Using Provenance Analyzers to Improve the Performance of Scientific Workflows in Cloud Environments

João Carlos de A. R. Gonçalves[1], Daniel de Oliveira[1], Kary A.C.S. Ocaña[1], Eduardo Ogasawara[2], Jonas Dias[1], Marta Mattoso[1]

[1] COPPE/Universidade Federal do Rio de Janeiro, Rio de Janeiro, Brasil
[2] CEFET/RJ, Rio de Janeiro, Brasil
`{jcg, danielc, kary, ogasawara, jonasdias, marta}@cos.ufrj.br`

Abstract. A major issue during scientific workflow execution is how to manage the large volume of data to be processed. This issue is even more complex in cloud computing 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, the scientist 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. These quality criteria can be evaluated based on provenance and domain-specific data. 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 paper, 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. Our approach introduces a component that enables sophisticated provenance analysis that allows for deciding at runtime if data have to be propagated or not for the subsequent activities of the workflow. 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 workflow on top of SciCumulus parallel workflow cloud execution 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.4.1 **[Information Systems Applications]**: Workflow management.

1. INTRODUCTION

In today's 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 [Deelman et al., 2009] play a fundamental role in many application domains [Hey et al., 2009]. Scientific Workflow Management Systems (SWfMS) [Deelman et al., 2009] are complex systems that provide ways for modeling and executing workflows. There are many SWfMS available [Callahan et al., 2006; Hull et al., 2006; Wilde et al., 2011]. Each of them 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 offer ways for running scientific workflows in parallel based on a pay per use model. Scientific workflows benefit from the elasticity and availability of virtual computing resources.

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

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 predict or can only be done based on runtime data. During a workflow execution, some produced data, which do not comply with quality criteria, may be discarded, avoiding unnecessary computations in the future. However, predicting the execution course of the workflow and when irrelevant data is produced 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 reduction may be obtained by scientists if they can analyze data generated during workflow execution, also known as workflow steering [Gil et al., 2007]. 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. In addition, a mechanism for interfering in the workflow execution plan during workflow execution, allowing scientists to inform filtering criteria during workflow execution, is needed. 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 paper.

Phylogenetic experiments aim at producing phylogenetic trees to represent existing evolutionary relationships. SciPhy [Ocaña 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, weeks, to produce the desired result. A typical phylogenetic analysis workflow (Fig. 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 Fig. 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 perhaps 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 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 filter criteria at run time, for example, the e-value (a random variable with respect to its probability measure). By doing this, a higher degree of quality will be presented on the activities´ output results. At Fig.1, this action would prevent Activity 4 from processing irrelevant data.

In this paper, 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 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 paper 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 paper 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.

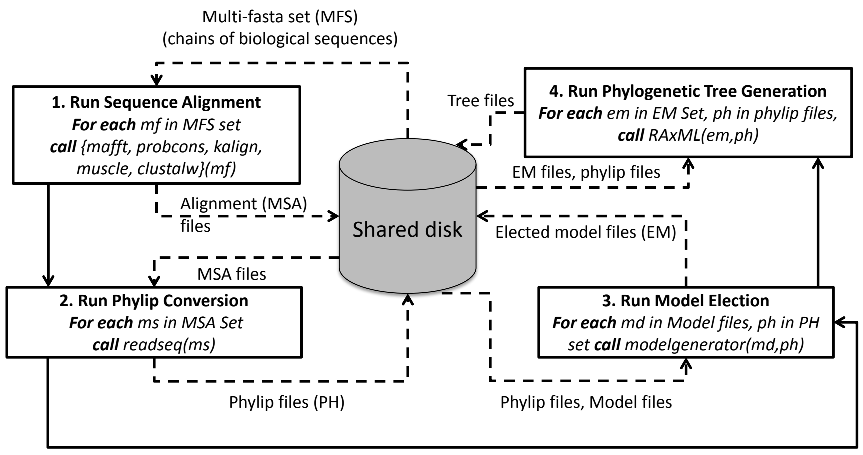

Multi-fasta set (MFS)
(chains of biological sequences)
1. Run Sequence Alignment
For each mf in MFS set
call {mafft, probcons, kalign, muscle, clustalw}(mf)
Alignment (MSA) files
Shared disk
MSA files
2. Run Phylip Conversion
For each ms in MSA Set
call readseq(ms)
Phylip files (PH)
Tree files
4. Run Phylogenetic Tree Generation
For each em in EM Set, ph in phylip files,
call RAxML(em,ph)
EM files, phylip files
Elected model files (EM)
3. Run Model Election
For each md in Model files, ph in PH set call modelgenerator(md,ph)
Phylip files, Model files

Fig. 1 SciPhy scientific workflow

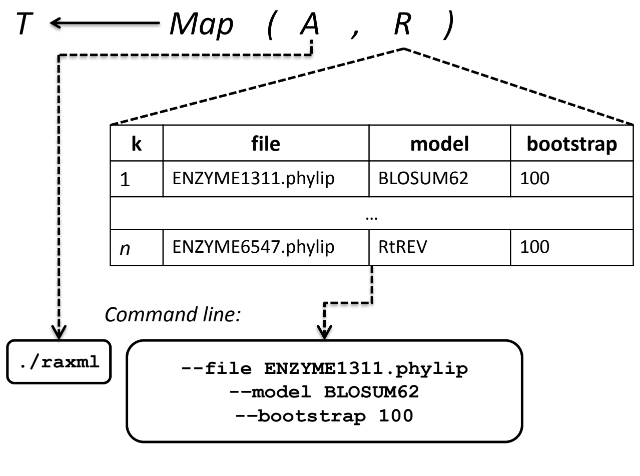
2. RELATED WORK

There are few approaches in the literature that improve parallel workflow execution performance by avoiding the unnecessary use of resources through provenance analysis. [Missier *et al.*, 2011] propose 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] concerns the Data Quality Monitor for the Kepler SWfMS, enabling users to inform the quality threshold value during the 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. This paper 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, thus enabling scientists to specify their own provenance quality evaluation criteria.

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 (Fig. 3). 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 (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 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. In addition, this algebra defines relations as one of its operands. The parameter values for the workflow activities are represented as attribute values 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. Fig. 2 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.



k	file	model	bootstrap
1	ENZYME1311.phylip	BLOSUM62	100
…			
n	ENZYME6547.phylip	RtREV	100

```
./raxml
```

```
--file ENZYME1311.phylip
--model BLOSUM62
--bootstrap 100
```

Fig. 2 Activity RAxML ruled by Map operator

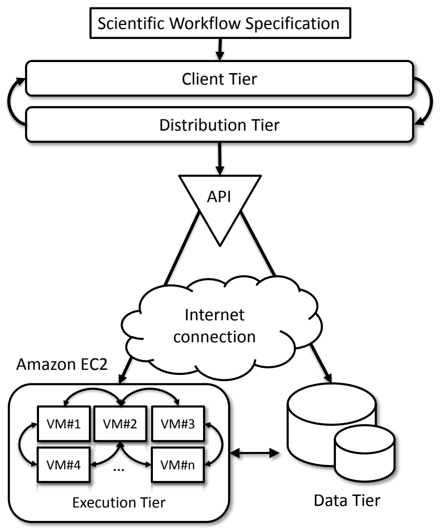


Fig. 3 SciCumulus conceptual architecture

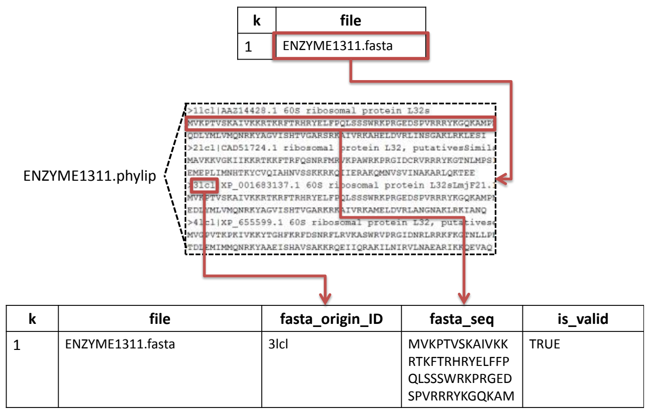
The execution tier is also responsible for capturing provenance data and to send it to a provenance repository (fundamental for the approach proposed in this paper). This provenance repository is also located in the cloud. The data tier is responsible for storing input data and provenance data. This data tier has information about the environment characteristics collected by an autonomous agent.

4. PROPOSED APPROACH: PROVENANCE ANALYZERS

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 reduction, 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. A 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. The execution of a PA follows a 2-phase procedure: provenance extraction and 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 [Goncalves 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 all 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 that is responsible for reading an input tuple (which contains a pointer to a file), 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, a 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 fit. 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 Fig. 4 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 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 (Fig. 5), 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).



k	file
1	ENZYME1311.fasta

k	file	fasta_origin_ID	fasta_seq	is_valid
1	ENZYME1311.fasta	3lcl	MVKPTVSKAIVKK RTKFTRHRYELFFP QLSSSWRKPRGED SPVRRRYKGQKAM	TRUE

Fig. 4 Provenance Analyzer execution

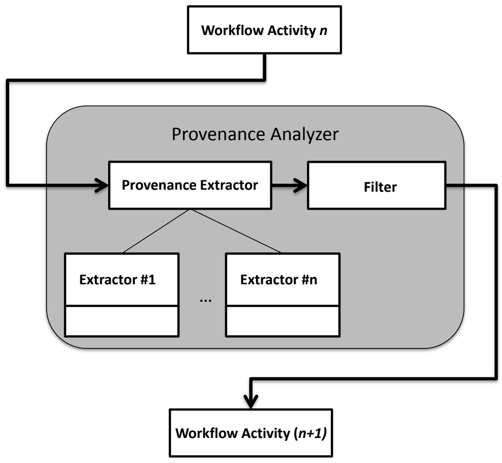


Fig. 5 SciCumulusPA implementation

5. EXPERIMENTAL RESULTS

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 paper we have instantiated 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, 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 (Pruitt et al., 2009). 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 activity, specialists analyze provenance data and the validation code.

In this experiment a provenance query Q1 is used for determining if the total number of phylogenetic trees expected was reached (i.e. if correspond to the number of multi-fasta files). Q1 offers information about the total execution time (TET) of the experiment. This can be done without PA, by browsing intermediate files. However, if the total number of trees was not reached, Q1 does not offer information about which trees or why these trees could not be generated. With PA, queries can be refined and further detailed going to specific steps of the execution, by querying parameters shown in Fig 4. For example, with further queries, the cause for no generating trees could be found, such as: empty files, incorrect format of files, insufficient number of sequences in files, or some specific problems of the execution environment. 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-doc way. By using current solutions, the analysis would be done only after the execution 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 Fig. 6 and Table I. The performance gains presented in Table I 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).

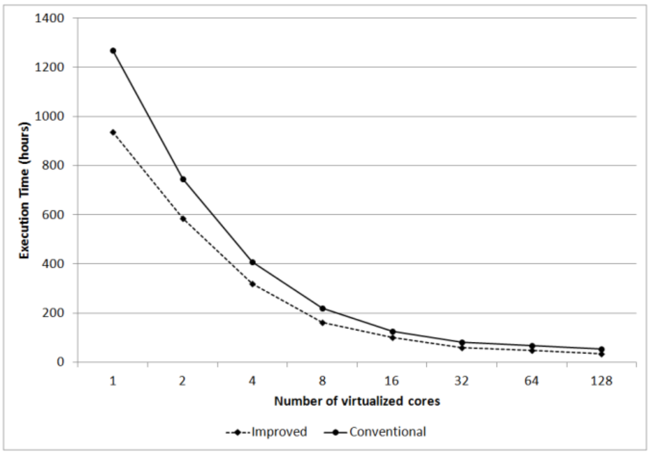


Fig. 6 Total execution time of SciPhy

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%

Table I Execution time of the SciPhy workflow

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 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. 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 936.77 hours (using one core) to 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.

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 paper 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 show benefits of up to 36.2% when using PA when filtering about 23% of the produced data. Although the proposed approach is an ongoing work, this paper 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. Future work includes the integration of the validation data framework with SciCumulus engine and its adaptive workflow execution model.

REFERENCES

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

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

Dias, J., Ogasawara, E., Oliveira, D., Porto, F., Coutinho, A., Mattoso, M., 2011. Supporting Dynamic Parameter Sweep in Adaptive and User-Steered Workflow, in: 6th Workshop on Workflows in Support of Large-Scale Science, WORKS ’11. Presented at the WORKS ’11, ACM, Seattle, WA, USA, pp. 31–36.

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

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

Gil, Y., Deelman, E., Ellisman, M., Fahringer, T., Fox, G., Gannon, D., Goble, C., Livny, M., Moreau, L., Myers, J., 2007. Examining the Challenges of Scientific Workflows. Computer 40, 24–32.

Goncalves, J., Oliveira, D., Ocaña, K.A.C.S., Ogasawara, E., Mattoso, M., 2012. Using Domain-Specific Data to Enhance Scientific Workflow Steering Queries, in: Proc. IPAW 2012. Presented at the IPAW, Springer, Santa Barbara, CA.

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

Hull, D., Wolstencroft, K., Stevens, R., Goble, C., Pocock, M.R., Li, P., Oinn, T., 2006. Taverna: a tool for building and running workflows of services. Nucleic Acids Research 34, 729–732.

Missier, P., 2011. Incremental workflow improvement through analysis of its data provenance, in: 3rd USENIX Workshop on the Theory and Practice of Provenance (TaPP ’11). Heraklion, Crete, Greece.

Moreau, L., Missier, P., 2011. The PROV Data Model and Abstract Syntax Notation [WWW Document]. W3C Working Draft. (Work in progress.). URL http://www.w3.org/TR/2011/WD-prov-dm-20111018/

Na’im, A., Crawl, D., Indrawan, M., Altintas, I., Sun, S., 2010. Monitoring data quality in Kepler, in: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC ’10. ACM, New York, NY, USA, pp. 560–564.

Ocaña, K.A.C.S., Oliveira, D., Ogasawara, E., Dávila, A.M.R., Lima, A.A.B., Mattoso, M., 2011. SciPhy: A Cloud-Based Workflow for Phylogenetic Analysis of Drug Targets in Protozoan Genomes, in: Norberto de Souza, O., Telles, G.P., Palakal, M. (Eds.), Advances in Bioinformatics and Computational Biology. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 66–70.

Ogasawara, E., Dias, J., Oliveira, D., Porto, F., Valduriez, P., Mattoso, M., 2011. An Algebraic Approach for Data-Centric Scientific Workflows. Proc. of VLDB Endowment 4, 1328–1339.

Oliveira, D., Ogasawara, E., Baião, F., Mattoso, M., 2010. SciCumulus: A Lightweight Cloud Middleware to Explore Many Task Computing Paradigm in Scientific Workflows, in: 3rd International Conference on Cloud Computing, CLOUD ’10. IEEE Computer Society, Washington, DC, USA, pp. 378–385.

Pruitt, K.D., Tatusova, T., Klimke, W., Maglott, D.R., 2009. NCBI Reference Sequences: current status, policy and new initiatives. Nucleic Acids Res 37, D32–D36.

Reiter, M., Breitenbuecher, U., Dustdar, S., Karastoyanova, D., Leyman, F., Truong, H.-L., 2011. A Novel Framework for Monitoring and Analyzing Quality of Data in Simulation Workflows, in: Proceedings of the 7th IEEE International Conference on e-Science.

Vaquero, L.M., Rodero-Merino, L., Caceres, J., Lindner, M., 2009. A break in the clouds: towards a cloud definition. SIGCOMM Comput. Commun. Rev. 39, 50–55.

Wilde, M., Hategan, M., Wozniak, J.M., Clifford, B., Katz, D.S., Foster, I., 2011. Swift: A language for distributed parallel scripting. Parallel Computing 633–652.