Journal of Computer Science 9 (12): 1661-1668, 2013
ISSN: 1549-3636

doi:10.3844/jcssp.2013.1661.1668 Published Online 9 (12) 2013 (http://www.thescipub.com/jcs.toc)

ADAPTIVE RESOURCE CONTROL MECHANISM THROUGH REPUTATION-BASED SCHEDULING IN HETEROGENEOUS DISTRIBUTED SYSTEMS

Masnida Hussin and Rohaya Latip

Department of Communication Technology and Networks,
Faculty of Computer Science and Information Technology, University Putra Malaysia Selangor, Malaysia

Received 2013-07-03, Revised 2013-09-02; Accepted 2013-10-29

ABSTRACT

The service-oriented distributed systems such as Grids and Clouds are unified computing platform that connect and share heterogeneous resources including computation resource, storage resource, information resource and knowledge resource. While these systems provide a vast amount of computing power their reliability are often hard to be guaranteed. It is due to the increased complexity of processing (e.g., overhead, latency) that can indirectly affect the system performance. In this study, we addressed the problem of dynamic control for resource management in distributed computing environment. Our dynamic resource control mechanism is designed based on reputation-based scheduling that aims for sustainable resource sharing. Particularly, each computational resource in the environment has its own reputation value that calculated online by considering the computing capacity and availability. The degree of resource reputation significantly helps in scheduling decisions in terms of successful execution while adaptively monitoring resource availability. Results demonstrate that our resource control mechanism significantly increases successful execution, while leading to robust resource management.

Keywords: Distributed Systems, Dynamic Control, Resource Sharing, Task Scheduling, Reputation

1. INTRODUCTION

The service-oriented distributed systems generally are high performance computational environment that are composed of heterogeneous resources. Some popular examples are computational Grids, Clouds, wireless sensor networks are recently the Internet of Things. The specific problem that underlies these systems is coordinated resource sharing. Resources are sacrificed on activities performed on each individual unit of service. The sharing activities concerned with file exchange, direct access to computers, software data and other resources. However, with the characteristics of dynamic and heterogeneity, there are issues to discover suitable resources for users' tasks. In addition, each administrative domain in the distributed systems has its own resource usage pattern (Hussin *et al*., 2011; Ping and Xingshe, 2011; Zunjare and Sahoo, 2012) made the evaluation of the resource behaviour varies. Another issue is because their behaviors change more rapidly than a controller can adapt that made the resource information for the controller (scheduler) delayed and potentially inaccurate. Such unpredictable and imprecise resource behaviour inherently results in unreliable task execution. Apparently, the resources cannot be assumed always reliable particularly in term of performance. In addition, the number of incoming tasks is unpredictable therefore it is difficult to determine the prior in task scheduling.

One of the methods to solve the resource sharing in dynamic environment is through efficient resource management (Beaumont *et al*., 2013). In particular, the resource management addresses the monitoring and controlling abilities that able to take into accounts the users' different needs and performance requirements. In

Corresponding Author: Masnida Hussin, Department of Communication Technology and Networks,
Faculty of Computer Science and Information Technology, University Putra Malaysia Selangor, Malaysia

this study, we focus on designing a reputation-based scheduling approach for which the decision is made taking into account conditions of heterogeneity and dynamism. With the reputation value of scheduler it helps to dynamically monitor a trust interaction between service providers and users that improves task successful execution. This is realized based on three aspects. First, we dynamically calculate reputation value of each scheduler that take into consideration the current resource performance. Second, we classify the reputation value for controlling resource allocation. Third, we address the issue of resource-task matched that can satisfy processing requirements. These strategies are adaptive in nature since they incorporate current task demand and resource availability to facilitate scheduling process. The results obtained from our comparative evaluation study clearly show that our adaptive resource management outperforms other schemes in terms of performance by a noticeable margin.

This study is organized as follows. A review of related work is presented in section 2. In section 3, we described the models used in the study. Our scheduling strategy is presented in section 4. Experimental settings and the experimental results are presented in section 5. Finally, conclusions are made in section 6.

2. RELATED WORK

In the service-oriented distributed systems, every application is completely different and independent. For example, some require more execution time to compute complex task and some others may need more memory to store data. Also, it appears in many distributed cases that schedulers have limited information about each other's resource state where some resource providers, if not all, might not always make public availability of information on their resources; this is especially true for Cloud (Nathani *et al*., 2012). Applying trust-based method to distributed systems is an interesting subject that yields more attention in distributed systems research recently. The concept of trust in resource behaviour for dealing with variability and instability of compute nodes was proposed in (Hussin *et al*., 2011; Ping and Xingshe, 2011; Zunjare and Sahoo, 2012), where reputation is defined based on various factors, such as prior performance, network capacity and resource availability. In the Cloud computing for example, the users needed some reputation mechanism to guarantee the safety of their investment in the specific service (Nathani *et al*., 2012; Song *et al*., 2009).

Generally, resource controller or scheduler can contribute much to the effective resource management. It has the ability to work within the processing requirements/constraints that put forth by the system users. The processing constraints must be effectively handled particularly in the present of dynamic computing; otherwise it may lead to load imbalance, over-provisioning of resources and system unreliability. The schedulers in (Ping and Xingshe, 2011; Gupta and Singh, 2012) adaptively deal with processing constraints for reliable execution while minimizing waiting time. The feedback control policy is proposed in (He *et al*., 2007) to solve resource allocation problems in asynchronous real-time distributed systems. The controller or scheduler is used to determine the number of replicas of each sub-task that are needed to adapt the application to workload changes; hence satisfies the task deadline. Yan *et al*. (2012) the Particle Swarm Optimization (PSO) algorithm is realized and used as controller. The PSO algorithm or controller adjusted the frequency of task according to the constraints of CPU utilization, lower bound of frequency and channel capacity. In that way, they optimized the Quality-of-Service (QoS). These approaches have demonstrated the effectiveness in minimizing response time. However, the efficacy of these approaches in dealing with system dynamicity is limited to a certain level.

Due to heterogeneity and dynamicity of resources, the dynamic resource control mechanism that is based on trustworthy can be a very effective approach for robust resource management. While most previous resource sharing solutions dealing with either workload changes or processing capacity, our approach in this work is explicitly taking into account diversity in both processing requirements and heterogeneous resources.

3. THE MODEL

In this section, we describe the application and system models used in our study.

3.1. System Model

The target system used in this study (**Fig. 1**) consists of several resource sites given as S_x where x = {1, 2…, s} which are loosely connected by a communication network. They are independently operated by different administrative domains. We assumed that the communication delay is negligible. The inter-site communication cost is insignificant. To simplify, we limit task scheduling to a global scheduler (Hussin *et al*., 2011; Gupta and Singh, 2012) that can handle tasks from users and map them onto resources.

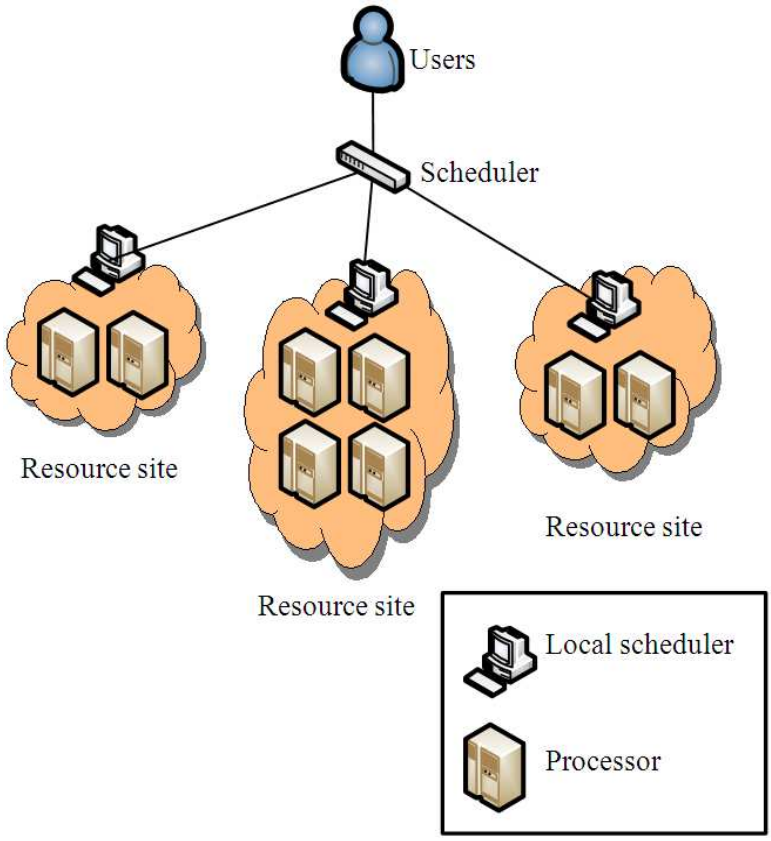

Fig. 1. System model

A resource site allows any type of tasks to arrive at its local scheduler. The local scheduler is responsible for scheduling tasks after allocation by the global scheduler. We assumed that the local scheduler has all necessary information about computational resources (processors) which is currently located at the site.

Each site x has a set R of r_j processors that are interconnected via a high-speed link. These processors are heterogeneous in terms of their processing speed. That is, each processor r has an associated processing capacity to complete a task of size s_i, as given in Equation 1:

$$P_r = \frac{C_i}{\sum exe} + q_t \quad (1)$$

where, C_i is a number of completed tasks, exe is an execution time of tasks i and q_t is an average waiting time of processor r, respectively. Hereafter, the terms resource and processor are used interchangeably.

As the nature of computation in distributed systems can be fluctuated, resources in this study are considered to be 'elastic' where a processor can go from being capable status (available to perform computation in a timely manner) to incapable status (unavailable to perform computation by deadline) at any time without further communication. Without the loss of generality, we also assume that a resource site has its own tasks (local workload) that need to be computed to service local users. As such, the background workload associated with it will affect the overall performance as well as the actual task completion time.

3.2. Application Model

Tasks considered in this study are computation-intensive and independent from each other (i.e., no inter-job communication or dependencies). Each task is a single arrival unit and associated with the set of parameters shown below Equation 2:

$$Ti = \{s_i, d_i\} \quad (2)$$

where, s_i is the computational size of task i that is specified by Millions of Instructions (MI) and d_i is the latest time (deadline) by which task Ti is supposed to be completed. For a given task T_i, the deadline d_i is computed by the computational size of task s_i divided by processing capacity of a referred (the slowest) resource. We assumed that the task's profile is available and can be provided by the user using job profiling, analytical models or historical information.

Tasks are assumed to be sequential applications and each of which requires not more than one processor for its execution. However, tasks may be considered to be assigned to processors with different processing capacities due to primarily to the urgency of task, i.e., task priority. This study considers three levels of task priority based on deadline: low, medium and high. The priority of a task Ti is set to high if its deadline d_i is at most 20% later than the expected execution time exe_i. The priority is set to low if d_i is 80% or more than exe_i. Otherwise, the priority is set to medium. Tasks arrival is in a Poisson distribution.

4. AUTOMATIC RESOURCE MANAGEMENT

This section begins by describing formation of resource reputation and gives the details of our dynamic resource control mechanism in the context of task scheduling.

4.1. Formation of Resource Reputation

The resource reputation is used to differentiate the resources with different qualities. Implicitly, it provides an effective strategy for high-level resource sharing amongst sites. Particularity, the resource expresses its valuation of processing as a function of a computing competence (resource reputation). To facilitate scheduling decision making, the prior performance of a resource must be accurately identified and determined. This type of performance is used to indicate the weight of a resource relative to other resources. More formally, the prior performance of a node is defined as summation of success rate and average execution time Equation 3:

$$PP_r = \sum sr + av\_exe;$$
$$\text{where } sr = \begin{cases} 1 & \text{if finish } T_i \le dt \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

Then, the resource reputation is computed by the prior performance divided by its processing capacity (i.e., $rrep_r = PP_r / P_r$). The basic idea behind this is that the higher reputation a resource is, the more reliable it can execute the users' tasks. For each site, the average reputation value over all its resources indicates its computing trustworthy.

4.2. Adaptive Scheduling

The adaptive scheduling approach is an effective means for selecting more reliable resource to complete the users' tasks. For this type of scheduling, the process to find the resources (resource-task matching) should take in optimal (minimum) time to make decision. In such a way, the tasks scheduled can be completed in the shortest time possible; provided that they are assigned to appropriate resources. The scheduler is responsible for scheduling the tasks to suitable resources that is based on the processing capability (availability and capacity) aiming to achieve reliable execution.

Our scheduling approach precisely represents resources in terms of reputation. The reputation value that mentioned in early section abstracts the property of resource sharing to support adaptive and (near) optimal task scheduling. We incorporate a resource classification scheme into the scheduling in order for a task to be assigned into the most appropriate resource. The resource classification is carried out in a systematic way by considering the resource reputation (**Fig. 2**). This work classified the resources according to the reputation value (more specifically, from the highest to the lowest) for task scheduling. In order to gain an up-to-date list of reliable resources, a (re)listing mechanism is carried out in the way that listed resources are further inspected with respect to their recent reputation value $rrep_r$ for possible relisting. This repeats until no further improvements in the resource list is possible. In some cases, for instance, if more than one resource has the same reputation value, the resources are then sorted according to their prior performance PP_r; the resource with higher PP_r, is listed first.

In particular, task is scheduled to the resource that gives the highest reputation value. However, this greedy approach leaves the true optimal decision and the resource superiority to never be discovered. In response to this, we calculated a deadline factor for each task. Given that each arriving task has its deadline; hence the deadline factor is determined based on the measurement in (Hussin *et al*., 2010) as given by:

$$df_i = \begin{cases} 1.00 & \text{if ataskmissed its 100\% deadline} \\ 0.50 & \text{if ataskmissed its 50\% deadline} \\ 0.25 & \text{if ataskmissed its 25\% deadline} \\ 0.05 & \text{if ataskbeneath its 25\% deadline} \\ 0 & \text{if askjust arrived} \end{cases}$$

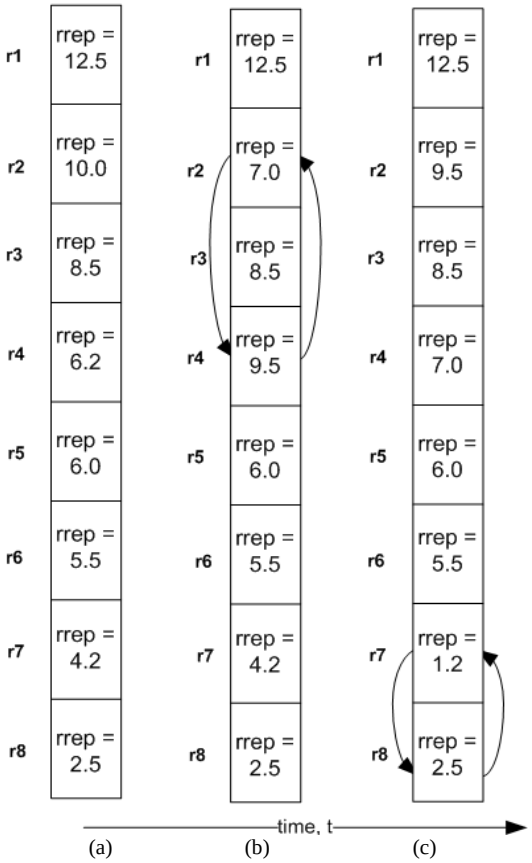

Fig. 2. Reputation-based list scheduling; (a) Initial schedule (b) The first revision and (c) the second revision

The scheduler constantly checks the task df_i to indicate which task needs to be mapped first. As the scheduling of tasks has to take place during runtime, both resource reputation and deadline factor are regularly updated.

Our resource-task matching algorithm aims to maximize successful execution while minimizing the response time. There are two different ways to assign the task onto the right resource. A task with highest deadline factor is always assigned to the resource that gives the highest reputation value. The resource with the second highest reputation value is allocated to a task with lower deadline factor. Each of the resource also can promote the other resources for task assignment when there is significant increase in waiting time in its task queue. Once the tasks are assigned, the scheduler updates resource reputation value and further inspected the reputation value for possible relisting. This repeats until no further processing requirements to be executed.

5. PERFORMANCE EVALUATION AND DISCUSSION

In this section, we describe the experiment configuration and present results. Performance metrics used for the experiment are success rate and utilization rate. Success rate is used to measure the degree of reliable execution for dealing with various priority tasks. We define the utilization rate of a resource as RU = $busy_j/(busy_j+idle_j)$ where $busy_j$ is the total time when the resource r_j is busy for servicing tasks and $idle_j$ is total idle time of r_j, respectively.

5.1. Experimental Settings

To evaluate the performance of our automatic resource management approach, we have conducted extensive simulations with a diverse set of tasks and computing resources (processors). Due to the large scale constitution of distributed environments, the parameters of simulation model used are assumed to be unbounded; that can be created. Specifically, there are five to ten resource sites and each contains a varying number of computing resources (processors) ranging from 5 to 8 processors. The relative processing power (speed) of processor j is selected within the range of 1 and 7.5. The number of tasks in a particular simulation is set between 1000 and 5000. Task inter-arrival times follow a Poisson distribution with a mean of 5 time units. The computational size s_i is randomly generated from a uniform distribution ranging from 20 to 500.

5.2. Results and Discussions

We first study how the performance of our resource control mechanism for effective resource management is influenced by a reputation factor. The experiments have been conducted with two different schemes: reputation-based scheduling (repRM) and without reputation (norepRM). With norepRM, the resources are chosen for task assignment by their processing capacity. Then, we compared our scheduling approach with extended versions of three other heuristic scheduling algorithms i.e., Min-Min, Max-Min and Max-Max. This is because their performance tends to produce competitive solutions with lower time complexity (Beaumont *et al*., 2013). Since our scheduling concerns deadline factor df_i, we have revised Min-Min, Max-Min and Max-Max to fit into our model. In the comparison, their mapping decisions are according to fitness value that calculated as the product of reputation value $rrep_j$ and deadline factor df_{JC}.

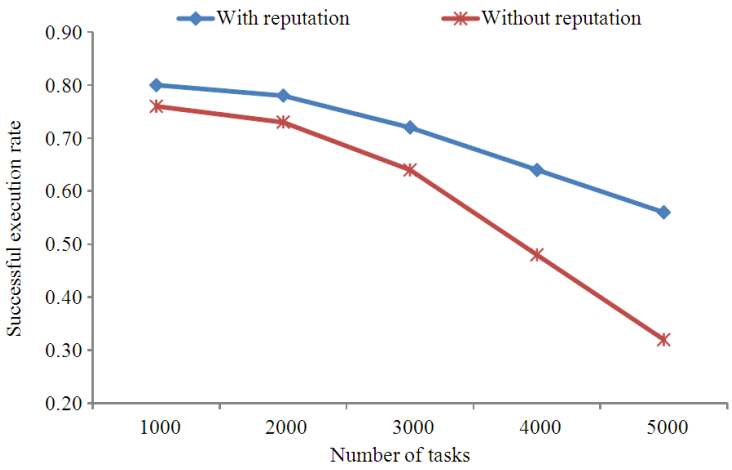

Fig. 3. Success rate with- and without- reputation factors

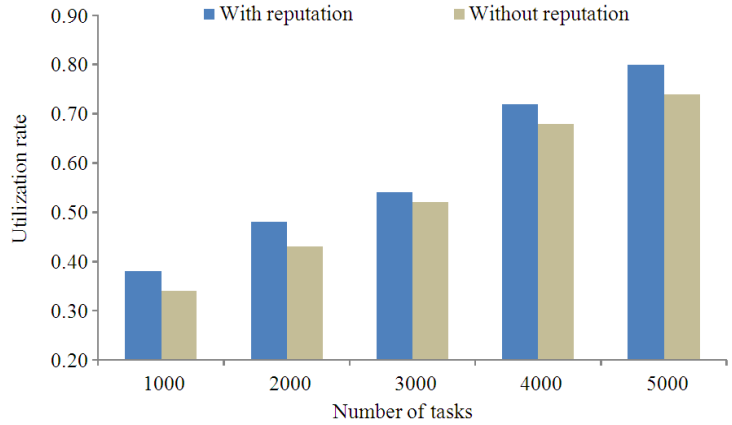

Fig. 4. Utilization rate with- and without- reputation factors

Experiment 1:

The impact of reputation towards adaptive resource scheduling.

Figure 3 illustrates the performance of resource management in the presence of reputation concerns. The success rate of reputation-based scheduling is higher than scheduling without reputation.

The reason is that the reputation-based scheduling directs the allocation of tasks with the resource reputation being the indication; hence contributes to minimize waiting time.

Next, we study the utilization rate between reputation-based scheduling and without reputation. As shown in the **Fig. 4**, both scheduling schemes (with and without reputation) have comparable performance with the difference about 20% on average. It indicates that the system is able to sustain better resource utilization when the characteristics of both resources and tasks are considered.

Experiment 2:

The impact of resource preference in scheduling towards robust resource management.

Figure 5 and 6 show that the benefit of reputation for a good resource management controller. As shown in **Fig. 5**, our scheduling scheme outperforms other schemes in terms of successful execution rate by a noticeable margin. There are more than 80% of tasks (on average) have completed their execution before their deadline. It also shows (**Fig. 6**) that the utilization rate using reputation-based scheduling is another compelling strength. This is mainly because a resource with a good offer (i.e., high availability) is more popular to be chosen to accommodate the input load that affects the results.

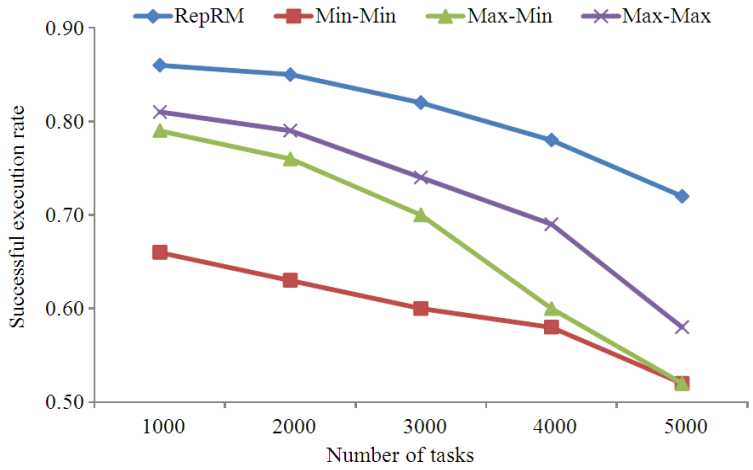

Fig. 5. Success rate of different list scheduling policies

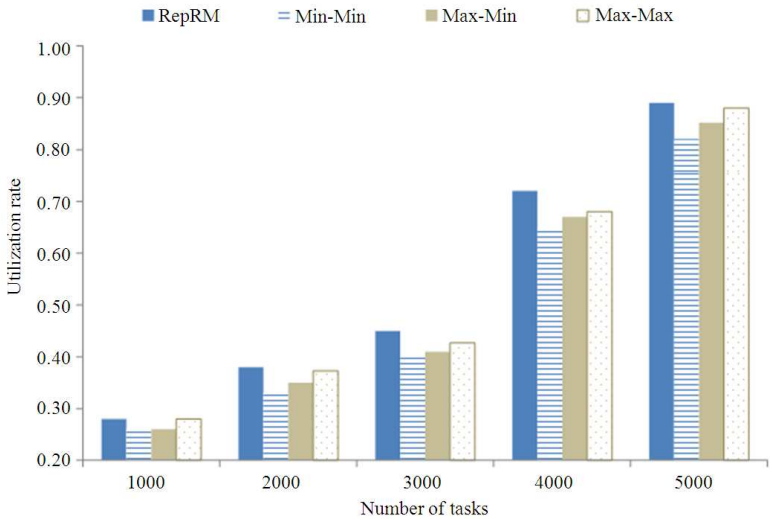

Fig. 6. Utilization rate of different scheduling policies

Although all scheduling schemes consider resource reputation and processing requirements during the mapping process, the prominence on resource behavior much benefit in order to lead for optimal decisions.

6. CONCLUSION

In this study, we address the dynamic resource control in the context of resource sharing. We have successfully developed a dynamic scheduler that explicitly considers resource and task heterogeneity. Dynamic discovery of resource behavior based on its reputation (availability and capacity) significantly helps improve scheduling decision with minimal possible performance degradation. Based on our extensive simulation results, we also found that the reputation-based scheduling algorithm contributes for minimizing task waiting time; hence our approach is capable of providing reliable processing in terms of successful execution time. We highlight that thorough analyzing of processing requirement and resource behavior in resource management are required and significantly important for dynamic-heterogeneous distributed systems. In future work, we will develop an adaptive scheduling framework for further solutions to robust computational.

7. REFERENCES

Beaumont, O., L. Eyraud-Dubois, C.T. Caro and H. Rejeb, 2013. Heterogeneous resource allocation under degree constraints. IEEE Trans. Parallel Distributed Syst., 24: 926-937. DOI: 10.1109/TPDS.2012.175

Gupta, K. and M. Singh, 2012. Heuristic based task scheduling in grid. Int. J. Eng. Technol., 4: 254-260.

He, T., J.A. Stankovic, M. Marley, C. Lu and Y. Lu *et al*., 2007. Feedback control-based dynamic resource management in distributed real-time systems. J. Syst. Software, 80: 997-1004. DOI: 10.1016/j.jss.2006.09.029

Hussin, M., Y.C. Lee and A.Y. Zomaya, 2010. ADREA: A framework for adaptive resource allocation in distributed computing systems. Proceedings of the International Conference on Parallel and Distributed Computing, Applications and Technologies, Dec. 8-11, IEEE Xplore Press, Wuhan, pp: 50-57. DOI: 10.1109/PDCAT.2010.19

Hussin, M., Y.C. Lee and A.Y. Zomaya, 2011. Reputation-based resource allocation in market-oriented distributed systems. Proceedings of the 11th International Conference Algorithms Architectures Parallel Processing, Oct. 24-26, Springer Berlin Heidelberg, Melbourne, Australia, pp: 443-452. DOI: 10.1007/978-3-642-24650-0_38

Nathani, A., S. Chaudhary and G. Somani, 2012. Policy based resource allocation in IaaS cloud. Future Generat. Comput. Syst., 28: 94-103. DOI: 10.1016/j.future.2011.05.016

Ping, X. and Z. Xingshe, 2011. Security-driven fault tolerant scheduling algorithm for high dependable distributed real-time system. proceedings of the 4th International Symposium on Parallel Architectures, Algorithms and Programming, Dec. 9-11, IEEE Xplore Press, Tianjin, pp: 29-33. DOI: 10.1109/PAAP.2011.14

Song, S., K.D. Ryu and M. Silva, 2009. Blue eyes: Scalable and reliable system management for cloud computing. Proceedings of the IEEE International Parallel and Distributed Processing Symposium, May 23-29, IEEE Xplore Press, Rome, pp: 1-8. DOI: 10.1109/IPDPS.2009.5161232

Yan, H.H., J.F. Wan and H. Suo, 2012. Adaptive resource management for cyber-physical systems. Applied Mechan. Mater., 157-158: 747-751. DOI: 10.4028/www.scientific.net/AMM.157-158.747

Zunjare, P. and B. Sahoo, 2012. Evaluating robustness of resource allocation in uniprocessor real time system. Int. J. Comput. Applic., 40: 13-18. DOI: 10.5120/5023-7168