

A Multi-Agent Architecture for QoS Support in Grid Environment

Ali Rezaee, Amir Masoud Rahmani, Saeed Parsa, Sahar Adabi
Department of Computer Engineering, Science and Research Branch,
Islamic Azad University, Tehran, Iran

Abstract: Grid computing is emerged as a new distributed computing technology that implements flexible and coordinated resource sharing among dynamic, heterogeneous, unpredictable and geographically distributed collections of resources owned by different individuals and organizations. Due to heterogeneous and dynamic nature of the grid, resource management and application scheduling is a complex undertaking. This paper proposed a multi-agent architecture that addressed resource management and application execution with support for Quality of Services (QoS) in grid environment. Five types of collaborative intelligent and mobile agents proposed to manage grid resources and applications in a decentralized, autonomous and intelligent manner. In the proposed architecture, negotiation, advanced reservation and QoS measurements handled by collaborative intelligent agents. Resources and applications are managed independently with respect to their defined policies. The simulation results showed that the multi-agent architecture is practical, flexible and effective.

Keywords: Application agent, grid architecture, intelligent agents, multi-agent, QoS, service agent

INTRODUCTION

Grid technology enables the sharing and dynamic allocation of distributed, high-performance computational resources while minimizing the associated ownership and operating costs, it also facilitates access to such resources and promotes flexibility and collaboration among diverse organizations^[4]. The resources are heterogeneous in terms of their architecture, power, configuration and availability. They are managed by different access policies and cost models that vary with time, users and priorities. Different applications have different computational models that vary with the nature of the problem. The resource owners and end-users have different goals, objectives, strategies and demand patterns^[3]. Due to large-scale heterogeneity present in resources and applications requirements in grid environments, resource management and application scheduling are complex undertaking^[2,6]. In respond to this heterogeneity, we proposed a new Multi-Agent Architecture for Grid Environment (MAAG), which fits in heterogeneous nature of grid's resources and applications. Multi-Agent System (MAS) is a distributed collaborative environment which allows a number of agents to cooperate and interact with each other in a complex environment^[10]. In a typical multi-

agent system, the agents work together to achieve a global objective in a distributed manner. In most cases, the interaction is asynchronous and decentralized. These characteristic features of MAS make them ideal for spontaneous and opportunistic collaborations using autonomous agents^[8,10] which continuously adapt to their environment.

The proposed architecture MAAG employs five types of collaborative agents to integrate and coordinate distributed resources in a computational grid environment and to support distributed and hybrid workflows with the aim to deliver expected QoS to applications. In the MAAG, each service is managed independently by an intelligent agent. The agent manages requests, negotiates with service consumers, performs advanced reservation^[4-6] and measures QoS metrics of all instances of a grid service in a specific grid site. Also each grid application has an intelligent agent who is responsible for application execution respecting to the QoS constraints defined by its owner. Thin mobile negotiator/watcher agents created and used by the application agent to negotiate with service agents in grid sites and to perform advanced reservation and other application's related task such as performance monitoring. The agents cooperate with each other to handle monitoring and management of the grid services and applications autonomously with aim to deliver user expected Quality of Services.

Corresponding Author: Ali Rezaee, Department of Computer Engineering, Azad University, Science and Research Branch, Tehran, Iran PO Box: 1991813831 Tel: +989123491692

The AgentScape project^[5] provides a multi agent infrastructure that can be employed to integrate and coordinate distributed resources in a computational grid environment. The objective of the AgentScape system is to provide a minimal but sufficient environment for agent applications. The A4 project at Warwick has likewise developed a framework for agent based resource management on grids. These multi-agent systems are used to trade with grid resources at the higher services level and not at the base resource level. Raw computational resources are normally scheduled using a more classical scheduler e.g., performance prediction or time based techniques.

The GrADS project focuses on building a framework for both preparing and executing applications in grid environment^[13]. Each application has an application manager, which monitors the performance of that application for QoS achievement. Failure to achieve QoS contract causes a rescheduling or redistribution of resources. GrADS monitor resources using NWS and use Autopilot for performance prediction^[12,13]. Also in^[14] mobile agent technology used to perform active monitoring in grid environment.

In^[1] a multi-agent framework for resource brokering and allocating in grid environment has been described. The goal of provided framework is to find select, reserve and allocate suitable resources to each submitted job with aim to optimize resource utilization and fairness. The main stress in ^[1] is to optimize resources utilization and there is no QoS support for grid's applications in this study.

Our proposed approach called MAAG employs intelligent and mobile agents to integrate and coordinate distributed resources in a grid environment with the aim to deliver user expected Quality of services and to optimize resource utilization.

MATERIALS AND METHODS

MAAG's Architecture: A general overview of MAAG's architecture with a sample grid in two sites is shown in Fig. 1. Each site is owned by an individuals or organizations and managed based its owners policies. The grid site provides some grid services and it also may provide an execution pool for local and migrant agents called Agent Holding Hosts (AHH). There is one static, intelligent and non-mobile agent in each site which is called Site Manager Agent (SMA). This agent monitors, measures and analyzes general QoS metrics of each site and also manages local AHH and allocate adequate execution host to migrant agents if available. The SMA is responsible for registering the

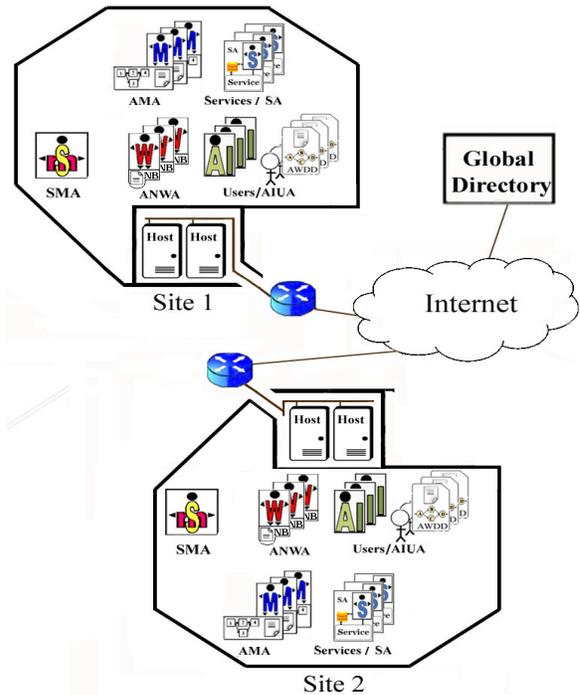


Fig. 1: MAAG architecture used in two grid site

identification information of its site and local available services on the grid's global directories. In response to site's user request for new application submission the SMA creates an Application User Interface Agent (AUIA) and assigns it to the user. The AUIA is an interface between user and the agent which is the main responsible for executing the application.

In the proposed architecture, each service has an intelligent and non-mobile agent called Service Agent (SA). This agent manages service factory and instances. Each request for using service will handle by its SA. Also the SA measures service's QoS parameters, performs negotiation and advanced reservation.

Each grid application has an intelligent agent which is called Application's Main Agent (AMA). The AMA is responsible for scheduling, executing and monitoring the application respecting to its owner's expected QoS defined in its Application's Workflow Definition Document (AWDD).

An AMA creates multiple thin mobile intelligent agents and sends each of them to one of its desired grid sites to act as its negotiator and watcher agent. These thin agents are called Application Negotiator and Watcher Agent (ANWA). Each ANWA migrates to its defined target site and negotiates with SA agents according to its Negotiation Bundle (NB) document-generated by its AMA- and sends back the results as a

negotiation report to AMA. The AMA uses all negotiation reports received from its ANWA agents to generate best possible Schedule regarding to application owner's QoS delivery expectations. The NB is generated by AMA for each of its ANWA agents specifically and it contains the orders describing ANWA objectives, negotiation constraints and directives for special conditions.

System Agents:

a) Intelligent Agents: Software agent is a component with capability of acting in order to accomplish its tasks on behalf of its owner^[10]. The ability of agents to work autonomously in heterogeneous environments^[10] utilized in the MAAG architecture to overcome complex tasks of monitoring, resource management and application scheduling in the large-scale heterogeneity presents in grid environment agents are shown. Common features of software agents are listed in the following^[11]:

- **Autonomy:** Agents operate without the direct intervention of humans
- **Social ability:** Agents co-operate with other agents toward the achievement of certain objectives
- **Re-activity:** Agents perceive their environment and respond in a timely fashion to changes that occur
- **Pro-activity:** Agents do not simply act in response to their environment; they are able to exhibit goal-directed behaviors by taking the initiative
- **Mobility:** Agents are able to travel through computer networks and to move from one computer to another computer during execution and may carry accumulated knowledge and data with them^[10,11]
- **Intelligence:** Informally, intelligent agents can be seen as software agents with intelligent behavior; that is, they are the combination of software agents with intelligent systems^[11]

b) Overview of MAAG Site and Agents: A Sample grid site using MAAG Architecture is shown in Fig. 2. Brief lists of duties assigned to each agent are introduced in Table 1. Also The SA, AMA and ANWA agents are described with more detail in part c, d and e from material and methods section respectively.

c) Service Agent (SA): While standard web services are persistent, grid services can be transient^[15]. OGSA^[9] provides a soft-service management by introducing the concept of grid service instances^[15]. An instantiation of a grid service can be dynamically created and

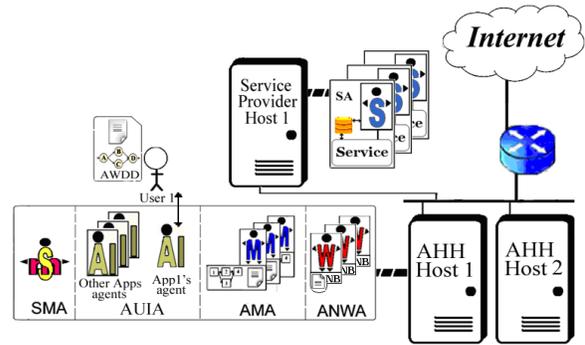


Fig. 2: Sample MAAG site

Table 1: Brief lists of duties assigned to each agent

| Site Manager Agent (SMA) | |
|--|---|
| | <ol style="list-style-type: none"> 1. Registering and updating identification information of site and local services in global directories. 2. Creating an AUIA agent in response to user's request for new application submission. 3. Analyzing and measuring general QoS specifications of the site. |
| Service Agent (SA) | |
| | <ol style="list-style-type: none"> 1. Monitoring and measuring QoS metrics for service instances. 2. Negotiating with ANWA agents and generating reservation offers in response to their requests. 3. Creating and destructing service instances. <p><i>This agent described in section c</i></p> |
| Application Main Agent (AMA) | |
| | <p>This agent described in section d</p> |
| Application User Interface Agent (AUIA) | |
| | <ol style="list-style-type: none"> 1. Creating an AMA for application 2. User interface for AMA. 3. Gathering Application execution status from AMA and informing the user. |
| Application Negotiator/ Watcher Agent (ANWA) | |
| | <p><i>This agent described in section e</i></p> |

destroyed. In OGSA a grid service that can create a service instance is called service factory^[9]. A client can request a factory to create many service instances, also it is possible that multiple clients access to same service instance^[15].

Service Agent (SA) is an intelligent and non mobile agent which is used in the MAAG to handle all management tasks related to a service including advanced reservation and service instances creation and destruction. Advanced reservation is a limited and restricted delegation of a particular Resource capability over a certain timeframe^[4,16]. General duties assigned to SA are as illustrated in Table 1. The SA handles the following tasks:

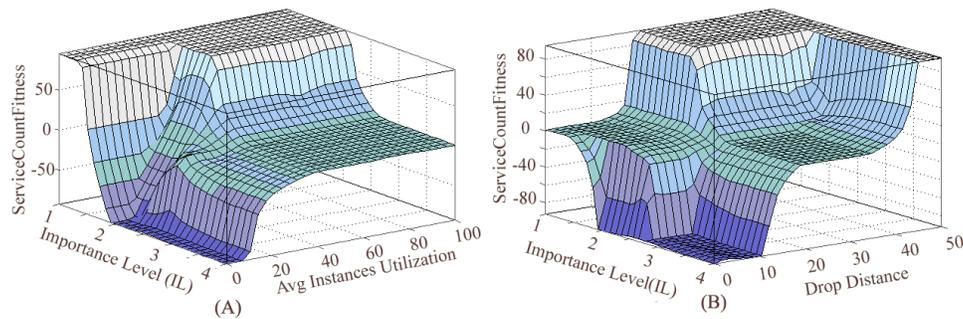


Fig. 3: Surface output of IGSCDA fuzzy inference system

- Accepting incoming advanced reservation requests from service consumers and classify the requests based on their Importance Level (IL)
- Selecting appropriate combination of incoming requests with different IL using a Novel proposed queue management algorithm which called FDWFQ
- Performing advanced reservation on service instances and dispatching the jobs as their reservation time slot become available
- Managing service factory and dynamically creating and destructing service instances using a Novel proposed Fuzzy based algorithm which called IGSCDA
- Monitoring and analyzing the utilization and some QoS related parameters for each service instance to adapt the agent to latest system's condition (such as workload, loss ratio in each IL class and etc)

Two Major components used in the SA are IGSCDA and FDWFQ. The IGSCDA and FDWFQ components are using two novel algorithms which are based on fuzzy inference systems. The Fuzzy Dynamic Weighted Fair Queue (FDWFQ) algorithm as its name indicates is Weighted Fair Queue (WFQ) algorithm in which queue weights are assigned dynamically using a fuzzy inference system (FDWFQ_FIS). The SA uses IGSCDA algorithm to balance between loss ratio and utilization by automatically create/destroy service instances [23]. In Fig. 3 the output surface of IGSCDA fuzzy inference system is plotted. The *CountFitness* indicates that the count of current active service instances is fit for current system conditions in the view of an specific IL class. The greater *CountFitness*, the more need to create new service instances and vice versa.

d) Application Main Agent (AMA): Each AMA is responsible for executing the application associated with it, regarding QoS constraints defined in Application's Workflow Definition Document

(AWDD). The AWDD used for job submission description (JSD^[16]) that consists of a set of constructs which used to specify workflow constraints as part of the WS-Agreement Specification. General duties assigned to AMA are as illustrated in Table. 1.

First part of application submission process for a sample grid application is illustrated in Fig. 4 as a sequence diagram. The AMA creates ANWA agents and sends them to other grid sites for negotiation, advanced reservation and performance monitoring.

In MAAG, each application has an Importance Level (IL) defined by organizational policy and can assumed as QoS class of the application^[7]. Table 2 depicts steps of negotiation, scheduling and reservation in an AMA. The AMA uses dynamic scheduling algorithm^[16] to generate QoS aware schedules.

e) Application Negotiator/Watcher Agent: The AMA creates ANWA agents and uses them as its agents in different grid sites. As described in part b from material and methods section and Algorithm 1, The AMA creates multiple ANWA agents and a Negotiation Bundle (NB) for each of them specifically, then the AMA requests from each ANWA to migrate to its target operating site.

The NB contains list of desired services, workflow constraints and QoS related expectations expressed in WS-Agreement format. ANWA should use the contents of mentioned list in its negotiations with SA agents in target site, also it contains directives that the ANWA should follow when special conditions are occurred. Each ANWA migrates to its target site and starts its operations defined in its NB (if target SMA permits its migration and allocates an operational resource in site's AHH to it). It requests from service agents for service reservation in the form of WS-Agreement^[16] templates and the SA generates a set of available reservation offers in the WS-Agreement format. The WS-Agreement is the proposed standard for the Grid Resource Allocation Agreement Protocol (GRAAP)^[15].

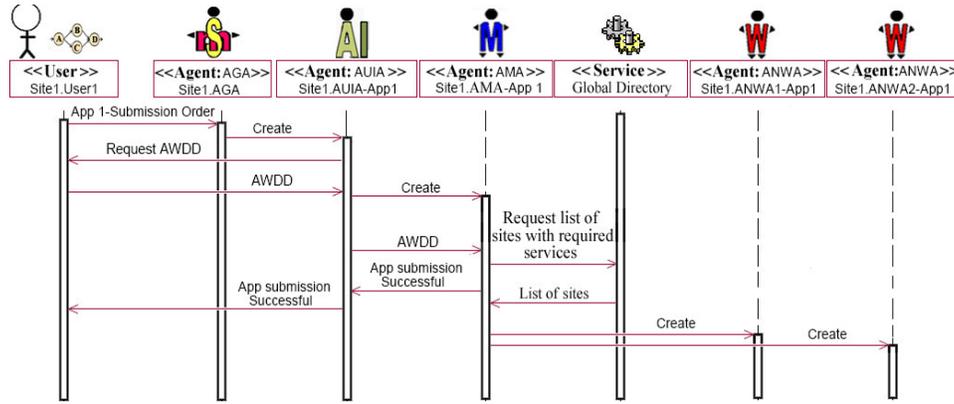


Fig. 4: First part of submission process for a sample application

Table 2: Application Main Agent (AMA) algorithm

- 1: Create *ReqServiceList* by processing the AWDD
- 2: For each service in *ReqServiceList* do Calculate reservation duration based on conservative estimate.
- 3: Request *TargetSitesList* from it.
- 4: For each site in *TargetSitesList* do
 - 4.1: Create an ANWA on local AHH
 - 4.2: Generate specific NB for the ANWA
- 5: Receive all *NegotiationReport* from ANWA agents
- 6: Generate *Schedule* based on received *NegotiationReport* using dynamic scheduling algorithm
- 7: For each task in *Schedule* do
 - 7.1: Order to respective ANWA to perform reservation
- 8: Store all advanced reservation results received from ANWA agents in *ReservationResults* set.
- 9: If any of reservations, failed then Reschedule
 - 9.2.2: If rescheduling failed then inform the AUIA to degrade QoS expectation or abort application)

Table 3: Brief duties of the ANWA

1. Migrating to target site, to negotiate with SA agents
2. Creating a negotiation report containing negotiation results and sending it to AMA.
3. According to AMA's command, performing advanced reservation with desired SA.
4. According to AMA's command, Renewing, extending or revoking a reservation (if possible).
1. Analyzing QoS provided to each of AMA jobs running in operating site and check QoS goals achievement using utility functions^[16].
2. Performing adequate action according to NB on reservation violation or service failure.

The ANWA will send these offers to AMA as its negotiation report. Two categories of duties assigned to the ANWA are mentioned in Table 3.

RESULTS AND DISCUSSION

Simulation software is developed to evaluate the performance of the Service Agent (SA) as an important part of the proposed architecture MAAG. The MATLAB fuzzy engine is used to implement fuzzy

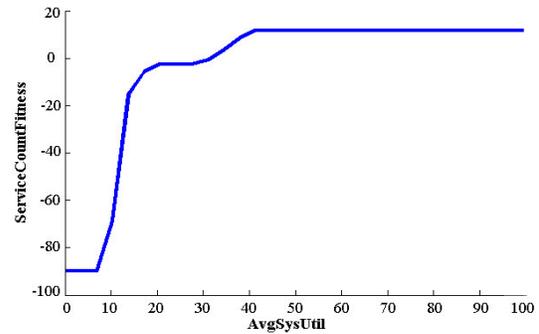


Fig. 5: Output of the service instance count controller fuzzy inference system for Average service utilities

based algorithms used in the SA (FDWFQ and IGSCDA algorithms). Various advanced reservation and queue management algorithms were implemented in the simulation software to compare the Service Agent performance with them.

In our simulation deadlines are hard, in that a user receives utility only if the job completes by its deadline. If D_j is the deadline of job j then it is uniformly distributed as shown in Eq. 1:

$$R_j + P_j \leq D_j \leq R_j + P_j + q (P_j) \quad (1)$$

where, q is a parameter to control the tightness of job deadline and it is between 0 and 1. In our simulation we let $q = 0.1$.

The output of the core fuzzy inference system used for controlling service instances count are plotted against Average system utility, Drop Ratio and Importance Level in Fig. 5, 6 and 7.

In Fig. 8 loss ratio against request arrival rate λ is plotted for the service agent (SA), WFQ and RR by

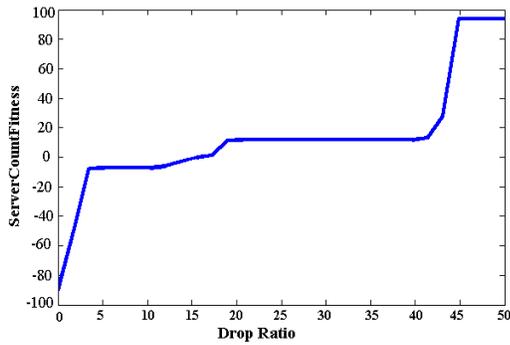


Fig. 7: Output of the service instance count controller fuzzy inference system for drop ratio

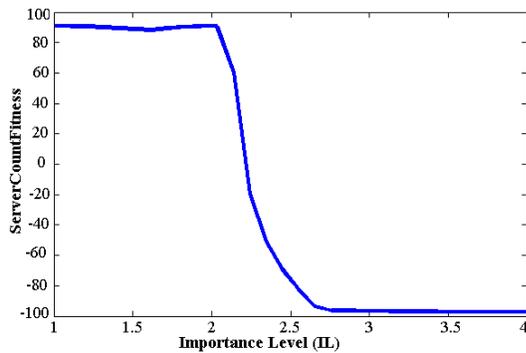


Fig. 8: Output of the service instance count controller fuzzy inference system for Importance Level

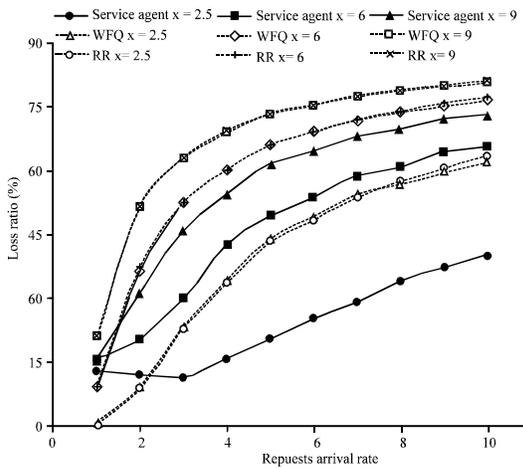


Fig. 9: Loss ratio vs. requests arrival rate λ , for RR and WFQ $n = 10$

considering job mean size $x = 2.5, 6$ and 9 also for WFQ and RR number of service instances considered fixed $n = 10$.

CONCLUSION

This research presented a new multi-agent architecture for grid computing environments. Five types of agents were introduced to handle grid services and applications management with aim to deliver users expected Quality of Services (QoS). The introduced agents work on top of the available Grid middleware (such as Globus Toolkit) and uses the services available with it. A brief discussion about the agents and their interactions is presented in this paper. In the proposed architecture, negotiation, advanced reservation and scheduling is performed in a decentralized manner. Using intelligent agents in service and application management adds flexibility and intelligence to negotiation and reservation interactions. Also using thin intelligent mobile agents makes the application's scheduling, monitoring and execution more autonomous and optimized. The simulation results illustrated that the SA as an important part of the MAAG gives better performance metrics compared with other commonly used techniques. The future work will focus on extending agents to make applications and services fault tolerant.

REFERENCES

- Roy, S., S. Dasgupta and N. Mukherjee, 2006. A multi-agent framework for resource brokering of multiple concurrent jobs in grid environment. In: Proceedings of the 5th International Symposium on Parallel and Distributed Computing (ISPDC'06). doi: 10.1109/ISPDC.2006.3
- Wolski, R., J. Plank, J. Brevik and T. Bryan, 2001. Analyzing market-based resource allocation strategies for the computational grid. J. High-Performance Comput. Applic., 15: 258-281. doi: 10.1177/109434200101500305
- Li, C., Z. Lu and L. Li, 2004. Multi-agent interaction for optimal resource allocation in computational grid. Proceedings of the 3rd International Conference on Machine Learning and Cybernetics, Shanghai, Aug. 26-29. doi: 10.1109/ICMLC.2004.1380596
- Castillo, C., G.N. Rouskas and K. Harfoush, 2007. On the design of online scheduling algorithms for advance reservations and qos in grids. International Parallel and Distributed Processing Symposium, IEEE, IPDPS. doi: 10.1109/IPDPS.2007.370226
- Wijngaards N.J.E., B.J. Overeinder, M. Van Steen, F.M.T. Brazier, 2002. Supporting Internet-scale multi-agent systems (2002). Data Knowledge Eng., doi:10.1016/S0169-023X(02)00042-3

6. Buyya, R., J. Giddy and D. Abramson, 2001. A case for economy grid architecture for service-oriented grid computing. 10th IEEE International Heterogeneous Computing Workshop, April 2001. doi: 10.1109/IPDPS.2007.370226
7. Doshi, B., L. Benmohamed and A. DeSimone, 2005. A hybrid end-to-end qos architecture for heterogeneous networks. Military Communications Conference, IEEE, 2005. doi: 10.1109/MILCOM.2005.1606078
8. Munindar, P.S. and N.H. Michael, 2005. Service-Oriented Computing. John Wiley and Sons. doi: 10.1002/0470091509.ch5
9. Foster, I., C. Kesselman, J. Nick and S. Tuecke, 2002. The physiology of the grid: An open grid services architecture for distributed systems integration. Open Grid Service Infrastructure WG, Global Grid Forum, Cambridge University Press, June 2002. doi: 10.1002/0470867167.ch8
10. Wooldridge, M., 2002. Introduction to Multi-Agent Systems. John Wiley and Sons. 10.2448/lr-univb-valentina-1099517115906
11. Yu, T., Y. Yuan, J. Li, F. Xiong and M. Fang, 2005. Multi-agent based approach for manufacturing grid workflows. Proceedings of the 4th International Conference on Machine Learning and Cybernetics, Guangzhou, Aug. 18-18. doi:10.1109/ICMLC.2005.1526944
12. Wolski, R., N.T. Spring and J. Hayes, 1999. The network weather service: A distributed performance forecasting service for metacomputing, Future Generat. Comput. Syst., (FGCS), 15: 757-768. doi:10.1016/S0167-739X(99)00025-4
13. Ribler, R.L., H. Simitci and D.A. Reed, 2001. The autopilot performance-directed adaptive control system. FGCS, 18: 175-187. doi:10.1016/S0167-739X(01)00051-6.
14. Ruoyun Yang and Mitchell D. Theys, 2005. RMF: Resource Monitoring Framework for integrating active and passive monitoring tools in Grid environments. J. Parallel Distributed Comput., 65. doi: 10.1016/j.jpdc.2005.05.012.
15. Li, M. and M. Baker, 2005. The Grid Core Technologies. John Wiley and Sons Ltd. ISBN: 13 978-0-470-09417-4.
16. Siddiqui, M., A. Villaz'On and T. Fahringer, 2006. Grid capacity planning with negotiation-based advance reservation for optimized QoS. Proceedings of the 2006 ACM/IEEE SC106 Conference (SC'06), 2006. doi: 10.1145/1188455.1188563.