Resource Scheduling in Hybrid Grid Environment

Dr. N. Malarvizhi
Professor & Head, Department of Information Technology
Jawahar Engineering College
Chennai, India
nmv_94@yahoo.com

Dr. N. Sankar Ram
Professor & Head,Department of Computer Science and Engineering ,
RMD Engineering College,Chennai.
n_sankarram@yahoo.com

Dr. V. Rhymend Uthariaraj
Professor & Director, Ramanujan Computing Centre
College of Engineering, Guindy
Anna University Chennai, India
rhymend@annauniv.edu

***Abstract*-- This paper deals with the resource scheduling algorithm for multi cluster hybrid grid environment. The combination of both the centralized and decentralized grid environments is collectively called as the hybrid grid environment. In this environment, each cluster has its own scheduler and cluster information system, and the whole organization is managed by one global grid scheduler with a grid information system. This mixed approach has the benefit of the shared management and administration of local and global schedulers. The local schedulers are responsible for the management of their own resources and the global scheduler manages the local schedulers. In the hybrid approach, as in the decentralized model, users submit their jobs to the appropriate cluster, and the cluster information is updated at a specific interval. The scheduler in the cluster then searches the nodes for executing the jobs in the originating cluster itself. If the number of nodes in the cluster is not satisfied, then the job will be transferred to the grid level scheduler. As in the centralized model, the grid scheduler schedules the jobs coming into it. The experimental results demonstrate that, the proposed hybrid algorithm effectively schedule the grid jobs thereby reduces total time of the jobs submitted in the grid. Also, it increases the percentage of jobs completed within the specified deadline and making the grid trustworthy.**

Keywords- Grid Resources; Total Time; Multi Clustering; Performance Benefit; Resource Scheduling

I. INTRODUCTION

The field of grid computing covers a wide scope of concepts and techniques, all of which related to the cooperation of heterogeneous computing resources separated by large distances and with differing administrative domains. Grid computing is defined as "coordinated resource sharing and problem solving in dynamic, multi institutional collaborations" [1]. The idea is using a large number of geographically distributed resources in order to solve a large problem that could not be solved with any single resource. One of the problems emerging from this concept is resource allocation, and therefore efficient job scheduling. The execution of jobs in a dynamic environment like grid often calls for efficient algorithms to schedule the resources required for successful execution of the jobs. One of the primary goals of grid computing is to share access to geographically distributed heterogeneous resources in a transparent manner. There will be many benefits when this goal is realized, including the ability to execute applications whose computational requirements exceed local resources and the reduction of total time the job spends in the grid system through job distribution across multiple computing facilities. Two important components of job distribution policy are a transfer policy and a location policy [2].The transfer policy determines whether a job is processed locally or remotely and the location policy determines the node to which a job, selected for possible remote execution, should be sent. Transfer policies use some kind of node information to determine whether the node satisfies the job requirement criteria.

In grid environment job scheduling is applied at two levels: grid and local. At grid level, a grid scheduler selects the appropriate systems for jobs, and at local level, local schedulers allocate jobs to specific resources according to a strategy. Grid and local schedulers constitute a scheduling framework which can be centralized or

decentralized. In the centralized architecture, a central scheduler is responsible for collecting system state information. Centralized policies performed well as the scheduler has the entire system state to make a job distribution decision. The obvious disadvantage of this type of policy is it can cause bottleneck problem when the grid size increases. The most common decentralized architecture is the hierarchical architecture which includes a grid scheduler, various distributed local schedulers and many resources.

In this paper, the grid comprising of multiple clusters is considered. A multi cluster system typically consists of different clusters of computers which most probably have completely different characteristics either in computational capabilities, network communication or simply the number of resources available. In each cluster there are users submitting jobs for execution. Each job begins execution only when enough machines are available to meet its needs. When a job terminates execution, all machines assigned to it are reclaimed.

II. LITERATURE REVIEW

The hierarchical architecture includes entities that belong to different levels. For example, a generic hierarchical tree model with four levels (grid, cluster, site, computing elements) is presented [3].Hierarchical scheduling strategies for grids are described, where 2-level scheduling strategies are presented [4]. The first level includes job and resource selection strategies, and the second level includes strategies for scheduling the job to the resource. To route a job to a resource for execution, the scheduler can use resource information for an effective resource selection. This information can be based on static or dynamic characteristics of resources [5]. Static characteristics do not change, for example, the number of processors. Dynamic characteristics change over time, for example, the length of resource queue. Obtaining real time global information from resources is costly and leads to high overhead [6]. This is because resources are distributed geographically, and the number may be large, and thus a large amount of communication traffic is required. An improvement could be the use of a fixed update interval [7]. In this case scheduler receives dynamic resource information only at specific times. In [8], techniques that support efficient task scheduling algorithms in real time distributed systems were studied, where deadline based task scheduling and resource allocation were considered jointly.

Some relevant work includes scheduling in distributed systems [9,10] and multi site scheduling [11] where scheduler's decisions are only based on predicted load values via time-series analysis. In some earlier studies it is assumed that the job run time to be known. This information can come from either estimates provided by users when the jobs are submitted, or predictions made by the system based on historical data [12]. None of the above can be completely accurate. In this study the job run time is calculated when the jobs are assigned to the resource for execution. However it has been shown that in general poor estimates of job run times do not significantly affect the overall system performance [13].The scheduling algorithms in [14] and [15] proposes a migration of job to balance the load when the nodes become an overloading node but does not consider the resource and network heterogeneity.

In the earlier study, MTTR scheduling strategy is implemented in centralized grid architecture [16]. In this study, the same approach is used in the hierarchical architecture so that when the grid size increases Total Time to Release (TTR) of the job can be reduced considerably. The hybrid approach reduces the bottleneck problem in the centralized architecture. The focus of this study is to reduce the TTR of the jobs submitted in the cluster. In order to reduce the communication traffic, users submit their jobs to the appropriate cluster and dynamic site information is updated at specific interval. The scheduler allocates jobs to node according to some selection criteria taking parameters of jobs and nodes into consideration. The consideration is limited to the scenario where user can submit job to the cluster in which the user resides.

III. HYBRID GRID ARCHITECTURE MODEL

The multi cluster hybrid grid architecture consists of heterogeneous clusters. Each cluster has a local scheduler, servers and nodes, while the whole is governed by a grid scheduler. The servers in the cluster have different number of nodes, and each node has a different number of Processing Elements (PE) with different processing capabilities. The servers within the cluster are interconnected through a local area network, while the clusters and the grid scheduler are connected through a wide area network.

The resource scheduling policy in the local scheduler selects the server in the cluster for job execution, based on the load, network transfer time and its computational speed. It is assumed that, once the job is allocated to the server in the cluster, it must be executed on the nodes within that server, i.e., migration between the servers within the cluster is not allowed. The user submits jobs to their own cluster. The proposed model is denoted as C/S/N, where C is the number of clusters that compose a grid, S is the number of servers in each cluster, and N is the number of nodes in each server. This model can be transformed into three specific models: C/S/N, 1/S/N, 1/1/N, depending on the value of C and S. Fig. 1 shows the model associated to a grid, with its three variants: C/S/N, 1/S/N and 1/1/N.

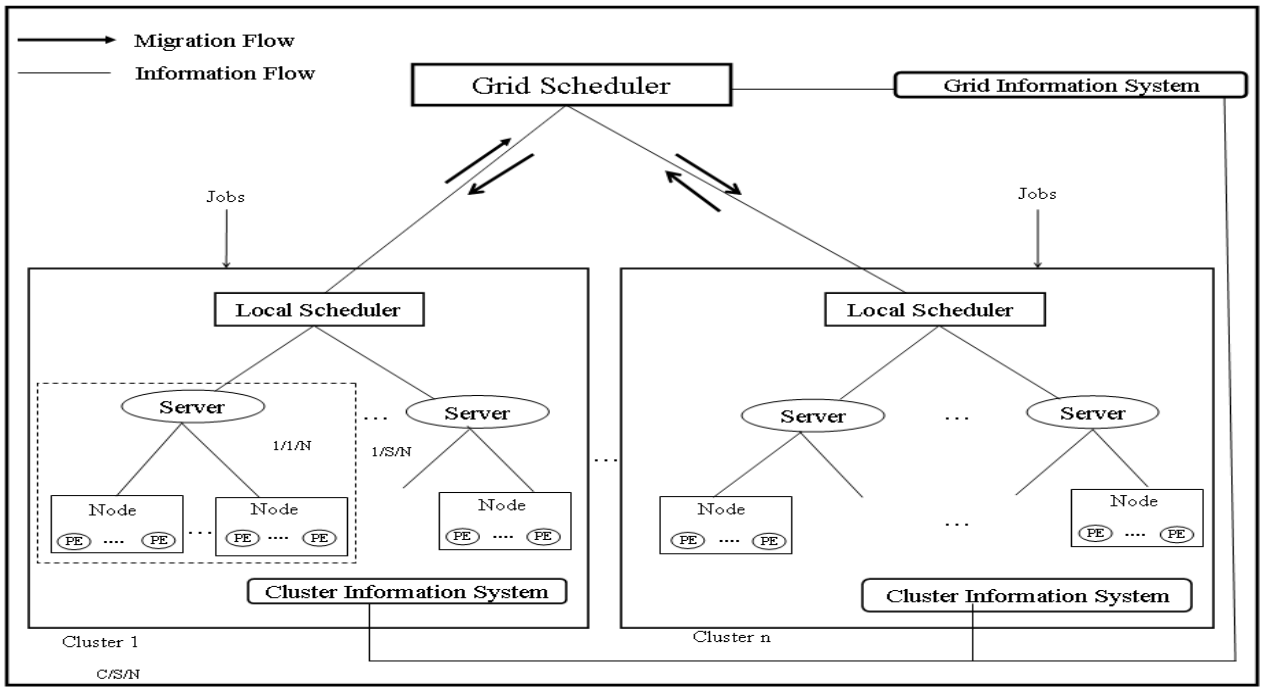

Figure 1. Hybrid grid architecture model

In a hybrid grid model, the grid information system maintains all the static and dynamic information about the clusters in the grid system. The clusters in the system update their information in the local cluster information system, which periodically sends this information to the grid information system. These updates use a finite update period T_s. At each periodic interval T_s, each component in the system exchanges its status information with its associated manager. The instant at which this information exchange takes place is called as the status exchange instant. The value of T_s is considered as 20 seconds.

A. Hybrid Resource Scheduling Algorithm

In the hybrid grid model, at any scheduling level, the following three steps strategy are used. As the description is generic, the concept of a Group (G) and an Element (E) is used. A group designs either a cluster or the grid and an element can be either a server in the cluster or the cluster in the grid respectively. The important steps in the proposed hybrid resource scheduling algorithm are given below:

Step 1: Information collection

- **For** every element E_i and at each status exchange instant period T_s
  Send the resource status information and workload to its associated manager.

  End For
- At each period T_s the group node does the following:
  - Computes its speed and capacity
  - Evaluates its current workload
  - Sends the elements the resource status information and the workload to its associated manager

Step 2: Decision making

For each job, the group node performs the following:

- Compare the minimum number of nodes requested (req) with the number of nodes available (avail) in each element.
- **If** (req < avail) **then**
  Make a decision according to approach 1; i.e., local execution

  Else

  Make a decision according to approach 2; i.e., HRS

  Endif
- **If** (req < avail) && (TTR > Deadline) **then**
  Make a decision according to approach 3; i.e., P-HRS

  Endif

Step 3: Job transferring

In order to transfer a job for execution, the following transferring criteria are proposed:

Switch

G= Cluster:

- Perform Min_TTR_Server policy with negligible transfer time
- Return;

G= Grid:

- Perform Min_TTR_Cluster policy with considerable transfer time
- Return;

End Switch

Min_TTR_Server policy

This policy is implemented in the local scheduler that determines a server for the jobs submitted to the local cluster. According to this policy, based on the collected resource status information, the local scheduler estimates the TTR of the satisfied servers under its control and the server which gives the least TTR is selected for execution. In case there are two servers with an identical TTR, any one is selected in a random manner.

Min_TTR_Cluster policy

This policy is implemented in the grid scheduler that determines the way a remote cluster is selected for a job migrated from the local cluster. Based on the collected resource status information, the grid scheduler estimates the TTR of each cluster under its control. The Grid scheduler selects the cluster that provides the minimum TTR value for submitting the job for execution.

B. Hybrid Resource Scheduling and Job Execution Approaches

This section briefs about the different resource scheduling and job execution approaches implemented in the hybrid grid environment. When a user submits a job to the local cluster, the scheduler in the cluster determines whether the job can be dispatched to the nodes in the cluster itself, or transferred to the grid scheduler for remote execution. The decision is based on the various job and resource characteristics. Three different approaches for selecting the resource and executing the job using the proposed Hybrid Resource Scheduling (HRS) algorithm are described below:

a) Local Execution

In this approach, jobs are executed only in the cluster they originate from. When the job enters the scheduler, it determines whether the minimum number of nodes required by the job is less than the available number of nodes in the servers of the associated cluster. If yes, the scheduler sorts out the servers that have a sufficient number of nodes to satisfy the job needs. For that set of servers it calculates the TTR value**.** The scheduler then filters out the servers whose TTR is greater than the deadline of the job. The job is then submitted to the server which gives minimum TTR. If the matched nodes are busy then the jobs wait in the Job Waiting Queue(JWQ) of the cluster till the nodes become free. If the deadline expires, the job will be discarded.

b) Remote Execution (labelled as HRS)

In this approach, jobs are executed either in the originating cluster, or migrated to any remote cluster through the grid scheduler. In case, the jobs cannot be handled by the originating cluster, without being discarded, the local scheduler transfers them to the grid scheduler. The Grid scheduler selects the cluster for executing the job and reschedules the job to it.

c) Performance Based Remote Execution (labelled as P-HRS)

A combination of the above approaches together with a performance benefit yields this approach. Here, the decision of job migration depends upon the performance benefit in terms of the TTR. The Performance Benefit (PB) is calculated by (1) as follows:

$$PB = \frac{Local\_Cluster\_TTR - (Remote\_Cluster\_TTR + Migration\ overhead)}{Local\_Cluster\_TTR} \quad (1)$$

The migration overhead is calculated by (2) as follows:

$$Migration\ overhead = Time\ for\ rescheduling + Transfer\ time + Time\ for\ starting\ job \quad (2)$$

If the performance benefit is greater than the migration threshold, the job will be migrated to the grid scheduler.

IV. PERFORMANCE EVALUATION AND DISCUSSION

A. Analysis of the proposed hybrid approaches based on system load

In this section, the impact of the proposed hybrid resource scheduling approaches is analysed by varying the system load. Performance metrics such as the TTR, the number of jobs completed and resource utilization are analysed. The system load is varied by varying the number of jobs submitted from 1000 to 9000, in the system having 40 clusters.

The performance of the proposed hybrid resource scheduling approaches is analyzed in terms of the TTR. Fig. 2 and Table I show the results of the simulation. For all the system loads tested, the performance of the P-HRS and HRS is better than that of the local execution strategy.

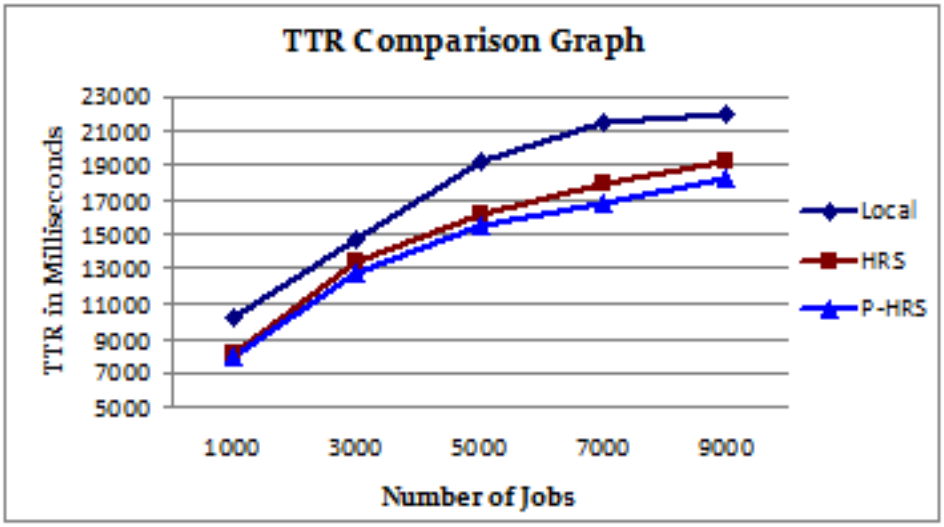

Figure 2. Effect of system load in TTR

TABLE I. PERCENTAGE OF IMPROVEMENT IN TTR OF P-HRS OVER OTHER APPROACHES WHEN VARYING SYSTEM LOAD

Approach / No. of jobs	1000	3000	5000	7000	9000
Local	22.3	12.9	19.3	21.3	16.8
HRS	1.9	4.5	3.8	6.1	4.9

It is found that the, P-HRS has an average improvement of 18.5% and 4.3% over the local and HRS approaches respectively. At a high load of 9000 jobs, the P-HRS yields an improvement of 4.9% over the HRS. This is due to P-HRS dispatches the jobs to a remote cluster, which yields a better performance benefit in terms of the TTR.

The performance analysis is focused on the comparison of number of the jobs completed by varying the system load. When the number of jobs increases, the number of jobs completed also increases in all the resource scheduling approaches. It is observed from Figure 3 that the HRS and P-HRS have more number of jobs completed than the local execution approach. This is because, in the local execution approach, some jobs get discarded due to non-availability of enough resources. In the P-HRS, the number of jobs finished within the specified deadline is more than that in the HRS approach. This is because, in the P-HRS approach the jobs can be migrated to a powerful cluster which provides a performance benefit in terms of a lesser TTR.

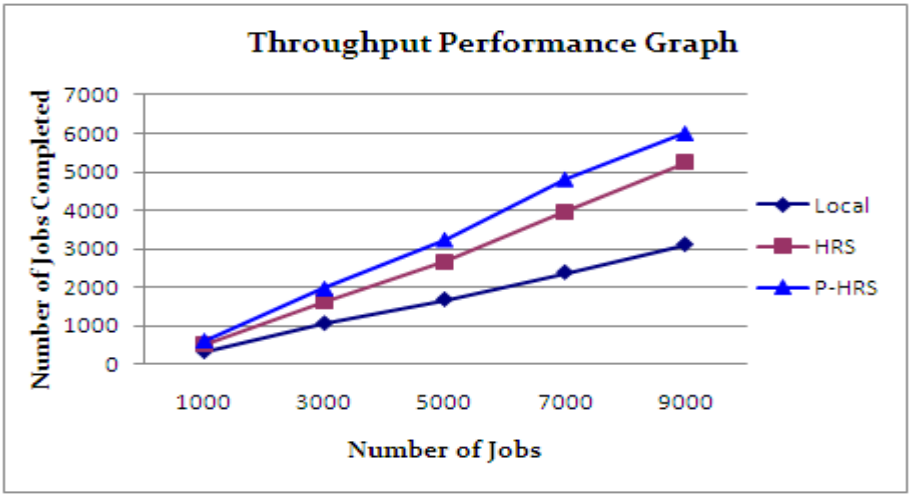

Figure 3. Effect of system load in job completion

TABLE II. PERCENTAGE OF IMPROVEMENT IN JOB COMPLETION OF P-HRS OVER OTHER APPROACHES WHEN VARYING SYSTEM LOAD

Approach / No. of jobs	1000	3000	5000	7000	9000
Local	46.9	46.2	48.3	50.6	48.4
HRS	16.5	16.9	17.9	17.5	12.7

From Fig. 3 and Table II, it is observed that the P-HRS gives more number of jobs completed across all values of the system load. The P-HRS has an average improvement of 48.1% and 16.3% over the local execution and HRS approaches respectively.

The percentage of resource utilization having 40 clusters in the system is examined. When the jobs are submitted to the system, the local execution approach executes the jobs only within the originating cluster, whereas in the HRS and P-HRS approaches, jobs are executed either in the originating cluster or in any of the remote clusters. Therefore, the number of resources that meets the job requirements is lesser in the local execution approach, than in the other approaches. Therefore, the percentage of resource utilization using the HRS and P-HRS is comparitively higher than that of the local execution approach. The resource utilization of the HRS is closely behind that of the P-HRS approach, since both the approaches utilize nodes in more than one cluster for job execution. Fig. 4 shows the resource utilization graph on varying the system load.

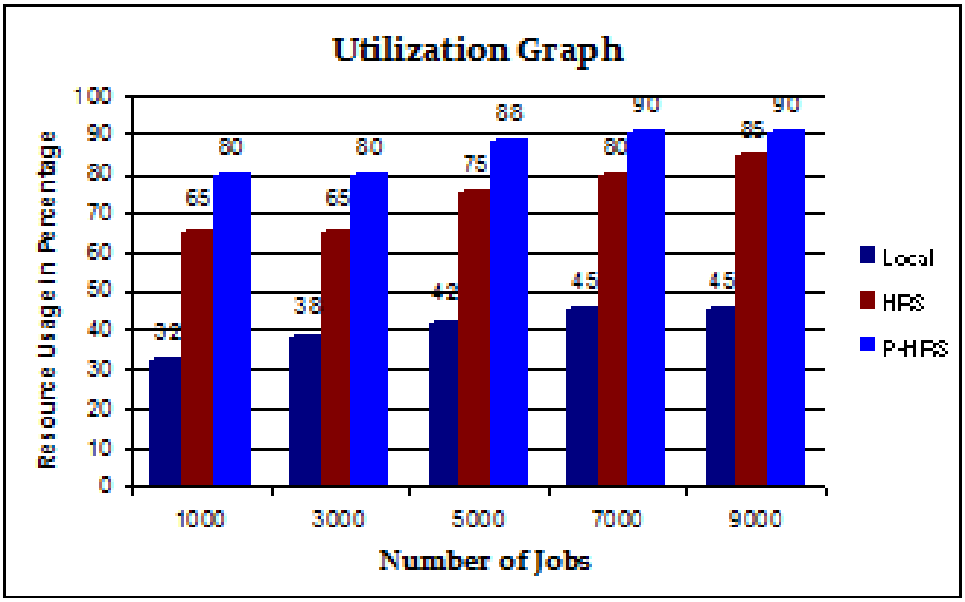

Figure 4. Effect of system load in resource utilization

At the system load of 1000 jobs, the HRS and P-HRS have an average improvement of 52.9% and 45.4% than the local execution approach respectively. The P-HRS has an average improvement of 13.8% over the HRS approach.

B. Analysis of the proposed hybrid approaches based on system size

In this section, the impact of the proposed resource scheduling approaches is analysed, based on the system size. Performance metrics such as the TTR, the number of jobs completed and resource utilization are examined. Initially, jobs are processed in the grid consisting of small number of a cluster like 10 and 20. Then, more clusters are added to the grid environment until all the 100 clusters are used.

The performance of the proposed hybrid resource scheduling approaches is analyzed in terms of the TTR. For all the system sizes tested, the performance of the P-HRS and HRS is better than that of the local execution approach. Fig. 5 depicts the simulation results for the TTR as a function of number of clusters. The results show that the P-HRS provides a lesser TTR for the same number of clusters, compared to the local and HRS approaches. This is because, the P-HRS assigns jobs to a remote cluster which provides better performance benefit in terms of a lesser TTR. The percentage of improvement of the P-HRS over the local and HRS approaches is given in Table III.

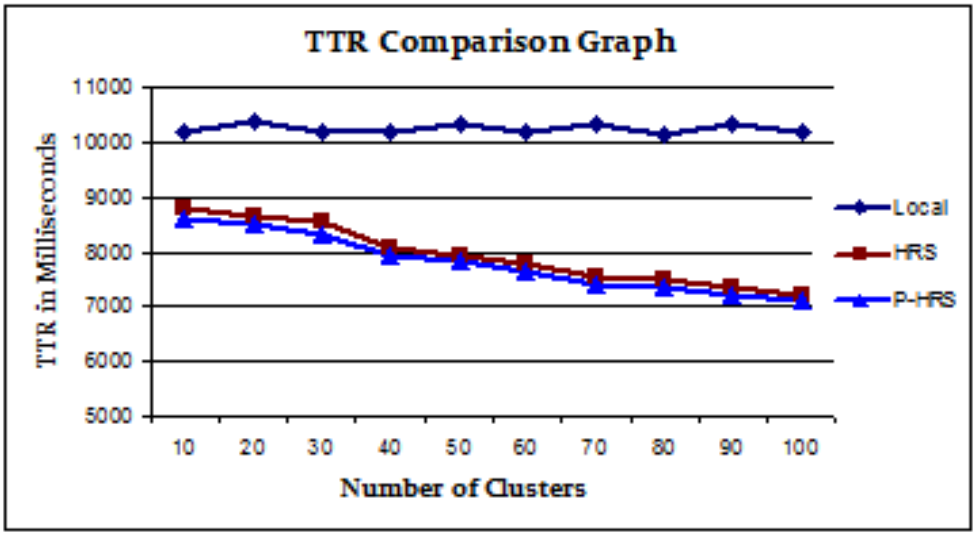

Figure 5. Effect of system size in TTR

TABLE III. PERCENTAGE OF IMPROVEMENT IN TTR OF P-HRS OVER OTHER APPROACHES WHEN VARYING SYSTEM SIZE

Approach / No. of clusters	10	20	30	40	50	60	70	80	90	100
Local	15.9	18.4	18.7	22.3	24.2	24.6	27.2	28.7	30.2	30.8
HRS	1.9	2.2	3.1	1.9	1.1	1.8	1.9	2.7	2.1	1.1

At the system load of 1000 jobs, the improvement that the P-HRS offers over local execution varies from 15.9% to 30.2% and that of the HRS varies from 1.1% to 3.1%. The P-HRS has an average improvement of 24.1% and 1.9% over the local execution and HRS approaches respectively.

The performance is analyzed by comparing the number of jobs completed in the system with 1000 jobs, and varying the number of clusters. From Fig. 6 and Table IV, it is observed that the HRS and P-HRS approaches have more number of job completions than local execution. This is because; in local execution some jobs get discarded due to the non-availability of enough resources, or the expiring of the deadline.

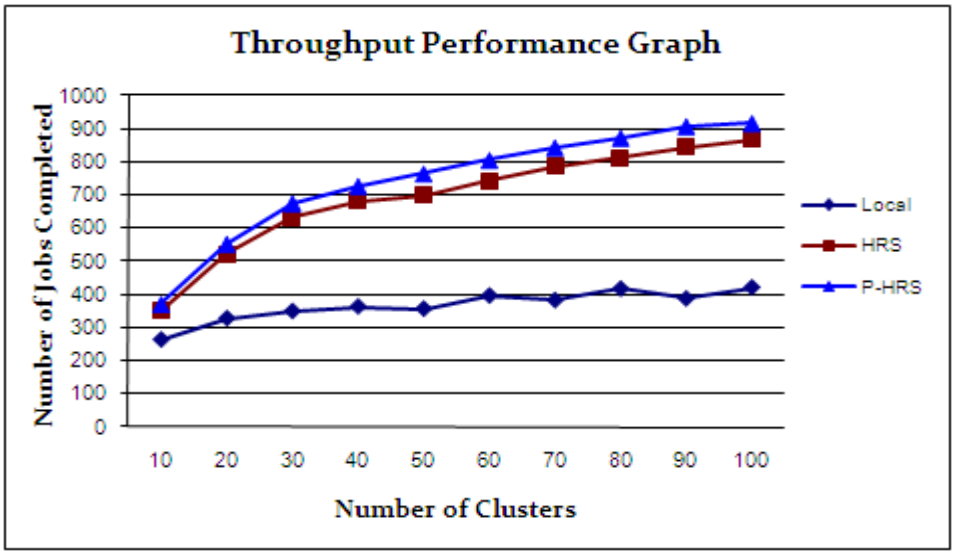

Figure 6. Effect of system size in job completion

TABLE IV. PERCENTAGE OF IMPROVEMENT IN JOB COMPLETION OF P-HRS OVER OTHER APPROACHES WHEN VARYING SYSTEM SIZE

Approach / No. of clusters	10	20	30	40	50	60	70	80	90	100
Local	29.6	40.9	48.4	50.1	53.5	50.9	54.7	52.2	57.4	54.1
HRS	5.9	5.8	6.5	6.2	8.8	7.8	6.9	6.7	6.9	5.5

With varying system sizes, the improvement that the P-HRS offers over local execution varies from 29.6% to 57.4%, and that of the HRS varies from 5.5% to 8.8%.The P-HRS has an average improvement of 49.2% and 6.7% over the local execution and HRS approaches respectively.

The percentage of resource utilization in the system with 1000 jobs and varying number of clusters is examined. When the number of clusters increases, the percentage of resource utilization decreases for all the approaches. This is because, the number of nodes utilized within the resource is increased, with an increased number of clusters for the given number of jobs. Fig. 7 shows the percentage of resource utilization for different number of clusters with the system load of 1000 jobs. For the same load, the HRS and P-HRS approaches show higher resource utilization than local execution approach as HRS and P-HRS use nodes in more than one cluster.

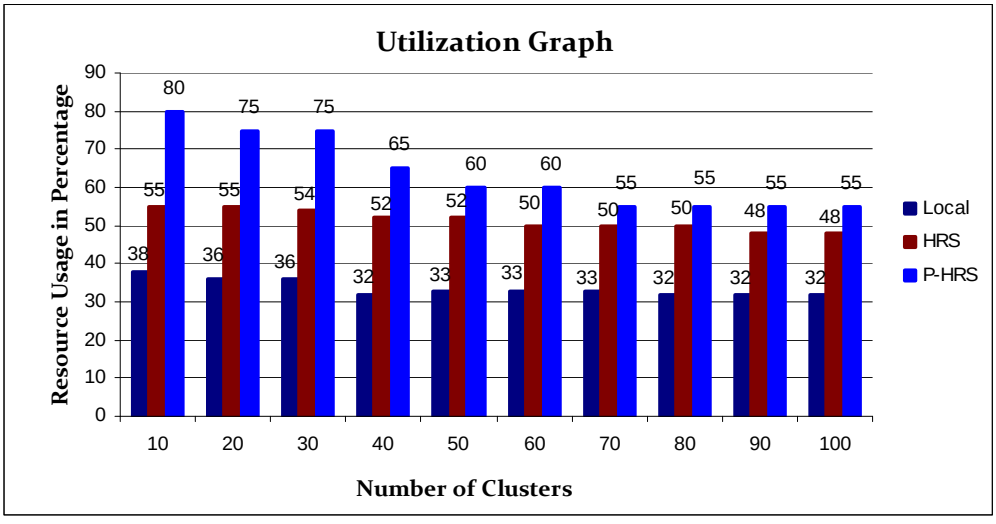

Figure 7. Effect of system size in resource utilization

The analysis illustrates that the local execution strategy underutilizes the resources regardless of the increase in the number of clusters. At the system load of 1000 jobs, the improvement that the P-HRS offers over local execution varies from 40% to 52.5% and that of the HRS varies from 9.1% to 31.3%.The P-HRS has an average improvement of 46.3% and 17.9% over the local execution and HRS approaches respectively.

V. CONCLUSION

In this paper, in the hybrid multi cluster grid environment, the proposed hybrid resource scheduling approaches are implemented and analysed at different levels of hierarchy, to optimise various performance metrics such as the TTR, the number of jobs completed and resource utilization.

REFERENCES

[1] Foster, I., C.Kesselman and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organization, International Journal of Supercomputer Applications" 15(2001).

[2] D.L. Eager, E.D, Lazowska and J.Zahorajan, "Adaptive Load Sharing in Homogeneous Distributed Systems", IEEE SE-12, No.5, 1986, pp. 662-675.

[3] B. Yagoubi, Y. Simani, "Dynamic Load Balancing Strategy for Grid Computing" Transactions on Engineering Computing and Technology, volume 13, May 2006, pp 260-265.

[4] A. Tchernykh, J.M. Ramirez, A. Avetisyan, N. Kuzjurin, D. Grushin, S. Zhuk, "Two Level Job Scheduling strategies for a Computational Grid", In Parallel Processing and Applied Mathematics, Springer- Verlag, 2006, pp 774-781.

[5] Y.Cardinale, H.Casanova, "An Evaluation of Job Scheduling Strategies for Divisible Loads on Grid Platforms", In Proceedings of the high Performance Computing & Simulation Conference, Germany, May 2006.

[6] Y.Patel, J. Darlington, "Allocating QoS-Constrained Workflow- based Jobs in a Multi-Cluster Grid Through Queuing Theory Approach", Parallel and Distributed Processing and Applications, 4th International Symposium, ISPA 2006, Springer-Verlag, pp. 499-510.

[7] E. Caron, V. Garonne, A. Tsaregorodtsev, "Definition, modeling and simulation of a grid computing scheduling system for high throughput computing", Future Generation on Computer Systems, Elsevier, Vol 23, Issue 8, 2007, pp. 968-976.

[8] Y. Chen, H. Huang, W.T, Tsai, "Scheduling Simulation in a Distributed Wireless Embedded System", Simulation Transactions of the Society for Modeling and Simulation International, Vol.81,Issue 6, June 2005, pp.425-436.

[9] H. D. Karatza, "Performance of Gang Scheduling Policies in the Presence of Critical Sporadic Jobs in Distributed Systems", Proceedings of the International Symposium on Performance Evaluation of Computer and Telecommunications Systems-SPECTS 2007, San Diego, CA, 2007, pp. 547-554.

[10] H. D. Karatza, "Scheduling Gangs in a Distributed Systems", International Journal of Simulation Systems, Science Technology, UK Simulation Society, Col 7, no: 1, 2006, pp. 15-22.

[11] M. Loannidou, H. Karatza, "Multi-site Scheduling with Multiple Job Reservations and Forecast Methods" , In Proceedings of International and Symposium on Parallel and Distributed Processing and Applications, Volume 4330 of Lecture Notes in Computer Science, Springer 2006, pp.894-903.

[12] Stavrinidis,G. H.D. Karatza, "Performance Evaluation of Gang Scheduling in Distributed Real Time Systems with possible software faults, Proceedings of the 2008 International Symposium on Performance Evaluation of Computer and Telecommunication Systems-SPECTS 2008,June 16-18,Scotland, UK, pp. 1-7.

[13] Tchernykh, A, M.Ramirez, A. Avetisyan, N.Kuzjurin, D. Grushin, S.Zhuk, "Two Level Job Scheduling Strategies for a Computational Grid, In Parallel Processing and Applied Mathematics, Springer-Verlag, 2006, 774-781.

[14] Y. Hu, R. Blake, and D. Emerson, "An optimal migration algorithm for dynamic load balancing," *Concurrency: Practice and Experienc*e, vol. 10, pp. 467–483, 1998.

[15] Zikos, S., Karatza, H.D., 2008. Resource allocation strategies in a 2-level hierarchical grid system. In: Proceedings of the 41st Annual Simulation Symposium (ANSS), April 13–16, 2008. IEEE Computer Society Press, SCS, pp. 157–164.

[16] N.Malarvizhi,V.Rhymend Uthariaraj , "A New Mechanism for Job Scheduling in Computational Grid Network Environments " in proceedings of 5th International Conference on Active Media Technology ,Volume 5820 of Lecture Notes in Computer Science, Springer, 2009,pp. 490-500.

AUTHORS PROFILE

Dr. N.Malarvizhi, Professor and Head of Information Technology Department holds a Doctorate in the faculty of Information and Communication Engineering, after an illustrious career in academia for nearly 12 years in reputed Engineering Colleges in Tamilnadu. Her professional qualifications include PhD in January 2012 from Anna University Chennai , Post graduation M.E Computer Science and Engineering in 2002 from Madurai Kamaraj University ,and B.E Computer Science and Engineering in 1994 from Madurai Kamaraj University. Her research interest includes Parallel and Distributed Computing, Grid Computing, Cloud Computing, Computer Networks, System Software, Computer Architecture. She has published more than 15 papers in the international journals and conferences.

Dr. N.Sankar Ram received the Bachelor of Engineering and Master of Engineering in Computer Science from M.K.University, Madurai, Tamilnadu, India and Ph.D degree at Anna University, Chennai, Tamilnadu, India. He is a Professor in the department of Computer Science and Engineering at Velammal Engineering College, Chennai, India. He is having 14 years of teaching experience and his research interest includes Software architecture, Computer Networks. He has published several papers in the international journals and conferences.

Dr. V.Rhymend Uthaiaraj received his PhD degree in computer science and Engineering from Anna University Chennai. He is currently working as a Professor and Director in the Department of Ramanujan Computing Centre, Anna University Chennai. His area of interest includes Computer Networks, Network Security, Computer Algorithms and Modeling, Mathematical Programming. He has published more than 40 papers in the international journals and conferences