and activities in the workflow execution that can last for weeks. There is no way to get information from these data in an *ad-hoc* way. By using current solutions, the analysis would be performed only after the execution, thus leaving no room for performance improvements. Actually, in several cases the whole execution would be interrupted and resubmitted with different inputs/parameters.

In this performance evaluation, we first measured the performance of bioinformatics programs on a single VM to analyze the local optimization. Then we measured the performance and scalability using up to 128 VMs. Two separate executions of SciPhy were performed: (i) the first was started running SciPhy without inserting PA in the workflow (*i.e.* conventional). This way, even when elements in the set of data are "irrelevant", they are processed; and (ii) the second one was using SciPhy with PA, which reduces the set of data to be processed (*i.e.* improved). The measurements of execution time (in hours) are summarized in Figure 4 and Table II. The performance gains presented in Table II refer to the improvements achieved with the reduction of the set of data to be processed in each execution. The filter criteria informed was the e-value superior to 1e-5 (cut-off value). In each execution of SciPhy 23% of produced data was discarded. By analyzing the results we can state that the use of PA in SciPhy introduced performance gains for each one of the executions. The smallest performance gain obtained was of 20.5%, which represents an absolute difference of more than one day of computations (25.93 hours of total processing) between the conventional and improved version of SciPhy running using 16 VMs. The overhead imposed by the insertion of PA can be considered negligible in this experiment since each PA execution (considering extraction and Filter phases) took about 10 seconds to finish in average for each activity execution. Since workflow activities are mostly CPU intensive (consuming at least 30 minutes), this time represents less than 1% of the total execution time of the activity. We can also state that the total execution time decreases, as expected, when SciCumulus provided more VMs for executing SciPhy. For example, the total execution time was reduced from approximately 39 days (936.77 hours - using one core) to approximately 4 days (100.58 hours - using only 16 cores). This led to a speedup of 9.31, which is still a very significant improvement for cloud computing. This behavior can be explained since this experiment is embarrassingly parallel, which is one of the most scalable patterns for clouds, yet very frequent in bioinformatics workflows.

Table II. Execution time of the SciPhy workflow

Number of Cores	Conventional	Improved	Performance gain
1	1269.21	936.77	26.1%
2	746.59	585.48	21.5%
4	408.96	320.24	21.6%
8	220.18	162.28	26.2%
16	126.51	100.58	20.5%
32	82.36	59.73	27.4%
64	68.16	49.16	27.8%
128	54.10	34.51	36.2%

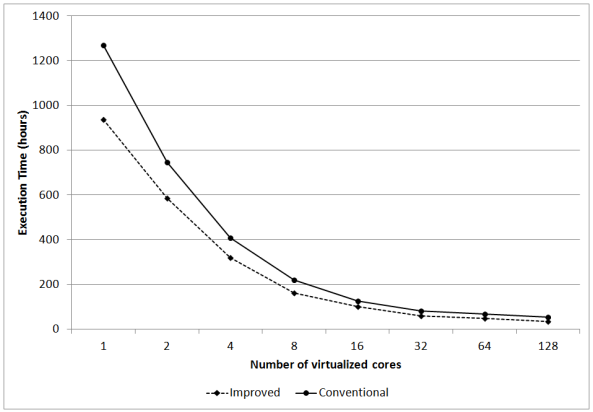

Fig. 4. Total execution time of SciPhy

Although the main focus of this article is related to performance analysis, we decided to measure the quality of the results, using precision/recall metrics [Baeza-Yates and Ribeiro-Neto 1999]. It is fundamental that PA not only reduces total execution time, but it eventually does that with quality improvement. To better understand the quality of produced results using PA and to know how much sequences needed are excluded before tree generation, we perform a comparison varying the e-value parameter during SciPhy execution. This way, we executed SciPhy using a 48 fasta files as input and using default e-value = 10.0 and e-value = 1e-05 during execution. After that we performed a manually verification of the results. When we associated default e-value (10.0) to Q1 all 48 sequences were considered. However, only 17 of them comply with the biological criteria needed to be included in the study, *i.e.* 35% of the input dataset. If we set PA to filter data using an e-value of 1e-5 we only consider 16 sequences to process. It means that using PA we only processed high quality results, however we neglected one sequence file that does not comply with e-value 1e-5 but has biological meaning. It means that using PA we achieved 100% of precision and 94% of recall. Although these results are promising, more complex analysis are effectively needed. For instance, this neglected sequence can belong to the group of distant homologues sequences that depending on the interest of the specialist, can be included (or not) in the study.

6. FINAL REMARKS

Large scientific experiments have long duration when many executions explore different parameters and input data. These executions are compute-intensive thus requiring techniques to improve performance especially in clouds where financial costs are involved and directly dependent of the total execution time. A phylogenetic analysis is one of several scientific workflows that need parallel mechanisms. In this article we introduce Provenance Analyzer (PA) to extract provenance from produced data and to use this information in runtime queries to reduce the set of data to be processed along the flow. The main advantage of using PA is that they are generic in representing scientists' quality criteria since scientists define the extraction and filter phases of PA according to the experiment execution behavior. Querying provenance at runtime is important since it allows for quality-based execution adjustments that otherwise would be impossible or too complex to be pre-programmed. In our experiments, SciPhy has been executed on top of the Amazon EC2 using SciCumulus engine. Performance results presented benefits of up to 36.2% when using PA with filtering level about 23% of the produced data. Note that this result is obtained using a specific domain of a specific phylogenetic analysis execution. Although the proposed approach is an ongoing work, this article contributes by using PA to show the potential of analyzing provenance data during execution time to improve the overall performance of the workflow when it involves thousands of activity executions and data products. Since PA may be considered a sub-workflow for provenance analysis it should be also defined in the workflow language as a new type

of activity. Future work includes the integration of the validation data framework with SciCumulus engine and its adaptive workflow execution model [Oliveira et al. 2012].

REFERENCES

Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*. Addison Wesley, 1999.

Callahan, S., Freire, J., Santos, E., Silva, C., Scheidegger, C., and Vo, H. T. VisTrails: visualization meets data management. In *International Conference on Management of Data*. Chicago, USA, pp. 745–747, 2006.

Dean, J. and Ghemawat, S. MapReduce: a flexible data processing tool. *Communications of the ACM* 53 (1): 72–77, 2010.

Dias, J., Ogasawara, E., de Oliveira, D., Porto, F., Coutinho, A., and Mattoso, M. Supporting Dynamic Parameter Sweep in Adaptive and User-Steered Workflow. In *Workshop on Workflows in Support of Large-Scale Science*. Seattle, Washington, USA, pp. 31–36, 2011.

Feldman, R. and Sanger, J. *The Text Mining Handbook*. Cambridge University Press, 2007.

Freire, J., Koop, D., Santos, E., and Silva, C. Provenance for Computational Tasks: A Survey. *Computing in Science and Engineering* 10 (3): 11–21, 2008.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, 1994.

Gonçalves, J., de Oliveira, D., Ocana, K., Ogasawara, E., and Mattoso, M. Using Domain-Specific Data to Enhance Scientific Workflow Steering Queries. In *International Provenance and Annotation Workshop (IPAW)*. Santa Barbara, California, USA, pp. 31–36, 2012.

Hey, T., Tansley, S., and Tolle, K. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, 2009.

Mattoso, M., Werner, C., Travassos, G. H., Braganholo, V., Murta, L., Ogasawara, E., de Oliveira, D., Cruz, S. M. S., and Martinho, W. Towards Supporting the Life Cycle of Large-scale Scientific Experiments. *International Journal of of Business Process Integration and Management* 5 (1): 79–92, 2010.

Missier, P. Incremental Workflow Improvement through Analysis of its Data Provenance. In *USENIX Workshop on the Theory and Practice of Provenance (TaPP)*. Heraklion, Crete, Greece, pp. 31–36, 2011.

Moreau, L. and Missier, P. The PROV Data Model and Abstract Syntax Notation, W3C Working Draft. http://www.w3.org/TR/2011/WD-prov-dm-20111018/, 2011.

Na'im, A., Crawl, D., Indrawan, M., Altintas, I., and Sun, S. Monitoring Data Quality in Kepler. In *Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing*. New York, NY, USA, pp. 560–564, 2010.

Ocana, K., de Oliveira, D., Dias, J., Ogasawara, E., and Mattoso, M. Optimizing Phylogenetic Analysis Using SciHmm Cloud-based Scientific Workflow. In *IEEE e-Science*. Stocholm, Sweden, pp. 745–747, 2011.

Ogasawara, E., Dias, J., de Oliveira, D., Porto, F., Valduriez, P., and Mattoso, M. An Algebraic Approach for Data-Centric Scientific Workflows. *Proceedings of the VLDB Endowment* 4 (1): 1328–1339, 2011.

Oliveira, D., Ocana, K., Baiao, F., and Mattoso, M. A Provenance-Based Adaptive Scheduling Heuristic for Parallel Scientific Workflows in Clouds. *Journal of Grid Computing* 10 (1): 521–552, 2012.

Oliveira, D., Ogasawara, E., Baiao, F., and Mattoso, M. SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows. In *IEEE Cloud Computing Conference*. Miami, FL, USA, pp. 378–385, 2010.

Reiter, M., Breitenbuecher, U., Dustdar, D., Karastoyanova, D., Leyman, F., and Truong, H. A Novel Framework for Monitoring and Analyzing Quality of Data in Simulation Workflows. In *IEEE International Conference on e-Science*. Stockholm, Sweden, pp. 378–385, 2011.

Taylor, I., Deelman, E., Gannon, D., and Shields, M. *Workflows for e-Science: Scientific Workflows for Grids*. Springer Verlag, 2007.

Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. A Break in the Clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review* 39 (1): 50–55, 2009.

Wilde, M., Hategan, M., Wozniak, J., Clifford, B., Katz, D., and Foster, I. Swift: a language for distributed parallel scripting. *Parallel Computing* 1 (1): 633–652, 2009.