http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 5 (5): 833-841, 2006
ISSN 1812-5638

Towards a Formal Model for Grid Architecture via Petri Nets

[1]Yuyue Du, [2]Changjun Jiang and [3]Yubin Guo
[1]College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China
[2]Department of Computer Science and Engineering, Tongji University, Shanghai 200092, China
[3]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510640, China

Abstract: Grid computing enables the virtualization of distributed computing and data resources to create a single system image, granting users and applications seamless access to vast IT capabilities. Grid architecture is a new software environment that developed to furnish users with grid computing platform that applications can access distributed resources through a well controlled, secure and mutually fair way. In this study a formal model based on Logical Petri Nets (LPNs) for grid architecture is proposed. Using this model, both static structure and dynamic behavior can be simulated and analyzed. The LPN model of grid architecture is constructed and causality relationship between the actions of grid components can be explicitly described. Based on this model, main properties of this grid structure is analyzed and verified formally.

Key words: Grid architecture, logical petri nets, modeling, correctness, verification, formal specification

INTRODUCTION

Grid computing enables the virtualization of distributed computing and data resources to create a single system image, granting users and applications seamless access to vast IT capabilities. Distributed computer systems are joined in a grid architecture (European Grid Forum, 2001; The Grid Forum, 2001), in which users can submit applications that are automatically assigned to and accomplished in suitable resources.

At present, considerable progress has been made on the concept and construction of grid architecture (Beiriger *et al*., 2000; The grid forum 2001). The Anatomy of the grid (Foster *et al*., 2001) presents a clear definition for grids in general. The real and specific problem that underlies the grid concept is coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations. However, a number of the approaches about grid settings are entirely informal and focus on how a grid system can be constructed. In this study, we try to introduce a formal model of what a grid system should provide, i.e., what functionalities are essential to creating a grid with no regard to actual components, protocols, any other details of implementation, or specific content of services.

The formal method used for modeling is Petri Nets (PN) that is a promising graphical and mathematical modeling tool (Murata, 1989). This method is superior to other formal technologies (Esteva *et al*., 2001; Ferraiolo *et al*., 1999) since both static structure and dynamic behaviors of grid architecture can be simulated. And the behavior of Grid architecture can be modeled and analyzed formally by reachability graph (Murata, 1989) and temporal logic analysis technologies (Du and Jiang, 2001). In order to apply Petri nets to the description and analysis of grid systems at conceptual level, we take Logical Petri Nets (Du and Jiang, 2002), a high level Petri Net, as a modeling tool.

The goal of this study is to provide a formal description and analysis approach for constructors and designers of grid architecture. According to this method, they can analyze and verify formally certain properties of the modeled grid architecture on the basis of petri net technologies such as correctness, liveness and robust properties. The outcome of our analysis is abstract declarative model. The model is declarative in the sense that it does not specify how to realize or decompose a given specific service functionality, such as computational resource services, cluster services and database services, but rather what it must provide in an entire grid environment. It also specifies formally how a user application is implemented and controlled in Grid architecture.

Corresponding Author: Yuyue Du, College of Information Science and Engineering, Shandong University of Science and Technology, Qingdao 266510, China Tel: 86-532-86057622

A GRID ARCHITECTURE

The Grid Computing (or Computational Grid) (Ernemann *et al*., 2002; Furmento *et al*., 2001; Wolski *et al*., 2001; Foster, 1998; Foster *et al*., 2001) is a software architecture for supporting high-performance distributed computing using resources that are chosen from multiple autonomous sites. Resources (such as machines, storage and network connections) are committed to a virtual, shared pool by their providers. The applications (consumers) under the control of Grid users draw the resources they need from the resource pool automatically. In defining a grid architecture, we should start from the perspective that the effective operations of the virtual organization require that we can establish sharing relationships among any potential participants.

We suppose that the grid computing architecture cannot force resource providers to remove or replace elements of their existing software base. That is to say source provider must possess ultimate control over their resources. The architecture must be accessible as a set of services layered on top of installed, extant software rather than as a replacement. The software platform is usually called middleware to describe its position between the application and the local system software.

The conceptual schematic of the grid architecture consists of three levels: user level, virtual level and physical level. User level consists of all users registered in the grid architecture. They first describe in detail application requirements by means of API, SDK and resource services provided by the virtual organization. Then the described applications are sent to the virtual organization where the applications are implemented under the control of grid protocols. Finally, they receive the results from the virtual organization and the payment of the sharing resources used is transferred to corresponding physical resource sites through an accounting system.

Physical level consists of all sharing (physical or logical) resources owned by the registered sites. Because the virtual organization complements rather than replaces existing institutions, sharing mechanisms cannot require substantial changes to local policies and must allow individual sharing resource sites to maintain ultimate control over their own resources. The sharing resources implement the local, resource-specific operations that occur on specific resources as a result of sharing operations at the virtual level. Thereby there is a tight and subtle interdependence between the functions implemented at the physical level or the sharing operations supported.

The main functionalities of this grid computing architecture are included in the middle tier, i.e., virtual level. In the virtual level, all sharing resources of the grid architecture are collected and managed based on grid protocols and user applications are implemented under its control. It provides dynamically the information about the structures and states of all physical resources in the physical level for the users of the user level, so its users can query in time for resources by name and/or by attributes such as type, availability or load. Virtual level also provides Application Programming Interfaces (APIs) and Software Development Kits (SDKs) to make its users develop easily an application program. Grid protocols implement interoperability between components. Virtual level contains the grid protocols required to control and monitor the behavior of the implement of user applications, such as communication and authentication protocols required for grid-specific network transactions, information protocols for obtaining information from the structure and state (e.g., its configuration, current load and usage policy) of a resource, management protocols for negotiating access to a shared resource, collective protocols for capturing interactions across collections of resources.

The grid architecture assumes a virtual pool of abstract resources rather than computational nodes (i.e., physical resource sites) at the Virtual Level. Although existing systems mostly focus on computational resources (e.g., CPU cycles, storage (Foster and Kesselman, 1999) that basically coincide with the notion of nodes, our grid architecture is expected to operate on a wider range of sharing resources like distributed file systems, computer cluster, network, software and so on. All these resources typically exist within sites that are geographically distributed and span multiple administrative domains at the physical level. The virtual machine implementing a user application is constituted of a set of resources taken from the virtual pool at the virtual level.

In the grid architecture, there exists a mapping from the abstract resources in the virtual pool to the physical resources at the physical level, when an application is implemented. Since the physical resources in sites can be added or withdrawn at any time according to their owner's discretion and their performance or load can change frequently over time, the virtual pool of abstract resources is dynamically changed with the addition or decrement of physical resources over time. Access to an abstract resource means that the user has some kind of credential accepted by the owner of the resource. A user may have the right to use a given resource. However it does not mean that he has login access to the site hosting the physical resource. Access to the site cannot be controlled in terms of login access by reason of a large number of resources in the virtual pool and the diversity of local security policies.

For the users registered in the grid architecture, the implementation processes of their applications under the control of grid protocols are shown in Fig. 1. The collective services are not associated with any one specific resource but rather are global in nature and focused on interactions across collections of abstract resources (i.e., the corresponding physical resources involved from multiple sites), while the resource services capture interactions with the abstract resources from a single site and map the abstract resources to corresponding physical resources in the site. The resource services are mainly used to negotiate access to a shared physical resource and to map the abstract resources involved in multiple sites to their corresponding physical resources through performing connective services in order to instantiate sharing relationships. The connective services implement core communication and authentication protocols required for grid-specific network transactions. They serve as enabling the exchange of data between the physical resources from different sites and providing cryptographically secure mechanisms for verifying the identity of users and physical resources.

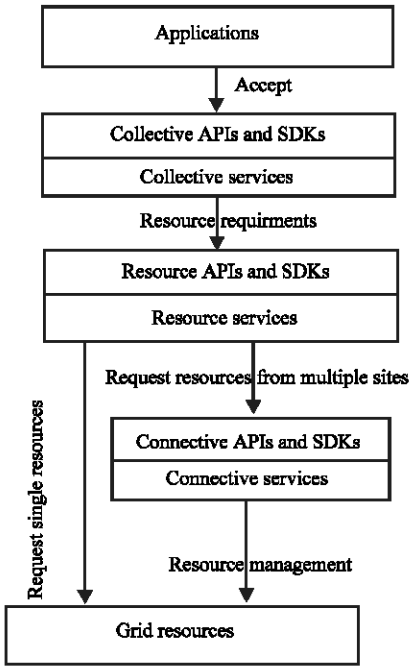

Fig. 1: The implementing procedures of user applications under the control of grid protocols in the grid architecture

LOGICAL PETRI NETS

To model the indeterminacy of data input and output, we will introduce below a type of transitions associated with logical expression constraints for their input or output, referred to as logical input or output transitions, respectively (Du and Jiang, 2002).

A logical input transition is enable only if the available tokens of its all input places satisfy a given logical expression and there exists no token in its all input places after it fires. The notation $f_I(t)$ ($f_O(t)$) is used to denote the logical expression of a logical input (output) transition t. Logical input or output transitions are represented graphically by the rectangles in which mark I or O is embedded, respectively. Figure 2 (a) means that transition (task) A can be enable only if there is at least one token in pace p1, or p2, or p3. Figure 2 (b) means that transition B can be enabled only if place p1 has at least one token and place p2 or p3 has at least one token. In fact, a logical input transition t can be enabled only if the value of its logical expression $f_I(t)$ is true.

The enabling conditions of logical output transitions are the same as ones of the transitions in classical petri nets. However, it is possible that the occurrence of a logical output transition will generate new data terms. The output places of a logical output transition must satisfy its logical expression and depends on the generated newly values of data items, after firing it. Thereby, the output of logical output transitions is non-determinate in static structures. Figure 3 (a) means that if task A start to be executed, there is one token in each of places p1, p2 and p3, or in one of them, or in two of them. In fact, one of previous cases is only determined based on the current generated values of data terms in a concrete modeled

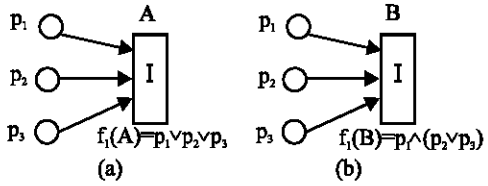

Fig. 2: PN representations of logical input transitions

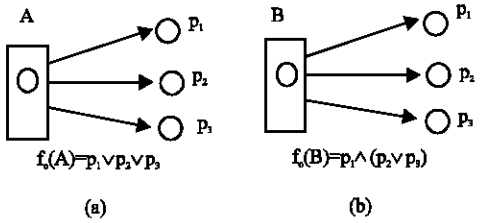

Fig. 3: PN representations of logical output transitions

system. Similarly, Fig. 3 (b) denotes that one token is generated in p1 and there is one token in one of places p2 and p3, or in each of them, after executing task B.

Therefore, in order to model the indeterminacy of data in grid architecture, we define the extended petri net as follows:

Definition 1: A Logical Petri Net (LPN) is a 9-tuple LPN=(P, TD, TI, TO, F, dt, I, O, M) where

P is a finite set of places.

TD is a finite set of the transitions associated with delay time dt.

TI is a finite set of the transitions that their input places are constrained by logical expressions $f_I(t)$.

TO is a finite set of the transitions that their output places are constrained by logical expressions $f_O(t)$. Sets P, TD, TI and TO are disjoined sets each other and their union set is non-empty.

$F \subseteq (P \times (TD \cup TI \cup TO)) \cup ((TD \cup TI \cup TO) \times P)$ is a finite set of arcs.

dt is a real function such that $\forall t \in TD$, $dt(t) \in R$, which denotes the delay time of the transition t.

I is a logical function such that $t \in TI$, I(t) is a logical input expression $f_I(t)$, while O is a logical function such that $\forall t \in TO$, O(t) is a logical output expression $f_O(t)$.

M: $P \rightarrow \{0,1,2,\ldots\}$ is a marking function, $\forall p \in P$, M(p) denotes the number of the tokens in p.

The transitions in LPNs are classified into three types: Delay time transitions, logical input and output transitions. The LPN representations of logical transitions Fig. 2 and 3.

Definition 2: Firing rules of the transitions in LPNs

$\forall t \in TD$, $dt(t)=\tau$, t is said to be enabled if $\forall p \in \bullet t$: $M(p) = 1$; t is said to be firable if its enabled time is equal to τ and firing t results in a new marking M': $\forall p \in \bullet t$: $M'(p)=M(p)-1$; $\forall p \in t\bullet$: $M'(p)=M(p)+1$.

$\forall t \in TI$, $I(t)=f_I(t)$, t is said to be enabled if $f_I(t)|_M = \cdot T \cdot$, i.e., all input places of t satisfy the logical input expression $f_I(t)$ at M; if t is enabled, it can fire and firing t results in a new marking M': $\forall p \in \bullet t$: $M'(p)=0$; $\forall p \in t\bullet$: $M'(p)=M(p)+1$

(3) $t \in TO$, $O(t)=f_O(t)$, t is said to be enabled if $\forall p \in \bullet t$, $M(p)=1$; if t is enabled, it can fire and its firing generates a new marking M': $\forall p \in \bullet t$: $M'(p)=M(p)-1$; for $t\bullet$, $f_O(t)|_{M'} = \cdot T \cdot$, i.e., all output places of t satisfy the logical output expression $f_O(t)$ at M'.

In Fig. 3 a, only one case satisfies the logical output expression $f_O(A) = p1 \vee p2 \vee p3$, based on the current generated values of the data terms in the input places of transition (task) A.

Delay time is relative time. A delay transition is said to be enabled only if each of its input places has an available token. An enabled delay transition may fire only after its delay time escapes.

LPN MODEL OF THE GRID ARCHITECTURE

For the grid architecture, we have discussed principally the functionalities of the virtual level and relationships between it and the other levels. The virtual organization consisting of the grid protocols at the virtual level manages, monitors and co-allocates shared resources and how an application is implemented by the virtual organization. In order to analyze and verify formally completeness, soundness and correctness of the grid architecture, in this section, we apply logical petri nets to its formal modeling and specification.

In the grid architecture, the users registered specify first in detail their specific requirements to program their applications in terms of the APIs and SDKs provided at the virtual level. Then user applications are implemented and managed by the virtual organization. The collection and co-allocation of the abstract resources are achieved at the virtual level and the virtual organization will map the abstract resources to corresponding physical resources. At the physical level, the owner of the shared physical resources in a site has the local control right for performing the authentication and communication of resources and users, when there is the exchange of data between the site and the other ones. But the implementation of this exchange is under the control of the virtual organization.

Figure 4 shows the LPN model of implementing a user application on grid architecture at the user level. $M_0(user\_req)=1$ denotes that the user has service requirements. An occurrence of transition reso_req checks the resource directory provided by the virtual organization in place reso_dir and generates real resource requirements in place req_resu. Firing transition construct will build an application in place application based on the APIs and SDKs. Then the user sends the application to the virtual organization by firing transition send_appl and waits to receive the results from the virtual organization through firing transition rece_resu. In Fig. 4, the double direction arc between a transition and a place is the abbreviation representation of two arcs that one is from the transition to the place and another is re-versed. This means that firing the transition does not change the number of tokens in the place. Gray places denote interface between different level. So do other figures in the rest of this study The result message is reposited

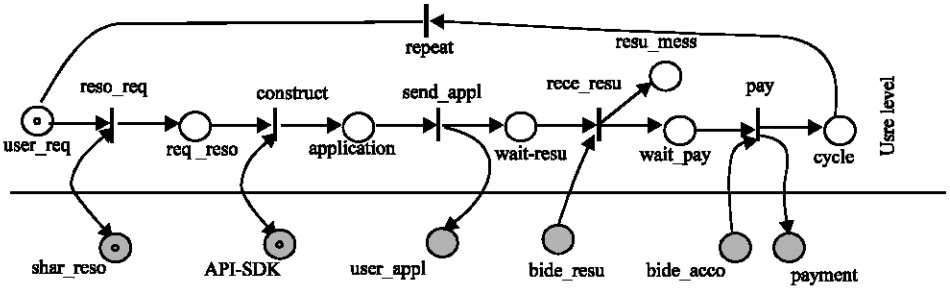

Fig. 4: LPN Model of process of applications at the user level

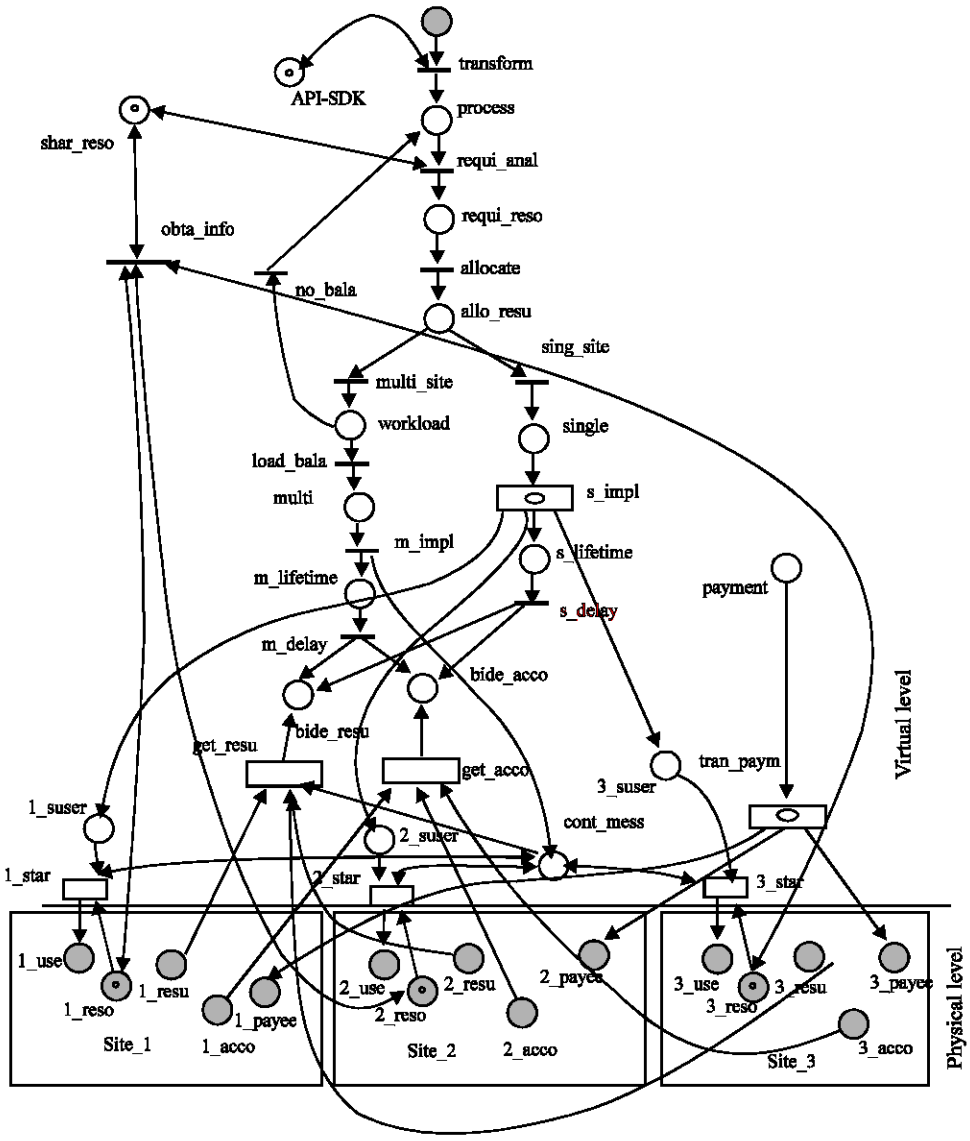

Fig. 5: The LPN model of virtual level of grid architecture

in place resu_mess. Finally, the service cost of the application is paid by occurrence if transition pay after get account information from virtual level. And the payment in place payment will be given to the corresponding service sites at the physical level via the virtual organization.

In Fig. 5, the LPN model of implementing a user application at the virtual level is presented. Some formulas in the graph areas are following.

f_O(s_impl)=s_lifetime/\ ((1_use/\ 1_acco) \/(2_use /\2_ acco)\/(3_use/\3_acco))

f_0(tran_paym)= 1_payee\/2_payee\/3_payee

f_I(get_resu)=bide_resu/\(1_use\/ 2_use\/3_use)

f_I(get_acco)=bide_acco/\(1_acco\/2_acco\/3_acco)

f_I(1-star)= 1_suser\/cont_mess

f_I(2-star) =2_suser\/cont_mess

f_I(2-star) =3_suser\/cont_mess

At the virtual level, the virtual organization performs the allocation and mapping of abstract resources and manages and monitors the running of the application by means of grid protocols. The directory of current shared resources is reposited in place shar_reso and is refreshed instantly once certain physical resources are added to and removed from the grid architecture by firing transition obta_info. The application submitted by the user may be transformed into several processes after transition transform fires. The requested resources of each process are analyzed and allocated by firing successively transitions requi_anal and allocate. If the allocated abstract resources are mapped into two or more sites, the workload of the sites is analyzed through firing transition multi_site. Once the workload is not balanced, the allocation of requested resources will be implemented again by firing transition no_bala. If the workload is balanced, the mapping plan that allocates corresponding physical resources for abstract resources will be created by firing transition m_impl. Then the application will be started on involved site by firing some transactions of 1_star, 2_star and 3_star. As to application involving only one site, the allocated resources are mapped into a single site after firing transitions singl_site and s_impl. One transition of 1_star, 2_star and 3_star can fire to start the application under the control of the virtual organization. When the lifetime of the application ends, delay transition s_delay or m_delay fires. The results and accounting of implementing the application are achieved by firing transitions get_resu and get_acco, respectively. Finally, the payment from the user is transferred to corresponding service sites after firing tran_paym.

For logical output transitions s_impl and tran_paym, their output is subjected to the constraint of logical expressions f_0(s_impl) and f_0(tran_paym), respectively, but which output places will generate tokens is indeterminacy and depends on the nature of the application. For example, if the physical resources only in Site_1 and Site_3 are used, each of places 1_payee and 3_payee has one token, after tran_paym fires, but place 2_payee has no token. And for logical input transitions

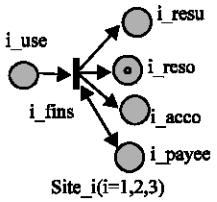

Fig. 6: LPN model of the physical level

1_star, 2_star and 3_star, firing constraint has been denoted by logical expressions f_I(1-star), f_I(2-star) , f_I(2-star), respectively. The intention of using there three logical input transitions is that a physical resource can be start for both applications that need single-site resources and applications that need multi-site resources.

From virtual level to each site in physical level, the interface including 5 places i_use, i_reso, i_resu, i_acco and i_payee with i=1,2,3. In the initial marking, one token has been placed in each of places 1_reso, 2_reso and 3_reso. This represents the sharing physical resources provided by the respective sites. If there is a token in place i_use, this means that respondent sharing resources are being used. If there is one token in place 1_acco, it denotes that the respondent site is accounting for its services. Place 1_payee is used to reposit the charge paid by the user. And the token in place i_resu means the result in this site is ready.

At the physical level, each site has local control right on the authentication and communication between three sites. After sharing resources having been started by virtual organization, the executing process of application is supervised by physical level. When the execution terminates results and accounting information can be produced and put into according places by firing transition i_fins. This i_fins will consume a token in place i_use and give out a token in place i_reso also that means transferring the status of sharing resources from busy into ready.

Therefore, the static and dynamic behavior of three level of the grid architecture can be extremely modeled in Fig. 4-6. Readers who want to get a whole impression of the grid architecture can put there three figures together by composing corresponding interface places. For a user application, an entire processing procedure from its submission and implementation to the acceptation of results can be represented definitely.

PROPERTIES ANALYSIS OF THE GRID ARCHITECTURE

The application of the existing analysis techniques of petri nets to the verification of some main properties, such as correctness and completeness, according to the logical petri net model of the grid architecture Fig. 4-6.

In present study, we analyze and verify formally certain dynamic properties of the grid architecture, in the following and present the corresponding formal description based on the petri net language. Let the initial marking M_0 of the logical petri net model in Fig. 4 be M_0(user_req)=M_0(shar_reso)=M_0(API_SDK) and for any place p∉{user_req, shar_reso, API_SDK}:M_0(p)=0. In Fig. 5, M_0(shar_reso) =M_0(API_SDK)=1 and M_0(p)=0 for all other places p. In Fig 6, M_0(1_reso)= M_0(2_reso)=M_0(3_reso)=1 and M_0(p)=0 for other places p.

Lemma 1: (If a user has service requests, the virtual organization will eventually allocate some abstract resources to it.) For any initial marking M_0 and M_0(user_req)=1, there is a firing sequence α and a marking reachable from M_0 such that M_0(α>M and M(multiple)=1 ∨ M(single)=1.

Proof: For any marking M reachable from the initial marking M_0, we have M(shar_reso)=M(API_SDK)= M(1_reso)=M(2_reso)= M(3_reso)=1, according to the suppositions and the net structure in Fig. 4. Thus, if M_0(user_req)=1, then transition reso_req is firable and M_0(reso_req>M_1∧ M_1(req_reso)=1. Here transition construct can fire, i.e., M_1(construct>M_2∧M_2 (application)=1. Similarly, M_2(send_appl>M_3∧M_3 (wait_resu)=1∧M_3(user_appl)=1.

At that time, in the corresponding marking M_3' in Fig. 5 M_3'(user_appl)=M '$_3$ (API_SDK)= 1. Thus M_3' (tranform>M_4 ∧ M_4(process)=1, M_4(requi_anal)>M_5∧M_5 (reque_reso)=1, M_5(allocate> M_6∧M_6(allo_resu)=1. If transition sing_site is firable at M_6, this means that the application requires only the shared resources on a single site. If transition multi_site is firable at M_6, it implies that the application wants the virtual organization to provide the services from several sites. Thereby, M_6(multi_site>M_7∧M_7(workload) =1, or M_6(sing_site>M_8 ∧M_8(single)=1. If transition no_bala is firable at M_7, this denotes that the workload assigned on each site involved is not balanced and the processes decomposed from the application demand to be re-allocated to sites. That is, M_7(no_bala>M_4(requi_anal >M_5(allocate> M_6(muti_site>M_7. But the procedure is eventually end, i.e. M_7(load_bala>M_9∧M_9(multiple)=1.

Therefore, there exists either a firing sequence α_1 at M_0 and

α_1=reso_req, construct, send_appl, transform, requi_anal, allocate, sing_site

such that M_0(α1>M8∧ M (single)=1, or a firing sequence α_2 at M_0 and

α_2= reso_req, construct, send_appl, transform, requi_anal, allocate, (requi_anal, allocate, no_bala)*, multi_site, load_bala.

Here* is Kleen closure of expression in the brackets that means transition sequence (equi_anal, allocate, no_bala) can fire 0 or any positive integer times.
such that M_0(α$_2$>M_9∧M_9(multiple)=1.

To save the space of this paper, the proofs of the following lemmas will be omitted. The proofs of the lemmas are similar to that of Lemma 1. And in the following, we built the whole LPN model for grid architecture from Fig. 4-6 by combining places that with the same names. And then the initial marking M_0 is defined as:
M_0 (user_req) = M_0(shar_reso) = M_0 (API_SDK)= M_0 (1_res o) =M_0(2_reso)=M_0(3_reso)=1 and M_0(p)=0 to all other places. Using this composed model all lemmas and theorems can be delivered in a much simple way.

Lemma 2: (Any application is eventually implemented at the physical level.) For any initial marking M_0 and M_0(application)=1 in Fig. 4, there is a firing sequence α and a marking M reachable from M_0 such that M_0(α>M and the corresponding marking in Figure 5 M′ is defined as M′(user_appl)= M(user_appl)∧M′(share_reso)= M (share_reso) ∧M′(API_SDK)= M (API_SDK), M′ (1_reso)= M′(2_reso)=M′(3_reso)=1 and there are no token in other place. Under marking M′ is a firing sequence α′ such that M′ (α′>M″ with (1_use)=1∨ M(2_use)=1∨ M(3_use) =1.

Lemma 3: (When the lifetime of any application is over, the user submitting the application will obtain the results.) Let M_1 be a marking reachable from M_0 and M_1(m_lifetime)=1∨M_1(s_lifetime)=1. Then there is a firing sequence α at M_1 and a marking M_2 reachable from M_1 such that M_1(α>M_2 and M_2(resu_mess)=1.

Lemma 4: (If a user receives a result sent by the virtual organization, the sites providing shared resources for the user must get the service payment.) For any marking M_1 reachable from M_0 and M_1(*resu_mess*)=1, there exists a firing sequence α at M_1 and a marking M_2 reachable from M_1 such that M_1(α>M_2 and M_2(*1_payee*)=1∨ M_2(2_payee)=1∨ M_2(3_payee)=1.

By means of the earlier lemmas, we can obtain easily and directly the following conclusions.

Theorem 1: (If any user registered in the grid architecture has service requirements, the user will eventually obtain service results and the sites providing shared resources for him will eventually get the payment.) For any marking M_0 and M_0 (user_req)=1, there are firing sequences α_1 and α_2 and the marking M_1 and M_2 reachable from M_0 such that M_0 (α_1> M_1∧ M_1(resu_mess)=1 and M_1(α_2>M_2∧ (M_2(1_payee)=1∨ M_2(2_payee)=1∨ M_2(3_payee)=1).

Theorem 2: (If the implementation of a user application demands to use the sharing resources from several sites, then the communication between different sites is managed and monitored all the time by the virtual organization within the lifetime of the application). For any marking M_1 reached from M_0 and M_1(multiple)=1, there is a firing sequence α and a marking M_2 reachable from M_1 such that $M_1(\alpha>M_2$ and M_2 (m_lifetime)=1$\wedge$ M_2(cont_mess)=1.

Firing m_impl must generate one token in m_lifetime and one token in cont_mess. Furthermore, if cont_mess is nonempty, two or all of logical input transitions 1_star, 2_star and 3_star are enabled. This means that the sites where corresponding transitions i_star, i=1,2,3, are enabled may communicate under the control of the virtual organization. Thereby, Theorem 2 can be proved.

CONCLUSIONS

Grid computing provides a ubiquitous and seamless computational power available from geographically distributed, heterogeneous resources. Thus we can develop and build a new software environment, grid architecture, on the basis of Grid Computing. The present study, we construct a grid architecture consisting of three levels: user level, virtual level and physical level. The core part of the grid architecture is at the virtual level. It realizes the allocation of abstract resources and a mapping from abstract resources to the physical resources at the Physical Level. It manages and monitors the implementation of any user application and the communication between the sharing resources from different sites.

The purpose of this study is to present a formal modeling and analysis approach for verifying the correctness and completeness of grid systems. For achieving this goal, we adopt logical Petri nets as a modeling tool of grid settings. First, we construct. Grid architecture according to existing grid environments and introduce the concept of virtual organizations in the architecture. We then apply logical Petri nets to the modeling of the grid setting. Finally, some significant properties of the grid architecture are described and analyzed formally based on its LPN model. The results obtained in this study demonstrate that the LPN model of the grid architecture can specify and model extremely its static and dynamic behavior. Since our approach is of graphical specification power, it is easily understood by engineers and designers and the causality between the actions in the grid system can also clearly expressed.

Further work will be to apply high-level petri nets (Jensen, 1996) and temporal logic (Du and Jiang, 2002) to the modeling and verification of grid architecture. The high-level petri net model can describe explicitly the individuality of user requirements. Thereby, we can analyze and verify more finely the properties of a grid system. Also, we will develop a formal tool for modeling and analyzing automatically grid architecture.

ACKNOWLEDGMENTS

This study was supported by the National Natural Science Foundation of China under Grants 60573018, 60125205, 90412013 and 60473094; Taishan Scholar Construction Project of Shandong Province, China; the National Basic Research Program of China (973 Program) under Grants 2003CB316902 and 2004CB318001-03; Shandong Natural Science Foundation of China under Grant 2004ZX17; the Open Project of Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences under Grants SYSKF0309 and SYSKF0604; and Shanghai Science and Technology Research Plan.

REFERENCES

Beiriger, J., H.P. Biven, S.L. Humphreys, W.R. Johnson and R.E. Rhea, 2000. Constructing the ASCI computational grid. In Proc. 9th IEEE Symp. on High Performance Distributed Computing, pp: 193-199.

Du, Y.Y. and C.J. Jiang, 2001. Formal analysis of an online stock trading system by temporal petri nets. In: Proc. Int. Workshop on Computer Networks and Mobile Computing, IEEE Computer Society Press, Beijing, China, pp: 197-202.

Du, Y.Y. and C.J. Jiang, 2002. Formal representation and analysis of batch stock trading systems by logical Petri nets workflows. Lecture Notes in Computer Science LNCS 2495, pp: 221-225.

Ernemann, C., V. Hamscher, U. Schwiegelshohn and R. Yahyapour, 2002. On advantages of grid computing for parallel job scheduling. In: Proceedings of 2nd IEEE International Symposium on Cluster Computing and the Grid (CC-GRID 2002), Berlin, Germany, pp: 39-46.

Esteva, M., J.A. Rodriguz-Aguilar, C. Sierra, P. Garcia and J.L. Arcos, 2001. On the formal specification of electronic institutions. Lecture Notes in Computer Science LNCS 1991, pp: 126-147.

European Grid Forum, 2001. http://www.egrid.org.

Ferraiolo, D.F., J.F. Barkley and D.R. Kuhu, 1999. A role based access control model and reference implementation within a corporate intranet. ACM Transactions on Information System Security, pp: 1.

Foster, I. and C. Kesselman, 1998. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Pubishers. Inc.

Foster, I. and C. Kesselman 1999. The Globus Toolkit. Morgan Kaufmann Publishers, pp: 259-278.

Foster, I., C. Kesselman and S. Tuecke, 2001. The anatomy of the grid: Enabling scable virtual organizations. Intl. J. High Perfor. Comp. Appl., 15: 200-222.

Furmento, N., A. Mayer, S. McGough, S. Newhouse, T. Field and J. Darlington, 2001. An integrated grid environment for component application. In: 2nd International Workshop on Grid Computing, Grid 2001, Lecture Notes in Comp. Sci. LNCS 2242, pp: 26-37.

Jensen, K., 1996. Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use. EATCS Monographs on Theoretical Computer Science. Springer-Verlag, Berline.

Murata, T., 1989. Petri nets: properties, analysis and applications. Proceedings of the IEEE, 77: 541-580.

The Grid Forum, 2001. http://www.gridforum.org

Wolski, R., J.S. Plank, J. Brevik and T. Bryan 2001. Analyzing market-based resource allocation strategies for the computational grid. The Intl. J. High Perfor. Comp. Appl., 15: 258-281.