Performance Prediction and QoS Based Resource Selection in Grid

Rajesh Kumar Bawa[1] and Mr. Gaurav Sharma[2]

[1]*Associate Prof., Deptt. Of Computer Sci, Punjabi University, Patiala*
[2] *Assistant Prof., Deptt. Of Computer Sci. & Engg., JMIT Radaur*

rajesh_k_bawa@yahoo.com, gauravsharma@jmit.ac.in

Abstract

Grid technology allows resource sharing among several entities, but selecting the best resource to run a specific job remains one of its main problems. We try to solve this problem by predicting the performance of the resources for a given job and submit our job to best suited one. So an appropriate resource selection mechanism is crucial for the success of job execution and performance of Grid. The QoS parameter is key factor in our selection process. The QoS ensure that high quality requested jobs which may be a communicational or computational based must be submitted to high quality resources. In this paper we proposed a resource selection mechanism which worked in two phases, first phase filter the resources based upon their trust value and second phase predict the performance of all the host for a given job by considering QoS and rank the resources according to user criteria and submit the job to top most resource.

Keywords: *Grid, Performance Prediction, Ranking, Resource Selection, QoS*

1. Introduction

In today's complex world of high speed computing, computers have become extremely powerful, even home-based desktops are powerful enough to run complex applications. But still we have numerous complex scientific experiments, advanced modeling scenarios, genome matching, astronomical research, a wide variety of simulations, complex scientific & business modeling scenarios and real-time personal portfolio management, which require huge amount of computational resources. To satisfy some of these aforementioned requirements, Grid Computing [9] is born. The Grid is emerging as a wide-scale, distributed computing infrastructure that promises to support resource sharing and coordinated problem solving in dynamic, multi institutional Virtual Organization [2].These features also increase the difficulty of Grid resources selection[9]. Resource selection is an especially critical step in data-intensive, decision making processes. Generally, the process includes foursteps: 1) sourcing; 2) selecting; 3) negotiating; and 4) deciding.In order to yield the best results, it is therefore essential to learn how to select the appropriate resources in a manner that is both efficient and cost-effective and provide QoS [11]. Qualities of service (QoS) are often associated with jobs submitted to a Grid system . An example of this is jobs with user-defined deadlines. The QoS of a job is satisfied if it finishes before the specified deadline, while the QoS decreases as the excess (of completion time over deadline) increases. Therefore, over-deadline can be used to measure the extent to which the QoS demands of a set of jobs are satisfied. Over-deadline is defined as the sum of excess time of each job's finish time over its deadline. The traditional algorithm of resource selection only consider the system performance, but they have neglected the user's grade of requirement also they cannot satisfy the demand of actual Grid environment very well.

The motivation of this paper is to develop a resource selection mechanism that can select the appropriate resource for a given job and releasing the burden from the user to select the best suitable resource for the task. There are two types of jobs in a Grid, Computational and communicational based. The Computational jobs are the jobs that require processing time more than communication time. Similarly, communicational jobs are the jobs that require communication time more than processing time. Job type is the critical factor in Grid because if the communication intensive job gets submitted to low bandwidth node then it will take more time to execute, which results in further poor performance of Grid and if computation intensive job gets submitted to a dual core or single core processor then it will also let down the efficiency of Grid. So while selecting the resource we must considered these factors. In the first phase of our proposed approach we check the reliability of the resources based upon their trust value, only reliable resources are allowed to participate in the selection process. In the Second phase we divide the resources into four different classes based upon QoS parameters, then according to user requirement we choose the class and then for that particular class we predict the performance of resources for given task, based on prediction resources are ranked and then job gets submitted to highly ranked resource of that class.

2. Related Work

Saurabh Kumar Garg, Rajkumar Buyya, et. al., [4] proposed Time and cost trade-off management for scheduling parallel applications on Utility Grids. They presented three novel heuristics for scheduling parallel applications on Utility Grids that manage and optimize the trade-off between time and cost constraints. They proposed 3 meta-scheduling online heuristics, such as Min-Min Cost Time Trade-off (MinCTT), Sufferage Cost Time Trade-off (SuffCTT), and Max-Min Cost Time Trade-off (Max- CTT), to manage the trade-off between overall execution time and cost and minimize them simultaneously on the basis of a tradeoff factor.

Xuan Wang and Ling-Fu Kong [5] proposed a resource selection algorithm based on the multi-goals. They proposed a resource credit model and then established resource provider model. Based on these models, a dual-credit is introduced to describe the behaviors of the resource and its provider objectively. In the next place, they considered a time-cost synthetic measurement function joins into the MGCS to meet the user requirements for response time and cost. In the end they proposed a improved scheduling architecture, which supports the credit evaluation.

Chen Chen, Gu Li-ze, NiuXin-xin, Yang Yi-xian [6] proposed a Resource selection technique based on trust management system in which a node is usually named an OP(operation point), the node requesting resources is called an CP(control point) and the node supplying resources is an PP(participate point). All entities can register into the trust management system. In Grid environment, each domain includes a trust agent, several trust subagents and other entities. In this system, the trust agent or subagent will collect the trust information in Grid, calculate the trust values according to a mathematic model and store the values in a database. When choosing resources, the subordinate of the trust agent would make use of trust service, negotiate with the agent, gain the trust value list and choose suitable resource according to trust values and local strategy. Also, further negotiation with the chosen resource should be proceeded in order to identify whether the resource is available and embed this trust value in a scheduler's scheduling algorithm in order to selecting the most trustful resource provider to the users.

Noorisyam Hamid, et. al., [7] proposes a resource selection technique based on resource ranking. Their resource selection technique i.e. Resource Ranking is based on Google's search technique PageRank.Resource can obtain higher ResourceRank score, if many users

from different organizations submit jobs to that resource. Maheswaran M, Ali S, Siegel H J, et. al., [8] proposed Min-Min heuristic is a simple heuristic which begins with the set MT of all unassigned tasks. It has two phases. In the first phase, the set of minimum expected completion time for each task in MT is found. In the second phase, the task with the overall minimum expected completion time from MT is chosen and assigned to the corresponding resource. Then this task is removed from MT and the process is repeated until all tasks in the MT are mapped. However, the Min-Min algorithm is unable to balance the load well, since it usually schedules small tasks first.

Xiao-Shan He, Xian-He Sun [1] QoS Guided Min-Min heuristic considers network bandwidth as QoS parameter. It divides the tasks in two QoS groups: high and low. The idea behind this division is that the tasks requiring high QoS can only run on high QoS providing hosts. The low QoS task can run on any hosts and if they are allocated to high QoS resources, the tasks with high QoS requirements will have to wait till they finish. This leads large makespan, wastage of resources and load unbalancing. At last, this reduces the overall performance of Grid. The QoS guided Min-Min heuristic first schedules the tasks from high QoS group on resources that can provide high QoS. Later it schedules the tasks from low QoS group.

N. Stakhanova, et. al., [10] describes a fully decentralized approach in computing the trustworthiness of an entity. This approach computes the trustworthiness of the target entity by taking into consideration the reputation value of the target entity and the credibility of the recommenders' providing feedback about the target entity. The credibility is estimated by the reputation value of the recommenders' stored in the local repository of the entity computing the trustworthiness.

C. -M. Wang, et. al., [3] proposed resource selection mechanism is based on two main algorithms, namely Minimum Execution Time (MET) and Minimum Completion Time (MCT) heuristics. In these algorithms, the processing time for all jobs that are queued at resources is expressed as resource availability which is defined as the earliest time that the resource can complete the execution of all jobs that has previously assigned to it. In MET, each job is assigned to a resource based solely on resource processing speed and job size without considering resource availability. In contrast, MCT assigns each job to a resource that has minimum completion time for that job by taking into account the resource availability. Indeed, there is little work on mechanisms for best resource selection especially in enterprise Grid systems and it is the first attempt that provides a mechanism on how to select the best resources in the presence of job that update data

3. Resource Selection Framework

We have implemented a general-purpose resource selection framework based on the reliability and performance prediction technique. It accepts the user request and identifies the trust full nodes from the available list of resources. From the list of trustful node it selects the best node to execute the task.

3.1. System Architecture

The architecture of our resource selection framework is shown in Figure 1. First phase check the reliability of the resources based upon their trust value and next phase select the best resource for given job by classified the resources into four different classes.

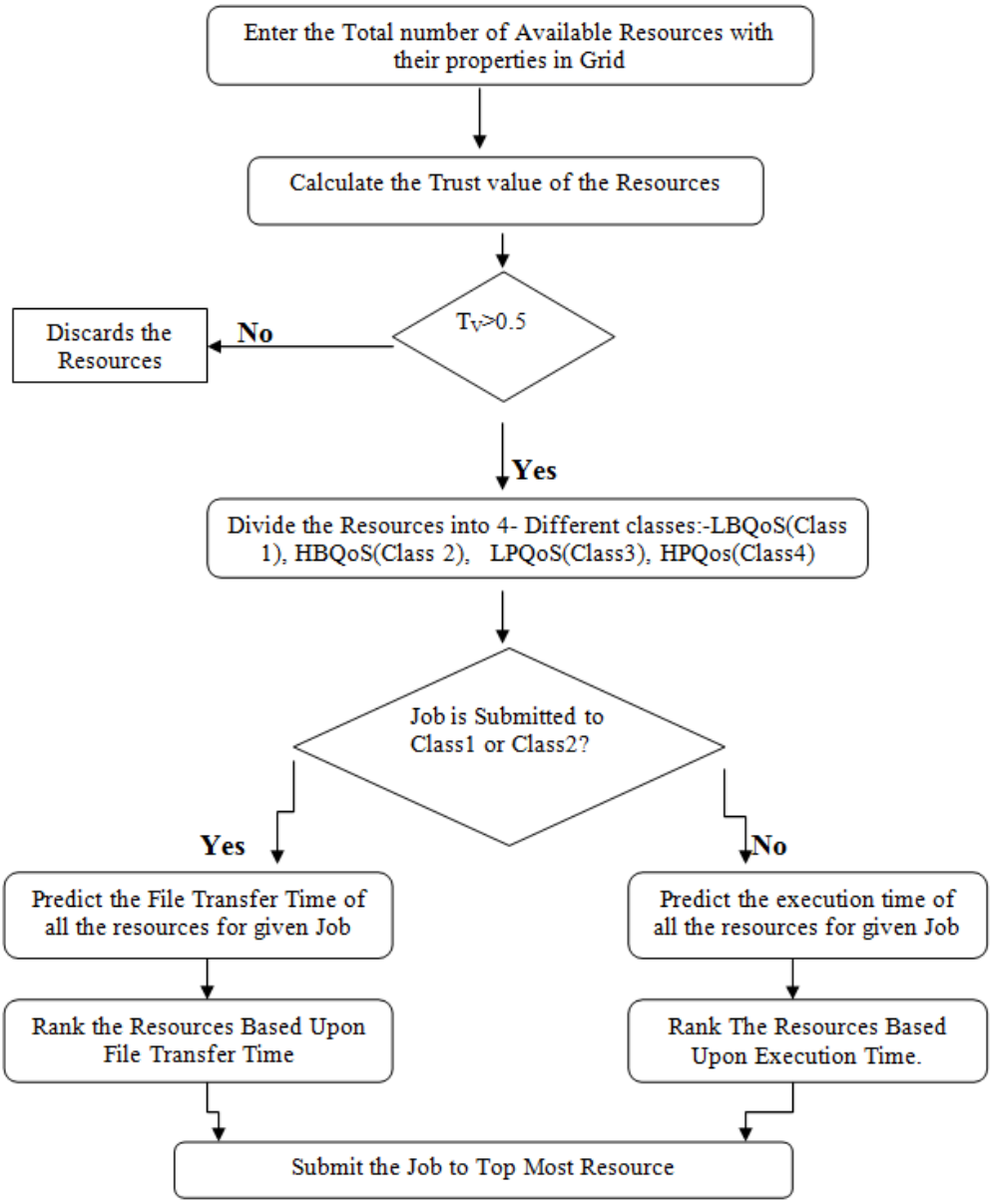

Figure 1. Architecture of Resource Selection Framework

3.1.1. First Phase (resource filtering phase): Resource Filtering phase is the first phase of our resource selection approach. Basically this step focuses on the reliability of the resource. In order to determine the reliability of the resources we evaluate the Trust value based upon certain factors which include Usage Score, Affordability, Success-Rate, Self protection capability..The resources having the trust value less than 0.5 are rejected and not considered for the next phase of resource selection. Following are the detail listing of factors:

(i)Usage score or Backlink score(U_s): A usage score is calculated on the basis of submission rate of the job. If the resource of an organization links to the resource of another organization means that it is casting a vote as an indication that the other resource is good. In other words,

higher the rate of submission of jobs means higher the usage score. In nutshell it is defined as the sum of the usage score of the resources which provides the job submission [12]. The equation is as follows:

$$US(A)=(1-d)+d\ (US(T_1)\ /C(T_1)\ +..+\ US(T_n)\ /C(T_n)\ ) \quad (1)$$

Where,

$US(A)$ = UsageScore of resource A

$US(T_1)$ = UsageScore of resource T1 which uses the resource A

$C(T_1)$ = the number of times user in the organization T1 submits job to current resource in an organization

d = damping factor which usually set to0.85

(ii)Affordability (A_A): The ratio between the number of times a resource being available to the Grid and the number of attempts made to access the resource.

It is defined as:

$$A_A= N_t / N_c \quad (2)$$

Where,

N_t = Number of times resource being available to Grid resource A

N_c=Number of attempts made to access the resource

(iii)Success Rate(S_A): The number of successful executions of jobs by a computational resource against the total number of jobs submitted to the resource that indirectly reveals the expertise of the resource provider.

It is defined as:

$$S_A= J_S / J_T \quad (3)$$

Where,

J_S = Number of jobs successfully executed by the resource.

J_T = Total Number of jobs executed by the resource.

(iv)Self-protection Capability (SPC): The self-protection capability of an entity is calculated by aggregating the security factors. The values of these factors differ in the range between 0 and 1, which are determined by trust strategy. Based on their contributions to security, a proportion is given to each security factor as a final point aggregated to compute the self-protection capability.

The security factors and their proportions are listed in the following table:

Table 1. Security Factors and their Proportion

Security Factors	**Proportion(P)**
IDS Capabilities	0.825
Anti-virus Capabilities	0.85
Firewall Capabilities	0.9
Authentication Mechanism	0.8
Secured File Storage Capabilities	0.7
Interoperability	0.6
Secured Job Execution	0.75

Based on these security factors SPC is calculated as:

$$SPC = \frac{\sum_{i=1}^{n} P(i)}{n} \quad (4)$$

Where,
n = the total number of factors.
P(i) = the proportion.

Based on all these factors we can calculate the Tvalue of the resource as follows:

$$Tvalue(A) = \frac{U_S * A_A * S_A * SPC}{4} \quad (5)$$

3.1.2. Second Phase (performance prediction and resource ranking): We can increase the performance of our Grid if we are able to predict the performance of resources corresponding to jobs. There are two types of jobs in Grid one is communicational based which require resources having good bandwidth another is computational based jobs which require resources having good processing speed of CPU. When the new job arrive in our Grid and we are able to predict which resource is most appropriate for that job then it will eventually increase the performance and Quality of Service of our Grid. The term quality of service (QoS) has varied meanings based on the different context while applying it to Grid resources. For example when QoS is used in the network context it means the desirable bandwidth for the application. Similarly when it use for CPU it means the requested speed like FLOPS or the utilization of the underlying CPU. In the second phase of our proposed work, both QoS of a network and QoS for CPU is considered. In current Grid job scheduling, it may be possible that a job with low QoS requested can be executed on a resource providing high quality of service. So when the job with high QoS arrives for execution it has to wait till the low QoS requested job completed, this will increase the jobs completion time, makespan time and decrease the overall performance of the Grid. To overcome this we classify the resources based upon their characteristics into four different classes i.e. Low Bandwidth Quality of Service (LBQoS),High Bandwidth Quality of Service(HBQoS), Low Processing Quality of Service (LPQoS), High Processing Quality of Service (HPQoS). Whenever a High Quality computational job arrives then it is directly assigned to HPQoS class and in this class we predict the execution time of each resources corresponding to given job and rank the resources based upon their execution time i.e. the resource having lower execution time will have higher rank. Now we submit our job to highest rank resource. Similarly Low Quality Computational jobs assigned to LPQoS class and we predict the execution time of each resource for given job and rank the resources of that class, job is submitted to highly ranked resource. Same procedure is applied for HBQoS and LBQoS but the only difference is that in this case we rank the resource based upon File Transfer Time instead of execution time i.e. lesser the File Transfer Time higher the Bandwidth. Our proposed algorithm is designed for both dedicated and non dedicated distributed systems which are shared asynchronously by both local and remote users. The proposed scheduling algorithm is shown in Figure 1.

Step1. Enter the total number of resources with their properties.

Step 2. Calculate the Trust value of the resources based upon certain factors i.e. usage score, Affordability, Sucessrate, self protection capability.

Step3. If Trust value of the resource is less than 0.5 discard the resource and if Trust value of the resource if greater than 0.5 go to step 4

Step4. We divide the resources in the four different classes. Class1 Low QoS in term of bandwith and Class2 high QoS in term of bandwith. Class3 Low QoS in term of CPU Processing speed.and Class4 high QoS in term of CPU Processing speed.

Step5. Calculate the Job requirement.

Step6. Based on the requirement calculate the class on which job is submitted.

If the job is requested for high processing speed then it is submitted to the class4 resources.

Else If the job requested for low processing speed then it is submitted to the class3 resources .
Else if If the job is requested for high Bandwith then it is submitted to the class2 resources
Else
Job is submitted to class1 resources.
End if

Step 7. If job is submitted to class1 or class2 then we calculate the File Transfer Time of all the resources of that class for given job and ranked the resources based upon File Transfer Time.
Else we calculate the execution time of all the resources of that class for given job and ranked them based upon execution time.

Step 8. Submit the job to top most resource.

The proposed algorithm predict the performance of the Resources corresponding to given job and submit the job to most suitable resource and also significantly decreases the completion time and makespan time and failure rate of the jobs as compare to Min-Min and Max-Min algorithm

4. Experimental Setup

We have setup a simulated Grid environment to evaluate the proposed algorithm. We used GridSim simulator for simulating Grid environment and the experimental results shown in Figs. (2),(3). We used Pentium-4 based system with CPU clock speed of 2.00 GHz, 2GB RAM running with Windows XP operating system. The behavior of the system with large no of jobs is analyzed and also compared with earlier Min-Min and Max-Min approach. In our experiment, we fixed the parameter for the hosts and used six task submission scenarios three for communicational based jobs and three for computational based jobs. The proposed algorithm and the conventional Min-Min, Max-Min are compared by their makespans , failure rate on the same set of tasks.

4.1. Communicational Based Jobs

Communicational based jobs are required bandwidth to complete their operations like transfer of file from one node to another node. So in this experiment we take three scenario on which jobs are required HBQoS (High Bandwidth Quality of Service).

First scenario:- Most of the jobs(80% jobs) require high QoS in term of bandwidth not less than 2Gb/s.

Second scenario:- half of the jobs(50% jobs) require high QoS in term of bandwidth no less than 2Gb/s.

Third scenario:- very few of the jobs(20% jobs) require high QoS in term of bandwidth no less than 2Gb/s.

For each of the scenarios, we compare the performance of the conventional Min-Min and Max-Min heuristics and proposed algorithm. For each scenario and each heuristic we create 100 tasks 100 times independently and get the average makespan of the 100 times. Table 1 and Figure 2 shows the comparison. The data is in seconds.

Table 2. Makespan for Three Scenarios for Three Heuristics

Scenario	Min-Min	Max-Min	Proposed	Improvement over Min-Min	Improvement over Max-Min
First	129.45	138.35	85.80	33.72%	37.98%
Second	182.60	198.58	105.25	42.36%	46.99%
Third	284.89	297.76	270.75	4.96%	9.07%

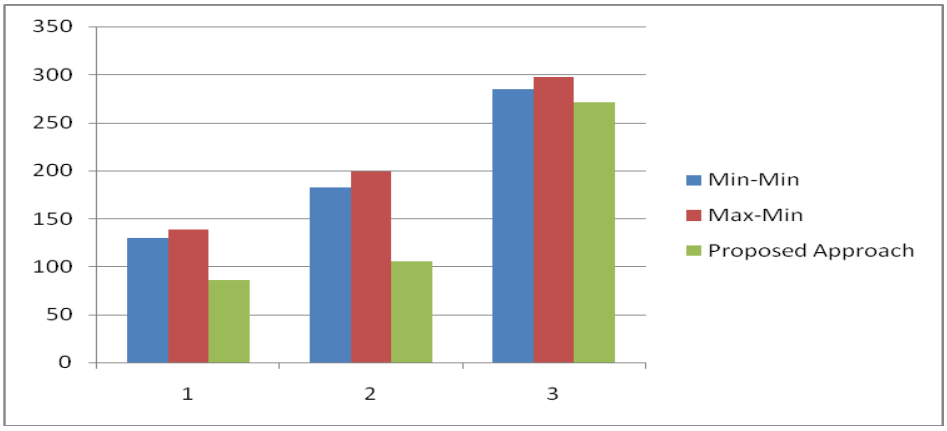

Figure 2. Makespan for Communicational based of Three Heuristics

As shown in Figure 2, for all three scenarios the proposed algorithm outperforms the traditional Min-Min and Max-Min heuristics by 22.63% and 27.24% shorter makespam. For scenario (a), where the tasks that require high QoS are in higher density (80%), a satisfactory performance gain 33.72% and 37.98% respectively. For scenario (b) where the tasks that require high QoS and the tasks that require low QoS are evenly distributed, the performance gain reaches as high as 42.36% and 46.99% respectively

For scenario (c), where the tasks that require high QoS is only 20%, the performance gain of the proposed approach is relatively small, i.e.,4.96% and 9.07% respectively better than the conventional Min-Min and Max-Min.

In the second scenario where the tasks are evenly distributed, we obtained the best results.

4.2. Computational Based Jobs

Computational based jobs required high proceesing speed to complte their operations. So in this experiment we take three scenario on which jobs are required HPQoS (High Processing Quality of Service).

First scenario:- Most of the jobs(80% jobs) require high QoS in term of CPU processing speed no less than 2.5Ghz

Second scenario:- half of the jobs(50% jobs) require high QoS in term of CPU processing speed no less than 2Ghz.

Third scenario:- very few of the jobs(20% jobs) require high QoS in term of CPU processing speed no less than 2Ghz.

Table 3. Makespan for Three Scenario for Three Heuristics

Scenario	Min-Min	Max-Min	Proposed	Improvement over Min-Min	Improvement over Max-Min
First	137.94	141.65	106.95	22.46	24.49
Second	180.60	202.28	129.25	28.43	36.10
Third	294.96	303.96	284.35	3.59	6.45

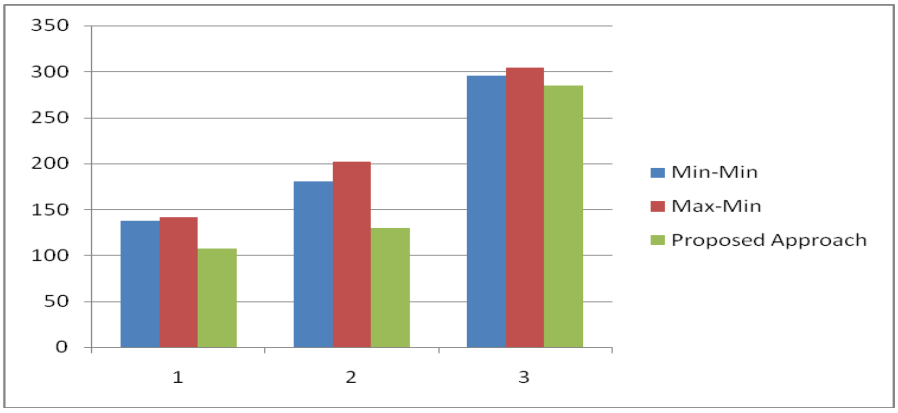

Figure 3. Makespan for Computational Based Scenario of Three Heuristics

For our test scenarios, we introduce the jobs based on computation and obtained the makespan using proposed approach 15.15% and 19.65% shorter than that using the conventional Min-Min and Max-Min. For scenario (a), where the tasks that require high QoS are in higher density (80%), a satisfactory performance gain 22.46% and 24.49% respectively is acquired. For scenario (b) where the tasks that require high QoS and the tasks that require low QoS are evenly distributed, the performance gain is quiet considerable and it sweeps as high as 28.43% and 36.10% respectively. For scenario (c), where the tasks that require high QoS is only 20%, the performance gain of the approach is relatively small, i.e., 3.59% and 6.45% respectively better than the conventional Min-Min and Max-Min

As clearly shown in Figure 2 & Figure 3 for all six scenarios the approach outperforms the conventional Min-Min and Max-Min heuristic. We find that for all scheduling frequencies, the makespan of the approach is approximately 18.89% and 23.44% respectively shorter than that of the conventional Min-Min and Max-Min on the same set of tasks. Also it is observed that it shows best results when the high and low QoS jobs are almost equal in percentage.

5. Conclusion

Resource Selection is one of the well-known problems in distributed computing systems such as Grid environments. In this paper we propose an approach for resource selection which considers QoS for network as well as CPU based upon the types of jobs. In proposed method we first filter the resources based upon their trust value and in the second phase we classifying resource into four different classes based upon their QoS parameters and predict the performance of the resources for given jobs. In our proposed approach we effectively utilized all the resource because the jobs is submitted to appropriate resources i.e. communication high quality requested job is submitted to higher bandwidth resource and similarly computational high quality requested job is submitted to higher processing speed resource. The experimental result shows that proposed algorithm outperforms the traditional Min-Min, Max-Min heuristic on the same set of task. Also in our proposed approach we filter the resources into very first phase so the malicious and bad nodes are not allowed to enter in the resource selection process which also decreases the failure rate of our approach.

Acknowledgement

We thankful to Punjabi University Patiala for unfailing support in the successful conduction of this work.

References

[1] X.-S. He and X.-H. Sun, "QoS Guided Min-Min Heuristic for Grid Task Scheduling", Journal of Computer Science & Technology, vol. 5, **(2003)**, pp. 442-451.
[2] I. Foster, C. Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations", International Journal of High Performance Computing applications, vol. 15, no. 3, **(2001)**, pp. 200-222.
[3] C. -M. Wang, H. -M. Chen, C. -C. Hsu and J. Lee, "Dynamic resource selection heuristics for a non-reserved bidding-based Grid environment", Future Generation Computer Systems, vol. 26, **(2010)**, pp. 183-197.
[4] S. K. Garg, R. Buyya and H. J. Siegel, "Time and cost trade-off management for scheduling parallel applications on Utility Grids", **(2009)**.
[5] X. Wang and L.-F. Kong, "Study On Grid Resource Selection Method Based On Multi-Goals", Proceedings Of The Sixth International Conference On Machine Learning And Cybernetics, Hong Kong, **(2007)** August 19-22.
[6] C. Chen, L. Gu, X. Niu and Y. Yang, "An Approach for Resource Selection and Allocation in Grid Based on Trust Management System", International Conference on Future Information Networks, **(2009)** IEEE.
[7] N. Hamid, F. Haron and C. H. Yong, "Resource Discovery using PageRank Technique in Grid Environment", Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID'06), **(2006)**.
[8] M. Maheswaran, S. Ali, H. J. Siegel, et. al., "Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems", 8th IEEE Heterogeneous Computing Workshop (HCW '99), **(1999)** April, pp. 30-34.
[9] L. Kleinrock, "MEMO on Grid Computing", University of California, Los Angeles, **(1969)**.
[10] N. Stakhanova, S. Ferrero, J. Wong and Y. Cai, "A reputation based trust management in peer-to-peer network systems", In International Workshop on Security in Parallel and Distributed Systems, **(2004)**.
[11] D. Zhu, Q. Meng and J. L. Zhao "Outsourcing Resource Selection: A Rough Set Approach", Proceedings of the 40th Hawaii International Conference on System Sciences, **(2007)**.
[12] N. Hamid, F. Haron and C. H. Yong, "Resource Discovery using PageRank Technique in Grid Environment", Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid (CCGRID 06), **(2006)**.

Authors

Dr. Rajesh K. Bawa holds Master's and Ph.D degree in the area of Numerical Computing from IIT Kanpur INDIA. Presently he is Associate Professor at Department of Computer science Punjabi University, Patiala, INDIA. His present area of interest is Parallel and Scientific Computation and has published many research papers in journals of reputed publishers and Also associated with review and editing work with many journals.

Gaurav Sharma received his B.Sc. and M.Sc .degrees in computer science from the Kurukshetra University, Kurukshetra in 1999 and 2001, respectively. Also, he holds the degree of M.Tech in Computer Science from Guru Jambeshwar University, Hisar. He is currently a Ph.D. candidate in Department of Computer Science, Punjabi University, Patiala. Also, he is working as Assistant Professor in JMIT Radaur Engineering College. His research interests include Grid computing, Software engineering, and artificial intelligence.