Scientific Workflow Scheduling with Provenance Support in Multisite Cloud

Ji Liu, Esther Pacitti, Patrick Valduriez, Marta Mattoso

Scientific Workflow Scheduling with Provenance Support in Multisite Cloud *

Ji Liu[1], Esther Pacitti[1], Patrick Valduriez[1], and Marta Mattoso[2]

[1] Inria, Microsoft-Inria Joint Centre, LIRMM and University of Monpellier, France
{ji.liu,patrick.valduriez}@inria.fr
{esther.pacitti}@lirmm.fr
[2] COPPE, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
{marta}@cos.ufrj.br

Abstract. Recently, some Scientific Workflow Management Systems (SWfMSs) with provenance support (*e.g.* Chiron) have been deployed in the cloud. However, they typically use a single cloud site. In this paper, we consider a multisite cloud, where the data and computing resources are distributed at different sites (possibly in different regions). Based on a multisite architecture of SWfMS, *i.e.* multisite Chiron, we propose a multisite task scheduling algorithm that considers the time to generate provenance data. We performed an experimental evaluation of our algorithm using Microsoft Azure multisite cloud and two real-life scientific workflows, *i.e.* Buzz and Montage. The results show that our scheduling algorithm is up to 49.6% better than baseline algorithms in terms of execution time.

Keywords: scientific workflow; scientific workflow management system; scheduling; parallel execution; multisite cloud

1 Introduction

Many large-scale *in silico* scientific experiments take advantage of scientific workflows (SWfs) to model data operations such as loading input data, data processing, data analysis and aggregating output data. SWfs enable scientists to model the data processing of these experiments as a graph, in which vertices represent data processing activities and edges represent dependencies between them. A SWf is the assembly of scientific data processing activities with data dependencies between them [5]. An activity is a description of a piece of work that forms a logical step within a SWf representation [11] and a task is the representation of an activity within a one-time execution of this activity, which processes a data chunk [11]. A Scientific Workflow Management System (SWfMS) is a tool

* Work partially funded by EU H2020 Programme and MCTI/RNP-Brazil (HPC4E grant agreement number 689772), CNPq, FAPERJ, and INRIA (MUSIC project), Microsoft (ZcloudFlow project) and performed in the context of the Computational Biology Institute (www.ibc-montpellier.fr).

to execute SWfs [11]. Some implementations of SWfMSs are publicly available, *e.g.* Pegasus [7] and Chiron [14]. A SWfMS generally supports provenance data, which is the metadata that captures the derivation history of a dataset [11], during SWf execution. In order to execute a data-intensive SWf within reasonable time, SWfMSs generally exploit High Performance Computing (HPC) resources obtained from a computer cluster, grid or cloud environment.

Recently, some SWfMSs with provenance support (*e.g.* Chiron) have been deployed in the cloud for the execution of a SWf at a single cloud site. However, the data necessary to run a SWf may well be distributed at different sites (possibly in different regions), which may not be allowed to be transferred to other sites because of big amounts or proprietary reasons. It may not be always possible to move all the computing resources (including programs) to a single site. In this paper, we consider a multisite cloud that is composed of several sites (or data centers) of the same cloud provider, each with its own resources and data. The difference between the multisite cloud and the classical large-scale distributed environment is that the data or the computing resources are well distributed and the network bandwidths among different sites are different.

To enable SWf execution in a multisite cloud with distributed input data, the execution of the tasks of each activity should be scheduled to cloud sites (or sites for short). The tasks of each activity can be scheduled independently. Then, the scheduling problem is how to decide at which sites to execute the tasks of each activity in order to reduce execution time of a SWf in a multisite cloud. The mapping relationship between sites and tasks is a scheduling plan. Since it may take much time to transfer data between two different sites, the multisite scheduling problem should take into account the resources at different sites and intersite data transfer, including the data to be processed by tasks and the provenance data, during SWf execution.

Classic scheduling algorithms, *e.g.* Opportunistic Load Balancing (OLB) [13], Minimum Completion Time (MCT) [13], min-min [10], max-min [10] and Heterogeneous Earliest Finish Time (HEFT) [18], and some other scheduling solutions [4][16][17] address the scheduling problem within a single site. A few multisite scheduling approaches are proposed [9], but they do not consider the distribution of input data at different sites and have no support for provenance data, which may incur much time for intersite data transfer. In [15], data transfer is analyzed in multi-site SWf execution, stressing the importance of optimizing data provisioning. However, this information is not yet explored on task scheduling. In a previous work [12], we proposed a solution of multisite activity scheduling of SWfs according to data location. However, it can schedule the execution of each activity to a site but cannot schedule tasks of one activity to different sites.

The difference between our work and others is multisite execution with provenance support. In the paper, we make the following contributions. First, we propose multisite Chiron, with a novel architecture to execute SWfs in multisite cloud environments with provenance support. Second, we propose a novel multisite task scheduling algorithm, *i.e.* Data-Intensive Multisite task scheduling (DIM), for SWf execution in multisite Chiron. Third, we carry out an experi-

mental evaluation, based on the implementation of multisite Chiron in Microsoft Azure using two SWfs, *i.e.* Buzz [8] and Montage [2].

This paper is organized as follows. Section 2 explains the design of a multisite SWfMS. Section 3 proposes our scheduling algorithm. Section 4 gives our experimental evaluation. Section 5 concludes.

2 System Design

Chiron [14] is a data-centric SWfMS for the execution of SWfs at a single site, with provenance support. We extend Chiron to multisite, *i.e.* multisite Chiron, which can manage the communication of Chiron instances at each site and automatically take advantage of distributed resources at each site to process the distributed data. In the execution environment of multisite Chiron, there is a master site (site 1 in Figure 1) and several slave sites (Sites 2 and 3 in Figure 1). The master site is composed of several Virtual Machines (VMs), a shared file system and a provenance database. The synchronization is achieved by master-worker model while the data transferring is realized by peer-to-peer model. A slave site is composed of a cluster of VMs with a deployed shared file system. In the multisite environment, each VM is a computing node (or node for short). A node is selected as a master node at each site. In this paper, we assume that there is a Web domain at each site for SWf execution and that all the resources related to the SWf execution are in the Web domain at that site.

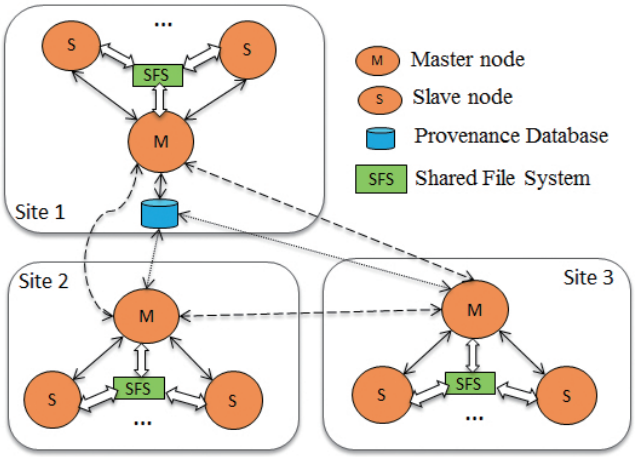

Fig. 1: **Architecture of multisite Chiron.**

During SWf execution, Chiron analyzes the data dependencies of each activity. When the input data of an activity is ready [14], Chiron generates tasks. Then, the tasks of each activity are independently scheduled to each site. All the

previous processes are realized at the master node of the master site. Then, the data of the scheduled tasks is transferred to the scheduled sites and the tasks are executed at the scheduled sites. After the execution of tasks, the provenance data [14] of each site are transferred to the provenance database. When the tasks of all the activities are executed, the execution of a SWf is finished.

3 Task Scheduling

In this section, we propose our two Level (2L) scheduling approach for multisite execution of SWfs and multisite scheduling algorithm, *i.e.* DIM. The first level performs multisite scheduling, where each task is scheduled to a site. DIM works at this level. The second level performs single site scheduling, where each task is scheduled to a VM by the default scheduling strategy (Round-Robin) of Chiron [14]. In this paper, we focus on the first level.

Algorithm 1 Data-Intensive Multisite task scheduling (DIM)

Input: T: a bag of tasks to be scheduled; S: a set of cloud sites
Output: SP: the scheduling plan for T in S
1: $SP \leftarrow \emptyset$
2: **for each** $t \in T$ **do**
3: $s \leftarrow GetDataSite(t)$
4: $SP \leftarrow SP \cup \{Schedule(t, s)\}$
5: EstimateTime(T, s, SP)
6: **while** MaxunbalanceTime(T, s, SP) is reduced in the last loop **do**
7: $sMin \leftarrow MinTime(S)$
8: $sMax \leftarrow MaxTime(S)$
9: TaskReschedule($sMin$, $sMax$, SP)
end

Our multisite scheduling algorithm, *i.e.* DIM, schedules a bag of tasks, *i.e.* the tasks of an activity, onto multiple sites (see Algorithm 1). First, the tasks are scheduled according to the location of input data (lines 2-5), which is similar to the scheduling algorithm of MapReduce [4]. Line 3 searches the site s that stores the biggest part of input data corresponding to task t. Line 4 schedules task t at site s. Line 5 estimates the execution time of all the tasks scheduled at site s with consideration of generating provenance data and intersite data transfer. Then, the execution time at each site is balanced by adjusting the whole bag of tasks scheduled at that site (lines 6-9). Line 6 checks if the maximum difference of the estimated execution time of tasks at each site can be reduced by verifying if the difference is reduced in the previous loop or if this is the first loop. While the maximum difference of execution time can be reduced, the tasks of the two sites are exchanged as described in lines 7-9. Line 7 and 8 choose the site that has the minimal execution time and the site that has the maximum execution time, respectively. Then, the scheduler calls the function $TaskReschedule$ to

exchange the tasks scheduled at the two selected sites to reduce the maximum difference of execution time.

In order to achieve load balancing of two sites, we propose $TaskReschedule$ algorithm. Let us assume that there are two sites, *i.e.* sites s_i and s_j. For the tasks scheduled at each site, we assume that the execution time of site s_i is bigger than site s_j. In order to balance the execution time at sites s_i and s_j, some of the tasks scheduled at site s_i should be rescheduled at site s_j. Algorithm 2 gives the method to reschedule a bag of tasks from site s_i to site s_j in order to balance the load between the two sites. Line 1 calculates the difference of the execution time of two sites with a scheduling plan. Line 2 gets all the tasks scheduled at site s_i. For each task t in T_i (line 3), it is rescheduled at site s_j if the difference of execution time of the two sites can be reduced (lines 4-8). Line 4 reschedules task t at site s_j. Line 5 calculates the execution time at sites s_i and s_j with consideration of the time to generate provenance data. Lines 6-7 updates the scheduling plan if it can reduce the difference of execution time of the two sites by rescheduling task t.

Algorithm 2 Exchange Tasks

Input: s_i: a site that has bigger execution time for its scheduled tasks; s_j: a site that has smaller execution time for its scheduled tasks; SP: original scheduling plan for a bag of tasks T

Output: SP: modified scheduling plan

1: $Diff \leftarrow CalculateExecTimeDiff(s_i, s_j, SP)$
2: $T_i \leftarrow GetScheduledTasks(s_i, SP)$
3: **for each** $t \in T_i$ **do**
4: $\quad SP' \leftarrow ModifySchedule(SP, \{Schedule(t, s_j)\}$
5: $\quad Diff' \leftarrow CalculateExecTimeDiff(s_i, s_j, SP')$
6: $\quad$ **if** $Diff' < Diff$ **then**
7: $\quad\quad SP \leftarrow SP'$
8: $\quad\quad Diff \leftarrow Diff'$

end

Let us assume n tasks to be scheduled at m sites. The complexity of the DIM algorithm $\mathcal{O}(m \cdot n \cdot \log n)$ is only a little bit higher than that of OLB and MCT ($\mathcal{O}(m \cdot n)$), but yields high reduction in SWf execution.

4 Experimental Evaluation

This section gives our experimental evaluation of the DIM algorithm, within Microsoft Azure [1]. We instantiated three A4 [3] (8 CPU cores) VMs at each of three site, *i.e.* Central US (CUS), West Europe (WEU) and North Europe (NEU). We deployed an A2 [3] (2 CPU cores) VM and install PostgreSQL database as provenance database in that VM. WEU is selected as master site. We compare our proposed algorithm with two representative baseline scheduling

algorithms, *i.e.* Opportunistic Load Balancing (OLB) and Minimum Completion Time (MCT). In the multisite environment, OLB randomly selects a site for a task while MCT schedules a task to the site that can finish the execution first.

Table 1: **Execution results.** The unit of execution time is minute. The data transfer represents the size of intersite transferred data. The unit of data is MB. The scheduling time represent the time to execute the scheduling algorithms. The unit of scheduling time is second.

Algorithm	SWf	DIM	MCT	OLB	SWf	DIM	MCT	OLB
Execution Time	Buzz 0	35.5	39.3	70.4	Montage 0	6.13	8.57	11.98
	Buzz 1	389	514	719	Montage 1	18.12	21.68	22.05
Data Transfer	Buzz 0	4.8	6.7	10.0	Montage 0	411	237	576
	Buzz 1	172	1408	1492	Montage 1	1299	1173	1951
Scheduling Time	Buzz 1	633	109	17	Montage 1	29.2	28.8	1.5

We performed experiments with the Buzz SWf using a DBLP 2013 XML file of 1.29GB as input data and the other with the Montage SWf using 5.5 GB input data. For Buzz SWf, 0 represents 60 MB input data and 1 represents 1.29 GB input data. For Montage, 0 represents 0.5 degree and 1 represents 1 degree [6]. The input data of each SWf is evenly partitioned and stored at the three sites while the configuration data is present at all the three sites. The execution results (see Table 1) show that the execution time corresponding to DIM is up to 24.3% smaller than that corresponding to MCT and up to 49.6% smaller than that corresponding to OLB although it corresponds to bigger time to execute the scheduling algorithm, *i.e.* scheduling time. The size of the data transferred between different sites corresponding to MCT is up to 7.19 times bigger than that corresponding to DIM and the size corresponding to OLB is up to 7.67 times bigger than that corresponding to DIM.

Table 2: **Provenance data distribution of the execution of Buzz.** All represents the size of all the provenance data. The distribution represents the percentage of provenance data generated at each site. The unit of data is MB

Algorithm	DIM	MCT	OLB
All	301	280	279
WEU	43	22	35
NEU	34	36	33
CUS	23	42	32

Furthermore, we measured the distribution of the provenance data during the execution of the Buzz SWf with 1.29 GB input data, which shows bigger advantage of DIM over MCT and OLB compared with that of Montage SWf.

The amount of the provenance data corresponding to the three scheduling algorithms are similar (the difference is less than 8%). The bandwidth between the provenance database and the site is in the following order: WEU > NEU > CUS. As shown in Table 2, the percentage of provenance data at WEU corresponding to DIM is much bigger than MCT (up to 95% bigger) and OLB (up to 97% bigger). This indicates that DIM can schedule tasks with provenance data to the site (WEU) that has bigger bandwidth with the provenance database. This reduces the time to generate provenance data in order to reduce the overall multisite execution time of SWfs.

Since the DIM algorithm considers the time to transfer intersite provenance data and makes optimization for a bag of tasks, it can reduce the total execution time. Because DIM schedules the tasks to where the input data is located at the beginning, DIM can reduce the amount of intersite transferred data. MCT only optimizes the load balancing for each task among different sites without consideration of the time to transfer intersite provenance data. It is a greedy algorithm that can reduce the execution time by balancing the execution time of each site while scheduling each task. However, it cannot optimize the scheduling for the whole execution of all the tasks of an activity. In addition, compared with OLB, MCT cannot reduce much the transferred data among different sites. Since OLB simply tries to keep all the sites working on arbitrary tasks, it has the worst performance. Because of the interaction with the provenance database and higher complexity, the scheduling time of DIM is much bigger than MCT and OLB. However, the scheduling time of the three scheduling algorithms is always small compared with the total execution (less than 3%), which is acceptable for the task scheduling during SWf execution. Although the scheduling time of DIM is much bigger than MCT and OLB, the total execution time of SWfs corresponds to DIM is much smaller than that of MCT and OLB, which means that DIM generates better scheduling plans compared with MCT and OLB.

5 Conclusion

In this paper, we proposed a solution, based on multisite Chiron to execute SWfs in a multisite cloud with geographically distributed input data. We proposed the architecture of multisite Chiron and a global method to gather the distributed provenance data in a centralized database. Based on this architecture, we proposed a new scheduling algorithm, *i.e.* DIM, which considers the latency to transfer data and to generate provenance data in multisite cloud. The complexity of DIM ($\mathcal{O}(m \cdot n \cdot \log n)$) is quite acceptable for scheduling bags of tasks. We used the Buzz SWf and the Montage SWf to evaluate the DIM algorithm in Microsoft Azure with three sites. The experiments show that DIM is much better than two representative baseline algorithms, *i.e.* MCT (up to 24.3%) and OLB (up to 49.6%), in terms of execution time and that DIM can also reduce significantly data transfer between sites, compared with MCT (up to 719%) and OLB (up to 767%).

References

1. Microsoft Azure. `http://azure.microsoft.com`.
2. Montage. `http://montage.ipac.caltech.edu/docs/gridtools.html`.
3. VM parameters in Azure.
   `http://msdn.microsoft.com/en-us/library/azure/dn197896.aspx`
   `http://windowsazureguide.net/tag/auzre-virtual-machines-sizes-bandwidth/`.
4. J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *6th Symposium on Operating System Design and Implementation (OSDI)*, pages 137–150, 2004.
5. E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528–540, 2009.
6. E. Deelman, G. Singh, M. Livny, B. Berriman, and J. Good. The cost of doing science on the cloud: The montage example. In *Int. Conf. for High Performance Computing, Networking, Storage and Analysis.*, pages 1–12, 2008.
7. E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. S. Katz. Pegasus: A framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming*, 13(3):219–237, 2005.
8. J. Dias, E. S. Ogasawara, D. de Oliveira, F. Porto, P. Valduriez, and M. Mattoso. Algebraic dataflows for big data analysis. In *IEEE Int. Conf. on Big Data*, pages 150–155, 2013.
9. R. Duan, R. Prodan, and X. Li. Multi-objective game theoretic schedulingof bag-of-tasks workflows on hybrid clouds. *IEEE Transactions on Cloud Computing*, 2(1):29–42, 2014.
10. K. Etminani and M. Naghibzadeh. A min-min max-min selective algorihtm for grid task scheduling. In *The Third IEEE/IFIP Int. Conf. in Central Asia on Internet (ICI 2007)*, pages 1–7, 2007.
11. J. Liu, E. Pacitti, P. Valduriez, and M. Mattoso. A survey of data-intensive scientific workflow management. *Journal of Grid Computing*, pages 1–37, 2015.
12. J. Liu, V. Silva, E. Pacitti, P. Valduriez, and M. Mattoso. Scientific workflow partitioning in multi-site clouds. In *BigDataCloud'2014: 3rd Workshop on Big Data Management in Clouds in conjunction with Euro-Par 2014*, page 12, 2014.
13. M. Maheswaran, S. Ali, H. J. Siegel, D. Hensgen, and R. F. Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *8th Heterogeneous Computing Workshop*, page 30, 1999.
14. E. S. Ogasawara, J. Dias, V. Silva, F. S. Chirigati, D. de Oliveira, F. Porto, P. Valduriez, and M. Mattoso. Chiron: a parallel engine for algebraic scientific workflows. *Concurrency and Computation: Practice and Experience*, 25(16):2327–2341, 2013.
15. L. Pineda-Morales, A. Costan, and G. Antoniu. Towards multi-site metadata management for geographically distributed cloud workflows. In *2015 IEEE Int. Conf. on Cluster Computing, CLUSTER*, pages 294–303, 2015.
16. S. Smanchat, M. Indrawan, S. Ling, C. Enticott, and D. Abramson. Scheduling multiple parameter sweep workflow instances on the grid. In *5th IEEE Int. Conf. on e-Science*, pages 300–306, 2009.
17. H. Topcuouglu, S. Hariri, and M. Wu. Performance-effective and low-complexity task scheduling for heterogeneous computing. *IEEE Trans. on Parallel and Distributed Systems*, 13(3):260–274, 2002.
18. M. Wieczorek, R. Prodan, and T. Fahringer. Scheduling of scientific workflows in the askalon grid environment. *SIGMOD Record*, 34(3):56–62, 2005.