Evaluation Of Scheduling And Load Balancing Techniques In Mobile Grid Architecture

[1] Debabrata Singh, [2] Sandeep Nanda, [3]Sarbeswara Hota,[4]Manas Kumar Nanda

ITER,SOA UNIVERSITY,BBSR,ODISHA ,INDIA

Abstract

Recent advances in mobile communications and computing are strong interest of the scientific community in the *Grid* have led to research into the *Mobile Grid*. Based on a realistic Mobile Grid architecture we formulate the problem of job scheduling and load balancing techniques in a mobile environment and performance metrics. We extended the work by introducing a new scheduling policy and load balancing policy based on the notion of *installments and intra cluster load balancing algorithm* and continue the evaluation of the expanded set of scheduling and load balancing strategies in an effort to overcome the intermittent character of connectivity in a mobile environment. On real wireless traces, we demonstrated the superiority of the proposed policy, and showed the feasibility of a Mobile Grid system and design the efficient scheduling and load balancing policies subject to the underlying mobility were based a small part of the offered resources is wasted and a small part of the workload has to be processed again .The size of the installments increases as the size of the aborted fragments of the workload increases.

Keywords : **Mobile grid, replication, intermittent connectivity, wireless traces, heuristics**

1. Introduction

Grid computing emerged resource sharing and problem solving in dynamic, multi-institutional virtual organizations [1]. A grid computing system is essentially a large scaled distributed system designed to aggregate resources from multiple sites. Users of such systems have opportunity to take an advantage of enormous computational, storage or bandwidth resources that would be impossible to attain. In many cases these resources would be wasted if not aggregated inside a grid.

Grid systems [1] are very large-scale, generalized network computing systems that can scale to Internet size environments with resources distributed across multiple organizations and administrative domains. Mobile Grid Computing is making Grid Services available and accessible at anytime and anywhere from mobile devices. Advantages of mobile grid computing is mobile to mobile and mobile to desktop collaboration for resource sharing, improvement of user experience, convenience and some new application scenarios[2]. Increasing number of new autonomous , portable devices has become significant part of everyday life and work that leads to a decentralized , location independent wireless computing environment. It is natural to consider the extension of the idea of resource sharing to mobile and wireless communication environments[3] . There are different approaches on the exact character of the extension . whether mobile devices are considered as powerful enough to provide their resources or not. Here we discussed that as the number of available mobile devices is nowadays enormous and the computational power is increasingly, so then the aggregated sum of their resources can be exploited. The mobile devices are resource constrained as they can be incorporated in a grid as resource consumers. Mobile devices are increasingly becoming powerful enough to also participate in grid systems as resource providers. Mobile devices face resource limitations in comparison to their stationary counterparts , but the vast number mobile devices and their computational power constantly increases that lead us to the assessment that the aggregated sum of their resources can be exploited to overcome the limitations. So for this we intended to enrich the set of examined policy by proposing the installments policy and load balancing algorithms .

2. Scheduling in mobile grid

A. Problem Formulation :

The core functionality of an MGS in the proposed architecture is to receive a job and, in the context of *divisible load* applications, divide the submitted workload into tasks for submission to its descendant MGSs. At the lowest level, an L-MGS is responsible for distributing the received task load to the MNs currently residing in the WLAN it serves.

The number of available nodes N obviously varies in time as MNs roam in the wireless infrastructure of the campus. The whole process consists of three distinct steps: the transfer of the input workload to MN_i , i ε {0,1,…,N-1}. , the subtask execution and return of the results back to the L-MGS. In absence of disconnection events each step requires t_{in}^i, t_{exec}^i and t_{out}^i amount of time to complete. We define t_{total}^i as :

$t_{total}^i = t_{exec}^i + t_{in}^i + t_{out}^i$, i ε{0,1,…..,N-1}

The load will be distributed to available MNs :

$T_{TOTAL}=T_{IN}+T_{EXEC}+T_{OUT}$

We make the following assumptions :

- The total task load is equally distributed to all participating MNs.
- The execution of a subtask may begin only after the entirety of the input data has been received.
- The output data of a subtask can be returned only after the execution has completed.
- The output data of a subtask must be returned in whole in order to be usable.

The communication to computation ratio of a divisible load application is :

$$CCR=\frac{Communication\ \ Cost}{Computation\ \ Cost}$$

The Communication Cost factor denotes the time required by a MN to successfully receive a certain amount of workload in the absence of disconnection events and is subject to the available bandwidth[4]. The Computation Cost denotes the time required for the same amount of workload to be processed and is subject to the device characteristics and the usage of the device by its owner.

B. Performance Metrics :

In order to evaluate the performance of the scheduling strategies we need to define suitable metrics. These metrics include:

- *Response Time* (*RT*): the time required for the entire set of result data to be gathered at the scheduler.

$$RT = \max RT^i,\ i \in \{0,1,\ldots..,N-1\} \qquad (1)$$

Resource Waste (*RW*): the amount of resources wasted in the effort to compensate for intermittent connectivity. We further subdivide RW into RW_C and RW_N with respect to the type of resources wasted i.e. computational and network resources, respectively. *RW* is measured as a percentage of the actual resources required for the completion of the task.

Speedup : the comparison of the achieved *RT* with that of a single node execution. It reveals the actual degree of parallelism achieved.

3. Proposed Architecture

We have proposed a hierarchical, campus-wide computational Mobile Grid system architecture [6]. In the proposed architecture, depicted in Fig. 1, mobile nodes (MNs), willing to offer their computational resources, move between Access Points (APs) of the campus. This willingness is based on the expectation of *reciprocity* [7]. In our work we have considered *divisible load* applications [8] (e.g. query processing [9]) in which a job can be divided into tasks that can be carried out independently of each other. These tasks are distributed by the *Local-Mobile Grid Schedulers* (L-MGSs) to the collaborating MNs, which process them and return the results back. In other levels of the hierarchy *Intermediate-MGSs (I-MGSs)* may act as *meta-schedulers*. This work focuses on the last level of the hierarchy and more specifically on the load distribution performed by L-MGSs.

These traces provided us with realistic information on the mobility and connectivity characteristics of the MNs in the campus. We pointed out the important mobile networking parameters affecting the performance of a Mobile Grid system and showed that disconnection events impose a severe impact on the turn-around time of jobs executed by MNs. On an effort to smooth the effects of intermittent connectivity we examined the performance of a simple *task replication* scheme [8]. The results were very promising with respect to the achieved turn-around time. We continued our research by also investigating whether the execution of a task in a MN should be aborted upon disconnection or not, in order for the task to be rescheduled, coming up with a positive answer with respect to the resulting turn-around times [9]. In this work, we enrich the set of examined policies by proposing the *installments* policy, in which task load is further partitioned into small chunks, and demonstrate its effectiveness.

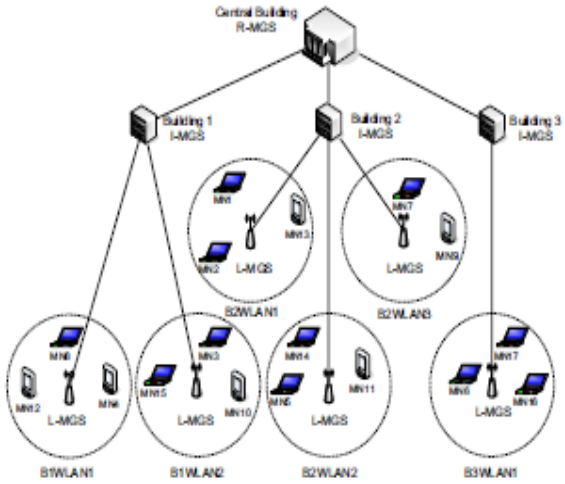


Fig. 1. Campus-wide Mobile Grid architecture.

4. Load Balancing Techniques

Load Sharing: This is the coarsest form of load distribution. Load may only be placed on idle resources, and can be viewed as a binary problem, where a resource is either idle or busy.

Load Balancing: Where load sharing is the coarsest form of load distribution, load balancing is the finest. Load balancing attempts to ensure that the workload on each resource is within a small degree, or balance criterion, of the workload present on every other resource in the system.

Load Levelling: Load levelling introduces a third category of load balancing to describe the middle ground between the two extremes of load sharing and load balancing[10][11]. Rather than trying to obtain a strictly even distribution of load across all resources, or simply utilizing idle resources, load levelling seeks to avoid congestion on any resource.

Resources are distributed in different geographic locations. Stability and performance of each resource is different. In other words, newly distributed system is dynamic and resources are composed of heterogeneous resources. Thus, an important problem is resources selection and task distribution when task are executed. This study proposed a hybrid load balancing policy, which selects effective node sets in the stage of static load balancing to lower the odds of selecting ineffective nodes and makes use of the stage of dynamic load balancing. [15]When the status of a node changes, a new substitute can be located in the shortest time to maintain the execution performance of the system.

It has four components :

1. Transfer policy determines whether a node is in a suitable state to participate in a task transfer.
2. Selection policy determines which task should be transferred.
3. Location policy determines to which node a task selected for transfer should be sent.
4. Information policy is responsible for triggering the collection of system state information.

A transfer policy requires information on the local nodes state to make decision. A location policy, is likely to require information on the state of remote nodes to make decisions.

We assume an application is composed of agents executable on any of P machines of the cluster. The structure of the application is modeled by the interdependence relationships among the agents. Specifically, we will use an undirected graph to model the application structure. An undirected graph is a generic model as a multi agent application executes perpetually and produces results continuously in response to user queries.

5. Scheduling Policies

A. Abort-Reschedule

In this policy a task executed by a mobile node must be aborted upon disconnection, in the sense that it will be rescheduled i.e. re-assigned to a new MN arriving at the WLAN served by the current L-MGS. In this policy, once a disconnection event has occurred, RTi is calculated as follows:

$$RT^{i}_{Abort} = t^{i}_{C} + t^{i}_{A} + RT^{i'}_{Abort} \quad , i \in \{0,1,\ldots\ldots,N-1\} \qquad (2)$$

In this policy, the computational and energy resources spent by a MN for the execution of a sub-task are obviously wasted if the assigned sub-task is aborted in the event of a disconnection[16]. In the following, we assume that an L-MGS is immediately aware of any disconnection event.

B. Installments

In this policy each sub-task is further partitioned into consecutive fragments (installments) of size *f*, with $f < t^{i}_{total}$ The major difference with the *Abort Reschedule* policy is that only the sub-task currently

executed by a mobile node must be aborted upon disconnection. All previously completed installments are not wasted since their results have been successfully returned to the scheduler in whole. Following the same notation, once a disconnection event has occurred, RT^i is calculated as follows:

$$RT^i_{Inst} = t^i_C + t^i_A + RT^{i'}_{ResInst} \quad , i \in \{0,1,\ldots\ldots\ldots\ldots,N-1\}$$

This policy, intends to alleviate the problem of resource wasting in the case of *Abort-Reschedule* by further fragmenting the task load. Even though the fragmentation and reassembly overhead of the scheduler increases, this policy results in a more efficient utilization of mobile resources. Upon disconnection, there is no need for the re-assignment of the completed part of the task to a new MN. Instead, only the remaining installments are submitted resulting in a decreased RT,, resource waste is inevitable in this policy, since a disconnection event may interrupt the completion of an installment.

However, the waste is limited to the size of the installments. Furthermore, it must be noted that this policy is better suited for mobile devices since it does not require large amounts of storage and memory for the computation of entire subtasks. It is also considered suitable for providing the MNs with the *flexibility* to specify the amount of resources they offer, expressed in number of installments and/or installment size (f). Moreover, *installments* can be used to implicitly inform the scheduler about the networking conditions and the processing capabilities of a MN. This can be considered as a form of quick (not waiting for the complete task) implicit feedback which can guide the scheduler to important decisions regarding scheduling and load distribution. In this case, the estimation can be used to decide whether to abort the installment or not. Finally, the proposed policy presents the advantage of quickly providing *partial results* to the scheduler. Since the results are usable only if they refer to the entirety of the input workload, it is not necessary to wait for the completion of the whole task as in the case of the *Abort-Reschedule* policy. Instead, the results from the completed installments may be returned to the Mobile Grid user. The aforementioned features of this policy are considered as especially useful in application scenarios where quick, and possibly partial, results are desirable/acceptable such as SETI like [13] scientific search applications, and in environments where the mobile devices are particularly resource constrained e.g. cellular networks.

C. Groups

In this policy tasks are replicated so that a certain task is assigned to more than one of the MNs residing in the same WLAN. This policy results in the formation of distinct *groups* of MNs processing the same sub-tasks. The reasoning behind this approach is that not all MNs present the exact same networking behavior at the same time and therefore, if the same task is submitted to multiple MNs it is highly probable that one of them will eventually return the results earlier than the others. Obviously, this policy is especially suitable for applications in which fault tolerance and reliability are on the main focus e.g. [9]. We denote R as the *replication degree* with R ε {1, ..,N}, i.e. the number of replicas produced for each task. If R = 1 we obviously have no replication. Then we have:

$$RT^i_{Groups} = \max_j RT^i_j, i\varepsilon\{0,1,\ldots\ldots\ldots\ldots..,\frac{N}{R}\} \text{ , } j\{1,\ldots,R\}$$

The *Groups* policy unavoidably results in the waste of resources. If a certain MN returns the results of a sub-task earlier than the MNs which have received the same sub-task, then the resources of the remainder of the MNs are wasted. Apart from the apparent waste of resources, excessive task replication has another important side-effect. A job of a certain workload is split to each participating MN as follows:

$$t^i_{total} = \frac{T_{TOTAL}}{\frac{N}{R}}$$

Load Balancing Strategy:

Intra-Cluster load balancing: In this first level, depending on its current workload (estimated from workloads of its worker nodes), each cluster manager decides whether to start or not a load balancing operation. If a *cluster manager* decides to start a load balancing operation, then it tries, in priority, to load balance its workload among its worker nodes. Hence, we can proceed *C* local load balancing in parallel, where *C* is the number of clusters. Load of an agent executing on machine is defined as the sum of its computational load and communication load

$U_i = H_i + G_i$ Where: H_i – Communication Load, G_i – Computational Load The load L_k of machine m_k is defined as the sum of all its local agents load. More specifically

$L_k = \sum (w_i + u_i)$ Where: w_i–Communication Load , u_i – Computational Load

Goal of a load balancing algorithm is to minimize the variance of the load among all the machines in the cluster, this will turn minimize the average response time of serving users queries.

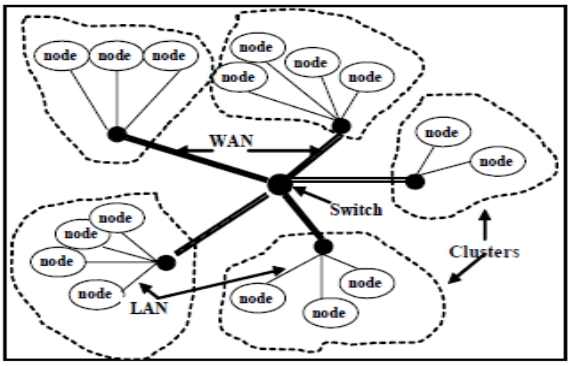


Figure 2: Example of a Grid topology

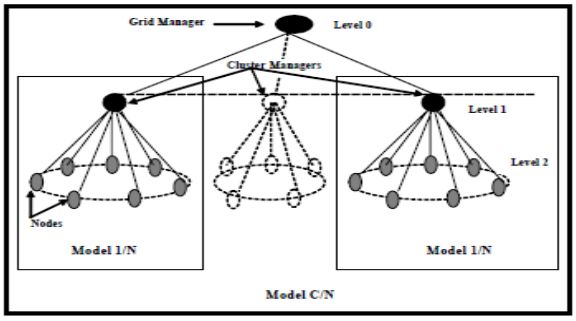


Figure 3: Tree-based representation of a Grid

6.Intra cluster load balancing algorithm

This algorithm is considered as the kernel of our load balancing strategy. The neighborhoods load balancing used by our strategy makes us think that the imbalance situations can be resolved within a cluster. It is triggered when any cluster manager finds that there is a load imbalance between the nodes which are under its control. To do this, the cluster manager receives periodically workload information from each worker node. On the basis of these information's and the estimated balance threshold ε, it analyzes the current workload of the cluster. According to the result of this analysis, it decides whether to start a local balancing in the case of imbalance state, or eventually just to inform its Grid manager about its current workload. At this level, communication costs are not taken into account in the task transfer since the worker nodes of the same cluster are interconnected by a WLAN network, of which communication cost is constant.

Step 1: Workload Estimation

1.For Every element E_i of G and according to its specific period **Do** Sends its workload LOD_i to its group manager

Endfor

2.Upon receiving all elements workloads and according to its period the group manager performs:

a- Computes speed SPD_G and capacity SAT_G of G

b- Evaluates current load LOD_G and processing time TEX_G of G

c- Computes the standard deviation σ_G over processing times

d- Sends workload information of G to its associated manager: in case where G is a cluster.

Step 2: Decision Making

3. Balance criteria

a. Cluster**: If** ($\sigma_G \leq \varepsilon$ **) Then** Cluster is balanced; Return **EndIf**

b. Grid**: If (**# (overloaded clusters)) ≤ given threshold **Then** Grid is in balance state; Return **EndIf**

4. *Saturation criteria*

If $\left(\frac{LOD_G}{SAT_G} > \delta\right)$**Then** Group G is saturated; Return **EndIf**

5. *Partitioning group G into overloaded (GES), under-loaded (GER) and balanced (GEN)*

GES←Φ; GER← Φ; GEN ← Φ

For Every element E_i of G **Do**

If (E_i is saturated) **Then** GES GES U E_i /* Saturated Overloaded */ **Else**

Switch

- $TEX_i > TEX_G + \sigma_G$: GES GES U E_i /* Source */

- $TEX_i < TEX_G - \sigma_G$: GER GER U E_i /* Receiver */

- $TEX_G - \sigma_G \leq TEX_i \leq TEX_G + \sigma_G$: GEN ← GEN U E_i/*Balanced*/

Step 3: Tasks Transfer

Test on Supply and Demand

$Supply = \sum_{E_r \, \varepsilon \, GER} \frac{LOD_G}{SPD_G} * \frac{SPD_r}{SPD_G} - LOD_r$

$Demand = \sum_{E_s \in GES} LOD_s - \frac{SPD_s}{SPD_G}\frac{LOD_G}{SPD_G}$

If$\left(\frac{Supply}{Demand} \geq \rho\right)$ **Then** local load balancing Fail; Return **EndIf**

7. Perform intra-group task transferring:

If (G = Cluster) **then** Perform Heuristic1 **else EndIf**

Heuristic 1: Intra-Cluster tasks transfer

a- Sort GES by descending order of their elements processing times.

b- Sort GER by ascending order of their elements processing times.

c- While (GES ≠ Φ .**AND.** GER ≠ Φ) **Do For** i = 1 **To** # (GER) **Do**

(i) Sort tasks of first node belonging to GES by selection criterion,

(ii) Transfer the higher priority task from first source node of GES to ith receiver node of GER

(iii) Update the current workloads of receiver and source nodes,

(iv) Update sets GES, GER and GEN,

(v) If (GES = Φ **OR** GER =Φ) **then** Return **Endif,**

(vi) Sort GES by descending order of their processing times.

Endfor

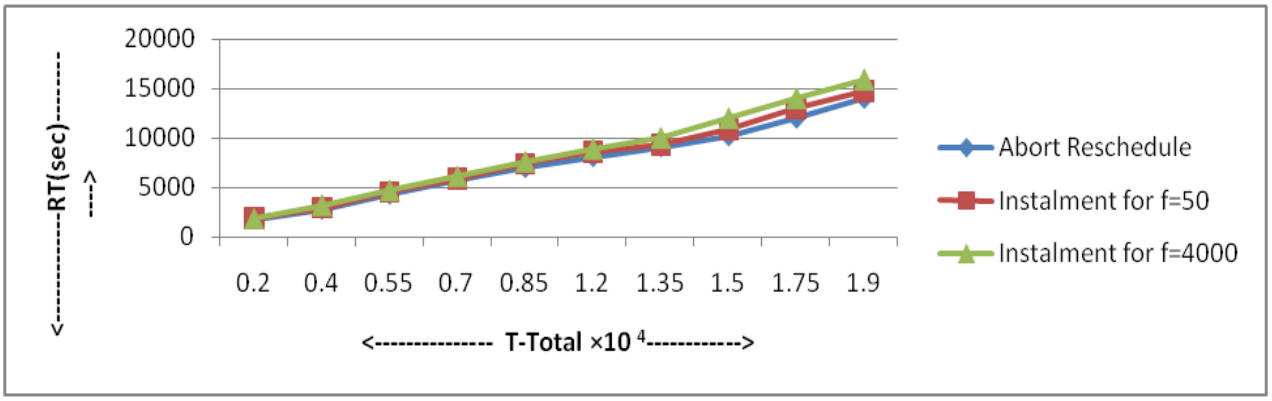
7. Experimental Study of Scheduling and Load Balancing techniques

It shows the performance of the proposed *installments* policy both it terms of RT and RW. We have examined a wide range of values for the size of the installments (f), measured in seconds. These values varied from a small portion to the entirety of the workload in order, first, to reveal the effects of the installment size on the resulting RT and RW, In both cases, the superiority of the *installments* policy is evident. As the size of the installments increases, the performance of the *installments* policy approaches that of the *Abort-Reschedule* policy .We expect that the superiority of the *installments* policy will be more evident in environments with more intermittent connectivity. Similar results were derived for the achieved Speedup. In all, for small installment sizes, any disconnection results in the abortion of a small portion of the overall subtask. Therefore, a small part of the offered resources is wasted and a small part of the workload has to be processed again. As the size of the installments increases so does the size of the aborted fragments of the workload. Hence, the increased RW and the extended re-processing leading to a higher RT.

In the first set of experimentations, we have focused on the response time, the waiting time and the processing time, according to various numbers of tasks and clusters. We have considered different numbers of clusters and we suppose that each cluster contains 50 worker nodes. For every node, we generate a random speed varying between 10 and 30 computing units per time unit. The number of instructions per task has varied between 300 and 1500 computing units. Our strategy has allowed to reduce in a very clear way the mean response time of the tasks. We obtain a gain in 100% of cases, varying between 3.09% and 24.44%. In more than 60% of cases, this gain is greater than 11%. The lower gains have been obtained when the number of clusters was fixed at 32 on the one hand and when the number of tasks was 10000 and 20000 on the other hand. We can justify this by the instability of the Grid state (either overloaded or idle).

Best improvements were obtained when the Grid were in a stable state: (for Clusters {8, 16} and for Tasks {14000, 16000}.In some infrequent cases, we have noted that the variation of the gain changes abruptly. We believe that this situation comes from the fact that the number of tasks and/or the number of clusters varies suddenly and generates instability in the Grid.

Fig .a Response Time (RT)



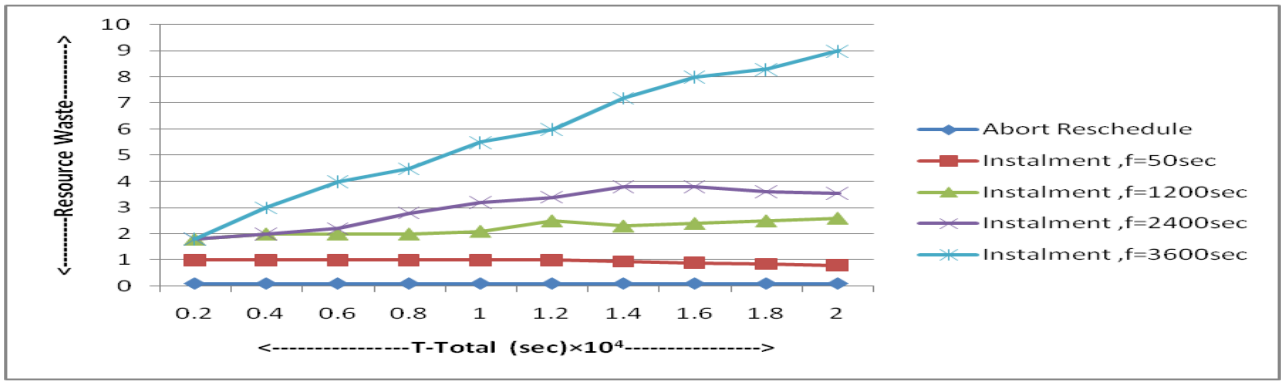


Fig b Resource Waste (RW)

Fig 4.Performance of installments policy

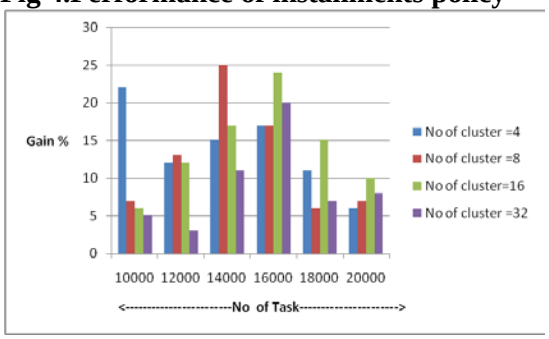


Fig 5: Gain according to various numbers of clusters by varying the number of tasks

8. Conclusion

Scalability is the precondition of algorithm to avoid poor allocation decisions. To assess stability we can measure hit-ratio, the ratio of remote execution requests concluded successfully. Another measure of stability is percentage of remote execution in the system. Activities related to remote execution should be bounded and restricted to a small proportion of the activity in the system. In both cases, the superiority of the *installments* policy is evident. As the size of the installments increases, the performance of the *installments* policy approaches that of the *Abort-Reschedule* policy. We expect that the superiority of the *installments* policy. In all, for small installment sizes, any disconnection results in the abortion of a small portion of the overall subtask. Therefore, a small part of the offered resources is wasted and a small part of the workload has to be processed again. As the size of the installments increases so does the size of the aborted fragments of the workload. Hence, the increased RW and the extended re-processing leading to a higher RT.

References

[1] K.Q. Yan1, S.C. Wang1 "The Anatomy Study of Load Balancing in Distributed System", proceeding of the seventh international conference on parallel and distributed computing, Application and Technologies (PDCAT'06)

[2] Reinhard Riedl and Lutz Richter "Classification of Load Distribution Algorithms ", CH-8057,1066-6192/96 $5.00 0 1996 IEEE Proceedings of PDP '96

[3] Ka-Po Chow and Yu-Kwong Kwok "On Load Balancing for Distributed Multiagent Computing", ieee transactions on parallel and distributed systems, vol. 13, no. 8, august 2002.

[4] G. Banavar, J. Beck, E. Gluzberg, J. Munson, J. Sussman, and D. Zukowski, "Challenges: an application model for pervasive computing," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, New York, NY, USA, 2000, pp. 266–274, ACM Press

[5] A. Litke, D. Skoutas, and T. Varvarigou, "Mobile grid computing: Changes and challenges of resource management in a mobile grid environment," in *Proceedings of Practical Aspects of Knowledge Management(PAKM 2004)*, 2005.

[6] T. Phan, L. Huang, and C. Dulan, "Challenge:: integrating mobile wireless devices into the computational grid," in *MobiCom '02: Proceedings of the 8th annual international conference on Mobile computing and networking*, New York, NY, USA, 2002, pp. 271–278, ACM Press.

[7] S. Kurkovsky and Bhagyavati, "Wireless grid enables ubiquitous computing.," in *ISCA PDCS*, 2003, pp. 399–404.

[8] K. Katsaros and G.C. Polyzos, "Optimizing operation of a hierarchical campus-wide mobile grid for intermittent wireless connectivity," in *Proceedings of the 15th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*, Princeton, NY, USA, 2007.

[9] K. Katsaros and G.C. Polyzos, "Optimizing operation of a hierarchical campus-wide mobile grid," in *Proceedings of the 18th IEEE Personal Indoor and Mobile Radio Communications conference (PIMRC)*, Athens, Greece, 2007.

[10] V. Bharadwaj, T.G. Robertazzi, and D. Ghose, *Scheduling Divisible Loads in Parallel and Distributed Systems*, IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.

[11] K. Katsaros, I. Niarhos, and V. Vassalos, "DAIMON: Data Integration for a Mobile Network," in *MobiDE '05: Proceedings of the 4th ACM international workshop on Data engineering for wireless and mobile access*, New York, NY, USA, 2005, pp. 57–64, ACM.

[12] D. Kotz, T. Henderson, and I. Abyzov, "CRAWDAD trace set dartmouth/campus/movement,"*http://crawdad.cs.dartmouth.edu/dartmouth/campus/movement, Mar*.2005.

[13] "Seti@home homepage," http://setiathome.ssl.berkeley.edu.

[14] A. Kamerman and G. Aben, "Throughput performance of wireless LANs operating at 2.4 and 5 GHz," in *Personal, Indoor and Mobile Radio Communications, 2000. PIMRC 2000.*, 2000, pp. 190–195.

[15] A. Litke, D. Skoutas, K. Tserpes, and T. Varvarigou, "Efficient task replication and management for adaptive fault tolerance in mobile grid environments," *Future Generation Computer Systems*, vol. 23, no. 2, pp. 163–178, 2007.

[16] Sang-Min Park, Young-Bae Ko, and Jai-Hoon Kim, "Disconnected operation service in mobile grid computing.," in *ICSOC*, 2003, pp. 499–513.

[17] S. Kurkovsky and Bhagyavati, "Wireless grid enables ubiquitous computing.," in *ISCA PDCS*, Seong-Moo Yoo and Hee Yong Youn, Eds. 2003, pp. 399–404, ISCA.

[18] S. Kurkovsky, Bhagyavati, and A. Ray, "A collaborative problem-solving framework for mobile devices," in *ACM-SE 42: Proceedings of the 42nd annual Southeast regional conference*, New York, NY, USA, 2004, pp. 5–10, ACM Press.

Author Information

Debabrata Singh is an Assistant Professor in the Department of Information Technology, holds M.Tech in Computer Science & Engg. (BPUT,BBSR) He has nearly five years experience in teaching, software development and research. Presently, he is working as Assistant professor in ITER,SOA University, Bhubaneswar, Orissa He has published 17 papers on Multi agent technologies, Sensor Network, & Grid Computing environment in national & international journals and conferences.

Sandeep Nanda is a research scholar in the Department of Computer Applications ,holds M. Tech in CSDP branch (ITER, Bhubaneswar) He has nearly 2 years experience in teaching, and research. Presently, he is working as a lecturer in Jawahar Navodaya Vidyalaya, Sora, Odisha ,India, under central government of India. He has published 5 papers on Wireless Mesh Networks, Grid Computing environment in national & international journals and conferences.

Sarbeswara Hota is an Assistant Professor in the Department of Computer Applications, holds M. Tech in Computer Science & Engg. (ITER, Bhubaneswar) He has nearly 8 years experience in teaching, and research. Presently, he is working as Assistant professor in ITER,SOA University, Bhubaneswar ,Orissa His area of interest are Wireless Mesh Networks, MANETs, multi agent technologies & Grid Computing environment.

Manas Kumar Nanda is an Assistant Professor in the Department of Computer Applications, holds M.Tech in Computer Science & Engg. (Berhampur University, Berhampur) He has nearly 7 years experience in teaching, and research. Presently, he is working as Assistant professor in ITER,SOA University, Bhubaneswar, Orissa .He has published 5 papers on Wireless Mesh Networks, sensor network, multi agent technologies & data mining in national & international journals and conferences.