Scientific Workflow Partitioning in Multisite Cloud*

Ji Liu[1,2,3,5], Vítor Silva[4], Esther Pacitti[2,3,5], Patrick Valduriez[1,2,5], and Marta Mattoso[4]

[1] Microsoft-Inria Joint Centre, Paris, France
[2] LIRMM, Montpellier, France
[3] University Montpellier 2, France
[4] COPPE/UFRJ, Rio de Janeiro, Brazil
[5] Inria, Montpellier, France
{ji.liu,patrick.valduriez}inria.fr,
{silva,marta}@cos.ufrj.br,
{esther.pacitti}@lirmm.fr

Abstract. Scientific workflows allow scientists to conduct experiments that manipulate data with multiple computational activities using Scientific Workflow Management Systems (SWfMSs). As the scale of the data increases, SWfMSs need to support workflow execution in High Performance Computing (HPC) environments. Because of various benefits, cloud emerges as an appropriate infrastructure for workflow execution. However, it is difficult to execute some scientific workflows in one cloud site because of geographical distribution of scientists, data and computing resources. Therefore, a scientific workflow often needs to be partitioned and executed in a multisite environment. Also, SWfMSs generally execute a scientific workflow in parallel within one site. This paper proposes a non-intrusive approach to execute scientific workflows in a multisite cloud with three workflow partitioning techniques. We describe an experimental validation using an adaptation of Chiron SWfMS for Microsoft Azure multisite cloud. The experiment results reveal the efficiency of our partitioning techniques, and their superiority in different environments.

Keywords: scientific workflow, scientific workflow management system, workflow partitioning, parallel execution, multisite cloud.

1 Introduction

Scientific experiments generally contain multiple computational activities to process experimental data and these activities are related by data or control

* Work partially funded by CNPq, CAPES, FAPERJ, INRIA (Hoscar and Music projects) and Microsoft (Zcloudflow project), and performed in the context of the Institut de Biologie Computationnelle (www.ibc-montpellier.fr).

L. Lopes et al. (Eds.): Euro-Par 2014 Workshops, Part I, LNCS 8805, pp. 105–116, 2014.

dependencies. Scientific Workflows (SWfs) enable scientists to model these data processing activities together to be automatically executed. A SWf is the assembly of scientific data processing activities and data dependencies between them [9]. An activity is a description of a piece of work that forms one logical step within a scientific workflow representation. A data dependency is a relationship that represents the fact that one activity (output activity) consumes the output data of another activity (input activity). One activity may consist of several executable tasks for different parts of experimental data during SWf execution. A task is the representation of an activity within a one-time execution of this activity, which processes a part of its input data. As the amount of experimental data becomes huge, data-intensive scientific workflows become an important issue.

A Scientific Workflow Management System (SWfMS) is a tool to manage SWf representation, execution and data sets in various computing environments. SWfMSs may exploit High Performance Computing (HPC) to execute SWfs within reasonable time. The HPC environment may be provided by a cluster, grid or cloud. Cloud computing, which promises virtually infinite resources, scalable services, stable service quality and flexible payment policies, has recently become a viable solution for workflow execution.

In general, one cloud site is sufficient for executing one user application. However, in the case of SWfs, some important restrictions may force their execution in a multisite cloud, i.e. a cloud with multiple distributed data centers, each being explicitly accessible to cloud users. For instance, some activities need to be executed at a cloud site trusted by the scientists for workflow monitoring without malicious attack, i.e. scientist security restriction; some activities need to be moved to another cloud site because of stored big input data at that site and the cost of transferring this big data to another site is very high, i.e. data transfer restriction; some activities have to be executed at a site where more computing resources are available, i.e. computing capacity restriction; some other activities may invoke special programs (instruction data), which are located at a specific cloud site and cannot be moved to another site because of proprietary reasons, i.e. proprietary restriction. The configuration data, which includes workflow representation files or workflow parameters, can be located at one site or distributed at different sites. In this situation, multisite cloud is appealing for data-intensive scientific workflows.

For a given application, a multisite cloud configuration, which is the configuration of Virtual Machines (VMs) at each site, can be homogeneous, with homogeneous computing capacity at each site, e.g. 8 VMs at sites 1 and 2, or heterogeneous, e.g. 8 VMs at site 1 and 2 VMs at site 2. The homogeneous configuration is obviously easier to deal with in terms of workflow partitioning and execution. However, even the homogeneous configuration makes it difficult to reproduce experiments as the allocation of VMs to real resources is typically controlled at runtime by the cloud provider. For instance, at time t_1, one VM may be allocated to a processor that is already very busy (e.g. running 16

other VMs) while at time t_2, the same VM may be allocated to an underloaded processor (e.g. running 2 other VMs).

In order to execute SWfs in a multisite cloud environment, a SWfMS can generate a Workflow Execution Plan (WEP) for workflow execution. Similar to the concept of Query Execution Plan (QEP) in distributed database systems [16], the WEP is a program that captures optimization decisions and execution directives, typically the result of compiling and optimizing a workflow. Since the multiple sites are interconnected but share nothing, the WEP includes a workflow partitioning result, which is the decision of partitioning a workflow into workflow fragments for independent execution. A workflow fragment (or fragment for short) can be defined as a subset of activities and data dependencies of the original workflow (see [14] for a formal definition). In order to execute a fragment within reasonable time at one site, the WEP generally contains a workflow parallelization plan, which parallelizes the workflow execution.

We formulate the problems addressed in this paper as follows. A SWf $W = \{V,E\}$ consists of a set of activities V and a set of dependencies E. A multisite cloud $MS = \{S_1, S_2, ..., S_n\}$ is composed of multiple cloud sites, each of which has multiple computing nodes and stores its own data (input data, instruction data or configuration data of W). The workflow execution time is the entire time to execute a SWf at a given execution environment. Given a SWf W and a multisite cloud MS, the multisite cloud execution problem is how to execute W in MS in a way that reduces execution time while respecting restrictions.

We propose a non-intrusive approach to execute a SWf in a multisite cloud. We propose three partitioning techniques with consideration of restrictions and present the existing parallel execution techniques for the execution within one site. Note that workflow pre-partitioning technique does not mean we cannot explore the elasticity within one cloud site. We validate our approach with a data-intensive SWf using Chiron [15] SWfMS in Microsoft Azure [2] cloud. The experiment results reveal that our approach can reduce execution time. Since the occupied computing resources do not change, the reduction of execution time may lead to less lease time, which corresponds to less monetary cost.

This paper is organized as follows. Section 2 discusses worflow parallelism and related work on workflow parallel execution. Section 3 introduces our system model based on an adaptation of Chiron for multisite cloud. Section 4 presents the SWf we use for experimentation. Section 5 details our three workflow partitioning techniques. Section 6 presents our experimental validation with Microsoft Azure. Section 7 concludes.

2 Related Work

Workflow partitioning and execution in a multisite cloud remains a challenge and little work has been done. Deng *et al.* [10] adopt a clustering method based on data-data, data-activity and activity-activity dependencies. This method is adapted for workflow structures, but it may have big amount of data to be transferred between workflow fragments. Chen *et al.* [7] present workflow partitioning based on storage constraints. Since a cloud environment can offer big

storage capacity and the VMs can be mounted additional storage resources before or during workflow exeuciton, the storage capacity limitation is not general in a cloud environment. In addition, this method do not take data transfer cost and different computing capacity at each site into consideration. Tanaka and Tatebe [17] use a Multi-Constraint Graph Partitioning (MCGP) algorithm [12] to partition a workflow. This approach partitions a workflow by minimizing the removed dependency and balancing the activities in each fragment. However, this approach is appropriate only for homogeneous execution sites. In this paper, we propose several partitioning techniques to address data transfer restriction and computing capacity restriction in the multisite cloud. Because of workflow partitioning, distributed provenance data is supported in the multisite cloud. In addition, data compression and file archiving is proposed to accelerate the data transfer between different cloud sites.

3 System Model

In this section, we present our workflow parallelization and scheduling methods, system model based on Chiron SWfMS, its adaptation for multisite cloud and a SWf partitioner.

A SWfMS exploits workflow parallelization and scheduling methods to enable parallel execution within one cloud site. Through parallelization, the WEP can achieve activity parallelism or data parallelism. Activity parallelism parallelizes the execution of different activities. It has two types: independent parallelism and pipeline parallelism. Independent parallelism is achieved when independent activities are executed in different computing nodes. When the execution of one activity does not dependent on the result of another activity, these two activity are independent activities. Otherwise, these two activities are dependent activities. Pipeline parallelism is obtained when dependent activities are executed in parallel. Data parallelism is realized by having the execution of the same activity on different parts of data in different computing nodes. Moreover, hybrid parallelism combines two or more types of parallelism to achieve better performance.

After parallelizing workflow execution, a SWfMS should schedule executable tasks to available computing nodes. A SWfMS can generate a static Scheduling Plan (SP) to schedule the tasks before workflow execution. Since the static SPs are generated before workflow execution, the workflow execution time is reduced. There are some existing static scheduling algorithms presented in [6,19]. However, since it is difficult to precisely predict the variation of execution environment or the workload of the executable tasks, static SPs do not achieve good performance on load balancing for dynamic changing execution. A SWfMS can also exploit dynamic scheduling method according to runtime environment features. Some dynamic scheduling methods are detailed in [5,13]. Although it takes time to generate SPs during workflow execution, the dynamic scheduling method can achieve better load balancing performance for a dynamic changing environment. Moreover, SWfMSs can use hybrid scheduling methods, i.e. the combination of static and dynamic scheduling method to reduce execution time and achieve load balance at the same time.

Chiron implements an algebraic approach for data-intensive scientific workflows proposed by Ogasawara *et al.* [15], to perform workflow parallelization and scheduling. This approach associates each activity with an operator, which has a semantic meaning for parallel execution. Since it models workflow data as relations similar to relational database management systems, this approach can optimize the entire workflow parallel execution based on well-founded relational algebra query optimization models [14].

The algebraic approach also allows online provenance data to be managed (and stored in a database by Chiron) for workflow activity monitoring [8]. Provenance data is the metadata that captures the derivation history of a dataset, including the original data sources, intermediate datasets, and the workflow computational steps that were applied to produce this dataset [8].

Chiron was initially developed for a one site execution environment as shown in Fig. 1-*A*. In a one site environment, a database is installed in a master node and all the computing nodes share storage resources through Network File System. Chiron achieves activity parallelism, data parallelism and dynamic scheduling for workflow parallelization as explained in Section 2. Chiron was modified to gather necessary produced data at the end of SWf execution at one site.

In a multisite cloud environment (see Fig. 1-*B*), all the sites have the same configuration as one site environment, i.e. a database installed in a master node and shared storage resources, while each site can have different numbers of slave computing nodes. We developped a SWf partitioner to automatically partition a processed SWf representation file into workflow fragment representation files when the first activity in each fragment is given. All the activities in a fragment are placed together in the processed SWf representation file. The SWf partitioner removes the dependencies in the original SWf and generates corresponding configuration files for each fragment. The execution of the generated fragments should respect the dependencies removed from the original SWf. Let us suppose that, in a SWf, activity A_2 consumes the output data produced by activity A_1. If these two activities are allocated to different fragments, their data dependencies will be removed from the explicit data dependencies. In this case, the fragment that contains activity A_2 should be executed after the execution of the fragment that contains activity A_1.

In order to reduce data transfer volume between different sites, we can use data compression and file archiving techniques. Data compression can just reduce the volume of transferred data to reduce transmission time. Through file archiving, we can also transfer the data at a relatively high speed to reduce transmission time. When transferring one file between two sites and the default transfer speed is less than the highest transfer speed between two sites, the file transfer is accelerated (accelerating phase) at the beginning and decreased (decreasing phase) at the end. The data transfer rate remains high in the middle (high speed transfer phase). If we transfer several small files, there will be many accelerating and decreasing phases. But if we transfer a big file of the same data volume, there will be an accelerating phase and a decreasing phase while the high speed transfer phase will be longer. Therefore, the transmission speed of a

big file is higher than that of several small files of the same data volume. In the remainder of the paper, we note data refining as the combination of file archiving and data compression.

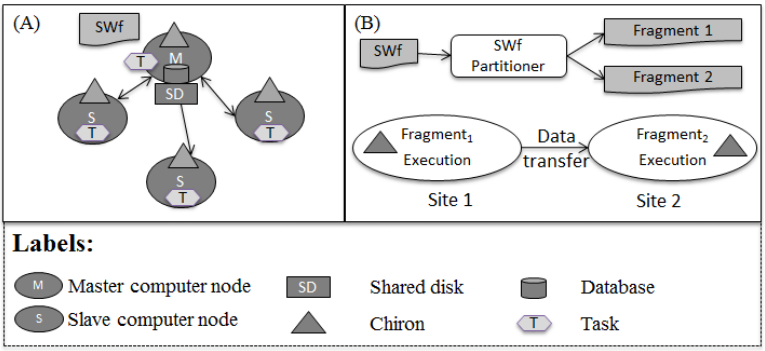


Fig. 1. The architecture of Workflow Execution in Chiron. Fig. *A* presents workflow execution in one site using Chiron before modification. Fig. *B* shows the multisite workflow execution using SWf partitioner and modified Chiron.

4 Use Case: Buzz Workflow

This section presents Buzz workflow [11], a data-intensive scientific workflow, as a use case to illustrate our partitioning techniques. Buzz workflow is modeled and executed using Chiron. Buzz workflow searches for trends and measures correlations in published papers from scientific publications. This scientific workflow uses data collected from bibliography databases such as the DBLP Computer Science Bibliography (DBLP) [1] or the U.S. National Institutes of Health's National Library of Medicine (PubMed). We used a DBLP 2013 XML file of 1, 29GB as input for Buzz workflow in our experiments.

Buzz workflow has thirteen activities (Fig. 2(a)). Each activity has a specific operator according to the algebraic approach. Boxes in the figure represent workflow activities together with the involved algebraic operators. *FileSplit* activity is responsible for gathering all scientific publications from bibliography databases. *Buzz* activity uses these publications to identify buzzwords (a word or phrase that can become popular for a specific period of time). *WordReduce* activity organizes these publications according to buzzword and publication year, and it also computes occurrences of identified words. Furthermore, *YearFilter* activity selects buzzwords that appeared in the publications after 1991, while *BuzzHistory* activity and *FrequencySort* activity create a history for each word and compute its frequency. With this information, *HistogramCreator* activity generates some histograms with word frequency varying the year. On the other hand, *Top*10 activity selects ten of the most frequent words in recent years,

whilst *ZipfFilter* activity selects terms according to a Zipf curve that is specified by word frequency values [18]. Moreover, *CrossJoin* activity merges results from *Top*10 activity and *ZipfFilter* activity. *Correlate* activity computes correlations between the words from *Top*10 activity and buzzwords from *ZipfFilter* activity. Using these correlations, *TopCorrelations* activity takes the terms that have a correlation greater than a threshold and *GatherResults* activity presents these selected words with the histograms.

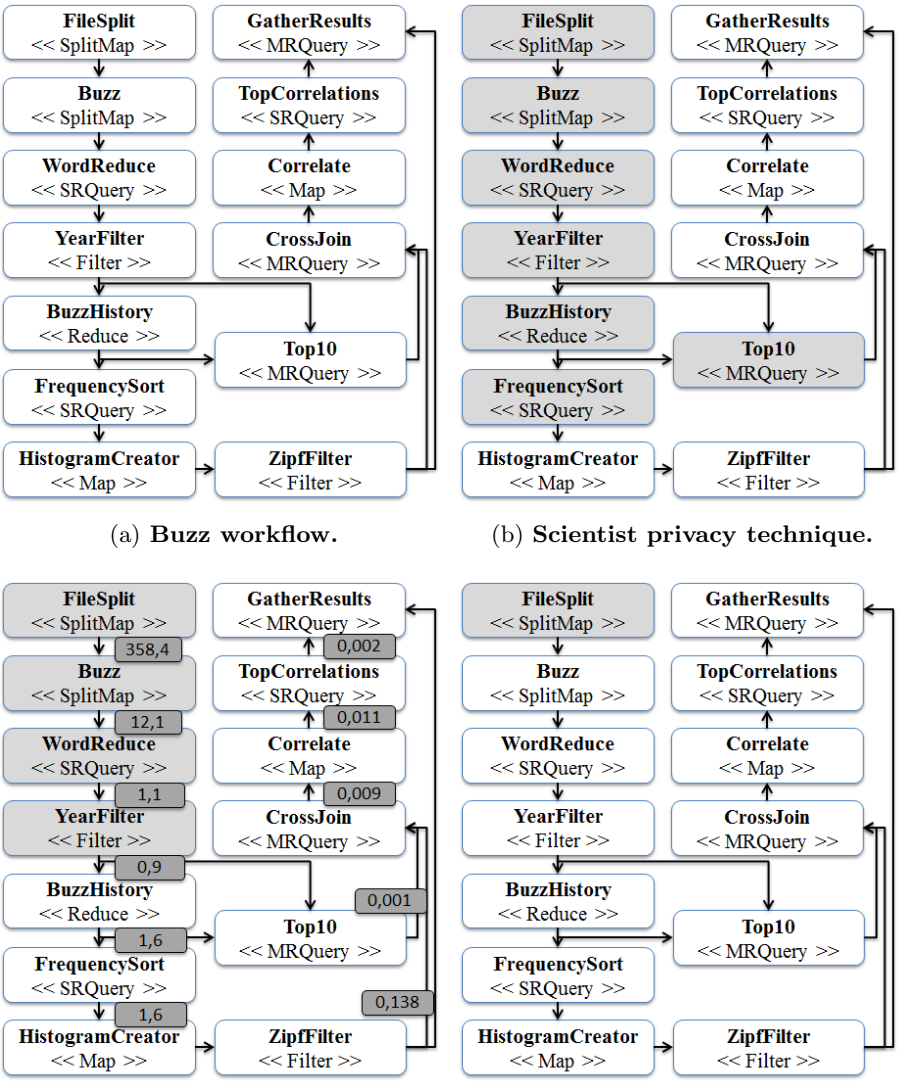
In the remainder of the paper, we assume that there are two cloud sites (S_1 and S_2) to execute Buzz workflow. A fixed activity is located at a specific site and cannot be moved to another site because of additional restrictions, i.e. scientist security, data transfer, computing capacity and proprietary issues. We also assume that the first activity (*FileSplit*) is a fixed activity at S_1 since the input data located at S_1 is very big. In addition, we assume that the last activity (*GatherResults*) is a fixed activity at S_2 because of proprietary issues. Finally, scientists at S_2 need to monitor the execution of activity *HistogramCreator* without malicious attack and thus, *HistogramCreator* becomes a fixed activity at S_2, which is trusted by scientists.

5 Workflow Partitioning Techniques

We propose three techniques for workflow partitioning, i.e. scientist privacy, data transfer minimizing and computing capacity adaptation.

The first technique, Scientist Privacy (SPr), is for better supporting workflow activity monitoring under scientist security restriction. When a SWf contains an activity that needs to be monitored by scientists, this activity is defined as a locking activity to be executed at a trusted cloud site to avoid malicious attack during workflow execution. A locking activity implies that this activity and all the following activities should be assigned to a same workflow fragment, in order to provide further workflow activity monitoring. The following activities represent the activities that process the output data or the data produced from the output data of the locking activity. In order to partition a workflow based on scientist privacy technique, a SWfMS identifies the locking activity. Then it can partition the workflow by putting the locking activity and its available following activities (the following activities that are not fixed activities) into a fragment. According to this technique, Buzz workflow is partitioned into two fragments as shown in Fig. 2(b). As scientists need to analyze some histogram files produced by *HistogramCreator* activity at runtime at S_2 (trusted by scientists), *HistogramCreator* activity is handled as a locking activity. This activity and the following activities (*ZipfFilter*, *CrossJoin*, *Correlate*, *TopCorrelations* and *GatherResults*) are assigned to the same fragment while the other activities stay in another fragment.

The second technique is Data Transfer Minimizing (DTM), which minimizes the volume of data to be transferred between different workflow fragments. It is based on the fact that it takes much time to transfer certain amount of data from one site to another site. If the amount of data to be transferred between



(a) **Buzz workflow.** (b) **Scientist privacy technique.**

(c) **Data transfer minimizing technique.** (d) **Computing capacity adaptation technique.**

Fig. 2. Buzz workflow partitioning

fragments is minimized, the time to transfer data between different sites can be reduced so as to reduce the entire execution time. During workflow design, the ratio between the volume of input data and output data can be offered. The scientists can estimate the data transfer for each data dependency based on the volume of input data of the SWf. There are five steps to partition a workflow based on this technique. The first step is to remove the dependencies that connect two activities at the same site in each route. The second step is to identify all the possible routes to connect the fixed activities at any two different sites. A route is a combination of a pipeline of activities and the dependencies between activities. The third step is to chose selected dependencies. The dependency that has the least data transfer volume in each route is put in the Selected Dependencies (SD). The dependencies that share the same input activity with the dependencies in SD are also put in SD. Fourth, we delete the dependencies from SD while ensuring that SD is enough to partition a SWf. The deletion begins from the dependency of the biggest data transfer volume to that of the smallest. Finally, the workflow can be partitioned to fragments by removing all the dependencies in SD from the original workflow. With this technique, Buzz workflow is partitioned as shown in Fig. 2(c). The dark gray boxes represent the data transfer volume for the corresponding dependencies. The data dependencies of each possible route between *FileSplit* and *HistogramCreator* is analyzed. The dependencies (*YearFilter* to *BuzzHistory* and *YearFilter* to *Top10*) are put in SD. Since it is necessary to remove the two dependencies to partition Buzz workflow, we do not delete any dependency from SD. Finally, Buzz workflow is partitioned by removing the selected dependencies (*YearFilter* to *BuzzHistory* and *YearFilter* to *Top10*).

The third technique is Computing Capacity Adaptation (CCA), which adapts SWfs partitioning to the computing capacity at different cloud sites. This technique is for the heterogeneous multisite cloud configurations, which may be incurred by the different configurations of different groups of scientists. If a scientific workflow is partitioned into two workflow fragments that are sequentially executed, i.e. one fragment begins execution after the execution of another one, a SWfMS can put all the possible activities to one fragment while leaving fixed activities in another fragment. As an example, Buzz workflow is partitioned into two fragments (WF_1 and WF_2) as shown in Fig. 2(d) . Since the input data of activity *FileSplit* is relatively big and located at a specific site, we keep this activity in the gray fragment, which is done by combining the computing capacity adaptation technique and the data transfer minimizing technique. Then, the white fragments can be scheduled to a cloud site that has more computing capacity.

6 Validation

In this section, we present experiments to validate our approach, by executing Buzz workflow using our partitioning techniques in Microsoft Azure cloud. The virtual machines (VMs) are distributed at two cloud sites: Western Europe (Amsterdam, Netherlands) and Eastern US (North Virginia). In the first experiment,

we use a homogeneous configuration by creating two A4 VMs at both of Western Europe site and Eastern US site. In the second experiment, we use a heterogeneous configuration by creating two A4 VMs at the Western Europe site and eight A4 VMs at the Eastern US site. Each A4 VM has 8 CPU cores, 14 GB of RAM memory, 127 GB of instance storage and a network of 800 Mbps [4,3].

We executed Buzz workflow with Chiron and the SWf partitioner. We used Linux tar command and Linux scp command for data refining and data transfer. We launched the workflow fragment execution by hand at each cloud site, which resembles to the cooperation between two scientist group. In our execution, Chiron exploits data parallelism and the dynamic scheduling method for workflow parallel execution within one site. Table 1 shows the experimental results. Elapsed time 1 represents the execution time without considering data transfer time. Elapsed time 2 shows the execution time plus the data transfer time. Elapsed time 3 is the execution time plus data transfer time with data refining. Data transfer 1 reveals the data transfer time without data refining. Data transfer 2 presents the data refining and data transfer time. The three techniques correspond to the three partitioning techniques as explained in Section 5.

Table 1. Experimental Results

Approach (Time in minutes)	Execution time 1	Transmission time 1	Execution time 2	Transmission time 2	Execution time 3
1^{st} experiment	2*A4 VM (EU) + 2*A4 VM (US)				
SPr technique	199	38	237	0	199
DTM technique	186	0	186	0	186
CCA technique	209	35	245	0.5	209
1^{st} experiment	2*A4 VM (EU) + 8*A4 VM (US)				
SPr technique	198	38	236	0	198
DTM technique	182	0	182	0	182
CCA technique	169	35	201	0.5	169

In the first experiment, the workflow execution time of the three techniques without considering data transfer time is different because there is a different amount of data loaded into RAM memory for the white workflow fragment execution. Since the DTM technique minimizes the data transfer between two fragments, it also reduces the data to be transferred from disk to RAM at the beginning of the second fragment (white fragment) execution. When the data is transferred without data refining, the execution time of the DTM technique is 21.5% and 24.1% less than the SPr and the CCA technique. When the data is transferred with data refining, the DTM technique saves 6.5% and 11.0% of the execution time compared to the SPr and the CCA technique. As the two sites have the same computing capacity and it incurs big data transfer volume, the CCA is the least efficient technique. In this experiment, the workflow execution with the best partitioning technique (DTM with data refining) takes 24.1% less time than the least efficient technique (CCA without data refining). In the second

experiment, because of the difference of computing capacity at two cloud sites, the execution of the CCA technique takes the least amount of time without considering data transfer. When the data is transferred without data refining, the DTM technique is still the best performance because of the minimized data transfer cost. This technique yields a gain of 22.9% and 10.4% of the execution time compared to the SPr and CCA technique. However, when we use data refining techniques, the third technique yields the best performance because of the adaptation of workflow partitioning to the computing capacity at each cloud site. In this case, the workflow execution of the CCA technique takes 14.6% and 7.1% less time compared to the SPr and DTM technique. In this experiment, the best partitioning technique (CCA with data refining) saves 28.4% time compared to the least efficient technique (SPr without data refining).

The experiments reveal that the DTM technique with data refining is the best for a homogeneous configuration and that the CCA technique with data refining has better performance for a heterogeneous configuration.

7 Conclusion

The cloud is emerging as an appropriate infrastructure for workflow execution. However, because of different restrictions, a scientific workflow often needs to be partitioned and executed in parallel in a multisite environment. In this paper, we proposed a non-intrusive approach to execute scientific workflows in a multisite cloud with three workflow partitioning techniques. We proposed a system model based on Chiron SWfMS and its adaptation to multisite cloud and a SWf partitioner. We presented the scheduling of workflow fragment execution by respecting all the data dependencies in the original SWf. We described an experimental validation using an adaptation of Chiron SWfMS for Microsoft Azure multisite cloud. The experiments experiments reveal the efficiency of our partitioning techniques, and their superiority in different environments. The experiment results show that data transfer minimizing technique with data refining, i.e. file archiving and data compression, has better performance (24.1% of time saved compared to computing capacity adaptation technique without data refining) for homogeneous configurations while computing capacity adaptation technique with data refining (28.4% of time saved compared to scientist privacy technique without data refining) is appropriate to heterogeneous configurations.

References

1. DBLP Computer Science Bibliography, `http://dblp.uni-trier.de/`
2. Microsoft Azure, `http://azure.microsoft.com`
3. VM bandwidth in Azure, `http://windowsazureguide.net/tag/auzre-virtual-machines-sizes-bandwidth/`
4. VM parameters in Azure, `http://msdn.microsoft.com/en-us/library/azure/dn197896.aspx`

5. Anglano, C., Canonico, M.: Scheduling algorithms for multiple bag-of-task applications on desktop grids: A knowledge-free approach. In: 22nd IEEE Int. Symposium on Parallel and Distributed Processing (IPDPS), pp. 1–8 (2008)
6. Blythe, J., Jain, S., Deelman, E., Gil, Y., Vahi, K., Mandal, A., Kennedy, K.: Task scheduling strategies for workflow-based applications in grids. In: 5th IEEE Int. Symposium on Cluster Computing and the Grid (CCGrid), pp. 759–767 (2005)
7. Chen, W., Deelman, E.: Partitioning and scheduling workflows across multiple sites with storage constraints. In: Wyrzykowski, R., Dongarra, J., Karczewski, K., Waśniewski, J. (eds.) PPAM 2011, Part II. LNCS, vol. 7204, pp. 11–20. Springer, Heidelberg (2012)
8. Costa, F., Silva, V., de Oliveira, D., Ocaña, K.A.C.S., Ogasawara, E.S., Dias, J., Mattoso, M.: Capturing and querying workflow runtime provenance with prov: a practical approach. In: EDBT/ICDT Workshops, pp. 282–289 (2013)
9. Deelman, E., Gannon, D., Shields, M., Taylor, I.: Workflows and e-science: An overview of workflow system features and capabilities. Future Generation Computer Systems 25(5), 528–540 (2009)
10. Deng, K., Kong, L., Song, J., Ren, K., Yuan, D.: A weighted k-means clustering based co-scheduling strategy towards efficient execution of scientific workflows in collaborative cloud environments. In: IEEE 9th Int. Conf. on Dependable, Autonomic and Secure Computing (DASC), pp. 547–554 (2011)
11. Dias, J., Ogasawara, E.S., de Oliveira, D., Porto, F., Valduriez, P., Mattoso, M.: Algebraic dataflows for big data analysis. In: IEEE Int. Conf. on Big Data, pp. 150–155 (2013)
12. Karypis, G., Kumar, V.: Multilevel algorithms for multi-constraint graph partitioning. In: ACM/IEEE Conf. on Supercomputing, pp. 1–13 (1998)
13. Maheswaran, M., Ali, S., Siegel, H.J., Hensgen, D., Freund, R.F.: Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In: 8th Heterogeneous Computing Workshop, p. 30 (1999)
14. Ogasawara, E., Oliveira, D., Valduriez, P., Dias, J., Porto, F., Mattoso, M.: An algebraic approach for data-centric scientific workflows. Proceedings of the VLDB Endowment (PVLDB) 4(12), 1328–1339 (2011)
15. Ogasawara, E.S., Dias, J., Silva, V., Chirigati, F.S., de Oliveira, D., Porto, F., Valduriez, P., Mattoso, M.: Chiron: a parallel engine for algebraic scientific workflows. Concurrency and Computation: Practice and Experience 25(16), 2327–2341 (2013)
16. Özsu, M.T., Valduriez, P.: Principles of Distributed Database Systems. Springer (2011)
17. Tanaka, M., Tatebe, O.: Workflow scheduling to minimize data movement using multi-constraint graph partitioning. In: 12th IEEE/ACM Int. Symposium on Cluster, Cloud and Grid Computing (Ccgrid), pp. 65–72 (2012)
18. Tarapanoff, K., Quoniam, L., Henrique, R., de Araújo, J., Alvares, L.: Intelligence obtained by applying data mining to a database of french theses on the subject of brazil. Information Research 7(1) (2001)
19. Topcuouglu, H., Hariri, S., Wu, M.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. on Parallel and Distributed Systems 13(3), 260–274 (2002)