Journal of Computer and System Sciences 76 (2010) 464–474

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Localising temporal constraints in scientific workflows

Jinjun Chen, Yun Yang *

CS3 – Centre for Complex Software Systems and Services, Faculty of Information and Communication Technologies, Swinburne University of Technology, PO Box 218, Hawthorn, Melbourne, Australia 3122

A R T I C L E I N F O

Article history:
Received 20 March 2009
Received in revised form 20 August 2009
Available online 22 November 2009

Keywords:
Scientific workflow
Temporal constraint
Localisation

A B S T R A C T

Temporal constraints are often set when complex e-science processes are modelled as scientific workflow specifications. However, many existing processes such as climate modelling often have only a few coarse-grained temporal constraints globally. This is not sufficient to control overall temporal correctness as we can not find temporal violations locally in time for handling. Local handling affects fewer workflow activities, hence more cost effective than global handling with coarse-grained temporal constraints. Therefore, in this paper, we systematically investigate how to localise a group of fine-grained temporal constraints so that temporal violations can be indentified locally for better handling cost effectiveness. The corresponding algorithms are developed. The quantitative evaluation demonstrates that with local fine-grained temporal constraints, we can improve handling cost effectiveness significantly than only with coarse-grained ones.

1. Introduction and motivation

Scientific workflows often sit in sophisticated scientific applications such as climate modelling and astrophysics simulation [2,3,11,20,23]. They enable complex scientific computation to be performed step by step [10,12,19,21,30].

In reality, scientific workflows are normally time constrained as temporal correctness is critical to ensure the usefulness of execution results [4,8,13,22]. Consequently, temporal constraints are often set. The types of temporal constraints mainly include: upper bound, lower bound and fixed-time [8,13]. An upper bound constraint between two activities is a relative time value so that the duration between them must be less than or equal to it. A lower bound constraint between two activities is a relative time value so that the duration between them must be greater than or equal to it. A fixed-time constraint at an activity is an absolute time value such as 6:00pm by which the activity must be completed.

Comparing the three types of temporal constraints, we can find that conceptually a lower bound constraint is symmetrical to an upper bound constraint while a fixed-time constraint is a special case of upper bound constraint. The reasons are as follows. For a lower bound constraint, we often check whether the duration between its start and end activities is greater than or equal to ($\geqslant$) its value while for an upper bound constraint, we often check whether the duration between its start and end activities is less than or equal to ($\leqslant$) its value. Therefore, they are symmetrical to each other. As for a fixed-time constraint, the first activity of a scientific workflow is actually its start activity. Hence, a fixed-time constraint can be viewed as a special upper bound constraint whose start activity is the first activity and whose end activity is the one at which the fixed-time constraint is. Nevertheless, an upper bound constraint is conceptually more general than a fixed-time constraint as its start activity can be an intermediate activity rather than the first activity. Besides, different upper bound constraints can have different start activities while all fixed-time constraints have the same one which is the first activity.

* Corresponding author.
E-mail addresses: jchen@swin.edu.au (J. Chen), yyang@swin.edu.au (Y. Yang).

0022-0000/$ – see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.jcss.2009.11.007

As such, in this paper, we focus on upper bound constraints only. The corresponding discussion and results can be symmetrically applied to lower bound constraints and adaptively simplified for fixed-time constraints.

In many scientific workflows such as climate modelling, we often have only a few coarse-grained upper bound constraints globally [2,24]. From the perspective of user needs, only a few coarse-grained upper bound constraints are intuitive and simple. However, from the perspective of specific scientific workflow execution, we cannot identify temporal violations locally in time for handling. Local handling affects fewer workflow activities than global handling with coarse-grained constraints, hence more cost effective. Therefore, we must investigate how to localise a group of fine-grained upper bound constraints based on coarse-grained ones so that we can identify temporal violations locally for better handling cost effectiveness. The existing related work has presented some background for the temporal aspect in scientific workflows. [4,24] analyses QoS (Quality of Service) including temporal QoS in scientific workflows on grid and discusses how to provide QoS including time. [14] examines key challenges in scientific workflow area including time aspect. [5] investigates multiple temporal consistency states in scientific/grid workflows. [16] discusses fault tolerance and recovery in scientific workflows including time management. [18] proposes a reference architecture for scientific workflow management in service computing environment. [25] analyses the overhead of scientific workflow execution in grid environment. [29] proposes a p2p based scientific workflow management architecture with time management included. [22] presents a method for dynamic verification of temporal constraints. [31] proposes a taxonomy for scientific workflow management. Several metrics are proposed to categorise scientific workflow management with time as one of them. [6–8] propose several strategies for selecting checkpoints for verifying temporal constraints.

However, the above existing work does not pay sufficient attention to how to localise fine-grained upper bound constraints. Hence, in this paper, we make an effort to fill this gap by systematically investigating the issue. We take one of coarse-grained upper bound constraints as the example to discuss how to localise fine-grained upper bound constraints within its timeframe. The corresponding results can be equally applied to each of other coarse-grained upper bound constraints. Based on the investigation, we develop the corresponding algorithms. With fine-grained upper bound constraints, we can achieve better cost effectiveness significantly than only with coarse-grained ones. The quantitative evaluation further demonstrates this result.

The paper is organised as follows. In Section 2, we summarise some time attributes of scientific workflows. In Section 3, we discuss how to localise fine-grained upper bound constraints including assignment and adjustment of them. The corresponding algorithms are developed. In Section 4, we conduct a quantitative evaluation which demonstrates that based on these algorithms we can achieve better handling cost effectiveness significantly than only based on coarse-grained upper bound constraints. Finally in Section 5, we conclude our contributions and point out future work.

2. Overview of timed scientific workflow representation

According to [1,17,27], based on the directed graph concept, a scientific workflow can be represented by a scientific workflow graph, where nodes correspond to activities and edges correspond to dependencies between them. To represent time attributes in a scientific workflow, we borrow some concepts from [13,22] such as maximum, mean or minimum duration as a basis. We denote the ith activity of a scientific workflow as a_i and its maximum duration, mean duration, minimum duration, run-time start time, run-time end time and run-time completion duration as $D(a_i)$, $M(a_i)$, $d(a_i)$, $S(a_i)$, $E(a_i)$ and $R(a_i)$, respectively. $M(a_i)$ means that statistically a_i can be completed around its mean duration. Other time attributes are self-explanatory. According to [9,28], $D(a_i)$, $M(a_i)$ and $d(a_i)$ can be obtained based on the past execution history which covers the delay time incurred at a_i such as setup delay, queuing delay, synchronisation delay, network latency and so on. The detailed discussion on how to obtain and set $D(a_i)$, $M(a_i)$ and $d(a_i)$ is outside the scope of this paper and can be found in [9,28]. For a specific execution of a_i, the delay time is included in $R(a_i)$. Normally, we have $d(a_i) \leqslant M(a_i) \leqslant D(a_i)$ and $d(a_i) \leqslant R(a_i) \leqslant D(a_i)$.

If there is a path from a_i to a_j ($i \leqslant j$), we denote the maximum duration, minimum duration, mean duration, run-time real completion duration between them as $D(a_i, a_j)$, $d(a_i, a_j)$, $M(a_i, a_j)$ and $R(a_i, a_j)$, respectively [9,28]. If there is an upper bound constraint between a_i and a_j, we denote it as $U(a_i, a_j)$ and its value as $u(a_i, a_j)$. For convenience, we only consider one execution path in the scientific workflow without losing generality. As to a selective or parallel structure, for each branch, it is an execution path. For an iterative structure, from the start to the end, it is still an execution path. Therefore, for the selective/parallel/iterative structures, we can also apply the results achieved from one execution path.

Besides the above time attributes, four temporal consistency states have been identified and defined in [5,8] which are SC (Strong Consistency), WC (Weak Consistency), WI (Weak Inconsistency) and SI (Strong Inconsistency). We summarise their definitions in Definitions 1, 2 and 3. The detailed discussion about the four consistency states can be found in [5,8].

Definition 1. At build-time stage, $U(a_i, a_j)$ is said to be of SC if $D(a_i, a_j) \leqslant u(a_i, a_j)$, WC if $M(a_i, a_j) \leqslant u(a_i, a_j) < D(a_i, a_j)$, WI if $d(a_i, a_j) \leqslant u(a_i, a_j) < M(a_i, a_j)$, and SI if $u(a_i, a_j) < d(a_i, a_j)$.

Definition 2. At run-time execution stage, at checkpoint a_p between a_i and a_j, $U(a_i, a_j)$ is said to be of SC if $R(a_i, a_p) + D(a_{p+1}, a_j) \leqslant u(a_i, a_j)$, WC if $R(a_i, a_p) + M(a_{p+1}, a_j) \leqslant u(a_i, a_j) < R(a_i, a_p) + D(a_{p+1}, a_j)$, WI if $R(a_i, a_p) + d(a_{p+1}, a_j) \leqslant u(a_i, a_j) < R(a_i, a_p) + M(a_{p+1}, a_j)$, and SI if $u(a_i, a_j) < R(a_i, a_p) + d(a_{p+1}, a_j)$.