International Journal of Computer Science and Informatics

Volume 1 | Issue 3 Article 10

January 2012

A Novel Approach for Load Balancing in Grid Computing

Swarna. M
Department of IT, MVGR College of Engineering, Vizianagaram, Andhra Pradesh, India., swarnamnkt@gmail.com

P.S.Sitharama Raju
Department of CSE, MVGR College of Engineering, Vizianagaram, India, vicky.poosapati@gmail.com

Nagesh V
Department of IT, MVGR College of Engineering, Vizianagaram, Andhra Pradesh, India., itsnageshv@gmail.com

Available online at www.interscience.in

A Novel Approach for Load Balancing in Grid Computing

Swarna.M, P.S. Sitharama Raju & Nagesh V
Department of IT, MVGR College of Engineering, Vizianagaram, Andhra Pradesh, India.
E-mail : swarnamnkt@gmail.com, vicky.poosapati@gmail.com, itsnageshv@gmail.com

Abstract - Computational grids are becoming increasingly vital in organizations with ever growing IT infrastructure and the need to increase the productivity of the computing infrastructure by ensuring optimal throughput for their computational jobs. Key to computational grids in the load balancer and job scheduler that is involved in decision making about when and which node to basically use to execute a job/task submitted to the grid. Most of the existing grids use a load function that evaluates the existing resources on the nodes, accesses the resource requirements of the task submitted and decide whether to withhold the job in the queue or schedule it on a node where the resources are available for the job. This decision making process becomes more challenging with jobs that are long duration, I/O intensive and resource requirements vary at different times during the task execution. Most current grid engines factor in the maximum requirement as stated at the time of job submission and are not good at analyzing the variation in resource requirements based on past history of the same job execution and use the information gathered in the decision making process. In this paper, we try to analyze how we can change the load balancing function to introduce more statistical analysis of history of past jobs in the scheduling decision process thereby ensuring we do not end up in trashing situations for I/O intensive jobs while at the same time utilize the available resources as efficiently as possible.

Keywords: *Scheduling policies, grid computing, memory utilization, Thrashing.*

I. INTRODUCTION

The term “grid Computing” refers to a distributed computing infrastructure for advanced Science & Technology [1]. Grid computing helps in the effective utilization of computing resources over the intranet/internet. The various existing grid software systems which are effective in job scheduling are Nimrod-G [2], Globus [3], Condor [4], Sun Grid Engine [5], Paryavekshanam [6], Vega PG[7] and many more. In the grid computing, the most essential aspect is Load balancing process which is handled by the Scheduler. The Scheduling System is responsible in selecting the best suitable machines in the grid for user jobs. When a job is submitted the scheduler schedules the job by considering static restrictions and dynamic parameters of jobs and machines. Depending on the availability of the resources, the scheduler maps the jobs to the selected resources. Thus, scheduling is the most important part for building an efficient grid infrastructure. But, the problems still persist with some applications viz.; increase in swapping and over utilization of few resources. Thus, an appropriate load balancing of tasks to the resources has large potentiality in enhancing the Grid’s functional capability.

The available literature on performance study of PBS[8], SGE and Condor has been reported in terms of system throughput, average turnaround time, average response time and resource utilization. All the Grid Engines use effective load balancing algorithms for the jobs entering into the queue by considering the job allocated for execution to be a new one. In this paper we have proposed our idea into SGE.

The rest of the paper is presented as, the section 2 explains about the SGE and section 3 discusses about the pitfalls in load schedulers. The section 4 explains the solution of the problem in load scheduling algorithms and Section 5 gives the results analysis. The section 6 concludes the paper.

II. ABOUT SGE

Though the SGE provides various features, the most promising issue is load balancing. The efficiency of any grid engine lies in its load balancing process. In the existing grid engine tools though the dynamism in load balancing is attained, the major issue is when a job which is already executed is once again made to execute, the existing SGE allocates the job based on existing scheduling policies(viz., Function policy, urgency based policy, override policy)without considering the past history of the job. Thus when submitting the jobs repeatedly which takes increased memory consumption from time to time, at a particular time number of swap-ins and swap-outs increases drastically. Finally this condition leads to a situation called **Thrashing.** To avoid this situation, we propose new approach.

Usually, the load schedulers are not aware of the required resources for a particular job prior to its execution. But the SGE6.1 provides a feature of advanced resource reservation for the jobs which will be submitted periodically. In this process, the administrator can configure a job to be executing periodically and when the job is about to submit, the master host reserves the required amount of resources in one of the execution host before the job is being submitted. Also an advanced feature introduced in SGE6.1 is that the administrator can reserve a specific host for execution of a job every time it is submitted. This process leads to preventing the periodically executing and prioritized jobs from waiting in the queue for availability of resources.

Though the scheduler verifies various parameters before allocating the job to the host, the parameters to be considered are limited to CPU availability, time to execute the job, etc. The scheduler daemon is not aware of how much memory is required by the job during the execution of the job. This may lead to serious problem because of the dynamism in memory utilization of jobs, i.e., the running jobs may not need the same amount of memory throughout its life cycle. This process is done by taking the snap shots of job status. Some Grid Engines maintain the information called as checkpoints. These checkpoints are used to re – schedule the task only when a job is terminated under abnormal conditions. In SGE also, the scheduler considers the system load of execution hosts and migrates the jobs when any one node is unavailable in middle of execution of jobs and re-schedules the job based on check points.

III. PIT-FALL:

But, the above discussed features are not sufficient for optimum scheduling of the jobs which are not periodical, but have a fair chance of being submitted repeatedly. When such jobs are submitted in batch and the scheduler daemon of master host is not aware of the required resources for the job, the current scheduling will submit the job to the execution host provided the necessary resources are available notwithstanding the fact that a combination of already executing jobs and the new job might need a lot more resident memory than available during the course of their execution resulting in T**hrashing.**

IV. EXPERIMENTAL RESULTS & ANALYSIS:

The experiments have been carried out on the nodes with a configuration of Intel Dual Core processor with 1GB RAM and having a storage capacity of 160 GB HDD. The Grid Engine System is of SGE6.1u6 installed under default configurations with 2 execution nodes namely exehost1.mvgr and exehost3.mvgr, where exehost1 is the Master Host and is also configured as submit host, administration host ,execution host as well. The exehost3 is configured just as submit and the execution host.

The experimentation has been carried out to show that various Grid engines are still having a serious problem of Thrashing. Thrashing is the situation which consumes more time in paging than execution. In this phenomenon, CPU is found to be in idle state for amount of time and the number of swap-ins and swap-outs also increases drastically. Thus, shortest jobs also consume more time to execute. To prove that there is a fair chance of causing a problem of Thrashing in Grid Engine System, we have submitted 3 I/O bound jobs where each job requires a memory of 500MB. The execution of all the 3 jobs exceeds the capacity of resident memory. The activity of above such submitted job is to open a file, read the contents of the file and then close the file. Hence, we can state that it is purely I/O bound jobs. The below figures clearly demonstrate that the Thrashing is caused for the jobs submitted.

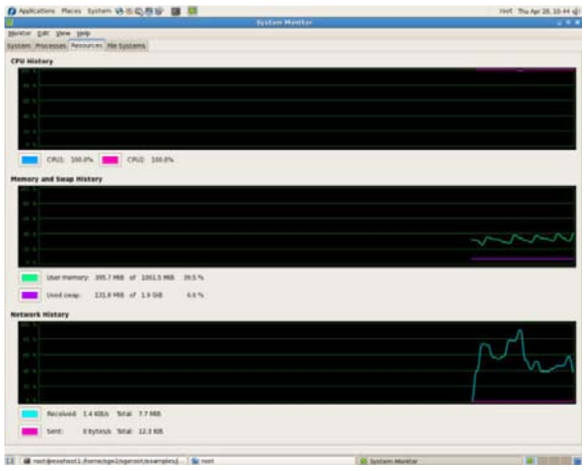

System Monitor
Monitor Edit View Help
System Processes Resources File Systems
CPU History
CPU1: 100.0% CPU2: 100.0%
Memory and Swap History
User memory: 395.7 MiB of 1001.5 MiB 39.5 %
Used swap: 131.8 MiB of 1.9 GiB 6.6 %
Network History
Received: 1.4 KiB/s Total: 7.7 MiB
Sent: 0 bytes/s Total: 12.3 KiB

Figure 1: immediately after submission of the jobs

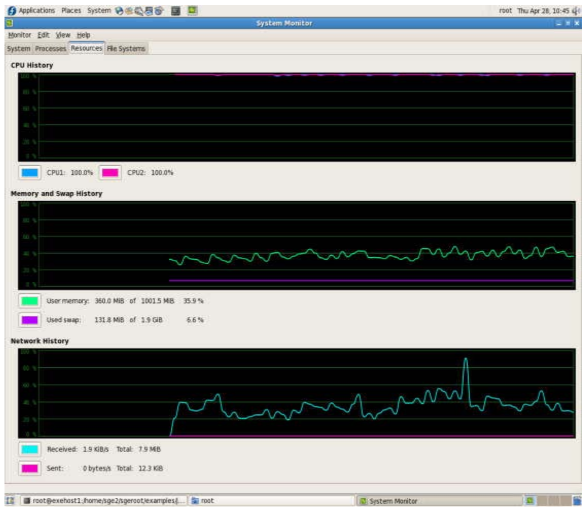

Applications Places System
root Thu Apr 28, 10:45
System Monitor
Monitor Edit View Help
System Processes Resources File Systems
CPU History
CPU1: 100.0% CPU2: 100.0%
Memory and Swap History
User memory: 360.0 MiB of 1001.5 MiB 35.9 %
Used swap: 131.8 MiB of 1.9 GiB 6.6 %
Network History
Received: 1.9 KiB/s Total: 7.9 MiB
Sent: 0 bytes/s Total: 12.3 KiB
System Monitor

Figure 2: after 1 minute

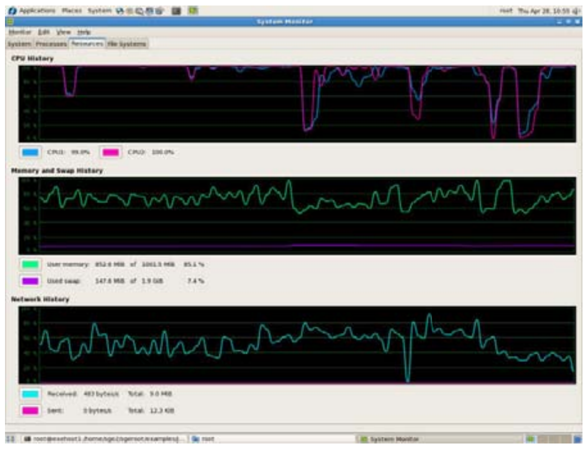


Figure 3: after 5 minutes (CPU is idle, increased swap-in, swap-out)

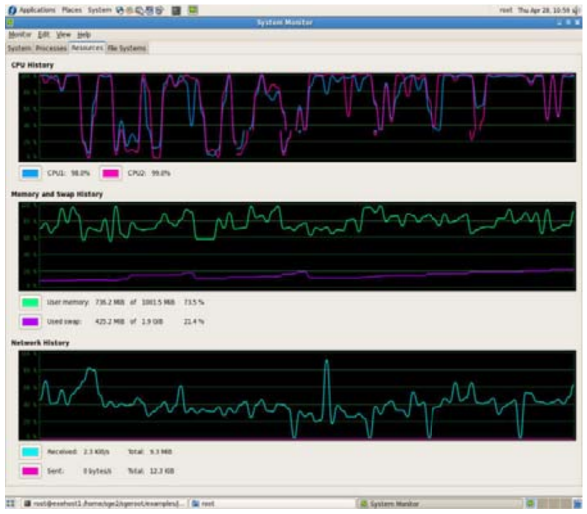


Figure 4: after 10 minutes

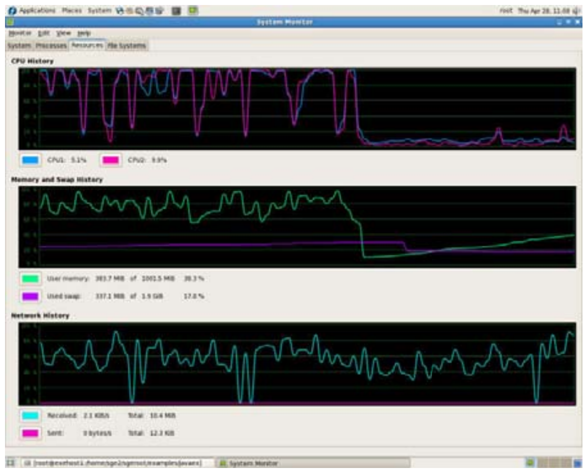
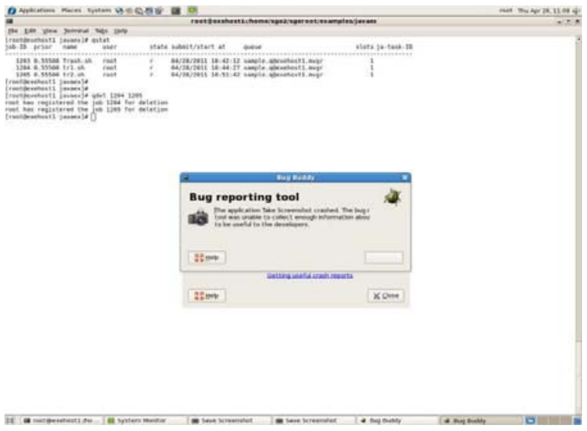


Figure 5 : after killing 2 jobs



```
[root@exehost1 javaex]# qstat
job-ID  prior   name       user         state submit/start at     queue                          slots ja-task-ID
-----------------------------------------------------------------------------------------------------------------
   1283 0.55500 Trash.sh   root         r     04/28/2011 10:42:12 sample.q@exehost1.mvgr             1
   1284 0.55500 tr1.sh     root         r     04/28/2011 10:44:27 sample.q@exehost1.mvgr             1
   1285 0.55500 tr2.sh     root         r     04/28/2011 10:51:42 sample.q@exehost1.mvgr             1
[root@exehost1 javaex]#
[root@exehost1 javaex]#
[root@exehost1 javaex]# qdel 1284 1285
root has registered the job 1284 for deletion
root has registered the job 1285 for deletion
[root@exehost1 javaex]# 
```

Bug reporting tool

The application Take Screenshot crashed. The bug reporting tool was unable to collect enough information about the crash to be useful to the developers.

Getting useful crash reports

Figure 6: System generated Bug tool (due to thrashing)

Table – 1: Statistical Analysis

Job status	**Paged-in**	**Paged-out**	**Swapped in**	**Swapped out**
Immediate submission	7602185	6091160	823199	1180388
After 2 minutes	9661133	8167407	1088774	1686586
After 5 minutes	9709741	8191239	1090833	1691936

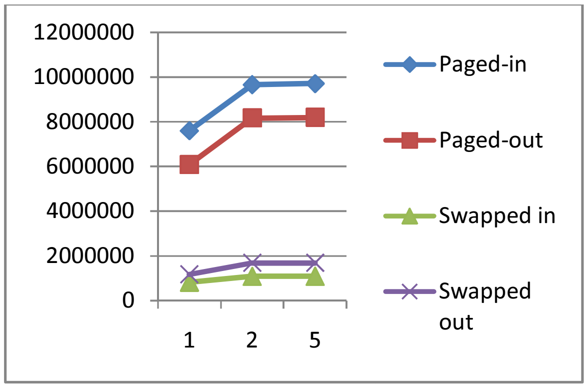


Figure : 7

The figure 1- 5 depicts the load caused on the CPU, Memory and the Network bandwidth at the time slots of immediately after submitting the jobs, after 2 minutes and after 5 minutes respectively. The figure 6 clearly specifies that the error is caused due to Thrashing. Also, table-1 shows the statistical analysis of the number of Swap-ins, Swap-outs, Page-ins and Page-outs caused at various time intervals. Thus we can clearly say that the Thrashing has occurred. The Figure 7 shows the graph of the number of Swap-ins/outs and Pagings. Hence, the following section discusses about the solution of Thrashing.

V. SOLUTION:

Hence, we propose a new approach to minimize the thrashing when considering the resident memory or hard limit of memory. In this approach, when a job is submitted to the grid, the scheduler checks whether it is being requested for the first time or been submitted earlier.

If it is submitted for the first time, scheduler follows the default load balancing function and allocates the job to execution host and takes the snap shots of resource utilization from time to time by the job. Otherwise considers already tracked out history of the submitted job and verifies the available resources and then allocates the job to the best suitable execution host.

Hence, by this approach, it theoretically minimizes the problem of thrashing caused due to the lack of required resources for the job at run time. This function is incorporated into the existing SGE prior to the load Scheduler function is performed. The Practical implementation of this approach is under process.

VI. CONCLUSION

In this approach, we discussed some load balancing features available in existing grid engine and also tried to project the drawback of existing and a probable solution too. This work is proposed to implement on existing SGE 6.1 and incorporate the new algorithm which is modified before the load balancing function is executed for the allocation of resources.

REFERENCE

[1]. Foster. I and Kesselman, C.: “The Grid2: Blueprint for a new computing infrastructure”.

[2]. Buyya,R, Abrason,D and Giddy(2000), “Nimrod-G: An architecture for a resource management and scheduling system in a global computational Grid”, The 4th International conference on High performance Computing in Asia-Pacific Region(HPC Asia 2000), Beijing, China,2000.

[3]. I. Foster and C.Kesselman. Globus: A metacomputing infrastructure toolkit.,1997

[4]. M.Litzkow, M.Livny and M.Mukta, “A hunter of idle workstations” In Proceeding of the 8th International Conference on Distributed Computing Systems, pp.: 104 – 111, 1988.

[5]. SunGridEngine: http://www.sun.com/ software/ Gridware/.

[6]. Karuna et al, “PARYAVEKSHANAM: A Status Monitoring Tool For Indian Grid GARUDA”

[7] Wei Li, Zhiwei Xu, Bingchen Li, Yili Gong, “The Vega Personal Grid: A Lightweight Grid Architecture”, Fourteenth IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2002)

[8]. Open PBS, http://www.openpbs.org

[9]. Rakesh Kumar & Navjot Kaur, “Job Scheduling in Grid Computers”.

[10]. Volker Hamscher et al, “Evaluation of job-scheduling strategies for Grid Computing”. Unpublished.

[11]. Lanier Watkins & Rahim Beyah, “A passive solution to the memory resource discovery problem in computational clusters”, IEEE Transactions on Network and Service management, Vol.:7, No.4, Dec 2010. Pp.:218-230.

[12]. Maozen Li and Marios Hadjini Colaou, “Curriculum development on Grid Computing”, International Journal of Education and Information Technologies, Issue 1, Vol. 2, 2008.

[13]. C Grigoras et al., “ MonALISA – Based Grid Monitoring and Control”, The European Physical Jouranl Plus.

[14]. GARUDA: www.garudaindia.in

[15]. Tinghuai Ma et al, “Grid Task Scheduling: Algorithm Review”, IETE Technical Review, Vol.:28, Issue 2, Mar-Apr 2011, pp.: 158 – 167.

[16]. Raihan ur Rasool, Guo Qingping and Zhou Zhen, “Users-Grid: A Unique and Transparent Grid – Operating System”, In Publication: Ubiquitous Computing & Communication Journal.

❑❑❑