# Performance Prediction for Running Workflows under Role-based Authorization Mechanisms

Ligang He, Mark Calleja[†], Mark Hayes[†], and Stephen A. Jarvis
*Department of Computer Science, University of Warwick, Coventry CV4 7AL, UK*
*[†]Cambridge eScience Centre, Centre for Mathematical Sciences, Cambridge CB3 0WA, UK*
*liganghe@dcs.warwick.ac.uk*

## Abstract

*When investigating the performance of running scientific/ commercial workflows in parallel and distributed systems, we often take into account only the resources allocated to the tasks constituting the workflow, assuming that computational resources will accept the tasks and execute them to completion once the processors are available. In reality, and in particular in Grid or e-business environments, security policies may be implemented in the individual organisations in which the computational resources reside. It is therefore expedient to have methods to calculate the performance of executing workflows under security policies. Authorisation control, which specifies who is allowed to perform which tasks when, is one of the most fundamental security considerations in distributed systems such as Grids. Role-Based Access Control (RBAC), under which the users are assigned to certain roles while the roles are associated with prescribed permissions, remains one of the most popular authorisation control mechanisms. This paper presents a mechanism to theoretically compute the performance of running scientific workflows under RBAC authorisation control. Various performance metrics are calculated, including both system-oriented metrics, (such as system utilisation, throughput and mean response time) and user-oriented metrics (such as mean response time of the workflows submitted by a particular client). With this work, if a client informs an organisation of the workflows they are going to submit, the organisation is able to predict the performance of these workflows running in its local computational resources (e.g. a high-performance cluster) enforced with RBAC authorisation control, and can also report client-oriented performance to each individual user.*

## 1 Introduction

Clusters and Grids are now popular platforms for processing large-scale scientific and commercial applications. Workflows typify such applications, consisting of multiple tasks with precedence constraints. When designing workflow management/scheduling strategies, or evaluate the performance of workflow execution on target resources, it is often assumed that when a task is allocated to a resource, the resource will accept the task and start the execution once the processor becomes available. In Grid and e-business/ commerce infrastructures, however, security mechanisms and policies may be deployed by individual participating organisations. When these security mechanisms are taken into account, the situation can become complex. For example, the user who submitted the task may not be authorised to use the target resource, or can only be authorised to do so by assuming certain roles, whose permissions are restricted according to the security policies implemented by the organisations.

A number of authorisation schemes have been presented in [1][15]. Currently, the RBAC (Role Based Access Control) scheme is one of most popular authorisation schemes. Under the RBAC scheme, the users are assigned to certain roles while the roles are associated with prescribed permissions. Therefore, the organisations can control the users' permissions through the roles. Furthermore, authorisation constraints can be imposed on the workflow itself. The following illustrate three authorisation constraints that are often encountered in workflow execution:

1) Task A in the workflow can only be run under Role 1 and 2 in the security mechanism;
2) If Task A is run under Role 1, then Task B must also run under Role 1;
3) If Task A is run under Role 1, then Task must not be run under Role 1.

The first constraint can be regarded as a static constraint, because the range of the roles that a particular task is allowed to assume is known prior to the execution of the workflow. The last two constraints are called *Binding of Duty* and *Separation of Duty*, respectively, and are considered two of the most important authorisation constraints in workflow execution [1][5][15]. Separation of Duty means that two different tasks must be run under

This work is supported in part by the UK Engineering and Physical Science Research Council (EPSRC) contract number EP/C538277/1.

different users/roles. To avoid conflicts of interests, for example, the task of utilising the resources (the actual computation) must not be run under the same user as the one who performs the accounting task (calculating how much resource has been consumed by the former, and how much should be paid by the associated user). Binding of Duty on the other hand, means that two tasks must be run under the same user/role.

The RBAC authorisation scheme has been further extended to incorporate temporal constraints [15]. For example, a particular role is only enabled from 9am to 6pm. If a task requesting this role arrives at some other time, it has either to wait until the role is enabled, or seek to run under other currently enabled roles if the security policy allows it to do so.

All these security considerations will have an impact on task and system performance (e.g. utilisation, mean response time, etc). This paper aims to model authorisation and the execution of workflows in cluster-based resources, and presents a mechanism to conduct theoretical calculations of the performance of the workflow execution in the system. Therefore, if the client informs the resource provider of the topology of the workflows they are preparing to submit, the arrival patterns of the workflows and the authorisation constraints imposed, the mechanism presented here can be employed to calculate/predict the performance of running these workflows.

Various performance metrics can be calculated, including those associated with system-oriented performance, such as utilisation, throughput of the cluster and, the overall mean response time of the workflows submitted to the system.

From the clients' perspective, they may be concerned with the mean response time of their particular tasks. Calculating such statistics is useful, since the clients can expect *a priori* notification of the execution performance of their workflows to the system and make decisions accordingly (e.g. whether to submit, increase the workload, or modify the settings).

In this paper, the Generalised Stochastic Petri-Net (GPSN) theory [8][18] is utilised to construct the models, which captures the authorisation constraints imposed by the resources/organisations and the workflows themselves. The models are also constructed to capture the workflow execution in a cluster environment. The authorisation models and the execution models are integrated to capture the workflow execution under authorisation constraints in cluster environments.

Furthermore, the authorisation constraints are analysed and resolved to guarantee that the markings representing completion of a workflow can be reached in the constructed GPSN model. Based on the above modelling and analysis of authorisation constraints and workflow execution, the standard GPSN analysis techniques and stochastic process theory can be employed to calculate the probability that the system stays in a certain state. After obtaining the probabilities of the states, we present methods to calculate the performance metrics we are interested in.

Although this work focuses on a single cluster environment, the results are also useful in multi-cluster/Grid environments; even if the execution of a workflow spans multiple clusters belonging to different organisations, the workflow usually has to be partitioned as multiple sub-workflows, each being run by a single cluster.

The remainder of this paper is organised as follows: related work is discussed in Section 2; Section 3 presents the Petri-net models to capture workflow execution under authorisation constraints in cluster-based environments; Section 4 presents the approach to calculating the performance in terms of utilisation, throughput and mean response time; the paper concludes in Section 6.

## 2 Related Work

Workflow management has been extensively studied and well documented in related literature [3][5][6][10][17]. Much of this research has aimed at enhancing the execution performance of workflows in parallel and distributed systems [6][10][18]. Some has also utilised Petri-nets to model workflow execution. However, this research does not take security requirements into account when designing execution strategies. Research has also been conducted to provide automatic workflow execution on distributed resources with minimal user intervention [2][4][12][16]. Some of this work has also considered security issues. For example, Grid Security Infrastructure (GSI) is utilised in [9][14] to provide secure authentication and communication in Grid environments. However, this work mainly focuses on enhancing the automata of workflow execution in distributed systems, and does not therefore investigate performance under security considerations.

Research has also been presented which imposes authorisation constraints on workflow execution [3][7][18][19]. Some of this research also uses Petri-nets to model authorisation constraints. The work presented in this paper differs from this work as follows:

First, the studies in [3][19] do not capture the temporal constraints of the roles' availability; in this work, the roles' temporal constraints are modelled.

Second, it is assumed in [3][7][13], that a task can only be run under one role (for example, the task of issuing checks can only be run by the role of a *Clerk*, while the task of approving checks must run under the role of a *Manager*). This assumption simplifies the modelling process and is not always true in distributed/Grid environments. It may well be the case that a task is allowed to run under a range of roles.

The relaxation of this assumption is especially necessary when the temporal constraints of a roles' availability is taken into account. In so doing, when the role that a task is requesting is not available, the task is given the opportunity to take on another currently accessible role and start execution once the resource (processor) is available. Without this the execution of the task, and therefore the execution of the entire workflow, would have to halt until the role is enabled. In this work, the task-role assignments and the user-role assignments are modelled in a flexible fashion. A task can be run under any selection of roles/users.

Third, the work presented in related literature focuses on modelling authorisation constraints [3][21]. Their models either miss the issue of resource saturation during workflow execution, or have an implicit assumption that the number of resources is unlimited. The work presented in this paper respects realistic scenarios in distributed computing and captures resource saturation (i.e. the resource is fully loaded), which may be caused either by different tasks in the same workflow (when these tasks can be run in parallel), or by different workflows entering the system.

Finally, much of the work presented in related literature only models the execution of a single workflow in the system at any one time [20][23]. In this paper, multiple workflows from different clients are permitted to enter the system. Moreover, the workflow authorisation and execution is modelled in a flexible fashion so that the performance of the workflows submitted by each client can be predicted.

## 3 Workflow Authorization/Execution Model

In this section, various authorisation constraints as well as task-role and role-user assignments are modelled using Petri-nets. This includes: the temporal constraints of roles; task-role assignment; role-user assignment; the binding-of-duty constraints for roles; the separation-of-duty constraints for roles; workflow execution and resource saturation;

### 3.1 Temporal Constraints

The temporal constraints of a roles' availability can be modelled using Petri-nets as in seen in Figure 1.

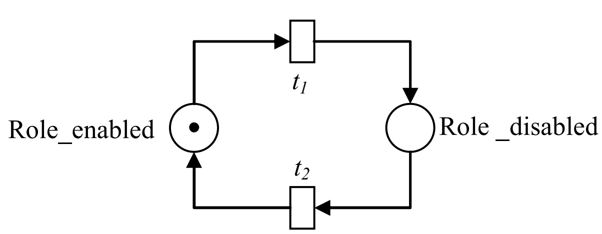


Fig. 1. Temporal constraints of the roles' availability

In this figure, $t_1$ and $t_2$ are timed transitions. In the current marking, the state *Role_enabled* contains a token, which means that the role is available. After a certain time period associated with transition $t_1$, $t_1$ fires and the token is deleted from the *Role_enabled* state and deposited into the state *Role_disabled*. Then, during the next time period associated with transition $t_2$, the role is unavailable. The time periods associated with transition $t_1$ and $t_2$ are variables, whose values are determined by the security policy implemented in an individual organisation. The work presented in [18] uses a Coxian distribution to approximate arbitrary distributions associated with the time variables of the timed transitions. A Coxian distribution of *s* stages is a weighted sum of convoluted exponential distributions and the approximation precision is controlled by *s*. Therefore, in this paper, we assume that the time variables associated with the timed transitions are all exponentially distributed. If there exist other distributions, the Coxian distribution can be used to perform the transformation so that all component timed transitions follow exponential distributions.

### 3.2 Task-role Assignment and task-user Assignment

Task-role assignment under authorisation constraints can be modelled as in Figure 2. In this figure, a task can assume any of a selection of roles from $R_1$ to $R_n$. Each role is associated with temporal constraints, which determine if the role is allowed to perform task-role assignment.

When the task is ready (for example, all its predecessors have been completed), a token will be deposited in the task state (this process will be discussed in the context of modelling workflow execution later in this section). Transitions $t_1$, …, $t_n$ are modelled as in conflict and therefore, at most one role will be assigned to the task at run time. When a token is deposited into the (task, role) state in the figure, the task-role assignment is established.

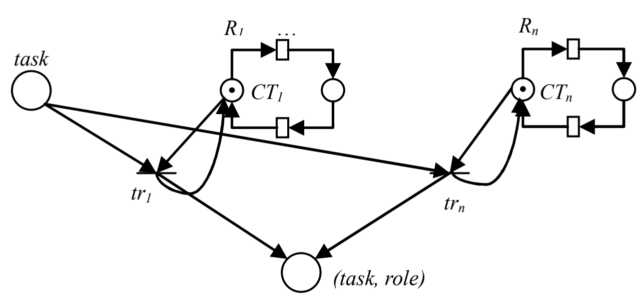


Fig. 2. Task-role assignment

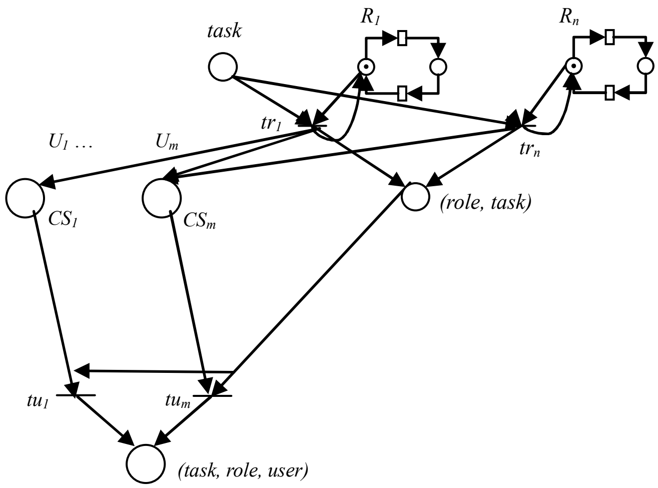


Fig. 3. User assignment to a (task, role) pair; in this example, role $R_1$ includes two users, $U_1$ and $U_m$, while role $R_n$ contains only one user $U_m$

Figure 3 models the assignment of users to a (*task, role*) pair (termed a task-user assignment for short). The top-right part of this figure is the task-role assignment, as shown in Figure 2. In this example, role $R_1$ includes two users $U_1$ and $U_m$, while role $R_n$ contains only one user $U_m$. Each user is associated with a state, *CS*, which determines if the user can proceed to role-user assignment. In the process of task-role assignment, if $t_{ri}$ fires, which means that role $R_i$ is assigned to the task, it will deposit a token to the *CS* state of every user belonging to $R_i$. Transitions $t_{u1}$, …, $t_{um}$ are in conflict and, therefore, at most one user will be assigned to the (task, role) pair. When a token is deposited into the *(task, role, user)* state, the assignment of a user to the (task, role) pair is established.

### 3.3 Binding-of-duty Constraints for Roles

Figure 4 models the binding-of-duty constraints for roles. The $CD_i$ state in the figure is used to enforce the binding-of-duty constraints. In this example, $task_1$ and $task_2$ must be run under the same role; and the role-user relations are the same as in Figure 3. In this figure, when role $R_i$ is assigned to $task_1$ (i.e. transition $tr_{1i}$ fires), $tr_{1i}$ will deposit a token into the *CD* state of $R_i$ for $task_2$ and the *CD* state of every user belonging to $R_i$ for $task_2$. In so doing, only $R_i$ can perform the task-role and role-user assignments for $task_2$. Therefore, the binding-of-duty constraint is satisfied. Please note that the temporal constraints of a role are modelled as being shared by all task-role assignments. This is reasonable because the roles' availability is a global parameter and should be applied to all tasks in the system.

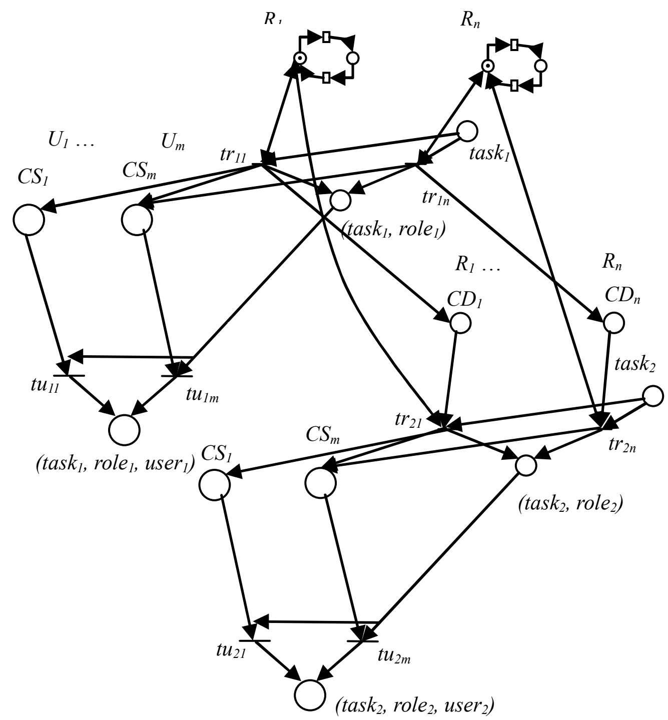


Fig.4. Binding of duty for roles; in this example, $task_1$ is the predecessor of $task_2$ and $task_1$ and $task_2$ must be run under the same role; as in Figure 3, user $U_1$ and $U_m$ belong to role $R_1$, while $U_m$ belongs to $R_n$

### 3.4 Separation of duty for roles

Figure 5 models the separation of duty for roles. In this figure, when role $R_i$ is assigned to $task_1$, $tr_{1i}$ will fire and deposit a token into the *CD* state of $R_j$ ($j \neq i$) for $task_2$ and also the *CD* states for all users belonging to $R_j$. According to this modelling, if $R_i$ is assigned to $task_1$, then the transition $tr_{2i}$ for the assignment of $task_2$ to $R_i$ will be disabled. Therefore, the separation of duty for roles is satisfied.

### 3.5 Modelling workflow execution under authorisation constraints

Given a workflow and the authorisation constraints among tasks, the individual components described in Figures 1 to 5 can be pieced together to obtain an *Authorisation Constraints Solver* (ACS). In this paper, multiple workflow instances (from multiple clients) are allowed to be authorised and executed in the system simultaneously. There will therefore exist different tokens generated by authorising the tasks from different workflow instances and these must be distinguished from one other. Failing this, when there exists a token in the *CD* state of a role/user for a task, it is not clear which workflow instance this token (and therefore, the corresponding dynamic constraint) is referring to.

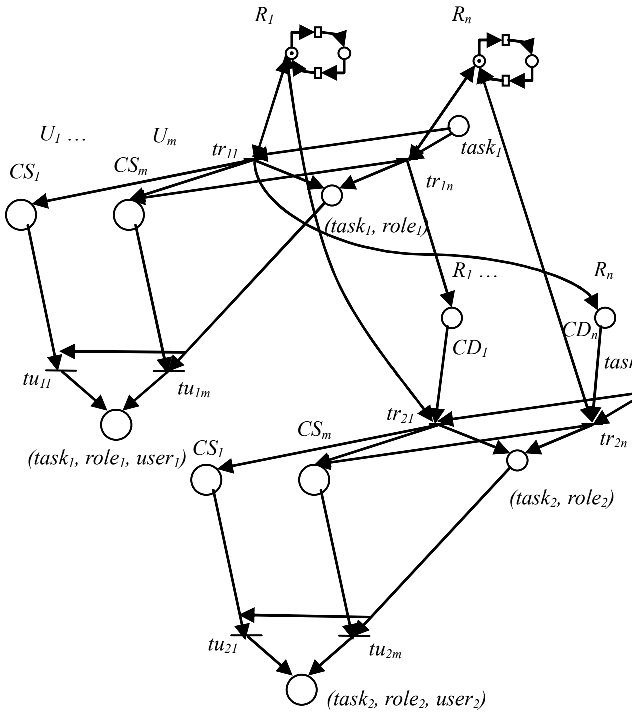


Fig.5. The separation-of-duty constraint for roles; $task_1$ and $task_2$ must not be run under the same role; the remaining settings are the same as those in Fig.4.

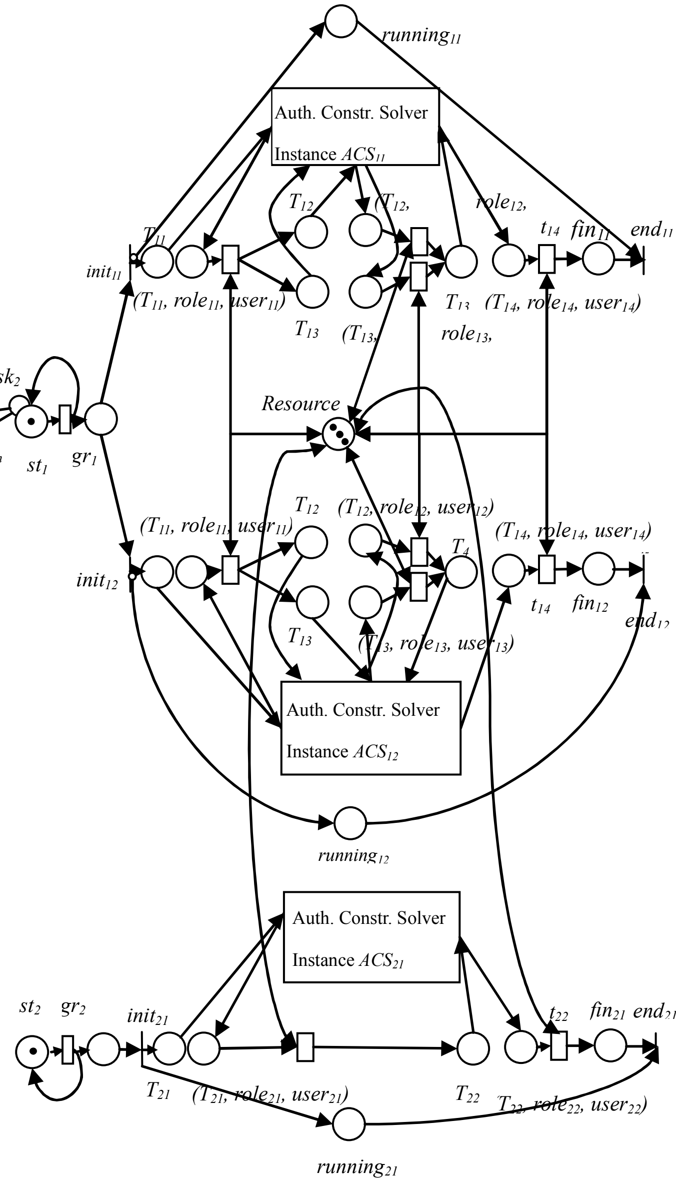


Fig.6c

Fig.6. Modelling workflow execution under authorisation constraints

Coloured Petri-nets are used in order to distinguish the tokens generated from authorising different workflow instances. However, when it comes to conducting the performance analysis, Coloured Petri-nets still need to be transformed to ordinary Petri-nets. In this paper therefore, Generalised Stochastic Petri-Nets are used to model and distinguish multiple workflow instances.

An ACS instance is created for an executing workflow instance. These have the same authorisation structure, and the temporal constraints of the roles are shared by the roles in all ACS instances.

A case study is provided below to illustrate the modelling of the workflow streams, submitted by different clients, under authorisation constraints in the cluster-based system.

We assume that two clients ($CL_1$ and $CL_2$) submit their individual workflow streams to the system. The topology of the workflow $WF_1$, submitted by client $CL_1$, is shown in Figure 6a, where the workflow consists of 4 tasks, while the topology of the workflow $WF_2$, submitted by client $CL_2$, is shown in Figure 6b.

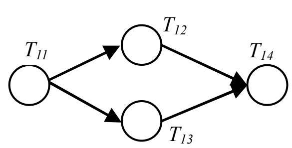


Fig.6a

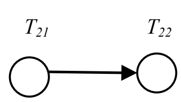


Fig.6b

Figure 6c models the execution of the two streams of workflow under authorisation constraints in a typical cluster-based system.

The individual authorisation components described in Figures 1-5 can be pieced together to obtain an ACS. For the sake of clarity, the ACS is only depicted as a black box in Figure 6c.

As can be seen from this figure, the model constructed in this paper presents a clear interface between the actual execution of the workflows and the ACS. When task $T_i$ is ready (i.e. it has no predecessor or its predecessors have been completed), a token is deposited into state $T_i$, which is the $task_i$ state in Figures 2-5, and it begins the authorisation process in the solver. After the authorisation is complete for $T_i$, a token is deposited into the ($T_i$, $role_i$, $user_i$) state, which corresponds to the (task, role, user) states in Figures 3-5. Only after this can the procedure of resource allocation start.

If there are tokens in the *resource* state (i.e. free processors), the execution of $T_i$ can begin, which means that the timed transitions are enabled and the associated timer begins to decrease. When $T_i$ is complete (i.e. the corresponding timed transition fires), a token is deposited back to the resource state (for the sake of clarity, this is represented by the two-headed arrows connecting the timed transitions with the resource state), which suggests the processor occupied by $T_i$ is released.

The workflow streams are generated by the states *st* and the timed transitions *gr*. The time associated with the *gr* transition follows an exponential distribution. This is used to simulate the arrival patterns of the workflow instances submitted by each client. It is assumed that for workflow $WF_i$, the maximum number of instances remaining in the system is $K_i$. In this case study, we assume that $K_1$ is 2 and $K_2$ is 1. Therefore, the number of the *Authorisation and Execution Instances* (*AEI* for short, including the ACS instance and workflow execution instance) generated for $WF_1$ and $WF_2$ are 2 and 1, respectively.

The *running* states in the figure are introduced to guide which AEI should be used to authorise and execute a newly generated workflow instance. When there is a token in the running state, it means that the corresponding workflow instance is still running. In the AEI instances for processing $WF_1$, there is an inhibitor arc connecting the running state to the *init* transition (for the sake of clarity, the inhibitor arc is merged with the output arc from the *init* transition to the running state). Therefore, when there is a token in the *running* state (which means that the corresponding workflow instance is still running), the newly generated workflow instance will not be dispatched to this instance of the ACS and workflow execution for authorisation and execution. Otherwise, a free choice of conflicts exists between this AEI instance and other free AEI instances. When the workflow instance is allocated to a workflow execution instance, a token is deposited into the state $T_1$ to start the process of authorisation and execution for the workflow instance; a token is also deposited into the running state so that no newly generated workflow instances can be admitted into this AEI instance until the running state is empty again. When a workflow instance is completed, a token is deposited into the *fin* state. The *end* transition fires immediately and removes the token from the running state, which means that the AEI instance can now accept a new workflow.

The procedure of authorisation and execution for $WF_2$ is similar, except that since there is only one AEI instance for $WF_2$, there is no inhibitor arc connecting the running state and the *init* transition.

# 4 Computing performance from the Petri-net models

Based on the constructed Petri-net models, this section describes the methods of calculating various performance metrics, including: system utilisation; system throughput; mean response time of all workflows submitted to the system and, the mean response time of the workflows submitted by each client

## 4.1 Constructing the Continuous Time Markov Chain (CTMC)

It has been extensively documented how to construct underlying stochastic process from a General Stochastic Petri-net model [8][18]. The model presented in this paper is a General Stochastic Petri-net model, and since the time associated with each transition is exponentially distributed, the underlying stochastic process is a Continuous Time Markov Chain [18].

There are two types of transitions in the GSPN: *timed transitions* and *immediate transitions*. When an immediate transition is enabled, it is fired immediately without delay. A marking in the GSPN is called tangible when there are no immediate transitions enabled. A tangible marking can be reached from another tangible marking by firing one timed transition and a (possibly empty) sequence of immediate transitions, until there are no more immediate transitions enabled. The Tangible Reachability Graph (TRG) is constructed as follows:

1. A node, denoted by $N_i$, in the TRG corresponds to a tangible marking, denoted by $M_i$, in the model;
2. There is an arc from node $N_i$ to $N_j$ in the TRG if $M_i$ can be changed to $M_j$ by firing an enabled timed transition $t_k$, (there may be a set of transitions that can trigger the change, denoted as $E_j(M_i)$); the weight of the arc equals the sum of the firing rates of all the transitions in $E_j(M_i)$, i.e., $\sum_{t_k \in E_j(M_i)} w_k$, where $w_k$ is the firing rate of $t_k$..

Since all timed transitions are exponentially distributed, the TRG graph is equivalent to a Continuous Time Markov Chain (CTMC).

The CTMC can be constructed from the TRG graph as follows:

1. Each state in the CTMC state space $S=\{s_i\}$ corresponds to a node $N_i$ in the TRG graph;
2. The transition rate from state $s_i$ (corresponding to the node $N_i$) to state $s_j$ (corresponding to $N_j$) is the weight of the arc from $N_i$ to $N_j$.

After the CTMC is constructed, we can calculate the infinitesimal generator (which is a two dimensional matrix, denoted by $Q$) of the CTMC. The element $q_{ij}$ in $Q$ can be determined by:

$$q_{ij} = \begin{cases} \sum_{t_k \in E_j(M_i)} w_k & i \neq j \\ -\sum_{t_k \in E(M_i)} w_k & i = j \end{cases} \quad (7)$$

Suppose there are $N$ states in the CTMC. $p_i$ is the probability that the CMTC stays in state $s_i$.

If we denote the row vector $P=[p_1, p_2, \ldots, p_i, \ldots, p_N]$, then the following linear equation system holds, where $Q$ is the infinitesimal generator of the constructed GPSN model. $P$ can be obtained by solving the equation system:

$$\begin{cases} PQ = 0 \\ \sum_{1 \le i \le N} p_i = 1 \end{cases} \quad (8)$$

## 4.2 Calculating performance

Based on the probability of the CTMC staying in each state, methods are presented to calculate the performance metrics of interest:

*1) Utilisation*

Assume in the initial marking that the number of tokens in the *resource* state is $V$, and denote $S_r(k)$ as the set of states in which the number of tokens in the *resource* state is $k$. The average number of tokens, *avgv*, in the *resource* state can be calculated as seen below. The calculation is correct because the number of tokens in the *resource* state can only be changed in the tangible markings since the *resource* state is only connected to the timed transitions in the model.

$$avgv = \sum_{k=0}^{V} k \sum_{s_i \in S_r(k)} p_i \quad (9)$$

After the task has completed, the processor will be reclaimed immediately. Therefore, the utilisation of the system, *ut*, equals

$$ut = 1 - \frac{avgv}{V} \quad (10)$$

*2) Throughput*

Let $S_{ij}(tr)$ denote the set of markings in which the last transition *tr* in the authorisation and execution instance $AEI_{ij}$ is enabled. In Figure 6c, *tr* in $AEI_{11}$ and $AEI_{12}$ is $t_{14}$ while in $AEI_{21}$, *tr* is $t_{21}$. For example, $S_{11}(tr)$ (or $S_{12}(tr)$) consists of the markings in which there exist the tokens in both the *resource* state and the ($T_{14}$ , $role_{14}$, $user_{14}$) state in $AEI_{11}$ (or $AEI_{12}$)).

Let the firing rate *tr* be $\mu$ (which is the execution time of task $T_{14}$ in the case of workflow $WF_1$ and the execution time of task $T_{22}$ in the case of workflow $WF_2$).

$th_{ij}$ denotes the throughput of the authorisation and execution instance $AEI_{ij}$. $th_{ij}$ can be calculated as follows:

$$th_{ij} = \mu \sum_{s_k \in S_{ij}(tr)} p_k \quad (11)$$

$th_i$ denotes the throughput of the system for processing workflow $WF_i$. $th_i$ can be calculated as follows:

$$th_i = \sum_{j=1}^{K_i} th_{ij} \quad (12)$$

*th* denotes the throughput of the entire system. th can be calculated as:

$$th = \sum_{i=1}^{CL} th_i \quad (13)$$

*3) Mean response time of the workflows*

In Figure 6c, only one workflow instance can enter an *AEI* instance at any time. Therefore, the mean response time of the workflow instances going through $AEI_{ij}$ is $1/th_{ij}$. $rt\_avg_i$ denotes the mean response time of the workflow $WF_i$ (that is, the workflows submitted by client $CL_i$). Then $rt\_avg_i$ can be calculated as follows, where $K_i$ is the number of *AEI* instances for workflow $WF_i$.

$$rt\_avg_i = \frac{(\sum_{j=1}^{K_i} \frac{1}{th_{ij}}) \times th_{ij}}{\sum_{j=1}^{K_i} th_{ij}} = \frac{K_i}{\sum_{j=1}^{K_i} (\mu \sum_{s_k \in S_{ij}(tr)} p_k)} \quad (14)$$

*rt_avg* denotes the mean response time of the workflows as a whole in the system. *rt_avg* can be calculated as follows, where *CL* is the number of clients submitting workflows to the system:

$$rt\_avg = \frac{\sum_{i=1}^{CL} (rt\_avg_i \times \sum_{j=1}^{K_i} th_{ij})}{\sum_{i=1}^{CL} \sum_{j=1}^{K_i} th_{ij}}$$

(15)

# 5 Conclusions

In this paper we employ Generalized Stochastic Petri Nets (GSPN) to model workflow execution under authorisation constraints in cluster-based systems. Various authorisation constraints are modelled in this paper, including temporal constraints of roles, separation of duty for roles and, binding of duty for roles. The model allows a task in the workflow to run under a selection of roles, and it also allows multiple workflows from different clients to be authorised and executed in the system simultaneously. The authorisation constraints are resolved so that any workflow admitted into the system will run to completion in the GSPN model. The GSPN is then transformed into a Continuous

Time Markov Chain (CTMC). In so doing, the probabilities of the states in the CTMC can be solved. Based on the state probabilities, methods are presented to theoretically calculate a number of performance metrics, including system utilisation, throughput, mean response time of all workflows submitted to the system, as well as the mean response time of the workflows submitted by each client.

## Reference


[1] G. Ahn and R. Sandhu, "Role-Based Authorization Constraints Specification," ACM Trans. Information and System Security, 3(4), 2000.

[2] R. Alfieri, R. Cecchini, V. Ciaschini, L. Agnello, A. Frohner, A. Gianoli, K. Lorentey and F. Spataro, "VOMS, an Authorization System for Virtual Organizations", in 1st European Across Grids Conference, Santiago de Compostela, Spain, February 13–14, 2003.

[3] V. Atluri, W. Huang, "A Petri net based safety analysis of workflow authorization models", Journal of Computer Security, 8(2/3): 209-240, 2000

[4] R. Baker, L. Gommans, A. McNab, M. Lorch, L. Ramakrishnan, K. Sankar and M. Thompson, "Conceptual Grid Authorization Framework and Classification", in 7th Global Grid Forum Workshop (GGF7), March, 2003

[5] R. Botha, and J. Eloff, "Separation of duties for access control enforcement in workflow environments", IBM Systems Journal, 40(3): 666–682, 2001

[6] J. Cao, S. A. Jarvis, S. Saini and G. R. Nudd. "GridFlow: Workflow Management for Grid Computing", Proc. 3rd IEEE/ACM Int. Symp. on Cluster Computing and the Grid, 198-205, 2003.

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

[8] C. J. Young and S. A. Reveliotis, "A generalized stochastic Petri net model for performance analysis and control of capacitated reentrant lines", IEEE Transactions on Robotics and Automation, 19(3): 474 - 480, 2003

[9] G. Della-Libera, P. Hallam-Baker, M. Hondo, T. Janczuk, et al. Web Services Security Policy Language (WS-SecurityPolicy), http://www-106.ibm.com/developerworks/library/ws-secpol/. 2002.

[10] L. He, S. A. Jarvis, D. P. Spooner, D. Bacigalupo, G. Tan, G. R. 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

[11] L. He, S. A. Jarvis, D. P. Spooner, H. Jiang, D. N. Dillenberger and G. R. Nudd, "Allocating Non-real-time and Soft Real-time Jobs in Multiclusters", IEEE Transactions on Parallel and Distributed Systems, 17(2):99-112, 2006

[12] S. Indrakanti and V. Varadharajan, "An Authorization Architecture for Web Services", 19th Annual IFIP WG 11.3 Working Conference on Data and Applications Security, pp. 222-236, 2005

[13] D. Ferrariolo, J. F. Barkley and D. R. Kuhn, "A Role-Based Access Control Model and Reference Implementation within a Corporate Intranet," ACM Trans. Information and System Security, vol. 2, no. 1, pp. 34-64, 1999.

[14] I. Foster, C. Kesselman, J. Nick and S. Tuecke, "The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration", in Open Grid Service Infrastructure WG (GGF), June, 2002.

[15] J. Joshi, E. Bertino, U. Latif and A. Ghafoor, "A Generalized Temporal Role-Based Access Control Model", IEEE Transactions on Knowledge and Data Engineering, 17(1): 4-23, 2005

[16] K. Keahey and V. Welch, "Fine-Grain Authorization for Resource Management in the Grid Environment", in 3rd International Workshop on Grid Computing, Baltimore, USA, November 18, 2002, pp. 199–206.

[17] 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.

[18] 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

[19] K. Tan, J. Crampton and C. Gunter, "The consistency of task-based authorization constraints in workflow systems", In Proceedings of 17th IEEE Computer Security Foundations Workshop, pp. 155–169, 2004

[20] J. Wainer, P. Barthelmess and A. Kumar, "W-RBAC – A workflow security model incorporating controlled overriding of constraints", International Journal of Cooperative Information Systems, 12(4), 455–486, 2003

[21] T. Ziebermayr and S. Probst. Web Service Authorization Framework. In International Conference on Web Services (ICWS), San Diego, CA, USA, 2004.

[22] W. Zhu and B. Fleisch, "Performance Evaluation of Soft Real-time Scheduling on a Multicomputer Cluster," Proceedings of the 20th International Conference on Distributed Computing Systems (ICDCS'2000), pp. 610-617, 2000.

[23] W. M. Zuberek, "Timed petri nets in modeling and analysis of cluster tools," IEEE Trans. on Robotics and Automation, vol. 17, pp. 562–575, 2001