Volume 5, Issue 2, February 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Basic Requirements for Temporal Constraint Management in Scientific Workflow Systems for High Quality Performance in Cloud Computing

Prof. Jasobanta Laha
Vice – Principal, SERC
Bhubaneswar, India

Prof. (Dr.) R. N. Satpathy
Principal, HIT,
Bhubaneswar, India

Prof. Kaustuva Dev
HOD – MCA, TACT,
Bhubaneswar, India

Abstract: ***The most important workflow QoS (Quality of Service) dimensions and the focus is the temporal constraint which is the foundation of the entire temporal framework. Temporal constraint setting is to deal with temporal QoS specifications at workflow build time. The high temporal QoS cannot be achieved without proper setting of high quality temporal constraints. In scientific workflow systems, it is critical to ensure the timely completion of scientific workflows. Therefore, temporal constraints as a type of QoS (Quality of Service) specification are usually required to be managed in scientific workflow systems. In this paper, with a systematic analysis of the above issues, we propose the basic requirements for temporal constraint which utilizes a novel probability-based temporal consistency model. Specifically for constraint setting, a negotiation process between the client and the service provider is designed to support the setting of coarse-grained temporal constraints and then automatically derive the fine-grained temporal constraints; for constraint updating, the probability time deficit/redundancy propagation process is proposed to update run-time fine-grained temporal constraints when workflow execution is either ahead of or behind the schedule.***

Key Words: Scientific workflow system, QoS, Temporal constraints, SLA, Coarse-grained temporal constraint, Fine-grained temporal constraint

I. INTRODUCTION

Temporal constraint setting is to deal with temporal QoS specifications at workflow build time. The high temporal QoS cannot be achieved without proper setting of high quality temporal constraints. Temporal constraint setting assigns both global temporal constraints (temporal constraints for entire workflow instances) and local temporal constraints (temporal constraints for local workflow segments and individual workflow activities) in scientific cloud workflow specifications at workflow build time and temporal constraints as a type of QoS requirements, are to be specified in cloud workflow definitions. With other QoS constraints such as cost and security, these temporal constraints serve as critical criteria for the selection of cloud services and the SLA (Service Level Agreement) management [1]. During cloud workflow runtime, service providers are obligated to complete workflow processes within the assigned temporal constraints, otherwise penalty may be enforced according to the service contracts signed. The setting of high quality temporal constraints is essential to the successful completion of scientific cloud workflows.

In scientific workflow systems, temporal consistency is critical to ensure the timely completion of workflow instances. The correctness of temporal consistency, temporal constraints are often set and then verified. The current work adopts, user specified temporal constraints without considering system performance, and may result in frequent temporal violations that deteriorate the overall workflow execution effectiveness. The task of setting temporal constraints described in this paper is to assign a set of coarse-grained and fine-grained upper bound temporal constraints to scientific workflows. Coarse-grained constraints refer to those assigned to the entire workflow or workflow segments, while fine-grained constraints refer to those assigned to individual activities. Though coarse-grained constraints can be deemed as collection of fine-grained constraints, they are not in a simple relationship of linear culmination and decomposition. To ensure on-time fulfillment of workflow instances, both coarse-grained and fine-grained temporal constraints are required. When scientific workflows are deployed on dynamic computing infrastructures like grid the performance of the underlying resources is highly uncertain [2]. The quality of temporal constraints can be measured by two criteria:

i) Well balanced between user requirements and system performance
ii) Well supported for both overall coarse-grained control and local fine-grained control

II. ASSIGNMENT OF QOS CONSTRAINTS:

Generally, there are two basic ways to assign QoS constraints,

I. Activity-level assignment and
II. Workflow-level assignment

Since the whole workflow process is composed by individual activities, an overall workflow-level constraint can be obtained by the composition of activity-level constraints. On the contrary, activity level constraints can also be assigned by the decomposition of workflow-level constraints [3]. However, different QoS constraints have their own characteristics which require in depth research to handle different scenarios.

For setting temporal constraint, the primary information which includes the workflow process models, statistics of activity durations and the temporal consistency model. Scientific workflows require the explicit representation of temporal information, i.e. activity durations and temporal constraints to facilitate temporal verification. One of the classical modelling methods is the Stochastic Petri Nets [4, 5] which incorporates time and probability attributes into workflow processes that can be employed to facilitate scientific workflow modelling. Activity duration is one of the basic elements to measure system performance, is of significant value to workflow scheduling, performance analysis and temporal verification [6, 7, 8, 9]. Most work obtains activity durations from workflow system logs and describes them by a discrete or continuous probability distribution through statistical analysis [4, 10, 11]. In temporal consistency, traditionally, there are only binary states of consistency or inconsistency. However, the conventional consistency condition is too restrictive and covers several different states which should be handled differently for the purpose of cost effectiveness. Therefore, it divides conventional inconsistency into weak consistency, weak inconsistency and strong inconsistency and treats them accordingly. However, multiple-state based temporal consistency model cannot support quantitative measurement of temporal consistency states and lacks the ability to support statistical analysis for constraint management. Therefore, a probability based build-time temporal consistency model is set to facilitate the setting of temporal constraints.

The workflow modelling is highly related to the specification of temporal constraints. It also concerns with two aspects of the modelling language and the modelling tool (language-based, graph-based or both) in addition to the three aspects of whether they support the specification of temporal constraints (the specification of temporal constraints in workflow models), the management of temporal constraints (i.e. the setting and updating of temporal constraints) and the temporal verification (the verification of temporal constraints). In the Table - I, among the 10 representative scientific workflow projects (ASKALON [12], CROWN workflow component [13], DAGMan [14], GridBus [15], JOpera [16], Kepler [17], SwinDeW-G [18], Taverna [19], Triana [20] and UNICORE [21]), most projects use XML-like modeling language and support language-based or graph-based modelling tool. Therefore, in the modelling stage, a temporal constraint can either be inexplicitly specified as an element in the XML document or explicitly as a graphic component in the workflow template. As for the representation of temporal constraints, the management of temporal constraints and the support of temporal verification which are the most concerned, only some of the projects such as ASKALON, DAGMan, GridBus, JOpera, Kepler, Taverna and SwinDeW-G clearly stated that temporal constraints are supported in their system QoS control or performance analysis. Yet, to the best of our knowledge, only SwinDeW-G has set up a series of strategies such as the probabilistic strategy for temporal constraint management and the efficient checkpoint selection strategy to support dynamic temporal verification [2].

Table – I - Support of temporal qos constraints

Scientific Workflow Systems	***Modelling Language***	***Modelling Tool***	***Temporal Constraint Specification***	***Temporal Constraint Management***	***Temporal Constraint Verification***
ASKALON	AGWL	Language – Based Graph - Based	Supported	N/A	N/A
CROWN	GPEL	Language – Based	N/A	N/A	N/A
DAGMan	DAG Scripts	Language – Based	Supported	N/A	N/A
GridBus	xWFL	Language – Based Graph - Based	Supported	N/A	N/A
JOpera	JVCL	Language – Based Graph - Based	Supported	N/A	N/A
Kepler	SDF	Graph - Based	Supported	N/A	N/A
SwinDeW – G	XPDL/BPEL	Graph - Based	Supported	Supported	Supported
Taverna	SCUFL	Language – Based Graph - Based	Supported	N/A	N/A
Triana	WSFL	Language – Based Graph - Based	N/A	N/A	N/A
UNICORE	BPEL	Language – Based Graph - Based	N/A	N/A	N/A

Basic Requirements for Temporal Constraint Setting:

The quality of temporal constraints can be measured by two criteria:

i) Well balanced between user requirements and system performance

ii) Well supported for both overall coarse-grained control and local fine-grained control

Well balanced between user requirements and system performance:

Temporal constraints should be well balanced between user requirements and system performance. It is common that users often suggest coarse-grained temporal constraints based on their own interest while with limited knowledge about the actual performance of workflow systems. For example, it is not rational to set a 60 minutes temporal constraint to the segment which normally needs two hours to finish. Therefore, user specified constraints are normally prone to cause frequent temporal violations. To address this problem, a negotiation process between the user and the service provider who is well aware of the system performance is desirable to derive balanced coarse-grained temporal constraints that both sides are satisfied with.

Well supported for both overall coarse-grained control and local fine-grained control:

Temporal constraints should facilitate both overall coarse-grained control and local fine-grained control. As analysed above, this criterion actually means that temporal constraint setting should support both coarse-grained temporal constraints and fine-grained temporal constraints. Specifically, the task of setting build time temporal constraints includes setting both coarse-grained temporal constraints (a global deadline for the entire workflow instance and several milestones for local workflow segments) and fine-grained temporal constraints (temporal constraints for individual workflow activities). However, although the overall workflow process is composed of individual workflow activities, coarse-grained temporal constraints and fine-grained temporal constraints are not in a simple relationship of linear culmination and decomposition. Meanwhile, it is impractical to set or update fine grained temporal constraints manually for a large amount of activities in scientific workflows. Since coarse-grained temporal constraints can be obtained through the negotiation process, the problem for setting fine-grained temporal constraints is how to automatically derive them based on the coarse-grained temporal constraints.

III. CONCLUSION

In the concluding line, the basic requirements for temporal constraint setting in scientific Cloud workflow systems can be put as: effective negotiation for setting coarse grained temporal constraints and automatically derive fine-grained temporal constraints. Our temporal constraint management consists of a negotiation-based probabilistic strategy for setting temporal constraints at build-time and a probabilistic strategy for updating temporal constraints at run-time. The setting strategy aims to achieve a set of coarse-grained and fine-grained temporal constraints which are well balanced between the user requirements and system performance. With the probability based temporal consistency, well-balanced overall coarse-grained temporal constraints can be obtained through either a time-oriented or probability-oriented negotiation process. Thereafter, fine-grained temporal constraints for each activity can be propagated instantly in an automatic fashion.

REFERENCES

[1] X. Liu, J. Chen, and Y. Yang, "A Probabilistic Strategy for Setting Temporal Constraints in Scientific Workflows", Proc. 6th International Conference on Business Process Management (BPM08), pp. 180-195, Milan, Italy, Sept. 2008.

[2] J. Chen and Y. Yang, Taxonomy of Grid Workflow Verification and Validation", Concurrency and Computation: Practice and Experience, vol. 20, no. 4, pp. 347-360, 2008.

[3] J. Yu and R. Buyya, "A Taxonomy of Workflow Management Systems for Grid Computing", *Journal of Grid Computing*, no. 3, pp. 171-200, 2005.

[4] W. M. P. van der Aalst, K. M. V. Hee, and H. A. Reijers, "Analysis of Discrete-Time Stochastic Petri Nets", *Statistica Neerlandica*, vol. 54, no. 1, pp. 237-255, 2000.

[5] G. Bucci, L. Sassoli, and E. Vicario, "Correctness Verification and Performance Analysis of Real-Time Systems Using Stochastic Preemptive Time Petri Nets", *IEEE Trans. on Software Engineering*, vol. 31, no. 11, pp. 913-927, 2005.

[6] J. Chen and Y. Yang, "A Taxonomy of Grid Workflow Verification and Validation", *Concurrency and Computation: Practice and Experience*, vol. 20, no. 4, pp. 347-360, 2008.

[7] J. Chen and Y. Yang, "Temporal Dependency based Checkpoint Selection for Dynamic Verification of Temporal Constraints in Scientific Workflow Systems", *ACM Trans. on Software Engineering and Methodology*, in press, http://www.swinflow.org/papers/TOSEM.pdf, accessed on 1st Nov. 2010.

[8] J. H. Son, J. Sun Kim, and M. Ho Kim, " Extracting the Workflow Critical Path from the Extended Well-Formed Workflow Schema", *Journal of Computer and System Sciences*, vol. 70, no. 1, pp. 86-106, 2005.

[9] J. Yu and R. Buyya, "Workflow Scheduling Algorithms for Grid Computing", Metaheuristics for Scheduling in Distributed Computing Environments, F. Xhafa and A. Abraham (Eds.), Springer, 2008.

[10] X. Liu, J. Chen, and Y. Yang, "A Probabilistic Strategy for Setting Temporal Constraints in Scientific Workflows", *Proc. 6th International Conference on Business Process Management (BPM08)*, pp. 180-195, Milan, Italy, Sept. 2008.

[11] K. A. Stroud, *Engineering Mathematics (Sixth Edition)*: Palgrave Macmillan, 2007.

[12] Askalon Project, http://www.dps.uibk.ac.at/projects/askalon, accessed on 1st Nov. 2010

[13] CROWN Project, CROWN portal, http://www.crown.org.cn/en/, accessed on 1st Nov. 2010.

[14] DAGMan, Condor Project, http://www.cs.wisc.edu/condor/, accessed on 1st Nov. 2010.

[15] GridBus Project, http://www.gridbus.org, accessed on 1st Nov. 2010.

[16] JOpera Project, http://www.iks.ethz.ch/jopera, accessed on 1st Nov. 2010.
[17] Kepler Project, http://kepler-project.org/, accessed on 1st Nov. 2010.
[18] Y. Yang, K. Liu, J. Chen, J. Lignier, and H. Jin, "Peer-to-Peer Based Grid Workflow Runtime Environment of SwinDeW-G", *Proc. 3rd International Conference on e-Science and Grid Computing (e-Science07)*, pp. 51-58, Bangalore, India, Dec. 2007.
[19] Taverna Project, http://www.mygrid.org.uk/tools/taverna/, accessed on 1st Nov. 2010.
[20] Triana Project, http://www.trianacode.org/, accessed on 1st Nov. 2010.
[21] UNICORE Project, http://www.unicore.eu/, accessed on 1st Nov. 2010.

