Optimization of Load Balancing Algorithm Using Workload Simulator For a HPC Environment

AK Aggarwal[a], Haresh S. Bhatt[b], BK Singh[c]

[a] University of Windsor, Windsor, On., N9B 3P4, CANADA
akshaia@uwindsor.ca

[b] Space Applications Centre, Indian Space Research Organization, Ahmeadbad – 380015, INDIA
haresh@ipdpg.gov.in

[c] Space Applications Centre, Indian Space Research Organization, Ahmeadbad – 380015, INDIA
brajendra_kumar@indiatimes.com

It is difficult to devise a uniform load balancing strategy for heterogeneous distributed computing environment, designed to cater to a number of users and a variety of applications, with unpredictable arrival patterns of jobs. This paper investigates the effect of the type of workload on the performance of a load balancing algorithm by using a Workload Simulator. The Simulator helps identify various parameters for improving the performance of the Load Balancing Algorithm for different workload and resource availability conditions. The algorithm and the workload simulator have been applied along with WebDedip, a distributed computing environment.

Il est difficile de mettre au point une stratégie de répartition des ressources dans un environnement hétérogène de systèmes distribués, pour un certain nombre d'utilisateurs, une variété d'applications, et des enchaînements de travaux imprévisibles. Cet article étudie l'effet du type de charge (Workload) sur la performance d'un algorithme de répartion des ressources en utilisant un simulateur de charge. Le simulateur aide à identifier différents paramètres améliorant la performance de l'algorithme pour différentes charges et différentes disponibilités de ressource. L'algorithme et le simulateur ont été appliqués en utilisant l'environnement de calcul distribué WebDedip.

1. Introduction

A Load Balancing algorithm for a heterogeneous distributed computing environment attempts to improve the response time of a parallel or distributed application by ensuring optimum utilization of available resources. It should respond quickly to the changing workload and environmental conditions without causing much overhead. Hence it should customize its strategy as per the prevailing conditions. One method to do this is by having an efficient learning system [15,16,17,18]. Such a system requires a long history of run of load balancer and distributed computing environment involving a large number of combinations of workload and environmental conditions. Even if we have a long history of run, it may not include all the conditions, particularly those which are encountered rarely. Moreover to obtain the data for a large variety of load conditions may prove to be a costly process in terms of the systems resources. Hence we have devised a faster means of testing a load balancing algorithm. We have made a rule based fuzzy simulator [13] which generates a session with low, medium, high and burst workload conditions. These simulated sessions are given to the load balancer, which along with WebDedip [12], runs the session for 2 to N number of nodes. Each run is analyzed and corresponding changes are suggested for the load balancer to optimize its performance. These studies have identified a method of detecting important parameters for variable load conditions. We have used the method to work out the important parameters for two types of load conditions.

Various scheduling optimization algorithms [1,2,3,4] are discussed in the literature. Tang *et al.*[1] have given an optimization scheme for static job scheduling. They use a centralized load balancing policy like us, but allow round robin job dispatching and multiple processes simultaneously while we run one process at a time. The weighted workload allocation scheme introduces the reallocation strategy to get the performance benefit in terms of session duration and to accommodate high priority applications. We are able to avoid the need for separate weighted workload allocation scheme and job dispatching scheme by using a waiting queue and a prediction mechanism. The Simulator generates a real world scenario and obtains a workable load balance policy, rather than a set of fixed mathematical optimizations, consistent with the complexity and dynamic nature of the heterogeneous environment.

Various workload simulators [5,6,8,9,10] for testing are discussed in the literature. They are used for application and environment specific simulation. Incase of the generic distributed processing environment like WebDedip, it is

supposed to handle, nearly simultaneously, a large variety of applications. We have developed a simulator [13] for testing the WebDedip Load Balancer under controlled and reproducible conditions. It uses rule based fuzzy probabilistic randomization for simulating distributed processing applications.

2. Backround

Several distributed computing environments, providing different types of tools to the developer, have been developed. Many scientists have worked on improving the efficiencies of such environments. Ease of use, is the parameter which is normally untouched by most of the scientists. WebDedip [12], developed under a research project at Gujarat University, Ahmedabad, India, is the environment which was developed with ease of use as the core theme to support HPC users like physicists, mathematicians, civil engineers, etc. Initially, the performance issues were not addressed. Later, a hybrid application centric load balancer [11], consisting of 6 algorithms, a heuristic and a normalization process, was developed for optimizing the performance.

Testing and tuning of such a load balancer is always a complex and challenging task. It is very difficult to get the required number of nodes as well as different applications for testing. Hence a rule based fuzzy simulator [13] was developed to generate a variety of sessions with different applications. The simulator was used extensively to test the load balancer. It was observed during testing that the performance of the load balancer can be improved a great deal by choosing the parameters appropriately.

3. Goal and Issues

WebDedip load balancer uses a variety of algorithms and rules. Furthermore, it supports the multi-user environment. This is achieved at the cost of overheads, while handling sessions having a number of distributed applications, with different characteristics. The basic aim of the WebDedip load balancer is to improve the session throughput and to manage requests with different priority levels. Furthermore, WebDedip does not postpone an application beyond a certain limit to improve throughput. We aimed at optimizing such a complex algorithm to improve the performance and efficiency.

4. Load Balancer Complexity

We need to understand the WebDedip load balancer before proceeding further.

WebDedip uses a deterministic approach for normalizing the heterogeneous nodes into few classes to reduce the node (class) assignment complexity for an application. Each application has different processes connected to each other through Task Flow Graph (TFG). The critical branch of TFG is taken as the baseline to assign the class to each process. Other branches can be delayed to optimize the resource allocation until it doesn't cross the baseline. It has process completion and application completion predictors that use different permutations of available resources versus requirements. The load balancer uses dynamic sliding window technique, while assigning a class to reduce the complexity of permutations. Applications outside the sliding window, are placed in different FIFO queues based on priority. However, queues are reshuffled to optimize the session throughput. Simultaneously it increases the priority of the applications, that are delayed, to restrict the delay to predefined limits.

WebDedip load balancer is a pseudo dynamic design that applies a set of rules and algorithms on predefined events.

5. Load Balancer Improvement Model

We developed a simulator [13] to simulate a variety of sessions with different characteristics to test the efficiency of the load balancer. The session may be in low, medium, high or burst mode of loading. Each session was repeatedly submitted to the load balancer, at a time, varying 2 to 35 nodes of the WebDedip environment. Application of each case of each session was submitted to the load balancer dynamically (when it was expected to arrive) using threads and sockets.

Load balancer analyses all applications of a session for a case (say 5 nodes) and dynamically assigns the resources to all the processes of all applications. It generates the node allocation graphs and expected session completion time. A throughput graph for each session is generated based on the results available for 2 to 35 nodes.

All sessions were submitted one after another in the same manner. Three critical parameters affecting the performance are identified on the basis of node allocation graphs and session throughput graphs

6. Defining Performance Parameters

The First parameter for load balancer is 'Requirement Factor'. It is defined as the time required by any application on a new node in terms of fraction of session duration. If its value is kept high, then it may happen that in spite of having resources, we may not be using them. If its value is kept low, then overhead associated with inducting new resources may exceed the advantage of occupying resources for a small period of time.

The second parameter is ‘Level of Branch delay’. It is defined in terms of number of other nodes whose allocation needs to be altered by dependent as well as subsequent processes (may be of other applications also) of a process to accommodate another process ahead of it on a given node. If its value is higher, then load balancer overhead may increase more than the advantage gained by accommodating another process ahead. This is because the Branch Delay algorithm is implemented by recursive procedures. A higher value of 'Level of Branch delay' means less possibility of Branch delay. This would mean a number of small time intervals, during which the node remains unutilized, may be there after load balancing.

The third parameter is ‘Depth of Reallocation’. It is defined in terms of time duration up to the end of allocation window [11], within which all the processes of all the nodes need to be reallocated to accommodate a new application. Optimization of this parameter is required to minimize session completion time. If its value is higher, then load balancer overhead will increase for reallocating nodes and other resources. Moreover some application may get delayed more than their allowable duration. If it is less, then wastage of resources occurs.

7. Results and Discussion

We have taken only two types of sessions i.e. Medium load session and High load session to test and optimize our load balancer. We have not simulated the Light load session since it does not benefit much through load balancing. These test sessions are generated by a Fuzzy simulator. We have generated more than 50 sessions of each type for our exhaustive testing. A typical simulated Medium load session and High load session are shown in Figure 2 and Figure 3 respectively.

These sessions are given to the load balancer which gives a set of load balanced sessions for 2 to 35 number of nodes. The whole philosophy is shown in Figure 1. Figure 4 and Figure 5 show the Percentage CPU utilization and normalized session completion time for a typical set of values of optimization parameter i.e. 'requirement factor' increased by 5 percent,’ Level of Branch delay' is increased to 3 nodes and 'Depth of Reallocation' is increased by 20 percent.

As shown in Figure 5, Percentage CPU utilization is, in general, quite less for a Medium load session as well as for a High loaded session even for two nodes. The reason for this behavior is the arrival pattern of the applications. During some time interval within a session, there may be very less load or no load at all, even for High loaded sessions.

For less number of nodes, Percentage CPU utilization is less in case of Medium load session as the session may have less load intermittently or no load at all. In general, a High load session will have sufficient load all the time. Hence for this case percentage utilization will be more for less number of nodes.

Gradient of Percentage CPU utilization curve is more in case of High load session, as load balancer will assign the load optimally for less number of nodes. But as the resources increase, it will try to occupy new resources even for a very short duration. Induction of a new system for very short duration will decrease Percentage CPU utilization sharply, as most of the capacity of newly inducted node remains unutilized. In general, when the level of requirements is low, no new systems should be inducted into the system, as it increases network data transfer overhead, the overheads of managing resources of newly inducted system, polling overhead etc.

In Figure 4, Normalized session completion curve saturates earlier for Medium load session because a Medium load session requires less resources. Average node requirement for a given type of session is expressed in terms of the minimum number of nodes required to complete the session for a given session duration. It can be obtained from Normalized session completion curve. It is equal to the number of nodes at the starting point of saturation of the Session completion time curve. The system shows remarkable improvement when the number of available nodes is closer to the average node requirement. Improvement is less in case of scarcity as well as abundance of resources.

As shown in Figure 5, an increase in 'requirement factor' will not only increase CPU utilization for a given number of nodes but it also increases session completion time. Its effect is more pronounced on a High load session for the reason explained earlier in conjunction with Gradient of Percentage CPU utilization curve. If the 'Level of Branch delay’ is increased, the CPU utilization increases and Session completion time decreases. Similarly when 'Depth of Reallocation' increases, the session completion time decreases and CPU utilization increases. The effectiveness of parameters 'Level of Branch delay' and 'Depth of Reallocation' is high for Medium load session as a High load session requires extensive recursive calls or reallocation causing excess overhead to load balancer.

Interdependency between optimization parameters is also observed. For example, reallocation becomes quite frequent when optimization parameter 'Depth of reallocation' is more than 40 percent of the allocation window length in the reallocation algorithm. First this parameter was decided in terms of remaining session duration. After experimentation we made it dependent on allocation window length as well. This works fine for both

the sessions. Similarly branch delay algorithm responds slowly when the value of optimization parameter 'Level of branch delay' is raised by more than four levels. Hence the load balancer takes unusually large time for such cases. This becomes comparable to the average process duration for a given session. Hence we decided to restrict the value of this parameter to four.

8. Conclusion and Future Scope

We have carried out exhaustive testing using more than 100 sessions for our optimization. The simulated environment has enabled such a testing in a short duration. Otherwise such an exercise would require many days of work on about 35 nodes for each session in actual production environment.

During this exhaustive and rugged exercise, we could fix more than 85 programming bugs to improve the load balancer accuracy. We had to carry out eight major modifications in the algorithm to improve the efficiency and performance. We have improved the performance of the load balancer by 17%.

We have identified three parameters for improving the session through put. These should be usable for other similar algorithms also. It is only by using optimized values of these parameters, that a load balancer may be able to achieve an optimum throughput. We are also working to address the optimization of burst mode and priority management cases.

We are in the process of interfacing our optimized load balancer with WebDedip in production environment. We wish to conduct sample testing for verifying the optimization effectiveness in production environment.

References

1. Optimizing static job scheduling in a network of heterogeneous computers *Xueyan Tang; Chanson, S.T.* Parallel Processing Proceedings. 2000 International Conference on, 2000, Page(s): 373 –382
2. *Haddad, E.,* Real-time optimization of distributed load balancing, Parallel and Distributed Real-Time Systems, 1994. Proceedings of the Second Workshop on , 1994 Page(s): 52 –57
3. Cherkasova L., Ponnekanti S.R., Optimizing a "content-aware" load balancing strategy for shared Web hosting service, Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on , 2000 Page(s): 492 –499
4. Hamidzadeh B., Lau Ying Kit, Lilja D.J., Dynamic task scheduling using online optimization, Parallel and Distributed Systems, IEEE Transactions on , Volume: 11 Issue: 11 , Nov. 2000 Page(s): 1151 –1163
5. Kao W.I., Iyer R.K., A user-oriented synthetic workload generator, Distributed Computing Systems, 1992., Proceedings of the 12th International Conference on, 1992, Page(s): 270 –277
6. Gomez M.E., Santonja V., A new approach in the modeling and generation of synthetic disk workload, Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on, 2000, Page(s): 199 –206
7. Wen-Sheng Wang, Dujmovic, J.J., Mathews W., A tool for performance measurement of NT networks, Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on, 2000 Page(s): 281 –288
8. Jean-Marc Saglio and José Moreira, Oporto: A Realistic Scenario Generator for Moving Objects, Found in: 10th International Workshop on Database & Expert Systems Applications pp. 426, September 1999.
9. Darko Marinov, Davor Magdic, Aleksandar Melenkovic, Jelica Protic, Igor Tartalja, Veljko Milutinovic, Scowl: A Tool for Characterization of Parallel Workload and its Use on Splash-2 Application Suite, Found in: 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems pp. 207, August 2000
10. P. Mehra and B. Wah, Synthetic Workload Generation for Load Balancing Experiments, IEEE Parallel and Distributed Technology: Systems & Applications , Volume: 3 Issue: 3 1995, pp 4-19.
11. Bhatt H. S., Singh B.K., Aggarwal A.K., Application centric load balancing based on hybrid model, International conference for advance computing and communication ADCOM-2001, organised by IEEE and ACS India, Dec. 2001, Page(s): 231-238
12. Bhatt H.S., Aggarwal A.K., web enabled client-server model for development environment of distributed image processing, Proceedings of international conference on meta computing, GRID-2000, pp 135-145
13. Aggarwal A.K., Bhatt H. S., Singh B.K., Rule-based Fuzzy Simulator for testing Load Balancing Algorithms, transmitted for publication.
14. Gopal, S., Vajapeyam, U., Load balancing in a heterogeneous computing environment, System Sciences, 1998, Proceedings of the Thirty-First Hawaii International Conference on, Volume: 7, 1998, Page(s): 796 -804 vol.7
15. Mehra P., Wah B.W., Adaptive load-balancing strategies for distributed systems, Systems Integration, 1992. ICSI '92., Proceedings of the Second International Conference on, 1992, Page(s): 666 -675
16. Koch T., Rohde G., Kramer B., Adaptive load balancing in a Distributed Environments, 1994, Proceedings of 1st International Workshop on Services, 1994, pp 115 –121
17. Gil-Haeng Lee, Wang-Don Woo, Byeong-Nam Yoon, An adaptive load balancing algorithm using simple prediction mechanism, Database and Expert Systems Applications, 1998. Proceedings. Ninth International Workshop.1998 Page(s): 496 -501

18. Chin Lu, Sau-Ming Lau, An adaptive load balancing algorithm for heterogeneous distributed systems with multiple task classes, Distributed Computing Systems, 1996., Proceedings of the 16th International Conference on, 1996,Page(s): 629 –636
19. Joshi B.S., Hosseini S.H., Vairavan K., A methodology for evaluating load balancing algorithms, High Performance Distributed Computing, 1993., Proceedings the 2nd International Symposium on, 1993, Page(s): 216 –222
20. Sarje A.K., Sagar G., Heuristic model for task allocation in distributed computer systems, IEE proceedings-E, vol. 138, no. 5, SEPTEMBER 1991, Page(s): 313-318
21. Chulhye Park, Kuhl J.G., A fuzzy-based distributed load balancing algorithm for large distributed systems, Autonomous Decentralized Systems, 1995. Proceedings. ISADS 95, Second International Symposium 1995, Page(s): 266 –273

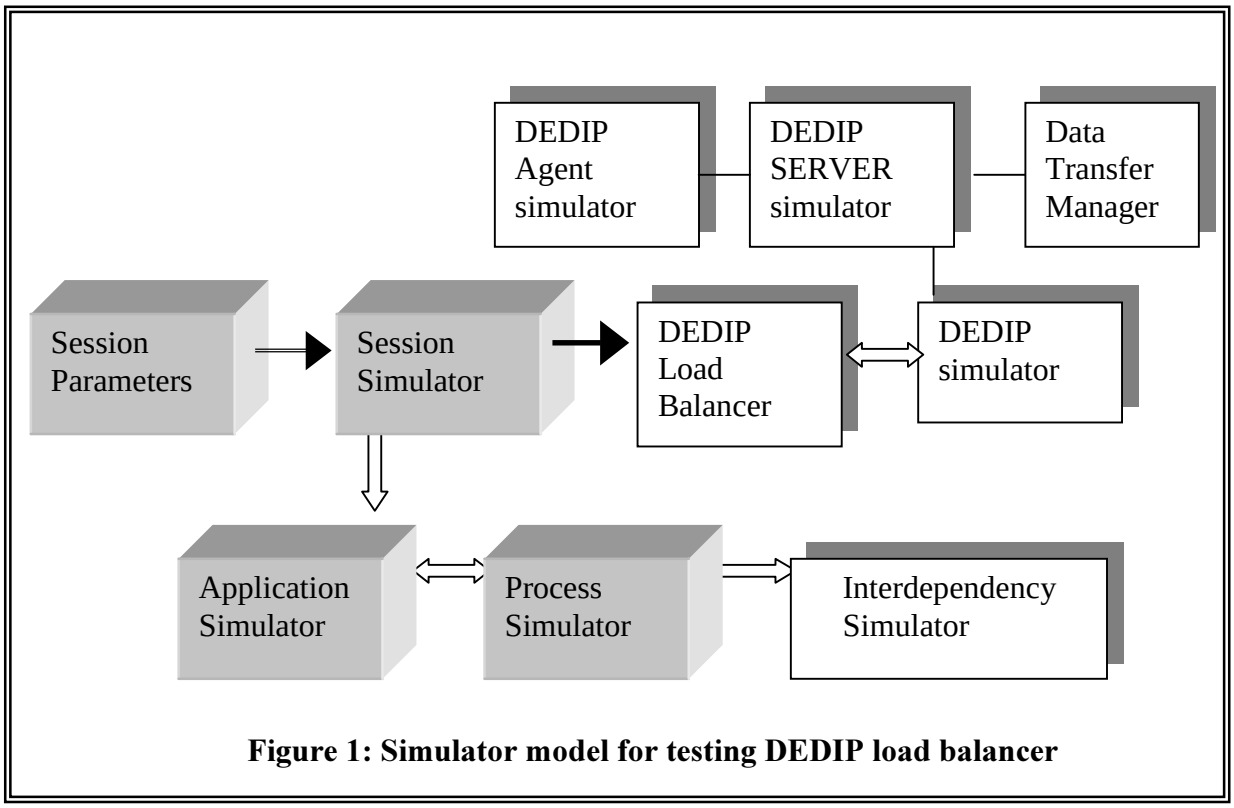

Figure 1: Simulator model for testing DEDIP load balancer

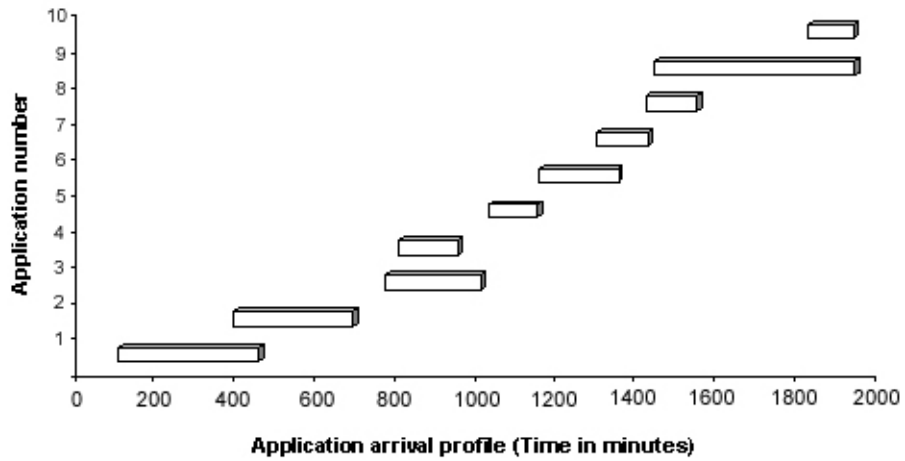

Figure 2: Simulated Application distribution for a medium loaded session

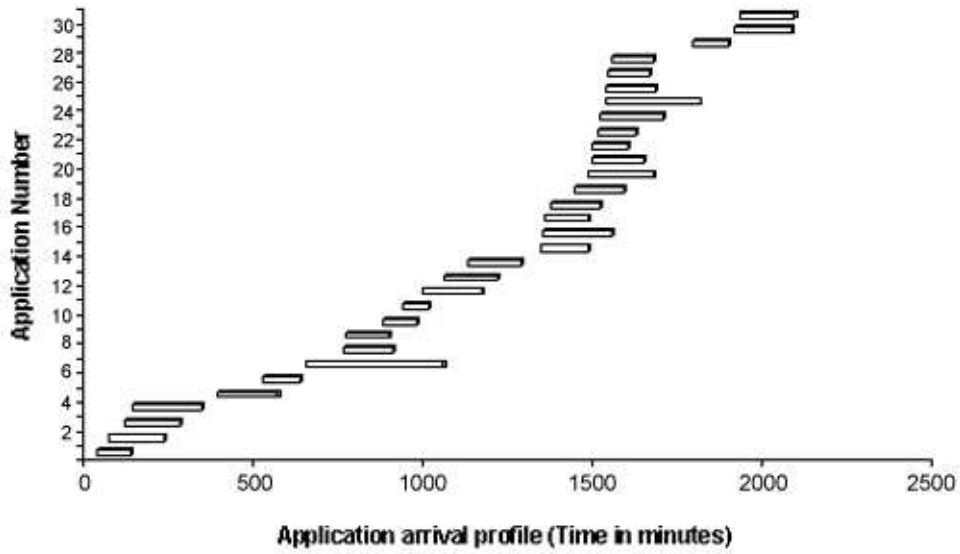

Figure 3: Simulated Application distribution for a heavily loaded session

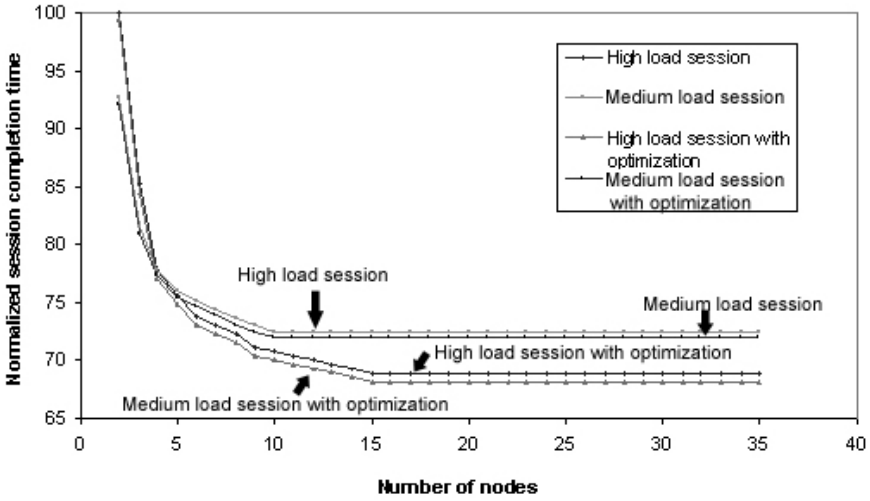

Figure 4: Normalized Session completion graph

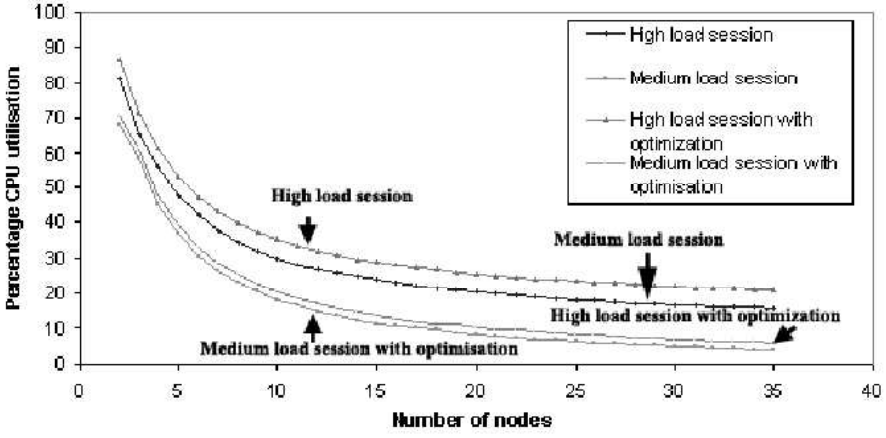

Figure 5: Percentage CPU utilization graph