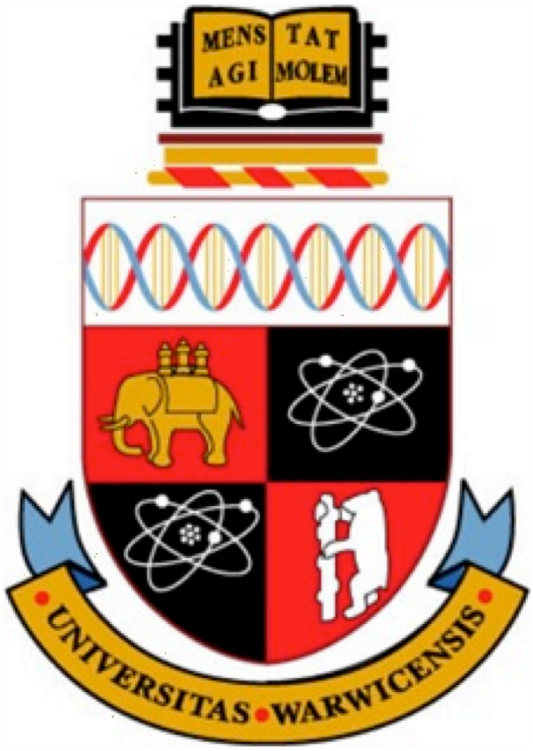
Optimizing Performance of Workflow Executions Under Authorization Control

by

Nadeem Chaudhary



Thesis

Submitted to the University of Warwick

for the Degree of

Doctor of Philosophy

Department of Computer Science

University of Warwick

June 2013

Contents

List of Tables

List Of Figures

Acknowledgements

I am very thankful to my supervisor Dr. Ligang He, for offering me the opportunity to work with him at the University of Warwick under his kind supervision. His invaluable guidance and help supported me to complete this research. He further encouraged me to continue from the Masters degree to the PhD degree programme and provided me partial financial support for my studies as well, this was very helpful for me to continue and keep focussing on my research.

I am also very thankful to my advisor Dr. Stephen A. Jarvis who helped a lot towards the partial financial support for my work. I would also like to thank all the past and present members at the Department of Computer Science, especially Dr. Alexandra I. Cristea, Dr. Jane Sinclair, Dr. Khalid Masood, Dr. Hammad Qureshi, Dr. Nasir Rajpoot, Dr. Simon Hammond, Dr. Mohammed Al-Ghamdi, Alaa Khadidos, Adnan Mujahid, Shan-e-Ahmad Raza and Malik Shahzad Kaleem Awan. I would also like to express my gratitude to Dr. Roger Packwood and Paul Williams for their technical support and Dr. Christine

Leigh for her kind support and help in administrative and financial affairs during the period.

Finally, and most especially, thanks to my family: my parents, wife, son - Shayaan, brothers, sisters and relatives for their love, support and kindness during this long period.

Declaration

This thesis is presented in accordance with the regulations for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. The work in this thesis has been undertaken by myself except where otherwise stated.

The various aspects of research that are documented in this thesis have also been published in the following publication: [He2012b] [He2013] [Chaudhary2013]. The publications on which each chapter is based and a full list of the 'research results' that relate to this work are summarised at the beginning of the thesis.

Research Context And Results

University of Warwick and the Department of Computer Science sponsored the research work presented in this thesis partially.

The work presented in this thesis is also supported by the Leverhulme Trust (grant number RPG-101). The research work has resulted in the following one journal and two conferences publications.

Journal Publications

- Ligang He, Nadeem Chaudhary and Stephen A. Jarvis "Developing Resource Allocation Strategies for workflows comprising both human and computing tasks", Future Generation Computer Systems, 2013, DOI: 10.1016/j.future.2013.09.030

Conference Publications

- Ligang He, Nadeem Chaudhary, Stephen A. Jarvis and Kenli Li. "Allocating resources for workflows

running under authorization control". Proceedings of the 13th IEEE/ACM International Conference on Grid Computing (Grid 2012), pp. 58-65, September 2012

- Nadeem Chaudhary and Ligang He. "Analyzing the Performance Impact of Authorization Constraints and Optimizing the Authorization Methods for Workflows" The 20th International Conference on High Performance Computing (HiPC 2013)

Abstract

"Business processes or workflows are often used to model enterprise or scientific applications. It has received considerable attention to automate workflow executions on computing resources. However, many workflow scenarios still involve human activities and consist of a mixture of human tasks and computing tasks.

Human involvement introduces security and authorization concerns, requiring restrictions on who is allowed to perform which tasks at what time. Role-Based Access Control (RBAC) is a popular authorization mechanism. In RBAC, the authorization concepts such as roles and permissions are defined, and various authorization constraints are supported, including separation of duty, temporal constraints, etc. Under RBAC, users are assigned to certain roles, while the roles are associated with prescribed permissions.

When we assess resource capacities, or evaluate the performance of workflow executions on supporting platforms, it is often assumed that when a task is allocated to a resource, the resource will accept the task and start the execution once a processor becomes

available. However, when the authorization policies are taken into account," this assumption may not be true and the situation becomes more complex. For example, when a task arrives, a valid and activated role has to be assigned to a task before the task can start execution. The deployed authorization constraints may delay the workflow execution due to the roles' availability, or other restrictions on the role assignments, which will consequently have negative impact on application performance.

When the authorization constraints are present to restrict the workflow executions, it entails new research issues that have not been studied yet in conventional workflow management. This thesis aims to investigate these new research issues.

First, it is important to know whether a feasible authorization solution can be found to enable the executions of all tasks in a workflow, i.e., check the feasibility of the deployed authorization constraints. This thesis studies the issue of the feasibility checking and models the feasibility checking problem as a constraints satisfaction problem.

Second, it is useful to know when the performance of workflow executions will not be affected by the given authorization constraints. This thesis proposes the methods to determine the time durations when the given authorization constraints do not have impact.

Third, when the authorization constraints do have the performance impact, how can we quantitatively analyse and determine the impact? When there are

multiple choices to assign the roles to the tasks, will different choices lead to the different performance impact? If so, can we find an optimal way to conduct the task-role assignments so that the performance impact is minimized? This thesis proposes the method to analyze the delay caused by the authorization constraints if the workflow arrives beyond the non-impact time duration calculated above. Through the analysis of the delay, we realize that the authorization method, i.e., the method to select the roles to assign to the tasks affects the length of the delay caused by the authorization constraints. Based on this finding, we propose an optimal authorization method, called the Global Authorization Aware (GAA) method.

Fourth, a key reason why authorization constraints may have impact on performance is because the authorization control directs the tasks to some particular roles. Then how to determine the level of workload directed to each role given a set of authorization constraints? This thesis conducts the theoretical analysis about how the authorization constraints direct the workload to the roles, and proposes the methods to calculate the arriving rate of the requests directed to each role under the role, temporal and cardinality constraints.

Finally, the amount of resources allocated to support each individual role may have impact on the execution performance of the workflows. Therefore, it is desired to develop the strategies to determine the adequate amount of resources when the authorization control is present in the system. This thesis presents

the methods to allocate the appropriate quantity for resources, including both human resources and computing resources. Different features of human resources and computing resources are taken into account. For human resources, the objective is to maximize the performance subject to the budgets to hire the human resources, while for computing resources, the strategy aims to allocate adequate amount of computing resources to meet the QoS requirements.

Glossary

RBAC	Role Based Authorization Control
DAG	Directed Acyclic Graph
ABAC	Attribute Based Access Control
PU	Processing Unit
CARD	Cardinality Constraint
TD	Time Domain
CP	Critical Parent
CT	Computing Task
HT	Human Tasks
ACT	Automated Computed Task
HCT	Human-Aided Computing Task
SoD	Separation of Duty
BoD	Binding of Duty
EST	Earliest Start Time
EFT	Earliest Finish Time
CTPN	Color Timed Petri Nets
PN	Petri-Net
CSP	Constraint Satisfaction Problem
FCP	Feasibility Checking Problem
CTC	Coverage of the Temporal Constraints
EAF	Earliest Activation First
GAA	Global Authorization Aware
TEMP	Temporal Constraints
rt	Response Time

1

Introduction

"Business processes or workflows are often used to model enterprise or scientific applications [Deelman2009] [He2006a] [Hsu2011] [WebBusinessProcess]. A workflow consists of multiple tasks with the order of execution, i.e., a task can only start execution after another task in the workflow is completed (the former task is called the latter's child). It has received considerable attention to automate workflow executions on computing resources, which has lead in part to BPEL being proposed as a standard for specifying and executing workflows [WebBusinessProcess]. However, many workflow scenarios still involve human activities and are comprised of a mixture of human tasks and computing tasks [Gaaloul2008] [Hara2009] [Schall2010] [Zhao2010] [VideoWorkflow]. For example, in IT-based video production workflows [VideoWorkflow], human interactions are still required for decision-making and artistic choices (e.g., video editing decisions). In mortgage business processes in banks [WebHumanTask], various human tasks (e.g., a manual approval step is required if the mortgage value exceeds some amount) could be involved in order to make the final decisions. Indeed, in many application domains, the completion of a task in a workflow replies on the subjective judgment of human. It would be very difficult,

if not possible, to use computers to completely replace human being in the foreseeable future.

In traditional workflow management systems, human interactions in a workflow are not well supported, and therefore a workflow with human involvement can be regarded as a semi-automated workflow [WS-BPEL]. Motivated by the requirements of integrating human interactions into business processes, research exists to support human tasks in workflow contexts. WS-HumanTask and BPEL4People, which have been proposed to overcome the lack of support for human activities in BPEL [WS-BPEL] [WebHumanTask], are the exemplar products of these research efforts. WS-HumanTask and BPEL4Peopl enables the integration of human tasks in business processes, and therefore the executions of the workflows containing human tasks can also be automated [WS-BPEL] [WebHumanTask].

Human involvement introduces security and authorization concerns, requiring restrictions on who is allowed to perform which tasks at what time. Research has been conducted to attach authorization information (such as roles and permissions) to activities, and to impose authorization constraints (such as separation of duty) on workflow executions [Ahn2000] [Bertino2006] [Crampton2012] [Joshi2005] [Lu2009] [Zhao2008] [zou2009]. For example, in BPEL4People, authorization concepts such as roles and permissions are defined, and various authorization constraints are supported, including cardinality constraints, separation of duty, binding of duty, etc. The authorization specified in BPEL4People can be categorized as Role-based Authorization Control (RBAC), under which users are assigned to certain roles,

while the roles themselves are associated with prescribed permissions.

When we assess resource capacities, or evaluate the performance of workflow executions on supporting platforms, it is often assumed that when a task is allocated to a resource, the resource will accept the task and start the execution once a processor becomes available. However, when human activities and authorization constraints are taken into account, the issue can become complex. The following example illustrates such a situation".

A bank will need both human activities and computing-based activities to support its business. A workflow will typically contain both Human Tasks (HT) and Computing Tasks (CT): A human task may consist of a person (or a user in the RBAC terminology) with an official position (or a role in RBAC, e.g., a branch manager) signing a document; a computing task may involve running an application on a computing resource to assess risk for an investment. Further, the computing applications may be hosted in a central resource pool (e.g. a cluster), and the invocation of an application may be automated without human intervention, which we term an Automated Computing Task (ACT), or for security reasons, can only be initiated by a user with a certain role and be executed under that role/user, which we term a Human-aided Computing Task (HCT). The following authorization constraints are often encountered in such scenarios [Zhao2008]: 1) Role constraints: A human task may only be performed by a particular role; a computing application may only be invoked by assuming a particular role; 2) Temporal constraints: A role or a user is only activated

during certain time intervals (e.g., a staff only works in morning hours); 3) Cardinality constraints: The maximum number of tasks (computing or other) running simultaneously under a role is N; 4) “Separation of Duty constraints: If Task A (HT or CT) is run by a role (or a user), then Task B must not be run by the same role (or user); 5) Binding of Duty constraints: If Task A is run by a role (or user), then Task B must be run by the same role” (or user).

Since a valid and activated role has to be assigned to a task before the task can start execution (to put security, tasks are assigned to rolls first), these authorization constraints may delay the workflow execution and consequently have negative impact on application performance. The following case study illustrates the situation.

Table 1.1 Execution times of the workflow tasks in the case study

Task	Execution time	Task	Execution time
T_1	30	T_2	30
T_3	36	T_4	42
T_5	48	T_6	42
T_7	30	T_8	36
T_9	42		

Assume a workflow consists of 9 tasks, T_1-T_9 as shown in Fig.1.1. The execution time of each task in the workflow is shown in Table 1.1.

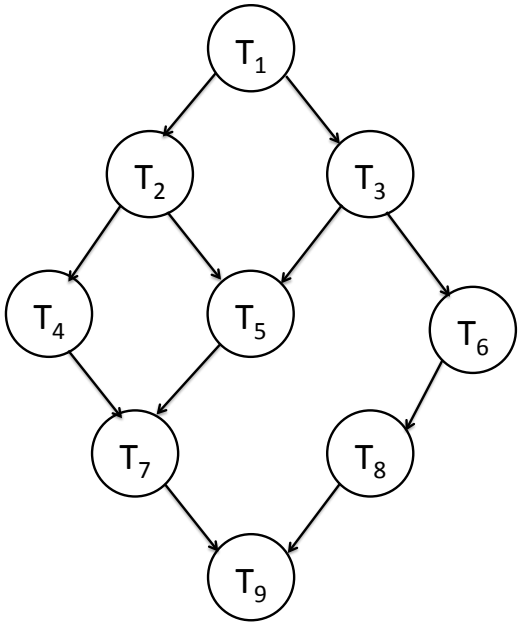


Figure 1.1. The workflow in the case study

There are 5 roles in the system. The temporal constraints of these roles are specified in Table 1.2, and illustrated in Figure 1.2, where the shaded area is the time duration when the roles are not activated.

Table 1.2 Temporal constraints of the roles in the case study

Role	Temporal Constraint	Role	Temporal Constraint
r_1	{[09:00, 17:00]}	r_2	{[12:00, 17:00]}
r_3	{[11:00, 17:00]}	r_4	{[09:00, 12:00], [14:00, 17:00]}

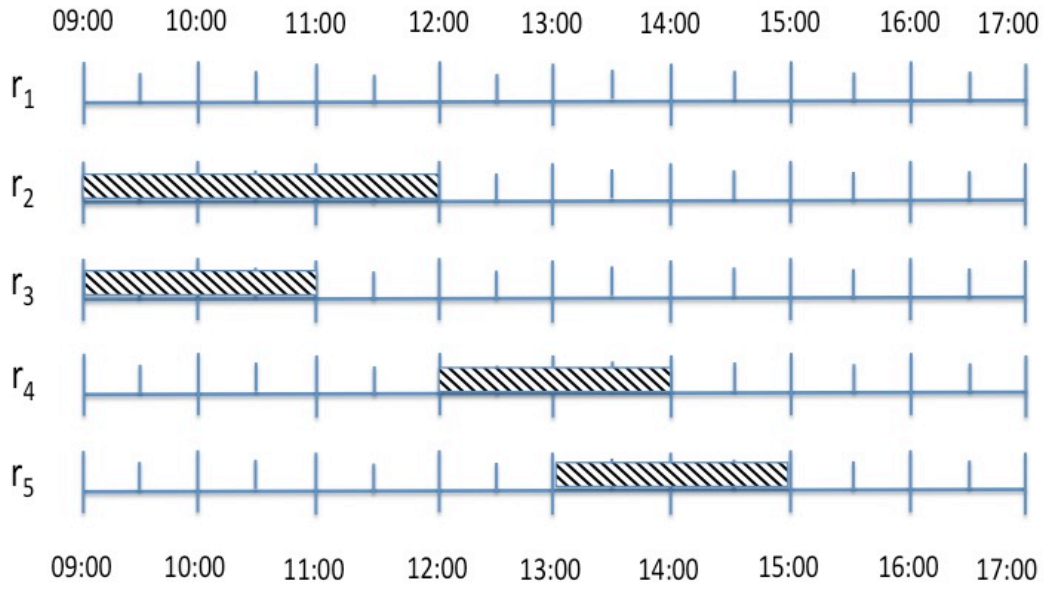


Figure 1.2. The temporal constraints of the roles

Assume the role constraints of the tasks are as follows.

T_1 -> {r_1}
T_2 -> {r_2, r_3}
T_3 -> {r_2, r_3}
T_4 -> {r_2, r_3}
T_5 -> {r_2, r_4}
T_6 -> {r_4}
T_7 -> {r_2, r_3}
T8 -> {r_2, r_3}
T9 -> {r_2, r_5}

When the first task of the workflow of figure 1.3 (i.e., T_1) arrives, it can be run under role r_1 according to the role constraints, and r_1 is always activated according to the temporal constraints. Therefore, T_1 starts execution immediately. After T_1 is completed, T_2 and T_3 are ready to run. T_2 and T_3 can be run under r_2 and

r_3. But when T_1 is completed, r_2 and r_3 are not activated. So the executions of T_2 and T_3 will be delayed by the authorization constraints. Similarly, when T_6 is ready to run, r_4, which is the role that T_6 has to assume, is not activated. Consequently, the execution of T_6 will also be delayed.

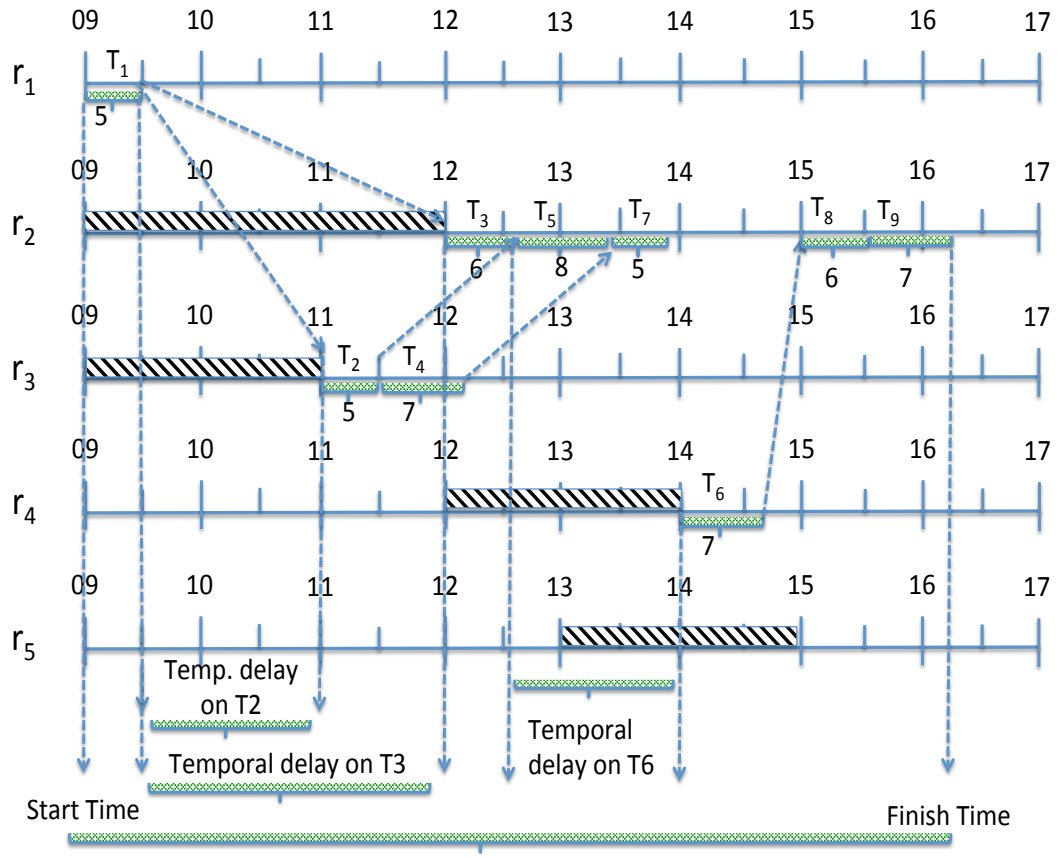


Figure 1.3. An exemplar scheduling solution of the workflow under the authorization constraints in the case study.

"It is common to find such authorization constraints and interaction between human and automated activities; our domains of interests include healthcare systems [Stuit2011], video management domain [VideoWorkflow] and

the manufacturing community [Hara2009] [Jin2003]. Human intervention and associated authorization clearly affects the processing of tasks and impacts on both application-oriented performance (e.g. mean response time of workflows) and system-oriented performance (e.g. utilization of the computing resource pool). Obtaining these performance data will be critical in capacity planning, designing authorization policies and developing workflow management strategies".

When the authorization constraints are present to restrict the workflow executions, it entails new research issues that have not been investigated yet in conventional workflow management.

First, it is important to know whether a feasible authorization solution can be found to enable the executions of all tasks in a workflow, i.e., check the feasibility of the deployed authorization constraints. The following example illustrates the situation. Assume a workflow consisting of 4 tasks as shown in Figure 1.4. Assume that the SoD (Separation of Duty) constraint is $r(T_2) \neq r(T_3)$, which means that the role assigned to task T_2 must be different from the role assigned to T_3, and that the BoD (Binding of Duty) constraints are $r(T_1)=r(T_2)$ and $r(T_1)=r(T_3)$. We cannot consider the roles only without mentioning the tasks to be involved and SoD and BoD constraints depend on the tasks to be assigned.

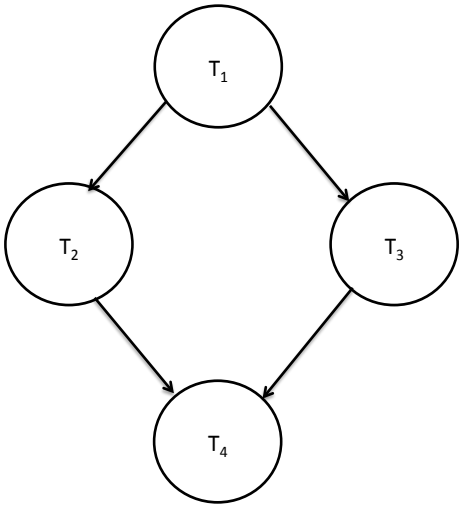


Figure 1.4. A case study for feasibility checking

In this situation the feasible solution is not possible as if task T_1 can run under role r_1 then according to the BoD constraints tasks T_2 and T_3 should run under r_1 as well. However, according to SoD constraint tasks T_2 and T_3 cannot run under the same role, which is the contradiction. Therefore, the feasible solution under these constraints is not possible.

Feasibility checking is important because if there are some tasks in a workflow that cannot be authorized subject to the deployed authorization constraints, there is no point to start the execution of the workflow at all. The request of the workflow execution should be rejected in the first place. Checking the feasibility of authorization constraints can help us design the authorization policy so that it will not cause the unnecessary rejections of the execution requests.

Second, the existence of the feasible authorization solutions for a workflow only means that the workflow can run to completion. Its execution performance may still be

negatively affected by the deployed authorization constraints. For example, roles may have temporal constraints, i.e., roles may only be activated during certain periods. When a task in a workflow is authorized to run under a particular role, but the role is not activated yet, the task may have to wait and consequently increase the execution time of the whole workflow. Therefore, it is useful to know when the performance of workflow executions will not be affected by the given authorization constraints. The first and second research issues are the focuses of Chapter 3 in this thesis.

Third, knowing the time durations when the authorization constraints will not have negative impact on performance is one way of shedding light into the impact of the authorization constraints. Another aspect of the impact is that when the authorization constraints have the performance impact, how to quantitatively determine the impact. For example, if the authorization constraints will cause the delay for the workflow execution, how can the delay be calculated? Chapter 4 in this thesis analyses the performance impact of the given authorization constraints. Further, based on the analyses, an optimal authorization method is proposed to select the authorization solution that can minimize the performance impact caused by the authorization constraints.

Fourth, a key reason why authorization constraints may have impact on performance is because the authorization control directs the tasks to some particular roles (i.e., the role assignment process). The authorization policy may specify the constraints on the roles, for example, role constraints or temporal constraints. Also, the quantity of resources allocated to support each role may

be different. For example, the number of the bank managers in a bank is normally different from the number of cashiers. This may cause the tasks assigned to different roles to have different response time, and consequently affect the execution performance of the workflow as a whole. Therefore, in order to examine the impact of the authorization constraints, it is desired to know the rate of the tasks arriving at each role, given the deployed authorization constraints.

Finally, after knowing the rate of the request arriving at each role, an important issue is to determine the amount of resources that need to be allocated to support the executions of the tasks assuming a particular role, so as to satisfy the desired Quality-of-Service. A workflow may consist of human tasks and computing tasks. Human resources and computing resources have different features and therefore require different considerations when determining the resource quantities.

Chapter 5 focuses on investigating the fourth and fifth research issues discussed above. The methods are proposed in Chapter 5 to calculate the rate of the tasks arriving at each role given a set of authorization constraints. Moreover, the resource allocation strategies are developed for both human resources and computing resource, aiming to optimize the performance under the current constraints.

To date, little attention has been paid to investigate the issues discussed above. This thesis aims to tackle these new research issues. The main contributions of this thesis are as follows.

- Proposing a method to check the feasibility of the authorization constraints, i.e., given a set of authorization constraints, checking whether there is a feasible authorization solution to enable the workflow execution (Chapter 3)

- Proposing a method to determine the time durations when the temporal constraints do not have negative impact on the performance of workflow executions (Chapter 3)

- Proposing the methods to conduct quantitative analyses about the delay caused by the authorization constraints for workflow executions (Chapter 4)

- Developing an optimal authorization method. The method is optimal in the sense that it can select the authorization solution that minimizes the delay caused by the authorization constraints

- Proposing a method to conduct theoretical analysis about the level of workloads assigned to individual roles, given the deployed authorization constraints (Chapter 5)

- Proposing the methods to determine the suitable amount of resources so that the performance of workflow executions is maximized, given the deployed authorization constraints and the resource budget (Chapter 5)

- Conducting the experimental studies to verify the effectiveness of the proposed methods in this thesis.

The rest of the thesis is organized as follows. Chapter 2 conducts the literature review relevant to the work in this thesis. Chapter 3 presents the methods to conduct the feasibility checking for the deployed authorization constraints and to calculate the time duration when the workflow executions will not be affected by the authorization constraints. Chapter 4 analyses the delay caused by the authorization constraints and further proposes an optimal authorization method. Chapter 5 present the methods to calculate the level of the workload directed to each role due to the deployed authorization constraints. Further, Chapter 5 presents the resource allocation strategies for both human and computing resources to optimize the performance.

2

Literature Review

This chapter will discuss the work related to workflow management, workflow scheduling and resource allocation, and security and authorization.

2.1 Workflow Management

2.1.1 Workflow Modeling

In general, "workflow is the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules [W3Workflow]. A workflow management system is a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engine, which is able to interpret the process definition, interact with workflow participants and, where required, invoke the use of IT tools and applications" [W3Workflow]. A workflow consists of a number of activities. An activity is different units of work to be done by a user or a program, requesting application programs [W3Workflow]. Activities are the smallest units of work. An activity is also called a

task. In a workflow, the execution of tasks has to follow the specified dependency, i.e., the acceptable relative orders of tasks executions. The typical elements of dependency are sequential execution and parallel execution of tasks.

However, the study in [Kim2003] has analyzed in more detail the dependency of workflows and its implications on distributed workflow systems. The authors have identified four types of workflow dependencies that could be useful in designing a distributed workflow system. These different types of dependencies are:

1) Data Dependency – used for modeling the effects of data flow on the behavioral aspects of different activities in a considered workflow. This modeling is further used for generating data-transition conditions associated with each activity along with the data-dependence information between the activities;

2) Activity (Control) Dependency – relates to the "control flows in a workflow procedure and is particularly used for modeling the effects of conditional and parallel branches" on the behavioral aspects of different activities in a considered workflow; For example, an activity in a workflow "can only start execution after another activity in the workflow has completed (the former activity is called the latter's child", and the latter is called the parent of the former).

3) Role Dependency – represents the role transition orders within a procedure. This could be modeled by

mapping the control flow part to the role assignment part in a workflow;

4) Actor Dependency – used to model the correspondent work-cases of the component jobs in the workflow; the dependency concepts are further embodied as objects in a distributed workflow architecture. While designing a workflow based on these four dependencies, the "actor-based workflow model and the role-based workflow model require actor-transition, activity-transition and data-transition conditions while activity-based workflow model and workcase-based workflow model only use the activity-transition conditions and data-transition conditions".

A task in a workflow is either processed by a computing resource or human being, which is called the actor of a task. Which actor will perform a task is typically decided by the workflow management system. However, when the security and authorization mechanisms are present, which actors, especially which human being actors, can perform a task may also be specified by the security and authorization mechanisms, which will be discussed in the later part (section 2.1.3) of this chapter.

Workflows are "often used to model enterprise or scientific applications [Deelman2009] [He2006a] [Hsu2011] [WebBusinessProcess]. Workflow management has been extensively studied and as a result is well documented in related literature [Atluri2000] [Chakraborty2007] [He2005] [Kim2003a]. Much of this research is aimed at automating the execution, and enhancing the performance, of workflows in parallel and distributed systems" [Chakraborty2007] [Manolachethesis].

2.1.2. Automation of Workflow Executions

A lot of research studies have been conducted to enable the automation of workflow executions, which lead to the proposal of "Web Services Business Process Execution Language (WS-BPEL). WS-BPEL is a language to specify the behavior of the business processes that are based on the Web Services". WS-BPEL's processes use the interface of the Web Service to import and export the functionality.

The business process "can be applied in one of the two ways: Abstract or Executable. The abstract process is partially specified". It is descriptive and cannot be executed. It is declared as an "abstract" process. The executable process is fully specified and can be executed as well. The abstract process can hide some operational details while the executable process shows all the details. Abstract processes are descriptive in role. WS-BPEL defines both abstract and executable processes. WS-BPEL defines a model and the grammar describing the behaviors for the business process, which bases on interactions between the process and its partners.

The WS-BPEL provides the language to specify the Executable and the Abstract processes. In this way it extends the Web Services interaction model and enables it to support business transactions. In order to support and facilitate the expansion of the automated process integration in both within the corporation and business-to-business spaces, WS-BPEL defines an interoperable integration model.

There is the work integrating multiple workflow management systems and automating the execution of the

workflows spanning multiple organizations. The study in [Chakraborty2007] has designed and implemented a so-called Heterogeneous Event Management Middleware (HEMM) to integrate the events across multiple siloed workflow management systems (WFMS). They have focused on the problem wherein a high-level change in workflow requires the running workflow (WF) instances in the WFM to handle external events necessitated by the high-level change. They argued that the existing solutions to handle such type of problems are cost-intensive mainly in terms of back-end downtime from an enterprise's viewpoint, as they require changing of either the workflow process definition or the workflow engine. The authors used their proposed HEMM to address this issue by: 1) the introduction of an overlay on the top of a WFMS; and 2) the abstraction of unforeseen event handlings from the workflow executions. Further, to adapt to new events whose process definitions could not be handled in a workflow execution, the event transformation is employed for mapping of events to event handlers associated with running workflow instances. The authors further demonstrated their prototype implementation by considering an example prevalent in the telecom industry.

2.1.3 Human Activities

Although a lot of research focuses on managing workflows in computing resources, "many workflow scenarios still involve human activities and will be comprised of a mixture of human tasks and computing tasks". Therefore, some studies investigate how to incorporate human activities into workflow execution. An exemplar product of these research efforts is

BPEL4People, “which has been proposed to overcome the lack of support for human activities in BPEL”.

BPEL4People introduces an extension to BPEL to support the human involvement in the business processes by adding the set of new elements in the standard BPEL [WS-BPEL]. The specification introduces human as a new basic activity, which supports human interaction in processes directly. In the language design, the extension in BPEL for people is defined in such a way that it makes a top layer on BPEL. This extension introduces the new elements and attributes to cover complex human interactions. The generic human roles are process initiator, process stakeholder and business administrators. The new basic activity elements use human tasks as an implementation and it allows the specification of the tasks for the processes.

The study in [Zhao2008] has proposed a formal model that adds support for human task support to Business Process Execution Language (BPEL) and have named it as BPEL4People. BPEL, which has been described as a standard for specifying and executing workflow of Web service composition invocation, has an inherent limitation of not providing any support for human workflow. The authors have used CSP process algebra to present a formal model of human workflow.

The study in [Stuit2011] has evaluated a “novel interaction-centric process modelling method using a case study” of a healthcare human collaboration processes (HCP) at a Dutch academic hospital. The HCPs in the healthcare domain involves interactions taking place “between healthcare workers representing different (para)

medical disciplines and departments. The existing workflow-based process modelling tools for healthcare process management" focus on defining task sequences rather than modelling the graphical description of human interactions in a HCP. The authors have considered a care pathway HCP performed by the head and neck oncology team for the evaluation of their interaction-centric process modelling method. The evaluation of the method has highlighted three significant results: 1) collection and formalization of the tacit domain knowledge of the interviewed healthcare workers in individual interaction diagrams; 2) support provided by the method for automatic integration of individual interaction diagrams into a global interaction diagram capable of reflecting the consolidated domain knowledge; and 3) utilizing a graphical modeling language to describe interactions between methods, their composition and routing relations, and their roles using an effective tree-based description; The proposed method showed good support for improving the healthcare collaborations.

2.2 Workflow Scheduling

Automating the execution of the tasks in a workflow is one of the main focuses in a workflow management system. Anther focus is to enhance the performance of workflow executions. In this aspect, the scheduling strategies employed by the workflow management system play a critical role.

Workflow can be modeled as a Directed Acyclic Graph (DAG). Therefore, the DAG scheduling strategies can be applied to workflow scheduling. There are two basic

stages in DAG scheduling. In the first stage, the scheduling order of the tasks in a DAG is determined, while the second stage decides which resource should be used to run a task. Most DAG scheduling algorithms is based on list scheduling, in which all tasks are prioritized and the scheduling order of the tasks follows their priorities. Two basic techniques to determine the scheduling order of the tasks are to calculate the t-level (top level) and b-level (bottom level) of the tasks in a DAG and use them to prioritize the scheduling order of the tasks. The t-level of task t_i is the length of the longest path from the first task (also called the entry task) in the DAG to t_i. The b-level of a task is the length of the longest path from the last task (also called the exit task) to the task. The t-level of a task correlates with the earliest start time of the task, while the b-level of a task correlates with the latest start time of a task if a deadline is set for the completion time of the whole DAG. Other list scheduling algorithms just use different approaches to determine the scheduling order of the DAG.

After the tasks' scheduling order is determined, the scheduling algorithm further performs resource selection for all tasks, i.e., decides among all resources which resource should be used to run each of the tasks. A basic method is to select the resource, which can offer the least finish time.

Different DAG scheduling algorithms essentially use different approaches to determining the tasks' scheduling order and selecting the resources to run the tasks. Reference [Kwok1999] conducts a survey of 27 DAG scheduling algorithms, which mainly aim to minimize the

scheduling length, i.e., the duration between the time when the first task in the DAG starts execution and the time when the last task completes execution.

The study in [van der Aalst2002] has characterized the scheduling principle for utilizing available computational resources. They have suggested that a scheduling principle should match each atomic task for proper resource management, which could finally lead to matching an atomic task to a corresponding suitable resource. Two decisions should be supported by the scheduling principle: firstly tasks should have some defined order for execution; and secondly, task assignment to the available resources should represent the most suitable match from the available set of resources. In nutshell, scheduling a workflow typically consists of two stages: 1) determining the execution order of the tasks in a workflow and 2) determining the resources that should be used to run each task. Numerous scheduling strategies have been proposed in literature.

The study in [Ranaweera2000] has proposed a novel scheduling algorithm, called TDS, to optimally schedule the tasks represented using a directed acyclic graph (DAG) onto an available set of heterogeneous processors with varying computing power. The TDS aims at minimizing the schedule length, also known as makespan, and scheduling time itself under task duplication based scheduling scenario. The algorithm further aims at minimizing the overall processing complexity to ensure reasonable runtime. The algorithm uses Earliest Start Time of a node (EST), Earliest Completion Time of a node (ECT), Latest Allowable Start Time of a node (LAST), Latest Allowable Completion Time of a node (LACT),

favorite predecessor task of a given task (fpred) and favorite processor of a given task (fproc). TDS runs in four steps, namely: 1) top-down traversal of the DAG to compute EST, ECT, fpred, fproc1 to fprocn and level of each task; 2) bottom-up traversal of DAG for calculating LACT and LAST for each node; 3) "an initial set of task clusters is generated using a reasonably small number of processors"; and 4) involves duplication of tasks and message forwarding, which represents forwarding of the results from that processor that has minimum completion time of a task amongst the available processors. The authors used three inputs: 1) Cholesky decomposition DAG; 2) Diamond DAG; and 3) the DAG for Gaussian elimination code; for their proposed algorithm, TDS, and compared the results with a similar scheduling algorithm, called Best Imaginary level scheduling (BIL). The comparison results showed better communication-to-computation cost ratios (CCR) of 0.2 as compared to 1 obtained using BIL and gave far more superior results than BIL for the scheduling time.

The study in [N'takpe'2007] has proposed a novel scheduling approach to execute mixed parallel applications on heterogeneous platforms. The static scheduling algorithms for online workflow applications are not feasible due to occurrence of multiple workflows submitted by different users and arriving at different times. In such a scenario, the task scheduling is done by maintaining waiting queues with an association of priorities with each of the workflow present in the queue. Such a scheduling becomes more difficult when a single processor is available for running each task. This makes dealing with workflows, which are composed of data-parallel tasks, as infeasible. The approach proposed

by authors has been found to be suitable for a single workflow involving mixed parallel applications, which combine task parallelism and data parallelism, on heterogeneous platforms.

The study in [N'takpe'2008] also extended their work in [N'takpe'2007] to develop a scheduling mechanism for dealing with concurrent mixed parallel applications. In general, there are two steps involved in concurrent scheduling for mixed parallel applications namely: 1) constrained resource allocation – to determine an optimal allocation for each task while determining the number of processors available; and 2) concurrent mapping – involving prioritizing of tasks of workflows for their execution. The authors have restricted the applicability of their scheduling mechanism to concurrent workflows submitted at the same time. Their scheduling mechanism does not deal with the online workflows submitted at different times.

The study in [Tarumi1997] has addressed the resource conflict problem by considering a resolution strategy at the runtime rather than at the build time. They associated agents with the resources and allowed mutual communication between agents for reserving office resources and checking their availability.

The study in [Senkul2002] proposed an interesting approach that considers resource allocation constraints while dealing with the scheduling problem. The approach deploys constraint logic programming (CLP) and integrates it with Concurrent Transaction Logic (CTR) to formulate a new logical representation.

The study in [Doulamis2011] has examined the resource allocation problem together with task scheduling. In a Workflow Management System (WfMS), resource allocation and task scheduling are two important issues that impose mutual constraints. Thus, optimizing resource allocation is subject to task scheduling and vice versa. They have highlighted the characteristics of an ideal algorithm for solving these fundamental issues in WfMS, which mainly include: “performance metrics of the infrastructure e.g., the number of resources and their utilization; and quality criteria” such as under temporal restrictions percentage of tasks undergoing violations. The authors have proposed an algorithm called Resource Conflicts Joint Optimization (Re.Co.Jo.Op), which aims at jointly optimizing resource allocation and task scheduling by minimizing resource conflicts subject to temporal constraints while simultaneously optimizing throughput or utilization subject to resource constraints. They used matrix for representing the two factors and applied the concepts of the generalized eigen value analysis for finding the optimal solution of the problem. They further proposed an agent-based architecture for integrating their proposed algorithm into a functional WfMS. The experimental results have established the superiority of their proposed strategy on the conventional approaches.

All of the above strategies of scheduling workflows do not consider the security and authorization issues. However, the security and authorization policies are deployed in many workflow applications in real worlds.

2.3. Security and Authorization for Workflow Executions

The workflow security and authorization constraints have been researched a lot in the literature and well documented in [Atluri2000] [Crampton2012] [He2011] [ManolacheThesis] [Wang2010] [Lu2009]. But different works have different focuses.

2.3.1. Enforcement of Security and Authorization Policies

Some studies focus on developing the methods to guarantee that in the processing of the workflows in the system, the authorization policies can be enforced properly.

XACML authorization engine is a popular product to achieve this. XACML stands for eXtensible Access Control Markup Language. It defines a declarative language to specify the access control policy and a processing model describing how to evaluate the authorization requests according to the rules defined in policies. XACML can be used to specify multiple authorization control schemes, such as Attribute Based Access Control system (ABAC) and Role-Based Access Control (RBAC). In ABAC, the attributes are associated with a user, an action or a resource and attributes are used to by the authorization control scheme to decide whether a given user may access a given resource in a particular way. There are multiple components in the XACML authorization engine, such as Policy Administration Point (PAP), Policy Decision Point (PDP), Policy Enforcement Point (PEP). PAP is the

component to manage authorization constraints. PDP is a component to evaluate and issue authorization decisions. PEP intercepts a user's access request to a resource and make sure that the request can only use the resources in the way consistent with the decision made by PDP.

The study in [Dagdee2011] has conducted a study to enhance the XACML standard so that it can support credential based hybrid access control. The standard XACML only supports attribute based access control mechanism. This work proposes credential based hybrid access in which any unknown user can have easy and immediate access to open access environment. The main extensions in the XACML policy specification are 1) addition of new element <Credential> for the representation of credentials 2) addition of the new element <CredentialRequirements> for logical combination of credentials 3) inclusion of CredentialId attribute in the <CredentialAttributeDesignator> to support conditions involving credential attributes 4) addition of <Credentials> in the XACML request context to get the credentials from the user. The extension in the XACML architecture is proposed in the form of credential manager in the context handler, which extracts the credential information submitted by the user. The access policy contains various conditions over credentials and the attributes associated with the credentials.

The study in [Liu2008] has also improved the XACML policy request processing engine. The growth in the web applications has improved the complexity and size of the XACML policies, which is the main cause of the slow processing of the requests. This work focuses on the performance of the request processing, which is a main

issue. Liu proposes a new XEngine, an efficient request processing schema. The XEngine follows the following steps: 1) it converts a textual policy into a numerical policy 2) it converts a numerical policy with complex structures to a numerical policy with normalized structures 3) it converts the numerical normalized structures to a tree structure for improved performance. To verify the effectiveness of the technique experiments have been conducted on both real life and synthetic XACML policies. The results verify the claim of improving the performance by orders of magnitude. The performance improves linearly with the number of policies. For small number of policies the XEnging is faster in one to two order of magnitude while for larger policies the XEngine is faster by three to four orders of magnitude than the Sun PDP.

The study in [Wang2010] has proposed a role-and-relation-based access control (R2BAC) model for workflow authorization systems wherein a user's role membership and his relationships with other users help in determining if the user, under the given conditions, could be allowed to perform a particular step of a considered workflow. The authors explored the computational complexity aspect of the workflow satisfiability problem to investigate if a set of users could complete a workflow. They further used parameterized complexity theory tools for understanding the problem complexities. They reduced the workflow satisfiability problem to SAT and applied SAT solvers for analyzing and solving this reduced problem. The experimental results have showed efficiency of the algorithm in solving instances of reasonable size. They further study the resiliency problem in workflow

authorization systems to investigate if a workflow could be completed when a number of users are absent. The authors further defined three resiliency levels in workflow systems and studied the associated computational problems.

The study in [Zou2009] combined the advantages of role-based access control (RBAC) and attribute-based access control (ABAC) mechanisms to propose a new access control model (CRBAC), which integrates all kinds of constraints into the RBAC model. The authors have analyzed the generic properties of the attribute constraints and have presented them into two constraint templates: a) authorization mapping constraint; and b) behavior constraint; for automating the user-role and role-permission mapping as well as restricting the behaviors of the authorization entities respectively. The authors have further introduced a state mechanism for building up the constraints in a group of statuses of the entities as well as reflecting the authorization control outcomes. They, based on the proposed constraint templates and the introduced state mechanism, have developed an execution model. Moreover, use cases have been proposed to describe the authorization process taking place in the proposed access control model (CRBAC). The authors have further analyzed the correctness, complexity, flexibility and compatibility of CRBAC to compare the multi-grained constraints of CRBAC with other models.

2.3.2. Feasibility Checking of Authorization Constraints

Some studies focus on checking whether the deployed authorization constraints can be satisfied

[Crampton2005][Atluri1999][Wang2010][Lu2009]. The work in [Crampton2005] conducted the theoretical analysis about the satisfiability of the authorization constraints for a workflow. The work conducted the theoretical analysis and found out that in order to check whether there is a valid the workflow authorization, it only needs to consider a single linear extension (i.e., a linear ordering) of the tasks in the workflow. There exists a valid workflow authorization if and only if there is also a valid authorization solution for the linear extension. However, the work cannot obtain all feasible authorization solutions. The modeling approach presented in Chapter 3 is able to obtain all feasible authorization solutions. Based on this, our work further develops the authorization methods, aiming to reduce the negative impact imposed by the authorization constraints.

Petri-net is a popular methodology in the literature to achieve this, partly because petri-nets is capable of capturing and modeling the dynamic behaviors in a system, and partly because there are well established techniques to conduct the theoretical and simulation analysis for the constructed Petri-nets models.

The work in [Atluri1999] conducts the safety analysis, i.e., analyzes whether a specified authorization state (i.e., the task-role assignments) can be reached under a set of authorization constraints, given an initial authorization state. The work uses the Color Timed Petri Nets (CTPN) to model roles, SoD and temporal constraints, and then converts the constructed CTPN model to an ordinary Petri-Net (PN) model so that the established PN analysis techniques can be applied to generate the results. The work can generate all possible authorization

solutions. However, this modelling approach is heavy since it needs to construct the CPTN model, covert the CPTN model to ordinary PN models, and analyze the PN models. In this thesis, we model the feasiblity checking problem concisely as a Constraint Satisfaction Problem (CSP).

The study in [Li2004] has proposed an extension to Petri Nets and have named that as the Time Constraint Workflow Nets. Their extension helps in identification and removal of conflicts associated with resources occurring in workflow specifications. They have added a notion of time to the Petri Nets for allowing the temporal validation of the conflicts associated with the resources. The method exhaustively searches for all tasks and has been found to be non-scalable.

The study in [Zhong2005] has extended the approach of [Li2004] and have proposed a new mechanism for identification of conflicts associated with resources under concurrent workflow settings and have also found their approach having scalability problems.

The study in [van Hee2005] has introduced a variation of Petri Nets, called the Resource-Constrained Workflow Nets, for dealing with the problem of resource conflicts. A method has been presented for assessing the minimum amount of resources to start up the process, which could guarantee that the started processes will be successfully terminated within the give constraints and no conflict will occur for the resources. They argued that their proposed method ensures calculation of sufficient amount of resources irrespective of the scheduling policy used afterwards and thus, guarantees the completion of tasks

on the correspondingly mapped computational resources. The calculation of sufficient amount of resources helps eliminate the resource conflicts on one hand but also results in wasteful architecture in the design process of the information system.

The study in [Lu2009] has used Colored Petri nets (CPN) for modelling and analyzing workflow with Separation of Duty (SoD) constraints. SoD represents the security principle wherein frauds and errors are prevented in collaborative environments. As the organizations achieve their business goals by interacting and collaborating between users through workflow, thus, during workflow design with SoD constraints, the correctness and consistency of workflow becomes crucial to verify and ensure. Keeping this problem in mind, the authors have combined control flow, authorization rules and SoD constraints in a single workflow and have used an integrated CPN model for representing this combination of constraints to a workflow. They used reachability tree analysis for deriving the execution paths of the integrated CPN model. The analysis of the derived execution paths resulted in identification of some latent deadlocks, which in turn were the results of the inconsistency between authorization rules and SoD constraints.

2.3.3. Analysis of Performance Impact of Security and Authorization Policies

Xie proposed the security aware model for workflows and focussed on three security aspects, which are: i) confidentiality, ii) integrity and iii) authentication.

He develop a security overhead model to measure the security overhead [xie2006]. Xie also developed resource allocation strategies TAPADS and SHARP by taking into account the security and precedence constraints for homogeneous clusters and heterogeneous cluster [xie2008]. Qiu used Security-Aware Task (SEAT) graph model to denote the constraints and relationship of tasks and on the basis of SEAT graph, he proposed an algorithm ILP-SOP and for special structures he proposed DPSOP-path/tree algorithm for security generation for tasks [Qiu2013]. However, these studies do not consider the impact of the authorization policies.

There are also the studies using Petri-nets to model and analyze the impact of authorization constraints.

The work in [He2009] “applied Generalized Stochastic Petri-Net (GSPN) theory to model workflow executions under Role-based Authorization Control, and then used standard Petri-net analysis techniques to theoretically calculate performance metrics from the constructed models. Although GSPNs are adequate for the scenarios investigated in [He2009], the work did not model the workflows consisting of both human tasks and computing tasks. Also, since GSPNs cannot express the temporal attributes associated with tokens, they cannot analyze the authorization overhead caused by temporal constraints. Moreover, the work in [He2009] did not investigate authorization methods to improve performance, given the specified authorization constraints”.

The study in [He2011] has presented a novel modelling scheme for workflow execution in cluster-based resource pools. The modelling mechanism works under a Role-based

Access Control (RBAC) scheme, which assigns certain roles to users and each role has an associated set of permissions. The authors have modeled various authorization constraint types including: 1) role constraints; 2) temporal constraints; 3) cardinality constraints; 4) Binding of Duty constraints; and 5) Separation of Duty constraints; using Coloured Timed Petri-Nets. The modelling scheme also captures the interaction between workflow authorization and workflow execution. The authors highlight the in-built automation support for their modelling scheme for workflow execution.

Generally, the Petri-net modeling approach is heavy and susceptible to state explosion problems.

"The Multi-layered State Machine (MLSM) is another method used in the literature [Gaaloul2008] [Hung2003] to model workflow authorization. However, the MLSM method is mainly used to guarantee that the authorization constraints are satisfied in the workflow environment, and the method itself cannot simulate and obtain the quantitative performance of the workflow execution. In order to obtain performance, the MLSM structure needs to be converted to Petri-nets before a performance analysis can be conducted [Gaaloul2008] [Hung2003]. Further, the work in [Gaaloul2008] [Hung2003] does not analyze the impact of the authorization constraints and does not investigate the authorization methods to improve performance".

3

Analyzing the Impact of Authorization Constraints

This chapter analyzes the impact of the deployed authorization constraints. More specifically, this chapter 1) checks whether all tasks in a workflow can be authorized so that the authorization constraints deployed in the system can be satisfied, and 2) determines such time durations in which the authorization constraints will not have negative impact on the performance of workflow executions. The notations used in this chapter are summarized in Table 3.1.

The rest of this chapter is organized as follows. Section 3.1 presents the methods to check the feasibility of role, SoD and BoD constraints deployed in the system. Section 3.2 presents the method to determine the time durations in which the workflow executions will not be delayed by the authorization constraints in the system. Section 3.3 presents a case study to illustrate the workings of the methods proposed in this chapter.

Table 3.1 Notations used in this thesis

Notations	Explanations
r_i	Role i
$C^t(\mathrm{r}_i)$	The temporal constraint of r_i
$C^r(\mathrm{s}_i)$	The role constraint of service s_i
P_i	Period i
E_i	The end time of the period P_i.
D_{s_i}	The domain of service s_i.
S_i	The start time of the period P_i
A_k	The k-th feasible role assignment solution
$r(t_i, A_k)$	The role assigned to task t_i in A_k
$EA_k(\mathrm{t}_i)$	Effective temporal constraint of $r(\mathrm{t}_i, \mathrm{A}_k)$ in A_k
ld_{ij}	Lower domain of role *i* for task *j*.
ud_{ij}	Upper domain of role *i* for task *j*.
e_i	The execution time of the tasks assigned to r_i.
w_i	The waiting time of the tasks assigned to role r_i
$np(r_i)$	The number of resources used to serve the tasks running under r_i
rp_i	The mean response time of the tasks running under role r_i
$C^c(r_i)$	Cardinality constraint of role r_i
$C^t(r_i)$	The temporal constraint of r_i
$C^r(\mathrm{s}_i)$	The role constraint of service s_i
$\lambda^x(r_i)$	The arrival rate of the tasks that are assigned to r_i when x constraints are considered.

$C^s(r_i)$	The set of services that role r_i can invoke
$rp(r_i, s_j)$	The mean response time of the tasks that assume r_i to invoke s_j
$\lambda^r(r_j)$	The arrival rate of all service requests allocated to r_j
h_i	The number of the human resources allocated for role r_i
D_i	The time duration when r_i is activated in the period P_i
α_{ij}	The proportion of processing capability allocated to run the requests that assume role r_j
$O_{(L \times M)}$	L services cross M rolls matrix.

3.1. Checking Feasibility of Role, SoD And BoD Constraints

$S = \{s_1,...,s_L\}$ denotes the set of services running on the resource pool.

$F = (T, E)$ denotes a workflow, in which $T = \{t_1,...,t_N\}$ is a set of tasks in the workflow and $E = \{(t_i, t_j) \mid t_i, t_j \in T\}$ is a set of directed edges linking task t_i to t_j. A task invokes one of the services in S.

$R = \{r_1,...,r_M\}$ denotes the set of roles defined in the authorisation control system. The role constraint specifies the set of roles that are permitted to run a particular service. $C^r(s_i)$ denotes the role constraint applied to service s_i. $r(s_i)$ denotes the role that is assigned to run s_i. The Separation of Duty (SoD) and the

Binding of Duty (BoD) constraint between s_i and s_j are represented as $r(s_i) \neq r(s_j)$ and $r(s_i) = r(s_j)$, respectively.

The problem of checking feasibility of role, SoD and BoD constraints is formulated as a Constraint Satisfaction Problem (CSP) [Brailsford1999] in this thesis.

A CSP consists of a triple $< V, D, C >$, where

$V = \{v_1, v_2, ..., v_n\}$ is a set of variables,

$D = \{D_{v_1}, D_{v_2}, ..., D_{v_n}\}$, and D_{v_i} is the domain of the value of v_i, and C is a set of constraints restricting the values that the variables can take.

The Feasibility Checking Problem (FCP) in this chapter is modelled as a CSP in the following way. The services in FCP are regarded as the variables in CSP. The role constraint of a service is regarded as the domain of the value of the service. The BoD and SoD constraints are regarded as the constraints restricting the values that the service variables can take. An example is given below to illustrate the modelling.

Assume the tasks in a workflow invoke 7 services, $s_1 - s_7$, and there are 6 roles, $r_1 - r_6$ in the authorization system. The role constraints are:

$s_1 = \{r_1\}$,

$s_2 = \{r_2, r_3, r_4\}$,

$s_3 = \{r_2, r_3, r_5\}$,

$s_4 = \{r_2, r_3, r_5\}$,

$s_5 = \{r_2, r_3, r_5\}$,

$s_6 = \{r_2, r_4\}$,

$s_7 = \{r_4, r_6\}$.

The SoD constraints are:

$r(t_2) \neq r(t_5)$,

$r(t_2) \neq r(t_7)$,

$r(t_6) \neq r(t_7)$.

The BoD constraints are:

$r(t_2) = r(t_4)$,

$r(t_3) = r(t_5)$.

Then the FCP can be formulated as CSP as follows.

$CSP = < V, D, C >$,

$V = \{s_1, s_2, s_3, s_4, s_5, s_6, s_7\}$,

$D = \{D_{s_1}, D_{s_2}, ..., D_{s_7}\}$, where

$D_{s_1} = \{r_1\}$,

$D_{s_2} = \{r_2, r_3, r_4\}$,

$D_{s_3} = \{r_2, r_3, r_5\}$,

$D_{s_4} = \{r_2, r_3, r_5\}$,

$D_{s_5} = \{r_2, r_3, r_5\}$,

$D_{s_6} = \{r_2, r_4\}$,

$D_{s_7} = \{r_4, r_6\}$.

$C = \{C_1, C_2, C_3, C_4, C_5\}$, where

$C_1 : r(t2) = r(t4)$,

$C_2 : r(t2) \neq r(t5)$,

$C_3 : r(t2) \neq r(t7)$,

$C_4 : r(t6) \neq r(t7)$,

$C_5 : r(t3) = r(t5)$.

There are the existing solvers to solve the CSP problem [Brailsford1999]. Some solvers only check whether a solution can be found to satisfy the problem, and if so, return one solution. Some solvers can return all solutions to the problem, i.e., all feasible role assignments to the tasks so that the specified SoD, BoD and role constraints are satisfied.

3.2. Analyzing the Coverage of Authorization Constraints for Workflow Executions

Roles have temporal constraints, i.e., when the roles are activated and can be assigned to tasks. It is useful to check the coverage of roles' temporal constraints in a given security setting for workflow executions. If the temporal constraints of the relevant roles cover the execution period of a workflow, then the temporal constraints will not delay the task executions in the workflow, and therefore will not have negative impact on the performance of the workflow.

According to the discussions in section 3.1, we can use the CSP solver to obtain all feasible role assignment solutions for the tasks in a workflow. A denotes the set of all feasible role assignments for the workflow, and

$$A_k = \{(t_i, r_j) \mid t_i \in T\}$$

denotes the k-th feasible role assignment, in which t_i is a task in the workflow and r_j is the role assigned to t_i.

In most cases, a role is activated periodically. For example, the role of bank manager is only activated from 9am to 12pm, and from 2pm to 4pm in a day. Therefore, the temporal constraint of role r_i, denoted as $\mathrm{C}^t(r_i)$, can be expressed as below,

$$\mathrm{C}^t(r_i) = (P_i,\ D_i,\ S_i,\ E_i) \qquad (3.1)$$

Where P_i is the period, $D_i = \{[ld_{ij}, ud_{ij}] \mid i \in \mathbb{N}\}$ is the time duration when r_i is activated in the period P_i, and S_i and E_i are the start and end time points when this period pattern begins and ends. E_i can be ∞, meaning the periodic pattern continues indefinitely.

Assume that the execution times of the tasks in a workflow and the scheduling algorithm used to schedule the tasks is known. Therefore, if we know the arrival time of the entry task in the DAG, we can calculate the start time of every task in the DAG. st_i denotes the start time of task t_i, $r(t_i, A_k)$ denotes the role assigned to task t_i in A_k. Assume t_0 is the entry task. Assume $r(t_0, A_k) = r_p$. $\mathrm{C}^t(r_p)$ represents the temporal constraint of role r_p. $\mathrm{C}^t(r_p) = (P_p,\ D_p,\ S_p,\ E_p)$ as shown in Equation (3.1). Assume $r(t_i, A_k) = r_q$ ($i \neq 0$). $\mathrm{T}(r_q)$ denotes the time durations

when r_q has to be temporarily available to run t_i.

Given $\mathrm{C}^t(r_p)$, $\mathrm{T}(r_q)$ can be determined by Eq. 3.2, where D_j is determined in Eq. 3.3.

$$\mathrm{T}(r_q) = (P_p, D_j, S_p + (st_i - st_0), E_0 + (st_i - st_0)) \quad (3.2)$$

$$D_j = \{[ld_{0k} + (st_i - st_0), ud_{0k} + (st_i - st_0)] \mid k \in \mathbb{N}\} \quad (3.3)$$

However, r_q is subject to the temporal constraint, $\mathrm{C}^t(r_q)$. Therefore, the intersection of $\mathrm{T}(r_q)$ and $\mathrm{C}^t(r_q)$, denoted by $I(t_i, A_k)$, is the time durations when task t_i can start execution immediately without being delayed by the temporal constraints, given the role assignment A_k.

$I(t_i, A_k)$ can be determined using Equation below;

$I(t_i, A_k) = (P^I_{ki}, D^I_{ki}, S^I_{ki}, E^I_{ki})$, where

P^I_{ki} is the least common multiple of P_p and P_q;

$S^I_{ki} = max(S_p, S_q)$; $E^I_{ki} = min(E_p, E_q)$;

Let $D^I_{ki} = \{[ld^I_{kij}, ud^I_{kij}] \mid j \in \mathbb{N}\}$.

As shown above, we calculate $\mathrm{T}(r(t_i, A_k))$ from $\mathrm{C}^t(r(t_0, A_k))$, and then calculate $I(t_i, A_k)$ from $\mathrm{T}(r(t_i, A_k))$ and $\mathrm{C}^t(r(t_i, A_k))$. $I(t_i, A_k)$ is a subset of $\mathrm{T}(r(t_i, A_k))$. This means that only when t_0 arrives in a subset of the time durations in $\mathrm{C}^t(r(t_0, A_k))$, t_i's start time falls into $I(t_i, A_k)$. In this thesis, such a subset of time durations in $\mathrm{C}^t(r(t_0, A_k))$ is called $r(t_0, A_k)$'s

effective time durations for t_i in the role assignment A_k, which is denoted by $ET_k(t_0,t_i)$. $ET_k(t_0,t_i)$ can be determined by Eq. (3.4).

$$ET_k(t_0,t_i) = (P^I_{ki},\{[ld^I_{kij}-(st_i-st_0),ud^I_{kij}-(st_i-st_0)] \mid j\in\mathbb{N}\},S^I_{ki},E^I_{ki}) \quad (3.4)$$

We can calculate $ET_k(t_0,t_i)$ for every task t_i in the workflow. $\bigcap_{t_i\in T} ET_k(t_0,t_i)$ is the time durations in $\mathrm{C}^t(r(t_0,A_k))$ that can ensure the start time of every task $t_i\in T$ ($i\neq 0$) in the DAG falls into the times durations specified in $\mathrm{C}^t(r(t_i,A_k))$. Only when t_0 arrives in these time durations, can every task in the DAG starts execution without being delayed by the temporal constraints of the role assigned to run the task in A_k. $\bigcap_{t_i\in T} ET_k(t_0,t_i)$ is called t_0's effective arrival time when the role assignment is A_k, denoted by $EA_k(t_0)$. Note that according to the calculation method of $EA_k(t_0)$, $EA_k(t_0)$ is a subset of $\mathrm{C}^t(r(t_0,A_k))$. Therefore, we also call $EA_k(t_0)$ the effective temporal constraint of $r(t_0,A_k)$ for the DAG in the role assignment A_k. Assume $EA_k(t_0)=\{[ld_{0j},ud_{0j}] \mid j\in N\}$. We can further determine the set of time durations for the start time of t_i, denoted by $EA_k(t_i)$, as in Eq. (3.5). Note that $EA_k(t_i)$ is a subset of $\mathrm{C}^t(r(t_i,A_k))$. Therefore, we call $EA_k(t_i)$ the effective temporal constraint of $r(t_i,A_k)$.

$$EA_k(t_i)=\{[ld_{0j}+(st_i-st_0),ud_{0j}+(st_i-st_0)] \mid j\in\mathbb{N}\} \quad (3.5)$$

We can calculate $EA_k(t_0)$ for every feasible role assignment. Assume $[S,E]$ is the time duration for which we want to check the coverage of the temporal constraints. If $\bigcup_{all\,k} EA_k(t_0)$ cover the entire range of $[S,E]$, then no matter when the workflow instance is initiated, we can always find a role assignment so that all tasks in the workflow can start execution without delay due to the roles' temporal constraints. Otherwise, $[S,E]-\bigcup_{all\,k} EA_k(t_0)$ is the time gap during which the execution of at least one task in DAG will be delayed by the current setting of the temporal constraints.

3.3. A Case Study

We now present a case study to illustrate the impact of the authorization constraints on the workflow performance. In the case study, a workflow consists of nine tasks, as shown in the figure 3.1 below. The workflow is run under the authorization constraints. Tasks may have to wait for the results from parent tasks and the number of tasks in a workflow are not fixed. Assume the authorization constraints are specified below.

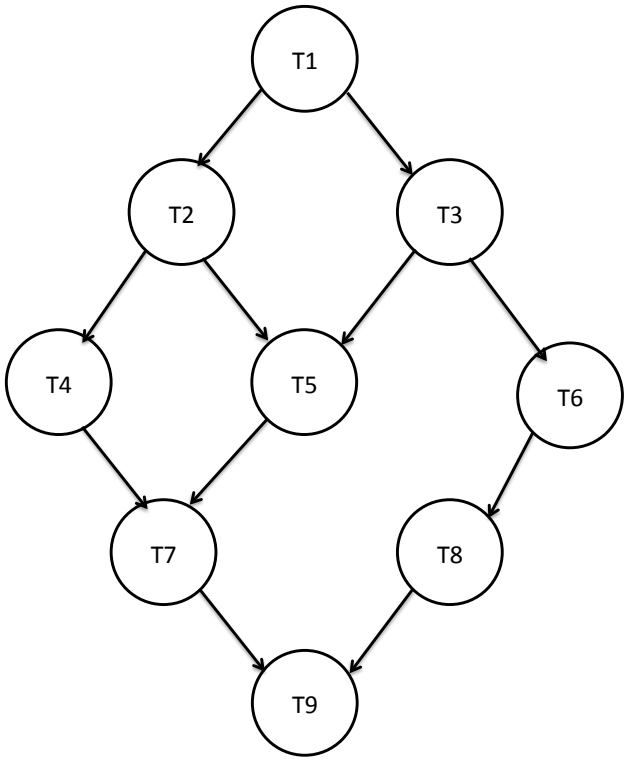


Figure 3.1. The workflow in the case study

1. There are five roles, r_1, r_2, …, r_5;

2. The temporal constraint of role r_i (i.e., $C^t(r_i)$) is as follows. The temporal constraints are illustrated in Figure 3.2, where the shaded area is the time durations when the role is not activated for service.

$C^t(r_1)$={[0900, 1700]},
$C^t(r_2)$={[1200, 1700]},
$C^t(r_3)$= {[1100, 1700]},
$C^t(r_4)$= {[0900, 1200], [1400, 1700]},
$C^t(r_5)$= {[0900, 1300], [1500, 1700]},

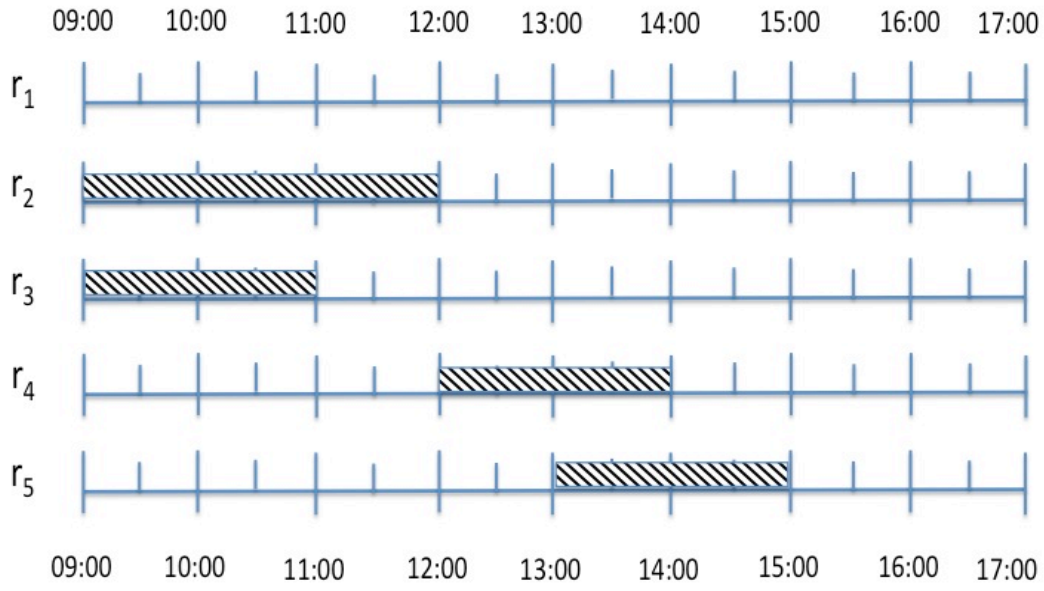


Fig. 3.2: The temporal constraints in the case study, the shaded areas in the timelines are the time durations when the roles are not activated

3. The role constraint of a task, denoted by $C^r(t_i)$, in the workflow is as follows.

$C^r(s_1)=\{r_1\}$

$C^r(s_2)=\{r_3, r_4\}$

$C^r(s_3)=\{r_1, r_2\}$

$C^r(s_4)=\{r_2, r_3\}$

$C^r(s_5)=\{r_2, r_3\}$

$C^r(s_6)=\{r_2, r_3\}$

$C^r(s_7)=\{r_3, r_4\}$

4. The Separation of Duty (SoD) constraints are as follows, where $r(T_i)$ is the role assigned to task T_i.

$r(T_2) \neq r(T_5)$

$r(T_2) \neq r(T_7)$

$r(T_6) \neq r(T_7)$

5. The Binding of Duty (BoD) constraints are as follows.

$r(T_2) = r(T_4)$

$r(T_3) = r(T_5)$

6. The tasks, t_i, in the workflow invoke the services in the following way.

t_0 -> s_1,

t_1 -> s_2,

t_2 -> s_3,

t_3 -> s_1,

t_4 -> s_6,

t_5 -> s_7,

t_6 -> s_4,

t_7 -> s_6,

t_8 -> s_5.

7. The execution times of the tasks t_0-t_8 are given in table 1.1. The execution times of the tasks have impact on the effective temporal constraints.

Applying the method proposed in Section 3.1, we can obtain that there are total 8 different authorization solutions for the workflow, [Brailsford1999] solver can be used to obtain CSP (Constraint Satisfaction Problem) solutions. Now we apply the method proposed in this section to calculate the coverage of the temporal constraints set in the case study.

The first authorization solution (A_1) is as follows.

Solution A_1:

$r(t_0,A_1) = r_1$,

$r(t_1,A_1) = r_3$,

$r(t_2,A_1) = r_1$,

$r(t_3,A_1) = r_1$,
$r(t_4,A_1) = r_2$,
$r(t_5,A_1) = r_3$,
$r(t_6,A_1) = r_2$,
$r(t_7,A_1) = r_2$,
$r(t_8,A_1) = r_2$.

Since 1) t_0 is authorized to r_1 in A_1, 2) r_1 is activated during [09:00, 17:00] in the period of [0900, 1700] and 3) t_0's execution time is 30 minutes, the possible start time of t_1, which is also the duration when the role assigned to t_1 in A_1 (i.e., r_3) has to be activated so that t_1 can start execution without being delayed by the temporal constraints, can be calculated by Equations given below:

$\mathcal{T}(r_3)$ = {[0930,1730]}
However, r_3's temporal constraint is
$C^t(r_3)$ = {[11:00,17:00]}
Consequently,
$I(t_1,A_1) = C^t(r_3) \cap \mathcal{T}(r_3)$
= {[11:00,17:00]}
Then,
$ET_1(t_0,t_1)$ = {[11:00 - 30min, 17:00 - 30mins]}
= {[1030, 1630]}

Similarly, t_2 is authorized to run under r_1 in A_1.
$T(r_1)$ = {[0930,1730]}
$I(t_2,A_1) = C^t(r_1) \cap T(r_1)$
= {[0930, 1700]}
$ET_1(t_0,t_2)$ = {[09:00, 1630]}

Similarly, $ET_1(t_0,t_i)$ for tasks t_3-t_8 can be calculated below.

$ET_1(t_0,t_3)$ = {[0900, 1600]},

$ET_1(t_0,t_4)$ = {[1054, 1554]},

$ET_1(t_0,t_5)$ = {[09:54, 15:54]},

$ET_1(t_0,t_6)$ = {[10:06, 15:06]},

$ET_1(t_0,t_7)$ = {[10:12, 15:12]},

$ET_1(t_0,t_8)$ = {[09:36, 14:36]}.

Then, the effective arrival time of t0 (i.e., the arrival time of the workflow), $EA_1(t_0)$, can be calculated as follows.

$$EA_1(t_0) = \bigcap_{t_i \in T} ET_k(t_0,t_i) = \{[10:54,\ 14:36]\}$$

This means that if the workflow arrives during [10:54, 14:36] and A_1 is used as the authorization solution, all tasks in the workflow can start execution without being delayed by the temporal constraints.

Given $EA_1(t_0)$, $EA_1(t_i)$ (i.e., the effective arrival time) for other tasks, t_1-t_8, can be calculated by using the equation below by using Matlab:

$$\begin{aligned} EA_1(t_1) &= \{[ld_{01} + (st_1 - st_0), ud_{01} + (st_1 - st_0)]\} \\ &= \{[10:54 + 30\,\text{min}], [14:36 + 30\,\text{min}]\} \\ &= \{[11:24, 15:06]\} \end{aligned}$$

$$EA_1(t_2) = \{[11:24, 15:06]\}$$

$$EA_1(t_3) = \{[12:00, 15:42]\}$$

$$EA_1(t_4) = \{[12:00, 15:42]\}$$

$EA_1(t_5) = \{[12:00, 15:42]\}$

$EA_1(t_6) = \{[12:48, 16:30]\}$

$EA_1(t_7) = \{[12:42, 16:24]\}$

$EA_1(t_8) = \{[13:24, 17:00]\}$

Similarly, we can calculate the value of $EA_k(t_0)$ ($2 \le k \le 8$) (i.e., other authorization solutions A_2-A_8).

The authorization solution **A_2** is below and $EA_2(t_0)$ are as follows.

$r(t_0, A_2) = r_1$,
$r(t_1, A_2) = r_3$,
$r(t_2, A_2) = r_1$,
$r(t_3, A_2) = r_1$,
$r(t_4, A_2) = r_2$,
$r(t_5, A_2) = r_3$,
$r(t_6, A_2) = r_2$,
$r(t_7, A_2) = r_3$,
$r(t_8, A_2) = r_2$.

$EA_2(t_0)$={10:54, 14:36}

For A_3:

$r(t_0, A_3) = r_1$,
$r(t_1, A_3) = r_3$,
$r(t_2, A_3) = r_2$,
$r(t_3, A_3) = r_1$,
$r(t_4, A_3) = r_2$,
$r(t_5, A_3) = r_3$,
$r(t_6, A_3) = r_2$,
$r(t_7, A_3) = r_2$,
$r(t_8, A_3) = r_2$.

$EA_3(t_0)$={[11:30, 14:36]}.

For A_4:

$r(t_0,A_4) = r_1$,

$r(t_1,A_4) = r_3$,

$r(t_2,A_4) = r_2$,

$r(t_3,A_4) = r_1$,

$r(t_4,A_4) = r_2$,

$r(t_5,A_4) = r_3$,

$r(t_6,A_4) = r_2$,

$r(t_7,A_4) = r_3$,

$r(t_8,A_4) = r_2$.

$EA_4(t_0)$={[11:30, 14:36]}.

For A_5:

$r(t_0,A_5) = r_1$,

$r(t_1,A_5) = r_4$,

$r(t_2,A_5) = r_1$,

$r(t_3,A_5) = r_1$,

$r(t_4,A_5) = r_2$,

$r(t_5,A_5) = r_4$,

$r(t_6,A_5) = r_2$,

$r(t_7,A_5) = r_2$,

$r(t_8,A_5) = r_2$.

$EA_5(t_0)$={[14:00, 14:36]}.

For A_6:

$r(t_0,A_6) = r_1$,

$r(t_1,A_6) = r_4$,

$r(t_2,A_6) = r_1$,

$r(t_3,A_6) = r_1$,

$r(t_4,A_6) = r_2$,

$r(t_5,A_6) = r_4$,

$r(t_6,A_6) = r_2$,

$r(t_7,A_6) = r_3$,

$r(t_8,A_6) = r_2$.

$EA_6(t_0)$={[13:30, 14:36]}.

For A_7:

$r(t_0,A_7) = r_1$,

$r(t_1,A_7) = r_4$,

$r(t_2,A_7) = r_2$,

$r(t_3,A_7) = r_1$,

$r(t_4,A_7) = r_2$,

$r(t_5,A_7) = r_4$,

$r(t_6,A_7) = r_2$,

$r(t_7,A_7) = r_2$,

$r(t_8,A_7) = r_2$.

$EA_7(t_0)$={[13:30, 14:36]}.

For A_8:

$r(t_0,A_8) = r_1$,

$r(t_1,A_8) = r_4$,

$r(t_2,A_8) = r_2$,

$r(t_3,A_8) = r_1$,

$r(t_4,A_8) = r_2$,

$r(t_5,A_8) = r_4$,

$r(t_6,A_8) = r_2$,

$r(t_7,A_8) = r_3$,

$r(t_8,A_8) = r_2$.

$EA_8(t_0)$={[13:30, 14:36]}.

Then,

$\bigcup_{A_k \in A} EA_k(t_0)$ ={[11:30, 14:36]}∪ {[10:54, 14:36]}∪ {[13:30, 14:36]}= {[10:54, 14:36]}

This suggests that whenever the workflow arrives in the time duration of [10:54, 14:36], there exists an authorization solution under which all tasks in the workflow can start execution without being delayed by the authorization constraints. When the workflow arrives in the time durations other than [1054, 1436], which can be calculated in Equation (3.6), it will be subject to the delay caused by the authorization constraints.

$$[S,E] - \bigcup_{A_k \in A} EA_k(t_0) = \{[09:00,17:00]\} - \{[10:54, 14:36]\}$$

$$= \{[09:00,10:53], [14:37, 17:00]\} \quad (3.6)$$

3.4 Summary

This Chapter investigates the issue of feasibility checking for authorization constraints deployed in workflow management systems. The feasibility checking problem is modeled as a constraint satisfaction problem in this chapter. Further, this chapter presents the methods to determine the time durations when the authorization constraints do not have negative impact on performance of workflow executions. A case study is given to illustrate the workings of the proposed methods.

4

Optimizing the Authorization Methods for Workflows

Chapter 3 presents the method to determine the time durations when the workflow execution is not affected by the authorization constraints, i.e., $\bigcup_{A_k \in A} EA_k(t_0)$. However, when a workflow arrives beyond $\bigcup_{A_k \in A} EA_k(t_0)$, the workflow will experience the delay. This chapter conducts the quantitative analysis of the delay. This chapter also proposes the optimal authorization method that can minimize the delay caused by the authorization constraints.

The rest of this chapter is organized as follows. Section 4.1 presents an intuitive authorization method, called the Earliest Available First (EAF) method, and the intuitive method will be used to compare against the optimal authorization method, called the Global Authorization Aware (GAA) method. The GAA method is presented in Section 4.2. Section 4.2 also conducts the quantitative analysis about the delay caused by the authorization constraints, given a workflow's arrival time, and proves that the proposed GAA method is optimal

in the sense that it can minimize the delay caused by the authorization constraints. Section 4.3 presents the experimental results to verify the effectiveness of the GAA method.

4.1. The EAF Authorization Method

The Earliest Available First (EAF) method is intuitive. Its fundamental idea is that when a task in the workflow is ready to run (i.e., all predecessors of the task has completed the executions), but all roles that can be assigned to the task (i.e., satisfy the authorization constraints) are not activated, a role with the earliest activation time will be assigned. The EAF method is outlined in Algorithm 1.

Algorithm 1. The EAF authorization method

1) For a ready task t_i in the workflow
2) Apply the role constraints, BoD and SoD to obtain a set of roles (denoted by $CA(t_i)$) that can be assigned to t_i;
3) If all roles in $CA(t_i)$ are not activated,
4) Assign to t_i a role with the earliest activation time;
5) If there are the roles in $CA(t_i)$ that are activated,
6) A role is randomly selected and assigned to t_i;

The delay caused the temporal constraints for a task is defined as the time that a ready task has to wait until the role assigned to the task become activated. The delay caused by the temporal constraints for a workflow (denoted by td) is defined as the sum of the delay caused

by temporal constraints for each individual task in the workflow. The workflows with different arrival times may have different td. $td(\tau)$ denotes the delay experienced by the workflow whose arrival time is τ. $td^{EAF}(\tau)$ denotes the delay experienced by all tasks in the workflow whose arrival time is τ when the EAF authorization method is applied.

4.2. The GAA Authorization Method

Assume a workflow arrives at time τ. $EA_k(t_0).next(\tau)$ denotes the beginning of the next duration after τ in $EA_k(t_0)$. If the workflow waits for $(EA_k(t_0).next(\tau)-\tau)$, then A_k can be used as the authorization solution of the workflow and the workflow execution can progress without further delay caused by the temporal constraints.

The GAA authorization method is proposed based on the above discussion. In the GAA method, the authorization solution that has the least value of $(EA_k(t_0).next(\tau)-\tau)$ is used to assign the roles to the tasks in a workflow. The GAA method is outlined in Algorithm 2. $td^{GAA}(\tau)$ denotes the delay caused by the temporal constraints for a workflow whose arrival time is τ under the GAA method, which equals to $(EA_k(t_0).next(\tau)-\tau)$.

Algorithm 2. The GAA authorization method

1) In all feasible authorization solution, find such a authorization solution, A_k, that A_k generates the minimal value of $(EA_k(t_0).next(\tau)-\tau)$;

2) The tasks in the workflow are authorized as designated in A_k;

Assume that a workflow arrives at the time point τ, and assume that it turns out that A_k is the authorization solution used for the workflow under the EAF method. We can prove that the delay caused by the temporal constraints for the workflow under the EAF method equals to $(EA_k(t_0).next(\tau)-\tau)$, as shown in Theorem 1.

Theorem 1: If a workflow arriving at time τ is authorized using the EAF method and the outcome is that the roles are assigned to the tasks in the workflow as in the authorization solution A_k, then Eq. 4.1 holds.

$$td^{EAF}(\tau)=(EA_k(t_0).next(\tau)-\tau) \quad (4.1)$$

Proof: If the role assigned to t_0 in A_k (i.e., $r(t_0)$) is only activated at time $EA_k(t_0).next(\tau)$, then t_0 starts execution at $EA_k(t_0).next(\tau)$ under the EAF method. Consequently, the delay caused by the temporal constraints on t_0 is $EA_k(t_0).next(\tau)-\tau$, and according to the definition of $EA_k(t_0).next(\tau)$, all successors of t_0 can start execution without further delay caused by the temporal constraints. Then

$td^{EAF}(\tau)=(EA_k(t_0).next(\tau)-\tau)$.

Therefore, Eq. 4.1 holds. We call $EA_k(t_0).next(\tau)$ t_0's effective start time (denoted by est_0).

When t_0 starts at $EA_k(t_0).next(\tau)$, we can calculate the start time of t_0's any successor t_i, which is called t_i's effective start time (denoted by est_i) because if t_i starts at time est_i, all successors of t_i can start execution without being delay by the temporal constraints of the roles assigned to the successors in A_k. est_i equals est_0 plus the length of the longest path from t_0 to t_i in the workflow.

If task t_0 starts execution at time τ_0' when the role assigned to t_0 in A_k becomes activated, then the delay caused by the temporal constraints on t_0 is $\tau_0' - \tau$. Assume t_k is t_0's child. If t_0 starts execution at τ_0', then t_k can be ready for execution (t_k's ready time is denoted by τ_k) at time τ_0' plus the length of the longest path from t_0 to t_k (i.e., all its predecessors have been completed), that is, $\tau_0' + (est_k - est_0)$, only subject to the availability of role $r(t_k)$.

If $r(t_k)$ is activated only at est_k, then t_k's delay caused by $r(t_k)$'s temporal constraints is $est_k - (\tau_0' + (est_k - est_0))$ = $est_0 - \tau_0'$, and all successors of t_k can start executions without being delayed by temporal constraints. Therefore, $td^{EAF}(\tau)$ can be calculated as:

$$\begin{aligned} td^{EAF}(\tau) &= (est_0 - \tau_0') + (\tau_{0'} - \tau) \\ &= est_0 - \tau \\ &= EA_k(t_0).next(\tau) - \tau \end{aligned}$$

It shows Eq. 4.1 holds. ■

If $r(t_k)$ is activated at time τ_k' ($\tau_k' < est_k$), then t_k starts execution at τ_k' in the EAF method. We can repeat the analysis similar as above only replacing t_0 with t_k, τ with τ_k and est_0 with est_k. Similarly, we can recursively conduct the analysis for the rest of all tasks in the workflow. It can be shown that Eq. 4.1 holds.

Besides the EAF method, other authorization method can be used to assign the roles to the tasks in a workflow. Based on Theorem 1, however, we can prove that no matter what authorization method is used to authorize the workflow, if it turns out that the workflow is authorized as in the authorization solution A_k, then the delay caused by the authorization constraints under the authorization method will be no less than the delay when using the EAF method to assign the roles to the tasks as in A_k. This relation is stated in Theorem 2. The proof of the theorem takes the similar steps as those in Theorem 1. The difference is that when using the EAF method to authorize the tasks as A_k, a task is authorized as soon as the role assigned to the task in A_k becomes activated, while under other authorization method, a task may be authorized (therefore start execution) later than the role's activation time.

Theorem 2: No matter what authorization method is used to assign the roles to the tasks in a workflow, if the outcome is that the tasks are authorized as the authorization A_k, then the delay caused by the authorization constraints under the authorization method

is no less than the delay when using the EAF method to authorize the tasks as in A_k.

Proof: Assume that a workflow arrives at time τ. Similar to Theorem 1, we can determine est_i for every task in the workflow.

If $r(t_0)$ in A_k is activated at time $EA_k(t_0).next(\tau)$, then the minimal delay caused by the authorization constraints is $EA_k(t_0).next(\tau)-\tau$, which equals to the delay generated when using the EAF method to authorize t_0. Any method that authorizes t_0 later than $EA_k(t_0).next(\tau)$ will generate a delay greater than that generated by the EAF method. The theorem holds.

If $r(t_0)$ becomes activated at time $\tau_{0'}$, but under the authorized method, task t_0 is authorized and starts execution at a later time $\tau_{0'}+\delta_0$ ($\delta_0>0$), then the delay caused by the authorization constraints on t_0 is $\tau_{0'}+\delta_0-\tau$.

Assume t_k is t_0's child. If t_0 starts execution at $\tau_{0'}+\delta_0$, then t_k can be ready for execution at time $\tau_k=\tau_{0'}+\delta_0+(est_k-est_0)$.

Assume $\tau_{0'}+\delta_0+(est_k-est_0)\geq est_k$. Then t_k can be authorized and start execution immediately and further, all successors of t_k can be authorized and start execution immediately when they are ready for execution. Therefore, the minimal delay for the workflow is $\tau_{0'}+\delta_0-\tau$. Since $\tau_{0'}+\delta_0+(est_k-est_0)\geq est_k$, we can have $\delta_0>est_0-\tau_{0'}$. Then the

following inequality holds, which shows that the EAF method generates the minimal delay.

$$\begin{aligned} \tau_{0'} + \delta_0 - \tau > \quad & est_0 - \tau \\ = \quad & EA_k(t_0).next(\tau) - \tau \\ = \quad & td^{EAF}(\tau) \end{aligned}$$

Assume $\tau_{0'} + \delta_0 + (est_k - est_0) < est_k$. We can repeat the same analysis on t_k as that on t_0, only replacing t_0 with t_k, τ with τ_k and est_0 with est_k. Similarly, we can recursively conduct the analysis for the rest of all tasks in the workflow. It can be shown that the theorem holds. ■

Based on Theorem 1 and 2, we can further prove that the GAA method is the optimal authorization method, as shown in Theorem 3.

Theorem 3: The GAA authorization method is optimal in the sense that under the GAA method, the delay caused by the authorization constraints for a workflow is not more than that under any other authorization method.

Proof: Given an authorization method and a workflow arriving at time τ, assume that the method authorizes the tasks as in the authorization solution A_k . From Theorem 2, we know that the delay generated by the authorization method is no less than the delay when using the EAF method to authorize the tasks as in A_k . From Theorem 1, we know that the delay generated by the EAF method can be calculated as $EA_k(t_0).next(\tau) - \tau$. Therefore, the given authorization method generates a delay greater than $EA_k(t_0).next(\tau) - \tau$. According to the algorithm of the

GAA method, the GAA method selects the authorization solution A_j that has the least value of $(EA_j(t_0)next(\tau)-\tau)$ from all possible authorization solutions. Therefore, the theorem holds. ■

4.3. Experimental Studies

4.3.1. Experimental Settings

This section conducts the simulation experiments (Graphical representation of the results by using Matlab) to evaluate the performance of the GAA method against that of the EAF method. The performance metrics evaluated in the experiments include the delay caused by the authorization constraints for a workflow (i.e., td defined in the first paragraph of Section 4.1) and the response time of a workflow (denoted as rt), which is defined as the duration between the time when a first task of the workflow arrives and the time when the last tasks of the workflow is completed.

In the experiments, the workflow is randomly generated. Each workflow containing TNUM tasks and each task in a workflow having the maximum of MAX_DG children. RNUM roles are assumed to exist in the system. “The tasks' role constraints (i.e., the set of roles that a task can assume) are set in the following fashion. The simulation sets a maximum number of roles that any task can assume in the role constraints, denoted as MAX_RCST , which represents the level of restrictions imposed on the role assignment for tasks. When setting the role constraint

for task t_i , the number of roles that can run t_i is randomly selected from [1, MAX_RCST], and then that number of roles are randomly selected from the role set".

NUM_SoD denotes the number of tasks associated with SoD constraints and NUM_BoD denotes the number of tasks that are associated with BoD constraints. "Duty constraints were set as follows. Each time, two tasks are randomly selected from the workflow to establish the BoD constraint between them until NUM_BoD tasks are covered. And then the same procedure is applied to establish the SoD constraints among tasks. In this process, the method presented in Section 3.1 (chapter 3) is used to make sure that the designated duty constraints on these selected tasks can be satisfied. We assume that the tasks execution times follow an exponential distribution". EX_H denotes the average execution time of the tasks in time units. In order to examine the delay caused by the authorization constraints, a workflow instance is only issued after the previous instance has been completed in the experiments. Unless otherwise stated, the value of td or rt depicted in the figure is the value averaged over all workflow instances issued within the period of the temporal constraints, which are set below.

"The temporal constraints on roles are set in the following way. For each role, time duration is selected from a period of P time units. The selected time duration occupies the specified percentage of the P time units, which is denoted as TEMP. The starting time of the selected duration is chosen randomly from the range of" [0, $P \times (1 - TEMP)$]. For example, if P = 100 and $TEMP$ =10%,

the starting point is randomly selected from 0 to $90\% \times 100$.

Unless otherwise stated, the parameters are set to be the values shown in Table 4.1.

Table 4.1. Experimental settings

Parameter	Value	Parameter	Value
TNUM	*15*	*MAX_DG*	*3*
EX_H	*5*	*RNUM*	*5*
MAX_RCST	*3*	*NUM_SoD*	*4*
NUM_BoD	*4*	*P*	*480*
TEMP	*20%*		

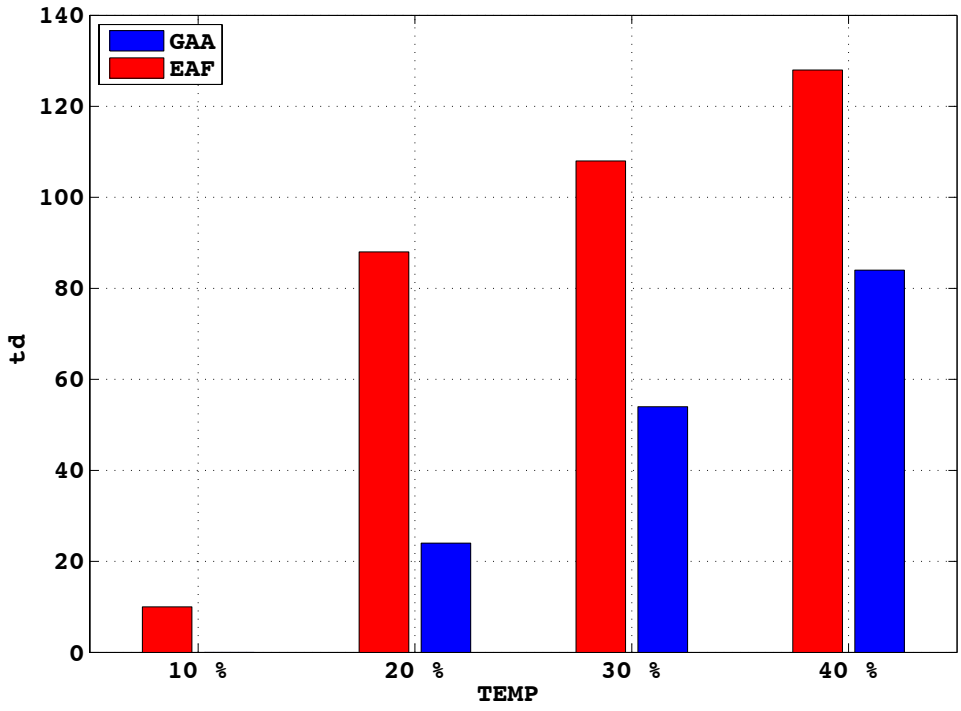


Figure 4.1. *td* under different TEMP

4.3.2. Temporal Constraints

Figure 4.1. shows the change of *td* as the temporal constraints (TEMP) changes. It can be seen from this figure that in all cases the GAA method achieves smaller *td* than EAF. For example, when TEMP is 10%, *td* is 0 under GAA while it is about 10 under EAF. The discrepancy becomes even bigger when TEMP increases. These results verify that the authorization method indeed matters and the GAA method is superior to the intuitive EAF method.

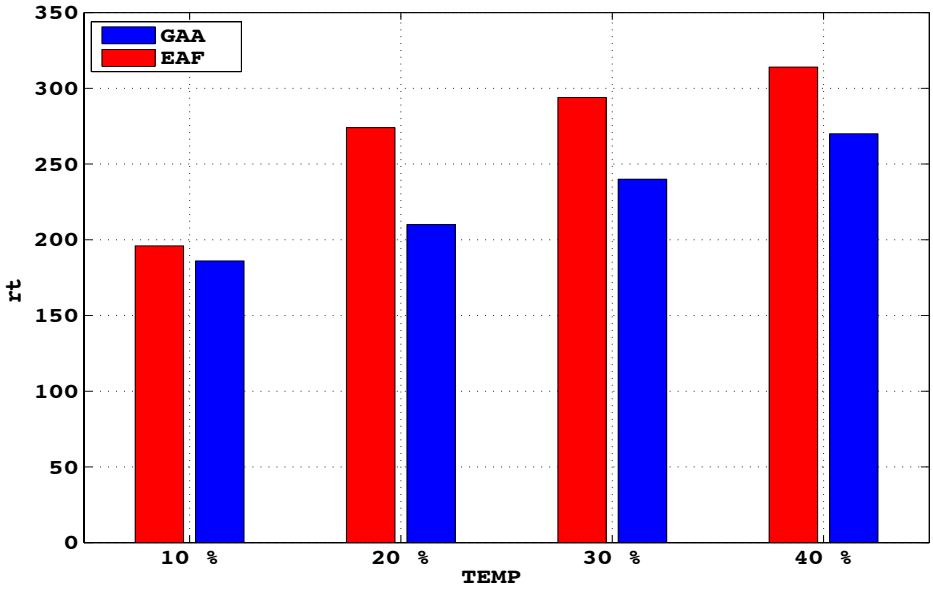


Figure 4.2. *rt* under different TEMP

Figure 4.2. compares *rt* achieved by GAA and EAF under different TEMP. It can be seen that GAA achieves the shorter *rt* than EAF in all cases. This is because GAA causes less delay and therefore achieves less response time than that under EAF.

4.3.3. Arrival Times of Workflows

The work in this chapter presents the method to determine the duration of the time for workflow arrivals within which the authorization constraints will not have negative performance impact. This shows that the arrival time of a workflow has impact on workflow performance. Figure 4.3 shows the value of td for different workflow arrival times under GAA and EAF. In these experiments, we set the period of all roles (i.e., P) as 480 time units, and then issue the workflow instances at the time points from 0 to 300 time units with increment of 60. It can be seen that once again, GAA incurs less td than EAF in all cases, except when the arrival time is 300 (whose will be explained later). Further, when the workflows arrive after 120, the GAA method does not cause any delay on workflow executions. These results verify that there indeed exist the durations for the workflow arrivals when the authorization constraints will not delay the workflow executions. The method proposed in this chapter is able to theoretically calculate such durations. A point to note is that when the arrival time is 300, no delay is caused under the EAF method either. This is because the time point 300 happens to be within the intersection of $EA_k(t_0)$ of all feasible authorization solutions. Therefore, the system can always find an activated role for any task to enable its execution.

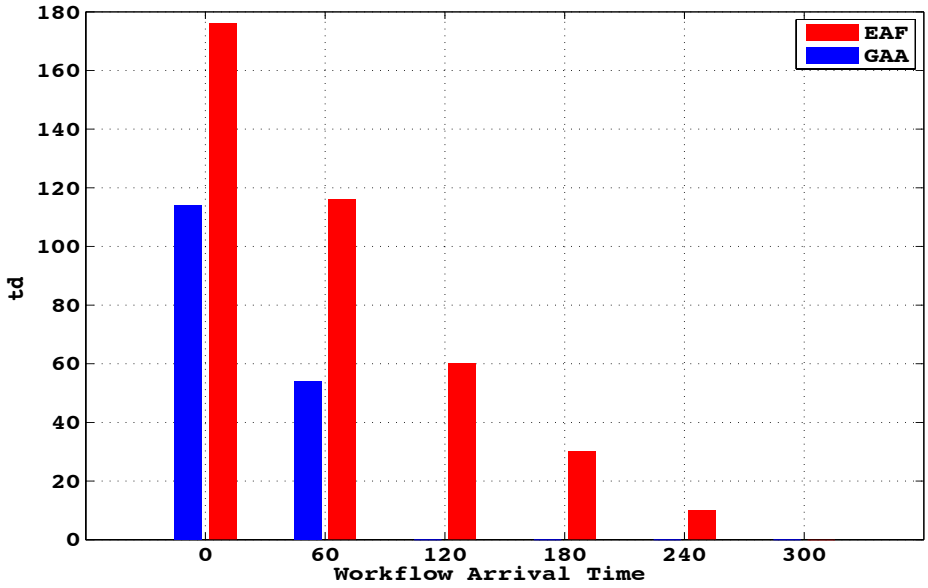


Figure 4.3. *td* under different workflow arrival times

Figure 4.4 shows that *rt* of the workflows with different arrival times. Again, GAA outperforms EAF in all cases. The *rt* trend is consistent with the *td* trend shown in Figure 4.3.

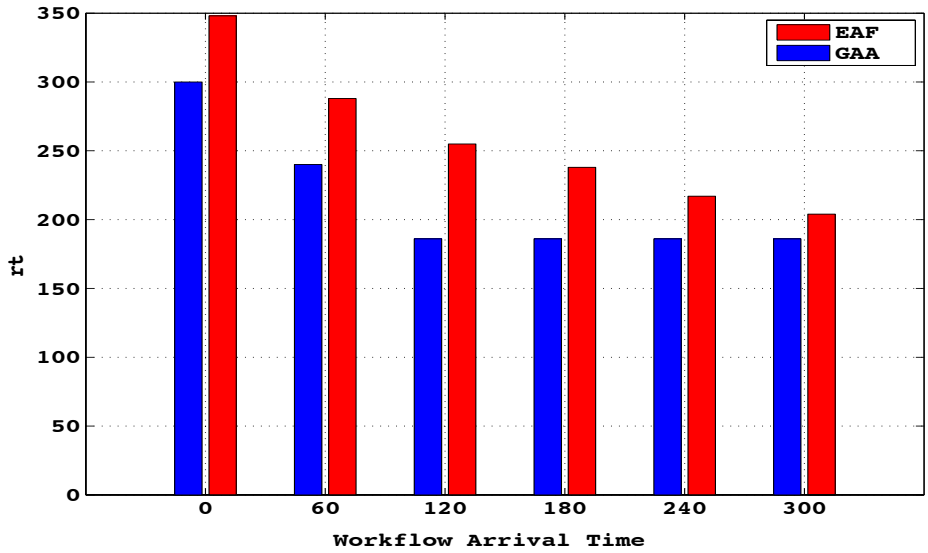


Figure 4.4. *rt* under different workflow arrival times

4.3.4. Execution Times of the Workflow Tasks

Obviously, increasing the execution times of the tasks in a workflow will increase the schedule length of the workflow. But do the execution times affect the authorization-related delay? Figure 4.5 shows the impact of the average execution time of the tasks in a workflow on the coverage of the temporal constraints (CTC), i.e., $\bigcup_{A_k \in A} EA_k(t_0)$.

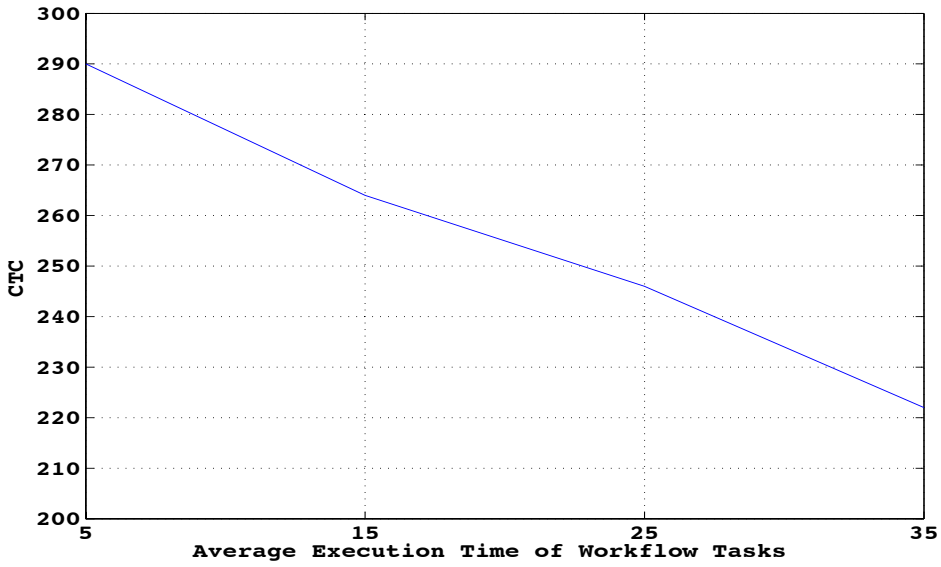


Figure 4.5. CTC under different average execution times of workflow tasks

As can be seen from this figure, CTC decreases as the average execution time increases. A reasonable explanation for this is that given a set of temporal constraints, the bigger the execution time of the tasks in a workflow is, the less likely the duration of the workflow execution fits into the temporal constraints. Therefore, CTC may become shorter. This result suggests that given a set of temporal constraints, a workflow with

longer tasks may be more likely to be delayed by the temporal constraints that a workflow with shorter tasks, which can be verified by the results, presented in Figure 4.6.

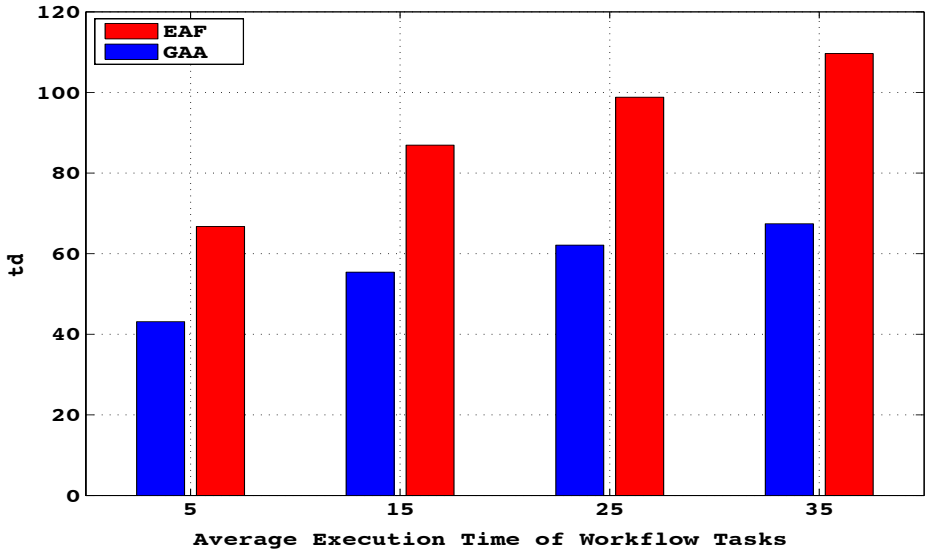


Figure 4.6. The coverage of temporal constraints (CTC) under different average execution times of workflow tasks.

Figure 4.6 demonstrates *td* under different average execution time of workflow tasks. Again, GAA causes less delay than EAF in all cases. It can also be observed from this figure that *td* increases as the average execution time of workflow tasks increases. The results coincide with the results in Figure 4.5. Indeed, when the execution times increases, CTC decreases. Then more workflow instances issued in the period of the temporal constraints will experience *td* . Consequently, *td* , which is the delay averaged over all workflow instances issued, is bigger.

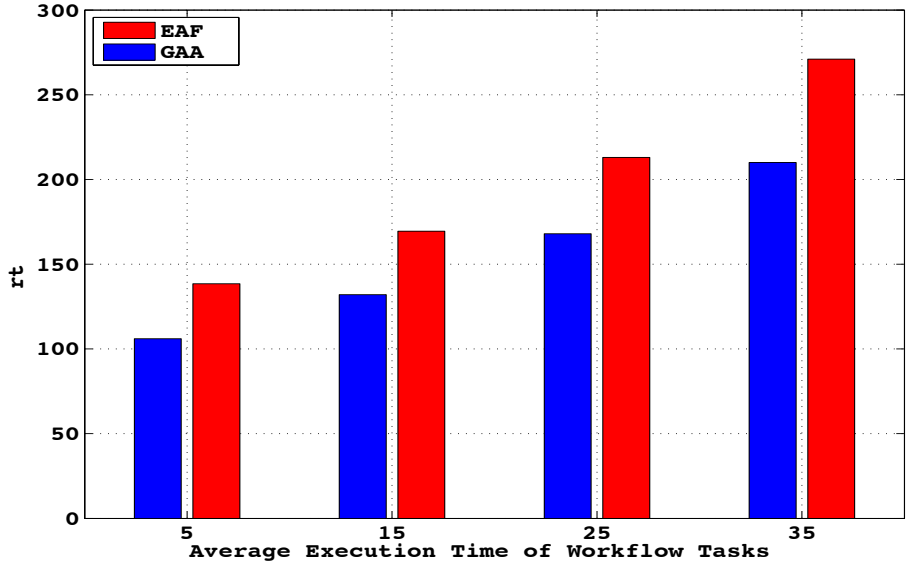


Figure 4.7. *rt* under different average execution times of workflow tasks.

Figure 4.7 shows *rt* generated by the GAA and the EAF method under different average execution time of workflow tasks. As can be observed, the GAA method generates shorter *rt* than EAF in all cases. This again verifies GAA causes less delay than EAF. There can be a situation when both GAA and EAF may produce the same delay. It can happen when EAF selects the roles as in optimal solution by choosing the roles randomly.

4.4 Summary

This chapter proposes two authorization methods. One is the intuitive method while the other is proved in this chapter to be the optimal authorization method. This chapter also quantitatively analyses the delay caused by the authorization constraints. The optimality proof for

the proposed method is based on the delay analyses. Finally, this chapter presents the simulation experimental results to verify the effectiveness of the proposed authorization methods.

5

Allocating Resources for Workflows Running under Authorization Control

Many workflow scenarios require human involvement. Therefore a workflow may consist of both human tasks, which are handled by human resources (e.g., employees in a company) and computing tasks, which are processed by computing resources. This chapter investigates the issue of allocating both human resources and computing resources for running workflows, so as to satisfy the QoS requirements of the workflows under the role-based authorization control deployed in the system.

The fundamental ideas of the work presented in this chapter are: 1) calculating the rate at which the requests arrive at each individual role under the deployed authorization constraints, and 2) based on the calculated arrival rates, presenting the methods to allocate the suitable amount of supporting resources for each role.

In the application domains of interest, the allocations of human resources and computing resources have different considerations. In the role-based

authorization control, a human resource is affiliated with a role. The human resources with different roles will incur different salary costs (e.g., hiring a branch manager is more expensive than hiring a cashier). The budget is often a major factor of determining the allocation of human resources in enterprise applications. Therefore, this chapter takes authorization constraints into account and develop an optimization method to allocate the proper amount of human resources for each role, so that the human tasks can achieve optimized performance subject to the budget limit for human resources.

Due to relatively low costs of computing resources, the cost is typically not a major concern for deploying low- or middle-end computing resources. When the workflows are running under authorization control, authorization constraints may incur performance penalty as discussed in the above workflow example in banks. Therefore, minimizing the overhead caused by the authorization constraints should be a main objective. In order to address this issue, this chapter develops a strategy of allocating computing resources. The strategy is able to calculate 1) a proper number of computing resources allocated to host each service, and 2) the processor sharing proportion in each resource allocated to run the tasks assuming a certain role.

In this chapter, a computing task involves invoking a computing service hosted in a central resource pool (e.g., a cluster or a Cloud). It is assumed that the invocation of computing services can only be initiated by a user with a certain role. A human task is executed by a human resource with a certain role. A human task can also

be regarded as invoking a human service provided by a user with a certain role. Therefore, for the simplicity of the presentation, we will discuss human tasks and computing tasks in a consistent manner in this chapter.

It is assumed that a set of services (human service or computing service) is hosted by the resources (human resources or computing resources). A task (human task or computing task) in a workflow invokes one of the hosted services.

The rest of this chapter is organized as follows. Chapter 5.1 presents the methods to calculate the arrival rate of the requests assigned to a role. Section 5.2 presents the method to allocate human resources, while Section 5.3 develops the method to allocate computing resources for hosting computing services. The experimental studies are presented in Section 5.4. Finally Section 5.5 gives the summary of the chapter.

5.1. Calculating the Arrival Rate under Authorization Control

In the workflow context in this chapter, a task in a workflow invokes one of the services running on the resources. In order to determine the amount of resources allocated to host services, this section first calculates the arrival rate of tasks for each service, which is the invocation rate of each service when there is no authorization control. However, under the authorization control, the tasks have to be assigned to a role before they can invoke the services, and the roles may have

temporal and cardinality constraints. Consequently, the services' invocation rates may be different from those when there is no authorization. This section derives the arrival rate of tasks for each role, i.e., the rate at which the tasks are assigned to each role under the authorization constraints. Table. 3.1 list the notations used in this thesis.

5.1.1. Calculating the Arrival Rates for Services

$S = \{s_1, \ldots, s_L\}$ denotes the set of services running on the resource pool.

$F = \{f_1, \ldots, f_N\}$ denotes the set of workflows, which has N types of workflows. Different types of workflow may have different topologies of tasks. A task in a workflow invokes one of the services in S. A service invocation matrix, denoted as $C_{L\,X\,N}$ (L cross N), can be used to represent which services are invoked by a workflow in F. The matrix has L rows and N columns. Row i represents service s_i, while column j represents workflow f_j. An element c_{ij} represents how many times service s_i is invoked by workflow f_j (different tasks in a workflow may invoke the same service). λ_i denotes the arrival rate of Workflow f_i.

The arrival rate of the requests for service s_i, denoted as $\lambda_{(si)}$, can be calculated from the service-calling matrix, $C_{L\,X\,N}$, as in Eq. 5.1.

$$\lambda_{(si)} = \sum_{J=1}^{N} (c_{ij} \times \lambda_i) \qquad (5.1)$$

5.1.2. Calculating the Arrival Rates for Roles

This subsection analyses how to calculate the arrival rates for the roles under three types of authorisation constraints: role constraints, temporal constraints and cardinality constraints [Zou2009].

5.1.2.1. Arrival Rates under Role Constraints

$R = \{r_1, \dots, r_M\}$ denotes the set of roles defined in the authorisation control system. The role constraint specifies the set of roles that are permitted to run a particular service. $C^r(s_i)$ denotes the role constraint applied to service s_i.

A role constraint matrix, denoted as $O_{(L \times M)}$, is used to represent which roles are permitted to invoke a particular service. The matrix has L rows and M columns. Row i represents service s_i, while column j represents role r_j. An element O_{ij} is 0 or 1, representing whether role r_j is permitted to run service s_i.

If only role constraints are considered and multiple roles are permitted to run a service, a role is randomly selected. In the requests for service s_i, the arrival rate of the requests allocated to role r_j, denoted as $\lambda^r(s_i, r_j)$, can be calculated using Eq. 5.2. Further, the arrival rate of all service requests allocated to r_j, denoted as $\lambda^r(r_j)$ can be calculated using Eq. 5.3.

$$\lambda^r(s_i, r_j) = \begin{cases} \frac{\lambda(s_i)}{\sum_{j=1}^{M} o_{ij}} & if \ \sum_{j=1}^{M} o_{ij} \neq 0 \\ 0 & if \ \sum_{j=1}^{M} o_{ij} = 0 \end{cases} \quad (5.2)$$

$$\lambda^r(r_j) = \sum_{i=1}^{L} \lambda^r(s_i, r_j) \quad (5.3)$$

5.1.2.2. Arrival Rates under both Role Constraints and Temporal Constraints

In most cases, a role is activated periodically. For example, the role of bank manager is only activated from 9am to 12pm in a day. Therefore, the temporal constraint of role r_i, denoted as $C^t(r_i)$ can be expressed as Eq. 5.4, where P_i is the period, D_i is the time duration when r_i is activated in the period P_i, and S_i and E_i are the start and end time points when this period pattern begins and ends. E_i can be ∞, meaning the periodic pattern continues indefinitely. A temporal function for role r_i is defined in Eq. 5.5. The value of the temporal function is 1 if the role is activated at the current time point t. Otherwise; the value of the function is 0. For example for role r_1 in Fig. 5.1 at time t = 3, eq. 5.5 will be 3 – 0 * 6 = 3 which is less than D_i = 4, so the role will be active at t=3 and similarly at t=5 the value of the eq. 5.5 will be 5 – 0 * 6 = 5 which is greater than Di=4 so the role is inactive at t=5.

$$C^t(r_i) = (P_i, D_i, S_i, E_i) \quad (5.4)$$

$$ft(r_i, t) = \begin{cases} 1 & if\ t - \left\lfloor \frac{t - S_i}{P_i} \right\rfloor \times P_i \leq D_i \\ 0 & if\ t - \left\lfloor \frac{t - S_i}{P_i} \right\rfloor \times P_i > D_i \end{cases} \quad (5.5)$$

The function $nr(s_i, t)$ defines the number of roles which are activated at time point t and are permitted to run

service s_i. nr(s_i, t) can be calculated using Eq. 5.6, which is based on the roles' temporal functions.

$$nr(s_i, t) = \sum_{r_j \in C^r(s_i)} ft(r_j, t) \quad (5.6)$$

$\lambda^{rt}(s_i, r_j)$ denotes the arrival rate of the tasks that are requesting service s_i and are assigned to role r_j when both role constraints and temporal constraints are considered. $\lambda^{rt}(s_i, r_j, t)$ denotes at time t, the arrival rate of the requests that assume r_i and invoke s_i. $\lambda^{rt}(s_i, r_j, t)$, can be calculated as Eq. 5.7. Then $\lambda^{rt}(s_i, r_j)$ (i.e., the average arrival rate of the requests that assume r_i and invoke s_i) can be calculated as Eq. 5.8, where P is the minimal common multiple of the periods of all roles that can run s_i.

$\lambda^{rt}(r_j)$ denotes the arrival rate of all tasks that are assigned to role r_j when both role constraints and temporal constraints are considered. $\lambda^{rt}(r_j)$ can be calculated as Eq. 5.9.

$$\lambda^{rt}(s_i, r_j, t) = \frac{\lambda(s_i)}{nr(s_i, t)} \quad (5.7)$$

$$\lambda^{rt}(s_i, r_j) = \frac{\int_0^P \lambda^{rt}(s_i, r_j, t)dt}{P} \quad (5.8)$$

$$\lambda^{rt}(r_j) = \sum_{i=1}^{L} \lambda^{rt}(s_i, r_j) \quad (5.9)$$

Figure 5.1 illustrates the temporal constraints of three roles, r_1, r_2, r_3, in which $t(r_1) = (6, 4, 0, \infty)$,

$t(r_2) = (4, 2, 0, \infty)$, and $t(r_3) = (3, 1, 0, \infty)$ by using equation 5.4.

Figure 5.2 illustrates $nr(s_i, t)$ for the three roles in Figure 5.1. As can be seen from this figure, the number of activated roles that can run s_i varies over time. Note that since the minimal common multiple of the periods of r_1, r_2, r_3 is 12, the pattern of $nr(s_i, t)$ will repeat in every time duration of 12.

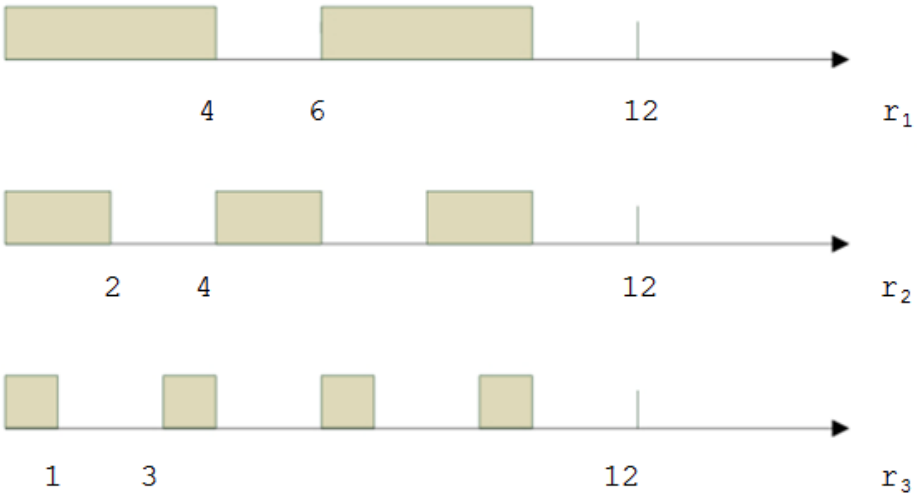


Figure 5.1. An Example of the temporal constraints of roles. Note that shaded area represent that role is available.

According to Eq. 5.8, $\lambda^{rt}(s_i, r_j)$ is $\frac{\lambda(s_i)}{3}$ and $\lambda(s_i)$ at time point 0 and 12, respective.

The analysis can be easily extended to the case where the temporal constraint of a role consists of multiple

different periodic patterns, each of which is specified by Eq. 5.4. The analysis for multiple periodic patterns is omitted in this chapter.

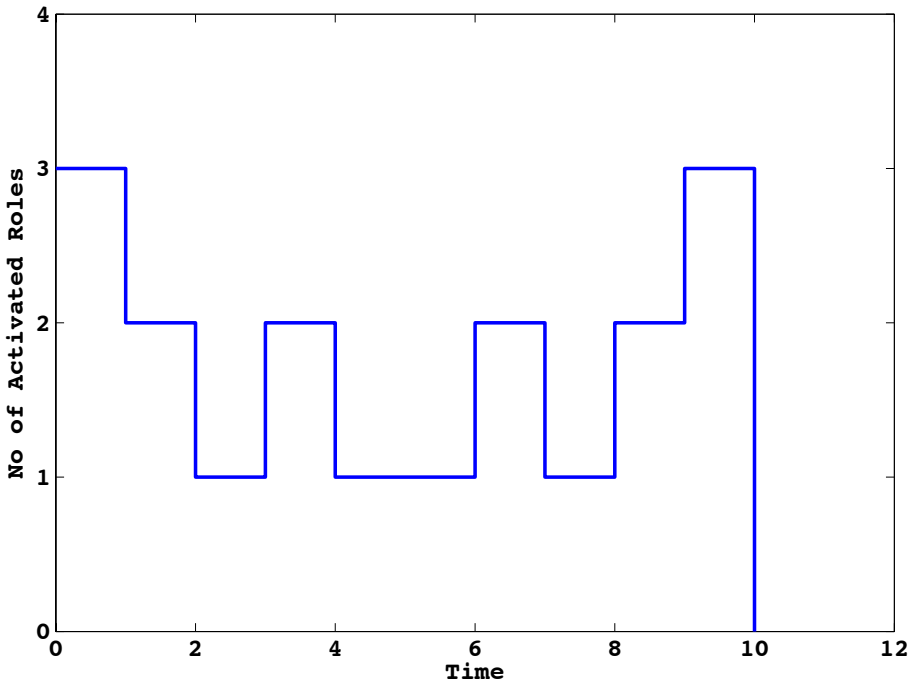


Figure 5.2. The function of the number of activated roles for the example in Figure 5.1.

5.1.2.3. Arrival Rates under both Role Constraints and Cardinality Constraints

The cardinality constraint of a role is defined as the maximum number of tasks that the role can run simultaneously. $C^c(r_i)$ denotes the cardinality constraint of r_i.

In order to avoid the execution delay caused by r_i's cardinality constraint, the number of the tasks running

under role r_i should be less than $C^c(r_i)$ when a new task arrives requesting role r_i. $\lambda^{rc}(s_i, r_j)$ denotes the arrival rate of the tasks that are requesting service s_i and are assigned to role r_j when both role constraints and cardinality constraints are considered. $\lambda^{rc}(r_i)$ denotes the arrival rate of all tasks that are assigned to role r_j when both role constraints and cardinality constraints are considered.

According to Littles Law [Kleinrock1976], we have Eq. 5.10, where rp_i is the mean response time of the tasks running under role r_i.

$$C^c(r_i) = \lambda^{rc}(r_i) \times rp_i \quad (5.10)$$

$np(r_i)$ denotes the number of resources used to serve the tasks running under r_i, and these resources are modelled as a $M/M/np(r_i)$ queuing model. w_i denotes the waiting time of the tasks assigned to role r_i. According to the queuing theory [Kleinrock1976], w_i can be calculated by Eq. 5.11, where e_i is the execution time of the tasks assigned to r_i.

$$W_i = \frac{\lambda^{rc}(r_i) \times {e_i}^2}{np(r_i)^2 - e_i \times np(r_i) \times \lambda^{rc}(r_i)} \quad (5.11)$$

Since Eq. 5.12 holds, Eq. 5.10 can be transformed to Eq. 5.13.

$$rp_i = w_i + e_i \quad (5.12)$$

$$C^c(r_i) = \lambda^{rc}(r_i) \times \left(\frac{\lambda^{rc}(r_i) \times {e_i}^2}{np(r_i)^2 - e_i \times np(r_i) \times \lambda^{rc}(r_i)} + e_i\right) \quad (5.13)$$

$\lambda^{rc}(r_i)$ can then be calculated by transforming Eq. 5.13 to Eq. 5.14, which is the maximum task arrival rate that r_i can tolerate in order to avoid the overhead caused by its cardinality constraint.

$$\lambda^{rc}(r_i) = \frac{np(r_i) \times \sqrt{np(r_i) + C^c(r_i)^2 - 4 \times C^c(r_i) \times (np(r_i) - 1)} + np(r_i) \times (np(r_i) + C^c(r_i))}{2 \times e_i \times (np(r_i) - 1)} \quad (5.14)$$

5.1.2.4. Arrival Rates under Role, Temporal and Cardinality Constraints

$\lambda^{rtc}(r_i)$ denotes the arrival rate of the tasks that are assigned to r_i when the role constraints, temporal constraints and cardinality constraints are considered. $\lambda^{rtc}(r_i)$ can be calculated as Eq. 5.15.

$$\lambda^{rtc}(r_i) = \min\left(\lambda^{rt}(r_i), \lambda^{rc}(r_i)\right) \quad (5.15)$$

5.2. Allocating Resources for Human Tasks

Since a human task in a workflow invokes a human service provided by a user with a certain role, we need

to allocate an appropriate amount of human resources for each role, so that the desired performance can be achieved for human tasks. In Section 5.1, we have derived the tasks' arrival rates for roles under the authorization constraints. This section models the problem of allocating human resources for roles, aiming to optimizing the average response time of the human tasks. Since the budget is often a major factor in hiring human resources, the allocation of human resources is subject to a budget constraint.

B denotes the budget that can be spent for human resources. b_i denotes the cost of hiring a human resource assuming role r_i (e.g., the salary for a staff taking the manager role). h_i denotes the number of the human resources allocated for role r_i. The budget constraint can be expressed as Eq. 5.16, where h_i is an integer.

$$\sum_{i=1}^{M}(b_i \times h_i) \leq \mathrm{B} \qquad (5.16)$$

We model the human resources allocated for role r_i as an $M/M/h_i$ queueing model. According to the queuing theory [Kleinrock1976], the average response time of human tasks over all roles, denoted as RH, can be calculated by Eq. 5.17.

$$RH = \sum_{i=1}^{M}\left(rp_i \times \frac{\lambda_i}{\sum_{j=1}^{M}\lambda_j}\right) \qquad (5.17)$$

Following the similar derivation as in Eq. 5.11 and Eq. 5.12, Eq. 5.17 can be transformed to Eq. 5.18.

$$RH = \sum_{i=1}^{M}\left(\left(\frac{\lambda_i \times {e_i}^2}{{h_i}^2 - e_i \times h_i \times \lambda_i} + e_i\right) \times \frac{\lambda_i}{\sum_{j=1}^{M}\lambda_j}\right) \quad (5.18)$$

From the analysis in Subsection 5.1.2.3, we know that in order to reduce the performance penalty caused by cardinality constraints, the tasks assigned to a role with a tighter cardinality constraint (i.e., less value of $C^c(r_i)$) should have a shorter average response time so that they can be turned around faster in the system. This relation can be represented in Eq. 5.19.

$$rp_i \leq rp_j, \ if\ C^c(r_i) \leq C^c(r_j) \quad (5.19)$$

The objective is to find $h_i(1 \leq i \leq M)$ subject to Eq. 5.16 and Eq. 5.19, such that RH in Eq. 5.18 is minimized. This is a constrained-minimum problem, and there do exist solvers to find its solution [Cuervo2010].

5.3. Allocating Resources for Computing Tasks

A computing task in the workflow invokes a service hosted in the central computing resource pool (e.g., a Cluster or a Cloud [He2011a]). This section aims to determine the suitable amount of computing resources allocated for hosting each service and for processing the tasks assuming each role, so that the overhead caused by the authorization constraints can be minimized.

n_i denotes the number of homogeneous nodes used to host service s_i. According to the role constraints, we know which roles can invoke the services. Using Eq. 5.8, we can calculate the arrival rate of the requests that assume r_j to invoke s_i. Applying Little's law, the desired average response time for a request assuming r_j (i.e., rp_j) can be calculated as Eq. 5.20. In order to satisfy rp_j, we need to find a minimal number of nodes for hosting each service (i.e., the minimal value of n_i, $1 \leq i \leq L(services)$, and to find the proportion of processing capability (in a node hosting s_i) allocated to run the requests that assume role r_j, which is denoted as α_{ij}.

$$rp_j = \frac{C^c(r_j)}{\lambda^{rtc}(r_j)} \tag{5.20}$$

We first calculate the desired response time for the requests that assume r_j to invoke s_i. es_i denotes the mean execution time of the requests invoking service s_i, which can be obtained by benchmarking the executions of service s_i. $rp(r_j, s_i)$ denotes the desired mean response time of the requests that assume role r_j to invoke service s_i. Then $rp(r_j, s_i)$ can be calculated from Eq. 5.21, where Eq. 5.21.(ii) expresses that the ratio among $rp(r_j, s_i)$ should be equal to the ratio among es_i ($s_i \in C^s(r_j)$).

$$\begin{cases} \sum_{s_i \in C^s(r_j)} \left(\frac{\lambda^{rtc}(s_i, r_j)}{\lambda^{rtc}(r_j)} \times rp(r_j, s_i)\right) = rp_j & (i) \\ \forall s_i, s_k \in C^s(r_j), rp(r_j, s_i) = \frac{es_i}{es_k} \times rp(r_j, s_k) & (ii) \end{cases} \quad (5.21)$$

The problem of finding α_{ij} in a node hosting service s_i relies on the analysis of multiclass queuing systems with Generalized Processor Sharing, which is notoriously difficult [Liu2001]. The analysis of the multiclass single-server queue can be approximated by decomposing it into multiple single-class single-server queues with the capacity equal to $\alpha_{ij}\mu_i$ [Liu2001], where μ_i is the processing rate of a node for serving service s_i (i.e., $\frac{1}{es_i}$). Finding α_{ij} and n_i can then be modelled as Eq. 5.22, where Eq. 5.22.i is constructed based on the equation of calculating average response time of the tasks in an M/M/1 queue [Kleinrock1976]. In Eq. 5.22, the number of unknown variables (i.e., n_i and α_{ij}, $r_j \in C^r(s_i)$) is the same as the number of equations in Eq. 5.22. Therefore, n_i and α_{ij} can be calculated.

$$\begin{cases} \forall r_j \in C^r(s_i), \frac{\alpha_{ij}}{es_i} - \frac{\lambda^{rtc}(r_j, s_i)}{n_i} = \frac{1}{rp(r_j, s_i)} & (i) \\ \sum_{r_j \in C^r(s_i)} \alpha_{ij} = 1 & (ii) \end{cases} \quad (5.22)$$

5.4. Experimental Studies

5.4.1. Experimental Settings

This section presents the simulation experiments to demonstrate the effectiveness of the resource allocation strategies developed in this chapter by using Matlab tool. The metrics used to measure the performance obtained by resource allocation strategies are mean response time of workflows and resource utilization.

In the simulations presented in this chapter, the workflows are randomly generated, each workflow containing TNUM tasks and each task in a workflow having the maximum of MAX_DG children. A workflow contains two types of task, Human Task (HT) and Computing Task (CT), following a certain ratio of the number of tasks in each type (denoted as |HT|:|CT|). Assume that all computing tasks can only be initiated by a user with a certain role (i.e., all computing tasks are human-aided computing tasks). RNUM roles and UNUM users are assumed to be involved in processing the workflows.

The role constraints (i.e., the set of roles that a task can assume) for each HT and CT are set in the following fashion. The simulation sets a maximum number of roles that any task can assume in the role constraints, denoted as MAX_RCST, which represents the level of restrictions imposed on the role assignment for tasks. When setting the role constraint for task t_i, the number of roles that can run t_i is randomly selected from [1, MAX_RCST], and then those numbers of roles are randomly selected from the role set. A similar scheme is

used to associate users to roles. The maximum number of users a role can be associated to is denoted as MAX_U2R. The number of users belonging to role r_i is randomly selected from [1, MAX_U2R]; and these users are then randomly selected for r_i from the user set.

The temporal constraints on roles are set in the following way. For each role, time duration is selected from a period of TD time units. The selected time duration occupies the specified percentage of the TD time units, which is denoted as TEMP. The starting time of the selected duration is chosen randomly from the range of [0, TD X (1-TEMP)]. For example, if TD = 200 and TEMP = 70%, the starting point is randomly selected from 0 to 30% X 200.

CARD denotes the cardinality constraint, i.e., the maximum number of the tasks that can be run simultaneously in the system by a role.

The arrivals of workflow instances are generated follow the Poisson process and that the tasks execution times follow an exponential distribution. The human tasks have the average execution time of EX_H time units, while the computing tasks, including HCT and ACT, have the average execution time of EX_C units.

In the experiments, we first plan the capacity of human resources and calculate the capacity of computing resources (i.e., the number of computing resources) and the allocation strategy of computing resources (i.e., the processing sharing fraction for each role). Then we generate the workflows and run them in the resources in the aforementioned fashion. The obtained performance is

recorded. In the experiments, we also compare the performance obtained by our strategies with the performance by conventional strategies. Conventional capacity planning and resource allocation strategies do not take authorization constraints into account, and often allocate the amount of resources proportional to the arrival rate.

5.4.2. Experimental Results

In order to demonstrate the effectiveness of the allocation strategy for human resources, we conduct the experiments using the traditional allocation strategy for human resources. In the traditional strategy, we don't impose authorization constraints, and assume particular types of human tasks are handled by a particular user. Based on the arrival rate of workflows, we can obtain the arrival rate of the requests for each human service. The number of human resources allocated for handling each human service is proportional to the arrival rate of requests for each service, subject to the constraint that the total cost of hiring all human resources is no more than the budget B. With the same budget constraint, we conduct the experiments using the allocation strategy for human resources developed in this chapter. Then we run the workflows consisting of only human tasks under authorization constraints on both resource allocation settings. Figure 5.3 shows their performance in terms of mean response time (i.e., RH) as the arrival rate of the workflow increases.

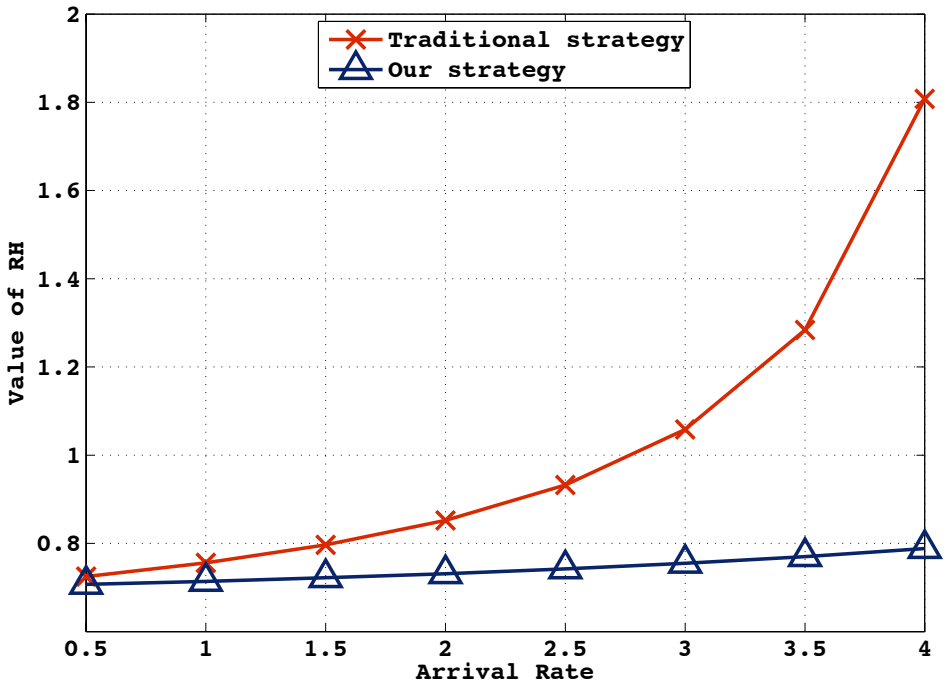


Figure 5.3. Comparing average response time of human tasks between our strategy and the traditional allocation strategy for human resources; TNUM=15, MAX_DG=10, EX_H=7, RNUM=5, UNUM=15, MAX_U2R=5, MAX_RCST=4, CARD=4, TEMP=70%, TD=200, B=200, b_1, ..., b_{RNUM} = 10, 8, 2, 5, 9. (Experimental setup and Variables are defined at page 87)

As can be seen from Figure 5.3, our strategy outperforms the traditional strategy in all cases and the trend becomes more prominent as the arrival rate of workflows increases. This is because our strategy takes into account authorization constraints and the arrival rate of requests, and establishes the optimization equations to calculate the allocation of human resources that can minimize the mean response time of human tasks. In the traditional allocation strategy, the resources are allocated only based on the arrival rate of the requests for services, not considering authorization constraints. Due to the existence of authorization constraints, the

incoming requests need to be first assigned to roles and then invoke the corresponding services. Consequently, the rate at which the services are invoked under authorization may be different from that without authorization. Therefore, the resources allocated by the traditional strategy may not be in line with the resource demands, and consequently the performance may be impaired. Further, as the arrival rate of workflows increases, it becomes more likely that the following situation may occur under the traditional strategy due to the fact that the amounts of resources allocated for different services have to maintain the proportion: the resources allocated for some services become saturated while the resources are over-provisioned for other services due to the extra authorization constraints. In our strategy, however, the authorization constraints are taken into account, and the amount of resources for each role is calculated accordingly. The effect is that the cost spent for allocating over-provisioned resources is now used to allocate more resources that are saturated under the traditional strategy.

Figure 5.4 compares resource utilizations between our strategy and the traditional strategy in the same experimental settings as in Figure 5.3. As can be seen from Figure 5.4, our strategy achieves higher utilization than the traditional strategy. This is still because the traditional strategy allocates resources based on the arrival rate of the requests for services, which causes the over-provisioned resources for some services after imposing authorization constraints.

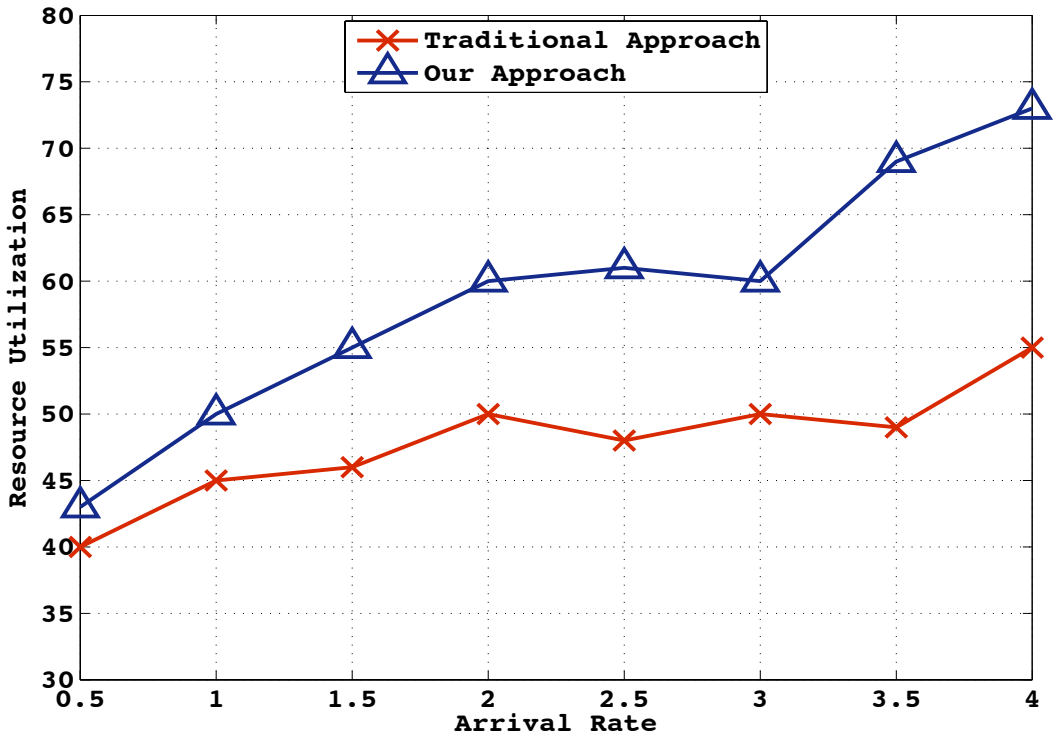


Figure 5.4. Comparing resource utilization between our strategy and the traditional allocation strategy for human resources; the experimental settings are the same as in Figure 5.3.

In order to demonstrate the effectiveness of the allocation strategy for computing resources, we conduct the experiments using the traditional allocation strategy for computing resources. In our strategy, the authorization constraints are taken into account, and the proportion of processing capability allocated for each role is calculated accordingly. In the traditional strategy, all tasks are treated equally and are put into the central waiting queue in the cluster of computing resources. When a computing resource is free and the authorization constraints are satisfied, the task at the

head of the waiting queue is put into execution in the free resource.

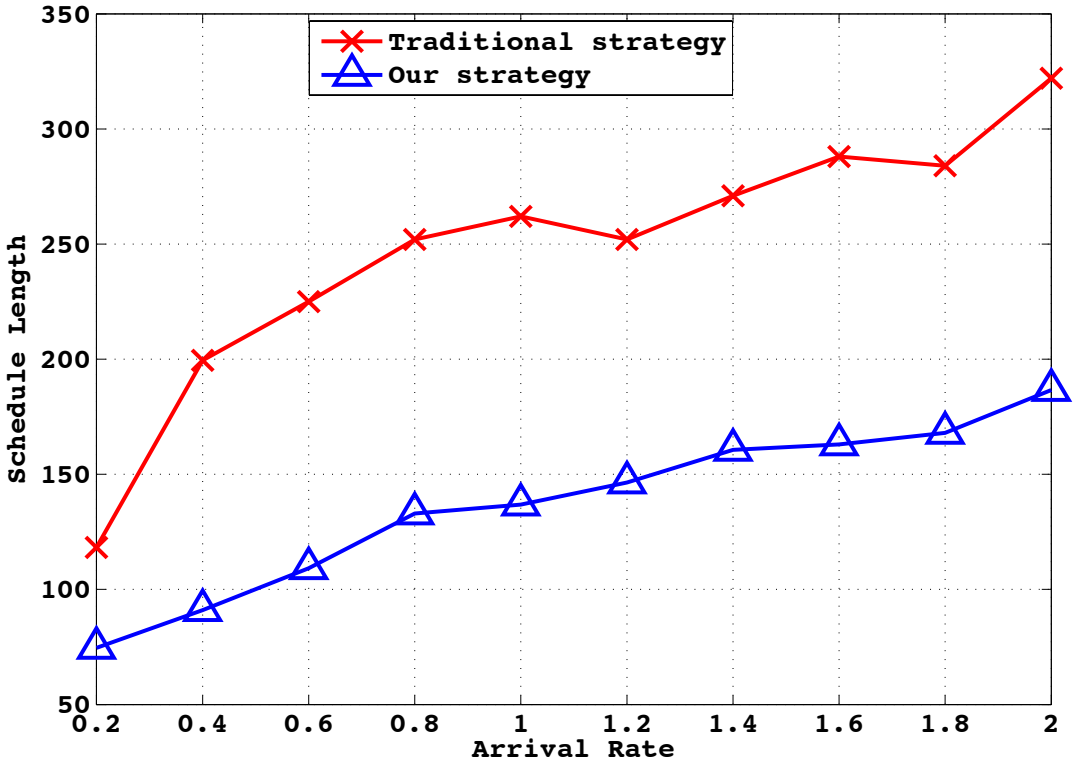


Figure 5.5. Comparison of performance in terms of average response time between our allocation strategy and traditional strategy for computing resources; NUM=15, MAX_DG=10, EX_C=7, RNUM=5, UNUM=15, MAX_U2R=5, MAX_RCST=4, CARD=4, TEMP=70%, TD=200

Figure 5.5 compares average response time of computing tasks between our strategy and the traditional allocation strategy for computing resources. In these experiments, all tasks in a workflow are computing tasks. In the traditional resource allocation strategy, authorization constraints are not taken into account, and the amount of resources allocated for a service is proportional to the arrival rate of the requests for the service. The

allocation strategy developed in this chapter calculates the arrival rate for each role and then further calculates the amount of resources allocated to serve the requests assigned to each role.

As can be seen from Figure 5.5, our strategy performs better than the traditional strategy. This can be explained as follows. In our allocation strategy, the authorization constraints are taken into account. For example, if role r_i has the tighter cardinality constraint (i.e., smaller value of $C^c(r_i)$, more proportion of processing capability will be allocated to serve the tasks assuming r_i, so that the number of those tasks in the system will be less and the performance penalty imposed by r_i's cardinality constraint can be reduced. In the traditional allocation strategy, the tasks assuming different roles are treated equally, and therefore cannot prioritize the tasks that are assuming the roles with tight cardinality constraint and therefore should be turned around faster. Tasks assuming tight cardinality have priority on other tasks to reduce the delay because of the tight cardinality.

Figure 5.6 compares the resource utilization between our strategy and the traditional allocation strategy for computing resources. It can be seen from this figure that our strategy can achieve higher resource utilization than the traditional strategy. This can be explained as follows. In the traditional strategy, it is more likely that the tasks have to wait in the waiting queue even if there are free resources in the system, because the tasks assuming the roles with tight cardinality constraints can be turned around faster in our strategy. This causes lower resource utilization.

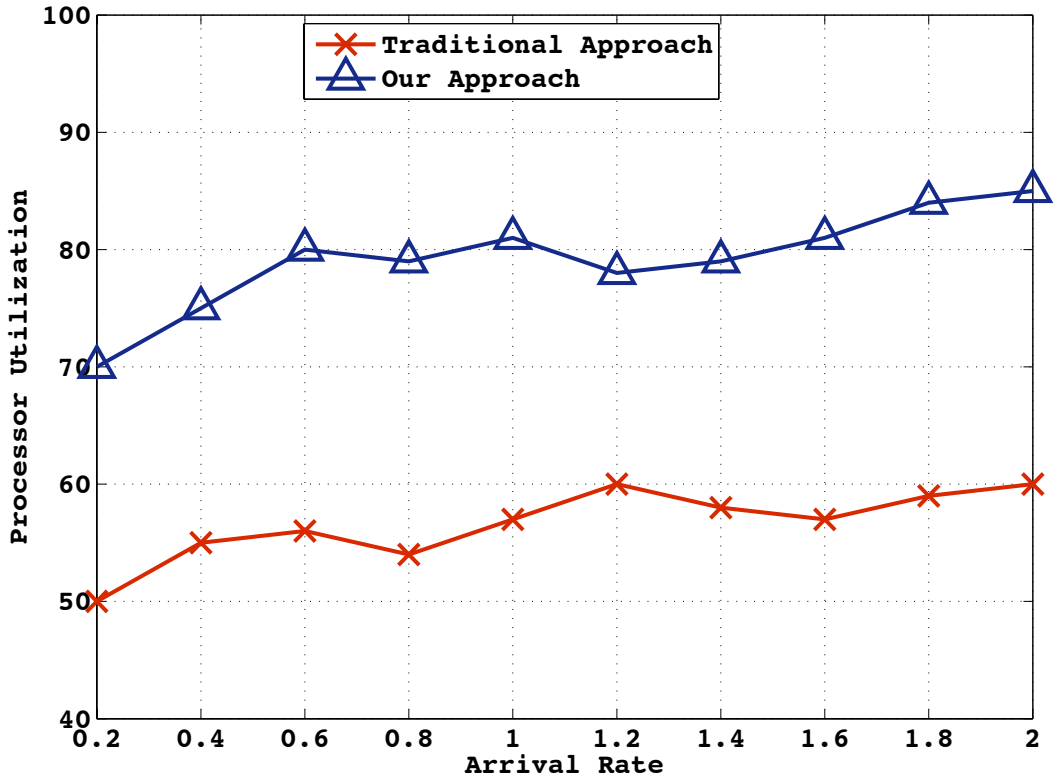


Figure 5.6 Comparing resource utilization between our strategy and the traditional allocation strategy for computing resources; the experimental settings are the same as in the Figure 5.5.

Figure 5.7 compares the schedule lengths of workflows achieved by our strategy and the traditional strategy. In these experiments, a workflow contains both human tasks and computing tasks. Then we run the workflows on human resources and computing resources allocated by our strategy as well as by the traditional strategy. Figure 5.7 shows that our strategy achieves shorter schedule length than the traditional strategy. Again, this is because our strategy takes authorization constraints into account and allocate suitable amount of resources for both human resources and computing resources.

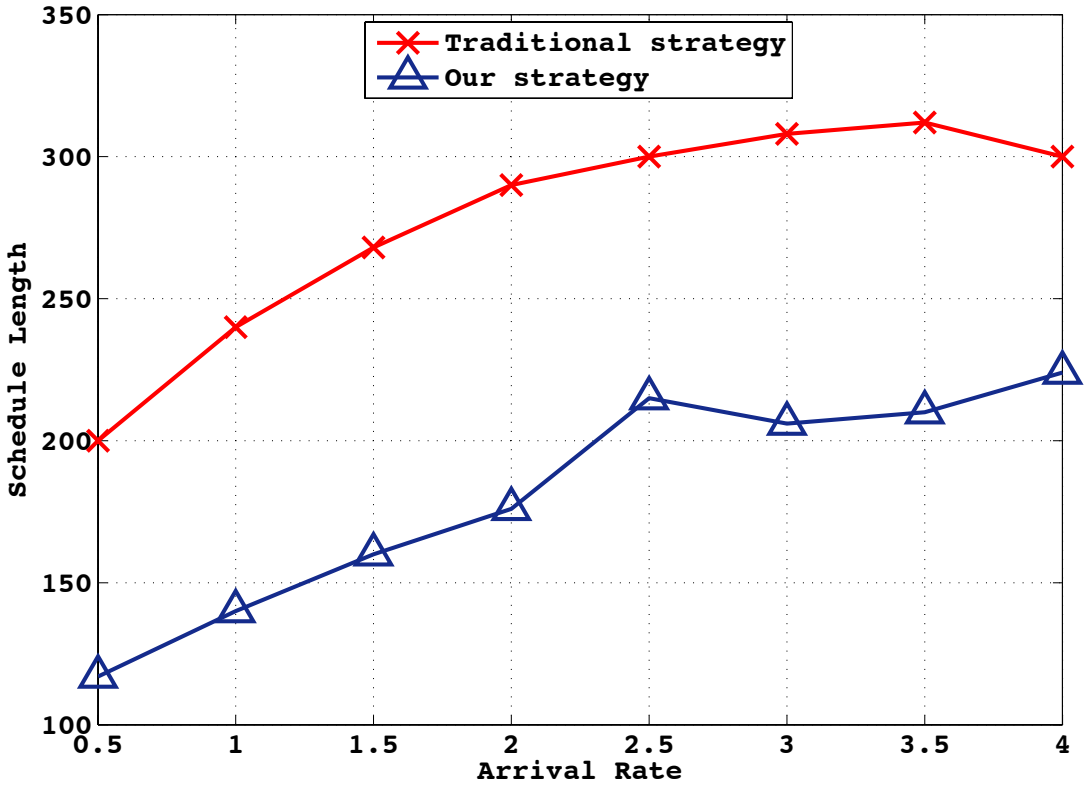


Figure 5.7 Comparing the schedule lengths of workflows achieved by our strategy and the traditional strategy; NUM=15, MAX_DG=10, EX_C=7, EX_H=7, RNUM=5, UNUM=15, MAX_U2R=5, MAX_RCST=4, CARD=4, TEMP=70%, TD=200, |HT|:|CT|=4:6, B=200, b_1, ..., b_{RNUM} = 10, 8, 2, 5, 9

Figure 5.8 compares the resource utilization achieved by our strategy and the traditional strategy. The depicted utilization is averaged over the entire system consisting of both human resources and computing resources. The figure shows that our strategy can achieve higher system utilization than the traditional strategy. The reason for this is similar as explained in Figure 5.6 and Figure 5.4.

The approach will be less beneficial when the execution times of the tasks are not known in advance because I assume that the execution times are known in advance.

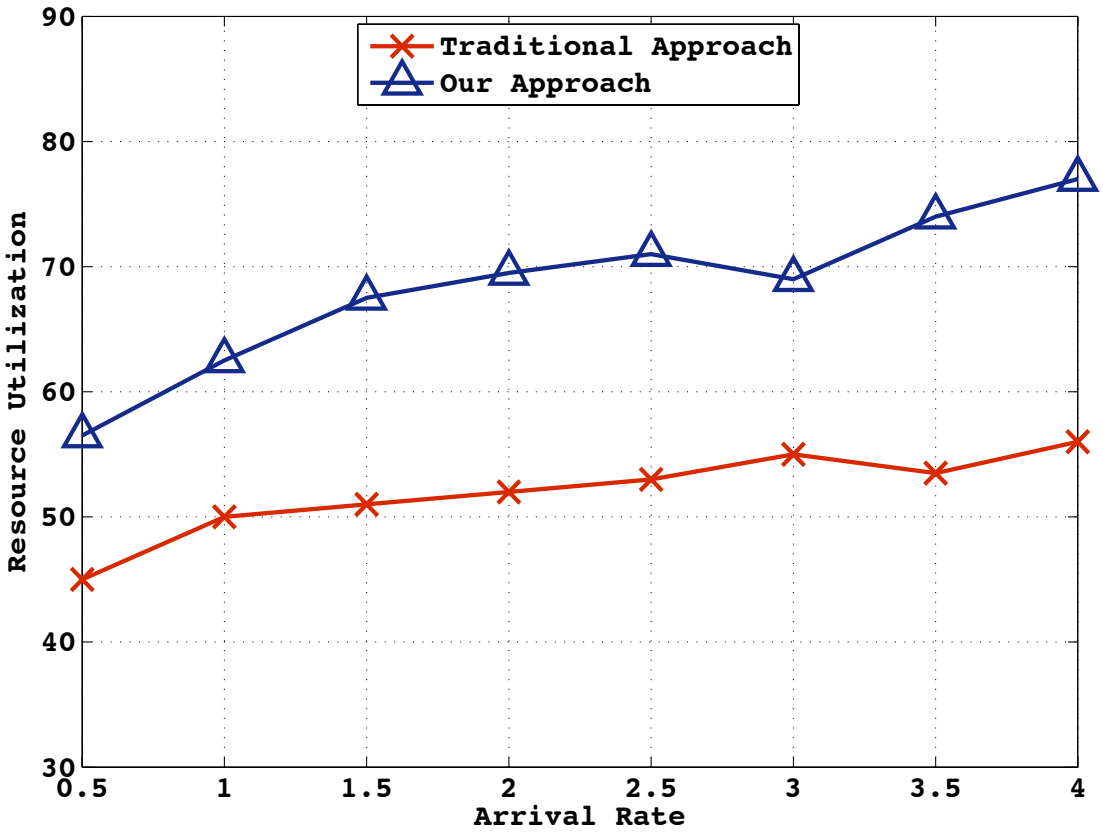


Figure 5.8 Comparing average resource utilization achieved by our strategy and the traditional strategy; the experimental settings are the same as in Figure 5.7.

5.5. Summary

This chapter investigates the issue of the allocation of the workflows running under the authorization control. The Chapter first calculates the rate of the requests arriving for each role under the role, temporal and

cardinality constraints. Further, this chapter present the methods to allocate the resource quantities for both human resources and computing resources. Different features of human resources and computing resources are taken into account. For human resources, the objective is to maximize the performance subject to the budgets to hire the resources, while for computing resources, the strategy aims to allocate adequate amount of computing resources to meet the QoS requirements. The simulation experiments have been conducted to compare the performance of the resource allocation strategies proposed in this chapter with a traditional strategy, which does not consider the authorization constraints and allocates the resource quantities proportional to the level of workload arriving at each service. The experimental results show that the proposed strategy is able improve the performance in terms of both schedule length and resource utilization.

6

Conclusions and Future Directions

6.1. Conclusions

The authorization control may be deployed in the workflow management systems in some application domain. However, the traditional workflow scheduling and resource allocation strategies do not take the authorization policies into account. This thesis investigates the impact of authorization policies on the execution performance of the workflows.

In Chapter 3, this thesis starts with investigating the issue of the feasibility checking for a set of authorization constraints deployed in workflow management systems. The feasibility-checking problem is modeled as a constraint satisfaction problem in this Chapter. The benefit of this modeling approach is that the solver for the constraints satisfaction problem can obtain all feasible authorization solutions. With knowing all feasible authorization solutions, the thesis further propose the method to determine the time durations when the workflow executions will not be impacted by the authorization constraints. A case study is given to illustrate the workings of the proposed methods.

In Chapter 4, this thesis proposes the method to analyze the delay caused by the authorization constraints if the workflow arrives beyond the non-impact time duration calculated in Chapter 3. Through the analysis of the delay, we realize that the authorization method, i.e., the method to select the authorization solution used to assign the roles to the tasks affects the length of the delay caused by the authorization constraints. Based on this finding, we propose an optimal authorization method, called the Global Authorization Aware (GAA) method. The GAA method is optimal in the sense that it can select the authorization method that minimizes the delay caused by the authorization constraints. We prove the optimality of the GAA method based on the delay analysis. We also conduct the simulation experiments to verify the effectiveness of this authorization method. The results show that compared with an intuitive authorization method, i.e., the Earlier Available First (EAF) method, the GAA method indeed greatly reduces the delay caused by the authorization constraints and the response time of the workflows.

A key reason why the authorization constraints may have impact on the execution performance is because the authorization constraints direct the incoming workload to different roles. Then the availability of the roles and the quantity of the resources allocated to each individual role will affect the execution performance of the workflows. In Chapter 5, we conduct the theoretical analysis about how the authorization constraints direct the workload to the roles. We propose the methods to calculate the arriving rate of the requests directed to each role under the role, temporal and cardinality constraints. Further, we present the methods to allocate

the appropriate quantity for both human resources and computing resources. Different features of human resources and computing resources are taken into account. For human resources, the objective is to maximize the performance subject to the budgets to hire the human resources, while for computing resources, the strategy aims to allocate adequate amount of computing resources to meet the QoS requirements. The simulation experiments are conducted to compare the performance of the resource allocation strategies proposed in this chapter with a traditional strategy, which does not consider the authorization constraints and allocates the resource quantities proportional to the level of workload arriving at each service. The experimental results show that the proposed strategy is able improve the performance in terms of both schedule length and resource utilization.

6.2. Future Work

This thesis conducted systematic studies about the impact of the authorization constraints on the execution performance of the workflows. However, the research work can be further extended in the following three folds.

First, when we calculate the delay caused by the authorization constraints, we assume that we know the exact value of the tasks' execution times. In real world, this assumption may not be always true. Therefore, we plan to conduct the further research in the following two aspects:

i) We plan to conduct the probability and statistical analysis about the delay caused by the authorization constraints, if the execution times of the tasks follow a certain probability distribution. For example, if the execution time follows the exponential distribution, then the time duration when the workflow executions will not be affected by the authorization constraints will not be a fixed value, but a random variable following certain probability attributes.

ii) The analysis of the delay caused by the authorization constraints requires knowing the prediction of the tasks' execution times. However, the prediction may not be exactly accurate. Therefore, we plan to study the impact of the inaccuracy of the prediction on the quality of the delay analysis.

Second, we propose the methods to allocate the appropriate quantity for human resources and computing resources. For human resources, the objective is to maximize the response time of the tasks subject to the resource budget, while the allocation strategy for computing resources aims to determine the adequate resources to meet the requirements in the tasks' response time. The response time of the tasks is the application-oriented performance metrics. There are also system-oriented performance metrics, such as resource utilization and system throughput. We plan to study the allocation strategies to maximize the performance or meet

the performance requirements in terms of the system-oriented metrics.

Finally, Petri-net is a popular approach to modelling the authorization constraints. Although the Petri-net modeling approach is heavy, it is especially useful if the system contains non-deterministic properties. For example, the resources may be dynamically added into or removed from the system, or the authorization control component may have dynamic interaction with the scheduling component in the workflow management system. Under such circumstances, we may still need to resort to the Petri-net modelling approach. A big problem of the Petri-net approach is that the constructed Petri-net is susceptible to the state explosion problem. Therefore, reducing the complexity of the Petri-net model will be very helpful. We plan to study whether and how the analysis method proposed in this thesis can simplify the Petri-net modelling approach.

BIBLIOGRAPHY

[Agarwal2009] A. Agarwal and Padam Kumar, "Economical Duplication Based Task Scheduling for Heterogeneous and Homogeneous Computing Systems", Proc. of 2009 IEEE International Advance Computing Conference (IACC'09), 2009.

[Agarwal2010] A. Agarwal, P. Kumar, Economical Task Scheduling Algorithm for Grid Computing Systems, Global Journal of Computer Science and Technology, Vol. 10 Issue 11 (Ver. 1.0) October 2010, 48-53.

[Ahmad1998] I. Ahmad and Y. K. Kwok, "On exploiting task duplication in parallel program scheduling," IEEE Trans. Parallel and Distributed Systems, 1998, vol. 9, no. 9, pp. 872-892.

[Ahn2000] G. Ahn, R. Sandhu, Role-based authorization constraints specification, ACM Transactions on Information and System Security 3 (4) (2000).

[Atluri1999] V. Atluri, and Wei-Kuang Huang. "A Petri net based safety analysis of workflow authorization models^ 1." Journal of Computer Security 8, no. 2 (2000): 209-240.

[Basu2003] P. Basu, W. Ke, and T. D. C. Little, Dynamic Task-Based Anycasting in Mobile Ad Hoc Networks. Mob. Netw. Appl., 8(5):593–612, 2003.

[Baxter1989] J. Baxter and Patel J H, “The LAST Algorithm: A Heuristic-Based Static Task Allocation Algorithm”, Int. Conf. on Parallel Processing, 1989, 2, pp. 217-222.

[Bertino2006] E. Bertino, J. Crampton, F. Paci, Access control and authorization constraints for ws-bpel, International Conference on Web Services (2006) 275–284.

[Bozdag2005] D. Bozdag, Fusun Ozguner, Eylem Ekici and Umit Catalyurek, “A Task Duplication Based Scheduling Algorithm Using Partial Schedules”, Proc. of the 2005 International Conference on Parallel Processing (ICPP’05), 630 – 637, August 2005.

[Brailsford1999] S. C. Brailsford, Chris N. Potts, and Barbara M. Smith. "Constraint satisfaction problems: Algorithms and applications. " European Journal of Operational Research 119, no. 3 (1999): 557-581.

[Brucker1995] P. Brucker, Scheduling Algorithms. Springer-Verlag, Inc., 1995.

[Chakraborty2007] D. Chakraborty, V. Mankar, and A. Nanavati, “Enabling runtime adaptation of workflows to external events in enterprise environments”, Proc. of IEEE International Conference on Web Services (ICWS’07), July 2007, pp.1112-1119.

[Chaudhary2013] N. Chaudhary and Ligang He. “Analyzing the Performance Impact of Authorization Constraints and Optimizing the Authorization Methods for Workflows”. Accepted in the 20th IEEE International Conference on High Performance Computing (HiPC2013).

[Choi2003] J. Y. Choi and S. Reveliotis, “A generalized stochastic petri net model for performance analysis and control of capacitated reentrant lines,” Robotics and Automation, IEEE Transactions on, vol. 19, no. 3, pp. 474 – 480, june 2003.

[Crampton2005] J. Crampton. "A reference monitor for workflow systems with constrained task execution." In Proceedings of the tenth ACM symposium on Access control models and technologies, pp. 38-47. ACM, 2005.

[Crampton2012] J. Crampton, M. Huth, On the modeling and verification of security-aware and process-aware information systems, Business Process Management Workshops 100 (6) (2012) 423–434.

[Cuervo2010] E. Cuervo, A. Balasubramanian, D. ki Cho, A. Wolman, S. Saroiu, R. Ch, and P. Bahl, “Maui: Making smartphones last longer with code offload,” in In In Proceedings of ACM MobiSys, 2010.

[Dagdee2011] N. Dagdee and R. Vijaywargiya. "Extending XACML to support Credential Based Hybrid Access Control." International Journal of Computer Science 8.

[Darbha1994] S. Darbha and Dharma P. Agrawal, “A Task Duplication Based Optimal Scheduling Algorithm For Variable Execution Time Tasks”, Proc. of the

International Conference on Parallel Processing (ICPP’94), 1994.

[Deelman2009] E. Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. "Workflows and e-Science: An overview of workflow system features and capabilities." Future Generation Computer Systems 25, no. 5 (2009): 528-540.

[Delias2011] P. Delias, Anastasios Doulamis, Nikolaos Doulamis, and Nikolaos Matsatsinis. "Optimizing resource conflicts in workflow management systems." Knowledge and Data Engineering, IEEE Transactions on 23, no. 3 (2011): 417-432.

[Dogan2004] A. Dogan and Fusun Ozguner, “LDBS: A Duplication Based Scheduling Algorithm for Heterogeneous Computing Systems”, Proc. of the International Conference on Parallel Processing (ICPP’02), 2004.

[Doulamis2011] A. Doulamis, N. Doulamis, and N. Matsatsinis, “Optimizing Resource Conflicts in Workflow Management Systems”, IEEE Trans. on Knowledge and Data Engineering, Vol. 23, Issue 3, 2011, pp. 417-432.

[El-Rewini1990] H. El-Rewini and T. G. Lewis, “Scheduling parallel programs onto arbitrary target machines, ” J. Parallel and Distributed Computing, vol. 9, no. 2, pp. 138-153, June 1990.

[Fang2007] D. Fang and Luo Junzhou, “A Heterogeneous Dynamic Critical Path and Duplication based Task Scheduling Algorithm for Pervasive Computing”, Proc. of

2nd International Conference on Pervasive Computing and Applications (ICPCA'07), 2007.

[Gaaloul2008] K. Gaaloul, A. Schaad, U. Flegel, F. Charoy, A secure task delegation model for workflows, in: The Second International Conference on Emerging Security Information, Systems and Technologies, 2008, pp. 10—15.

[Guodong2003] L. Guodong, Chen Daoxu, Wang Daming and Zhang Defu, "Task Clustering and Scheduling to Multiprocessors with Duplication", Proc. of International Parallel and Distributed Processing Symposium (IPDPS'03), 2003.

[Hara2009] T. Hara, T. Arai, Y. Shimomura, and T. Sakao, "Service cad system to integrate product and human activity for total value," CIRP Journal of Manufacturing Science and Technology, vol. 1, no. 4, pp. 262 — 271, 2009.

[He2005] L. He, S Jarvis, D. Spooner, D. Bacigalupo, G. Tan, G. Nudd, "Mapping DAG-based Applications to Multiclusters with Background Workload", Proceedings of the 5th IEEE International Symposium on Cluster Computing and the Grid (CCGrid'05), 9-12 May 2005, Cardiff, UK

[He2006a] L. He, S.A. Jarvis, D.P. Spooner, H. Jiang, D.N. Dillenberger, G. Nudd, Allocating non-real-time and soft real-time jobs in multiclusters, IEEE Transactions on Parallel and Distributed Systems 17 (2) (2006) 99—112.

[He2006] K. He and Yong Zhao, "A New Task Duplication Based Multitask Scheduling Method", Proc. of 5th

International Conference on Grid and Cooperative Computing (GCC'06), 2006.

[He2009] L. He, M. Calleja, M. Hayes, S.A. Jarvis, Performance prediction for running workflows under role-based authorization mechanisms, in: Proc. of the 2009 IEEE International Symposium on Parallel & Distributed Processing, IEEE Computer Society Press, 2009, pp. 1—8.

[He2011] L. He, K. Duan, X. Chen, D. Zou, Z. Han, A. Fadavinia, and S. Jarvis, "Modelling workflow executions under role-based authorisation control," in Services Computing (SCC), 2011 IEEE International Conference on, July 2011, pp. 200 —208.

[He2011a] L. He, D. Zou, Z. Zhang, K. Yang, H. Jin, and S. A. Jarvis, "Optimizing resource consumptions in clouds," in Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing, GRID '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 42—49.

[He2012] L. He, Chenlin Huang, Kewei Duan, Kenli Li, Hao Chen, Jianhua Sun and Stephen A. Jarvis, "Modeling and analyzing the impact of authorization on workflow executions", Future Generation Computer Systems, Vol. 28, Issue 8, October 2012, pp. 1177-1193.

[He2012a] L. He, Deqing Zou, Zhang Zhang, Chao Chen, Hai Jin and Stephen A. Jarvis, "Developing resource consolidation frameworks for moldable virtual machines in clouds", Future Generation Computer Systems (2012), doi: 10.1016/j.future.2012.05.015, Article in Press.

[He2012b] L. He, N. Chaudhary, S. Jarvis and K. Li, “Allocating Resources for Workflows Running under Authorization Control”, Proc. Of The 13th IEEE/ACM International Conference on Grid Computing (Grid 2012), 2012.

[He2013] L. He, N. Chaudhary and Stephen A. Jarvis “Developing Resource Allocation Strategies for workflows comprising both human and computing tasks” (Accepted and to appear in Future Generation Computer Systems).

[Hermenier2009] F. Hermenier, X. Lorca, J. Menaud, G. Muller, J. Lawall, “Entropy: a consolidation manager for clusters”, Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp. 41-50, 2009

[Hosseinzadeh2009] M. Hosseinzadeh and Hadi Shahriar Shahhoseini, “Earliest Starting and Finishing Time Duplication-based Algorithm”, Proc. of International Symposium on Performance Evaluation of Computer & Telecommunication Systems (SPECTS’09), 2009.

[Hsu2011] C. -C. Hsu, K. -C. Huang, and F. -J. Wang, “Online scheduling of work-flow applications in grid environments,” Future Generation Computer Systems, vol. 27, no. 6, pp. 860 — 870, 2011.

[Hung2003] P. Hung, and Kamalakar Karlapalem. "A secure workflow model." InProceedings of the Australasian information security workshop conference on ACSW frontiers 2003-Volume 21, pp. 33-41. Australian Computer Society, Inc., 2003.

[Jensen2007] K. Jensen, Lars Michael Kristensen, and Lisa Wells. "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems. "International Journal on Software Tools for Technology Transfer 9, no. 3-4 (2007): 213-254.

[Jin2003] Y. Jin, S. Reveliotis, A generalized stochastic petri net model for performance analysis and control of capacitated re-entrant lines, IEEE Transactions on Robotics and Automation 19 (3) (2003) 474–480.

[Joshi2005] J. Joshi, E. Bertino, U. Latif, A. Ghafoor, A generalized temporal role-based access control model, IEEE Transactions on Knowledge and Data Engineering 17 (1) (2005) 4–23.

[Kim2003a] S.H. Kim, J. Kim, S.J. Hong and S. Kim, “Workflow-based Authorization Service in Grid”, in 4th International Workshop on Grid Computing, Phoenix, USA, November 17, 2003, pp. 94–100.

[Kim2003] K. H. Kim, “Workflow Dependency Analysis and Its Implications on Distributed Workflow Systems”, Proc. of 17th International Conference on Advanced Information Networking and Applications (AINA’03).

[Kiyanclar2006] N. Kiyanclar, G. A. Koenig, and W. Yurcik. Maestro-VC: OnDemand Secure Cluster Computing Using Virtualization. In 7th LCI International Conference on Linux Clusters, 2006.

[Kleinrock1976] L. Kleinrock. "Queueing systems, volume II: computer applications." (1976).

[Kwok1996] Y.-K. Kwok and I. Ahmad. Dynamic critical path scheduling: An effective technique for allocating task graphs to multiprocessors. IEEE Transactions on Parallel and Distributed Systems, 7(5):506—521, May 1996.

[Kwok1999] Y.-K. Kwok and I. Ahmad. Benchmarking and comparison of the task graph scheduling algorithms. Journal of Parallel and Distributed Computing, 59(3):381—422, December 1999.

[Kwok1999a] Y.-K. Kwok and I. Ahmad. Static scheduling algorithms for allocating directed task graphs to multiprocessors. ACM Computing Surveys, 31(4):406—471, December 1999.

[Li2004] H. Li, Y. Yang, and T.Y. Chen, “Resource Constraints Analysis of Workflow Specifications,” J. Systems and Software, vol. 73, no. 2, pp. 271-285, 2004.

[Liu2001] Z. Liu, M. S. Squillante, and J. L. Wolf, “On maximizing service-level-agreement profits,” SIGMETRICS Perform. Eval. Rev., vol. 29, no. 3, pp. 43—44, Dec. 2001.

[Liu2008] A. X. Liu, F. Chen, J. Hwang & T. Xie (2008, June). Xengine: a fast and scalable XACML policy evaluation engine. In ACM SIGMETRICS Performance Evaluation Review (Vol. 36, No. 1, pp. 265-276). ACM.

[Lu2009] Y. Lu, L. Zhang, and J. Sun, “Using colored petri nets to model and analyze workflow with separation of duty constraints,” The International Journal of

Advanced Manufacturing Technology, vol. 40, pp. 179—192, 2009, 10.1007/s00170-007-1316-1.

[Manolache2002] S. Manolache, Schedulability Analysis of Real-Time Systems with Stochastic Task Execution Times, Ph.D. Thesis, Department of Computer and Information Science, IDA, Linkoping University, 2002.

[McCreary1989] C. McCreary and Gill H, “Automatic Determination of Grain Size for Efficient Parallel Processing”, Comm. Of ACM, 1989, 32(9), pp.1073-1078.

[Michael1996] P. A. Michael, Jing-Chiou Liou, and David S. L. Wei, “Task Clustering and Scheduling for Distributed Memory Parallel Architectures”, IEEE Trans. Parallel and Distributed Systems, 1996, 7(1), pp. 46-55.

[Muller2000] R. Muller and E. Rahm, Dealing with Logical Failures for Collaborating Workflows. In CoopIS, pages 210—223, 2000.

[N’takpe’2007] T. N’takpe’, F. Suter, A comparison of scheduling approaches for mixed-parallel applications on heterogeneous platforms, in: Proceedings of the 6th International Symposium on Parallel and Distributed Computing, IS-PDC, Hagenberg, Austria, July, 2007.

[N’takpe’2008] T. N’takpe’, F. Suter, Concurrent Scheduling of Parallel Task Graphs on Multi-Clusters Using Constrained Resource Allocations, Technical Report: Rapport de recherché no. 6774, December 2008.

[Park2001] C.-Ik Park and Tae-Young Choe, “An optimal Scheduling Algorithm based on Task Duplication”, Proc. of 8th International Conference on Parallel and Distributed Systems (ICPADS’01), 2001.

[Qiu2005] J. Qiu, C. Wang, and Y. He, “Research on Application of Intelligent Agents in the Workflow Management System,” Proc. 2005 IEEE Int’l Conf. Networking, Sensing and Control (ICNSC ’05), pp. 827-830, 2005.

[Qiu2013] M. Qiu, L. Zhang, Z. Ming, Z. Chen, X. Qin, and L.T. Yang, "Security-Aware Optimization for Ubiquitous Computing Systems with the SEAT Graph Approach," Journal of Computer and System Sciences, vol. 79, 2013.

[Ranaweera2000] S. Ranaweera and Dharma P. Agrawal, “A Task Duplication Based Scheduling Algorithm for Heterogeneous Systems”, Proc. of 14th International Parallel and Distributed Processing Symposium (IPDPS’00), 2000.

[Reijers2003] H.A. Reijers, “Resource Allocation in Workflows,” Design and Control of Workflow Processes: Business Process Management for the Service Industry, pp. 177-206, Springer, 2003.

[Rodriguez-Moreno2006] M.D. Rodriguez-Moreno, A. Oddi, D. Borrajo, and A. Cesta, “IPSS: A Hybrid Approach to Planning and Scheduling Integration,” IEEE Trans. Knowledge and Data Eng., vol. 18, no. 12, pp. 1681-1695, Dec. 2006.

[Rodriguez-Moreno2007] M.D. Rodriguez-Moreno, D. Borrajo, A. Cesta, and A. Oddi, “Integrating Planning and Scheduling in Workflow Domains,” Expert Systems with Applications, vol. 33, no. 2, pp. 389-406, 2007.

[Schall2010] D. Schall, S. Dustdar, M. Blake, Programming human and software-based web services, IEEE Computer 43 (7) (2010) 82–85.

[Senkul2002] P. Senkul, M. Kifer, and I. H. Toroslu, A Logical Framework for Scheduling Workflows Under Resource Allocation Constraints. In VLDB, pages 694–705, 2002.

[Stuit2011] M. Stuit, H. Wortmann, N. Szirbik, and J. Roodenburg, “Multiview interaction modelling of human collaboration processes: A business process study of head and neck cancer care in a dutch academic hospital,” J. of Biomedical Informatics, vol. 44, no. 6, pp. 1039–1055, Dec. 2011.

[Tarumi1997] H. Tarumi, K. Kida, Y. Ishiguro, K. Yoshifu, and T. Asakura, “Workweb System-Multi-Workflow Management with a MultiAgent System,” Supporting Group Work: the Integration Challenge, pp. 299-308, ACM Press, 1997

[van der Aalst2002] W.M.P. van der Aalst and K. van Hee, Workflow Management: Models, Methods, and Systems. MIT Press, 2002.

[van der Aalst2003] W.M.P. van der Aalst, A.H.M. ter Hofstede, B. Kiepuszewski, and A.P. Barros, “Workflow Patterns,” Distributed and Parallel Databases, vol. 14, no. 1, pp. 5-51, 2003.

[van Hee2005] K. van Hee, A. Serebrenik, N. Sidorova, and M. Voorhoeve, “Soundness of Resource-Constrained Workflow Nets,” Proc. Int’l Conf. Applications and Theory of Petri Nets 2005, pp. 250-267, 2005.

[VideoWorkflow] Video management workflow. http://www.telestream.net/pdfs/whitepapers/wp-video-workflowmanagement.pdf, 2010.

[W3Workflow] www.w3.org (workflow)

[Wang2010] Q. Wang and N. Li, “Satisfiability and resiliency in workflow authorization systems,” ACM Trans. Inf. Syst. Secur., vol. 13, no. 4, pp. 40:1–40:35, Dec. 2010.

[WebBusinessProcess] Web services business process execution language version 2.0. http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.pdf, 2007.

[WebHumanTask] Web services – human task (WS-HumanTask) specification version 1.1. http://docs.oasis-open.org/bpel4people/ws-humantask-1.1.html, 2010.

[WS-BPEL] WS-BPEL extension for people (BPEL4People) specification version 1.1. http://docs.oasisopen.org/bpel4people/bpel4people-1.1-spec-cd-06.pdf, 2009.

[Wu1990] M. Y. Wu and Gajski D D, “Hypertool: a Programming Aid for Message-Passing Systems”, IEEE Trans.

Parallel and Distributed Systems, 1990, 1(3), pp. 330-343.

[Xie2006] T. Xie and X. Qin, "Scheduling Security-Critical Real-Time Applications on Clusters," IEEE Transactions on Computers, vol. 55, no. 7, pp. 864-879, July 2006.

[Xie2008] T. Xie and X. Qin, "Security-Aware Resource Allocation for Real-Time Parallel Jobs on Homogeneous and Heterogeneous Clusters," IEEE Transactions on Parallel and Distributed Systems, vol. 19, no. 5, pp. 682-697, May 2008.

[Zhao2008] X. Zhao, Z. Qiu, C. Cai, and H. Yang, “A formal model of human workflow,” in Proceedings of the 2008 IEEE International Conference on Web Services, ICWS ’08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 195—202.

[Zhao2010] Q. Zhao, X. Liu, D. Sun, T. Liu, Y. Li, Mashing-up rich user interfaces for human interaction in WS-BPEL, in: The 2010 IEEE International Conference on Web Services, 2010, pp. 559—566.

[Zhong2005] J. Zhong and B. Song, “Verification of Resource Constraints for Concurrent Workflows,” Proc. IEEE Seventh Int’l Symp. Symbolic and Numeric Algorithms for Scientific Computing (SYNASC ’05), D. Zaharie, D. Petcu, V. Negru, T. Jebelean, G. Ciobanu, A. Cicortas, A. Abraham, and M. Paprzycki, eds., pp. 353-361, 2005.

[Zou2009] D. Zou, Ligang He, Hai Jin and Xueguang Chen, “CRBAC: Imposing multi-grained constraints on the RBAC model in the multi-application environment”, Journal of Network and Computer Applications 32 (2009), pp. 402-411.