ISSN (ONLINE): 2349-7084
GLOBAL IMPACT FACTOR 0.238
DIIF 0.867

INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 6, JUNE 2015, PP 392-394

A Concern towards Job scheduling In Cluster Computing

Savita Mahajan, Rasleen Kaur

Abstract— Cluster computing is one of the novel and effective approach in parallel processing. As cluster installation helps to satisfy ever-increasing computing demands, so advance schedulers can help in improving resource utilization rate and quality of service. In this paper, we focus on features for developing external schedulers that can complement features of resource manager and enable sophisticated scheduling.

Index Terms— Cluster Computing, Resource Management System, Scheduling, Scheduler.

1 INTRODUCTION

PARALLEL processing is a term used to denote a large class of techniques that are used to provide simultaneous data-processing tasks for the purpose of increasing the computational speed of a computer system. Instead of processing each instruction sequentially as in a conventional computer, a parallel processing system is able to perform concurrent data processing to achieve faster execution time. Clusters are one of the prominent trends in parallel processing nowadays. A cluster is composed of a number of PC's, workstations or servers connected together via a network. In a cluster, each node is a complete PC with its own processor(s) and memory. Cluster installations mainly involve two type of standard hardware components i.e. server and networking interconnects. Clusters can be categorised into two major classes: high-throughput computing clusters and high-performance computing clusters [9, 4]. High throughput computing clusters connect a large number of compute node using low-end interconnects. In contrast, high-performance computing clusters connect more powerful compute node using fast-end interconnects i.e. are designed to provide low latency and higher bandwidth.

There are different scheduling requirements for these two classes of clusters. In high-throughput computing clusters, the major objective is to maximize throughput (jobs completed per unit time) by reducing load imbalance among compute node in cluster. These clusters operate by routing all work through one or more load-balancing front-end nodes, which then distribute the workload efficiently between the remaining active nodes. Devoting a few nodes to managing the workflow of a cluster ensures that limited processing power can be optimized. Load balancing is of high concern for heterogeneous compute nodes. In high-performance computing clusters, the main goal is to minimize communication overhead associated with throughput by mapping properly to the available compute nodes. They are most commonly used to perform functions that require nodes to communicate as they perform their tasks – for instance, when calculation results from one node will affect future results from another. High throughput computing clusters are used for executing loosely coupled parallel or distributed applications as these applications do not have high communication requirements among compute nodes whereas high performance computing clusters are used for tightly coupled parallel applications that have communication and synchronization requirements [10].

2 RELATED WORK

Many research work has been carried out in relation to scheduling job in cluster computing. Cluster computing has come to distinction as a cost-effective parallel processing tool for solving many complex computational problems. In heterogeneous multi-cluster systems, scheduling methods can be classified into two categories, single-site allocation and multi-site co-allocation, depending on whether the system supports a job to be executed across different clusters [9]. J. Ramírez-Alcaraz, et al. [7, 12] states that central job scheduler is commonly used to dispatch all submitted jobs. In utility-driven cluster computing, cluster Resource Management Systems (RMSs) need to know the specific needs of different users in order to allocate resources according to their needs [3]. In single site allocation, Huang and Chang [8] studied many classical methods, such as First-Fit, Best-Fit, Worse-Fit, Median-Fit, and Random Fit. During scheduling, there are two major factors which would highly affect overall system performance: speed heterogeneity and resource fragmentation. Moreover, the relative effect of these two factors changes with different workloads and resource conditions. Processor allocation methods have to deal with this issue an intelligent processor allocation is proposed

- *Savita Mahajan is currently pursuing M.Tech in Computer Science & Engineering in GIMET, Amritsar, Punjab, India. E-mail: savita_eng1991@yahoo.in*
- *Rasleen Kaur is working as Asst. Prof. in Department of Computer Science & Engineering in GIMET, Amritsar, Punjab, India. E-mail: rasleenkaur@gmail.com*

ISSN (ONLINE): 2349-7084

INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING IN RESEARCH TRENDS
VOLUME 2, ISSUE 6, JUNE 2015, PP 392-394

[12]. An on-line dynamic scheduling policy has also been proposed that manages multiple job streams across both single and multiple cluster computing systems with the objectives of improving the mean response time and system utilization and has shown performance improvement up to 30% [1].

3 JOB SCHEDULING IN CLUSTERS

Job scheduling in clusters is based on typical resource management system, which manages the load imbalance by preventing jobs competing with each other for limited resources [1]. Typical resource management system consists of a resource manager and a job scheduler. Resource manager contains information about queues, loads on compute nodes and resources requested and availability to make scheduling decisions [7, 3]. Scheduler receives periodic input from resource manager queue and resource availability and decides which job will execute on which compute node and when.

In high-performance parallel computing, scheduling a parallel job is a difficult task to be operate because a parallel job consists of several subtasks. Each subtasks is assigned to distinct compute node during process of execution and nodes communicate with each other. The scheduler must map subtasks carefully, because mapping (i.e. manner in which subtasks are assigned to processors) affects the execution time. The responsibility of scheduler is to make sure that nodes scheduled to execute parallel jobs are connected to fast-end interconnects to minimize the associated communication overhead [4].

In high throughput parallel computing, during job execution, large jobs can occupy major portions of clusters processing and memories [4]. In heavy load conditions, providing fair share of resources to each user is an important task. This strength can be given only by using fair-share strategy, which allows the scheduler to collect historical information from previously executed jobs and make dynamic decisions to adjust the priority of the jobs in the queue, which ensure that resources are fairly distributed among users.

4 JOB SCHEDULING ALGORITHMS

Generally scheduling algorithms can be classified into two major classes: time sharing and space sharing. Time sharing algorithms are those algorithms that divide time on a processor into different discrete intervals or slots. These slots are then assigned to distinct jobs. Numerous jobs can use the same compute resources at any given time. They are pre-emptive in nature. In contrast, space sharing algorithms gives all requested resources to a single job until job completes the execution i.e. it is non pre-emptive in nature. Mostly cluster schedulers operate in space sharing mode.

The most commonly used time sharing algorithms are: gang scheduling and local scheduling. Local scheduling is used in scheduling concurrent processes to the time slices of a single processor. Global scheduling is the assignment of tasks to processors in parallel systems. The several space sharing algorithm are: first come, first served (FCFS); first in, first out (FIFO); round robin (RR); shortest job first (SJF); longest job first (LJF); best fit (BF); worst fit (WF); random fit (RF); advance reservation and backfilling [8].Advance reservation scheduling technique uses the execution time information provided the user in order to reserve resources i.e. CPU's and memory, and generate schedule accordingly. The backfilling technique is enhancement of space-sharing scheduling. Consider an advance reservation schedule having a list of high priority and low priority jobs, the function of backfill is to fit in the small jobs into scheduling gaps [11].

5 FEATURES OF JOB SCHEDULER

5.1 Scope

The job schedulers have broad perspective as the nature of jobs submitted to a cluster can vary significantly. The scheduler can support batch, sequential, parallel, distributed, interactive and non-interactive with equal efficiency [10].

5.2 Support for Algorithms

One of the extraordinary feature of the scheduler is support for several job-processing algorithms such as FCFS, FIFO, RR, SJF, LJF, BF, RF, WF, advance reservation and backfill. The scheduler are able to switch among different algorithms and apply different algorithms at different time intervals or apply different algorithms to different queues, or both [8].

5.3 Support for Pre-emption

Scheduler support for pre-emption can occur at various levels of job processing. For instance, jobs may be suspended while running. Checkpointing (the capability to stop running job), save intermediate results and restart the job later, which help to ensure that results are not lost for large jobs [9].

5.4 Fair-share Capability

The scheduler has the capability to provide fair share of resources to its users under heavy load conditions and at different time intervals.

5.5 Scalability

The tremendous feature of scheduler is its capability of scaling to thousands of nodes and processing of jobs simultaneously. As cluster sizes scale to satisfy ever-increasing computing demands.

5.6 Efficiency

The several advance scheduling algorithms available take time to execute. So to be efficient, the scheduling algorithm itself must spend less time running than the expected saving in application execution time from improved scheduling. The overhead associated with scheduling should be minimal and within acceptable limits [2].

5.7 Strength to Interface with Standard Resource Managers

The scheduler must have capability to integrate with resource manager in use and commonly available resource manager such as Sun Grid Engine, Platform LSF (Load Sharing Facility) and open PBS (Portable Batch System).

5.8 Dynamic in Nature

The scheduler must be capable of adding or removing compute resources to a job on a fly making assumption that the job can adjust and utilize the extra compute capability.

5.9 Sensitivity to Compute Node and Interconnect Architecture

The scheduler should match the appropriate compute node

architecture to the job profile. For example, by using compute nodes that have more than one processor to provide optimal performance for applications that can use the second processor efficiently [6].

6 CONCLUSION

Cluster computing is an on demand technology that helps to satisfy ever-growing computing demands. In heterogeneous clusters, either dedicated or non-dedicated, have individual computing characters. As heterogeneous node computing characteristics are unique, node finish times will differ for each. As a result, faster nodes will have to wait in idle states for slower nodes that lead to inefficient use of cluster resources. Increase in resource utilization rate is an important issue for resolution. Hence, there is need of effective scheduling policy which will overcome the problem like resource fragmentation, speed heterogeneity, system loading, and for which different workload source is required.

6 REFERENCES

[1] Abawajy, J.H..2003 Parallel job scheduling on multicluster computing system. Cluster Computing, IEEE 2003.

[2] C. Ernemann, V. Hamscher, R. Yahyapour. 2004 Benefits of Global Grid Computing for Job Scheduling. In proceedings of the Fifth IEEE/ACM International Workshop on Grid Computing (GRID'04), 374-379, November 2004.

[3] Chee Shin Yeo,Rajkumar Buyya. 2006 A taxonomy of market-based resource management systems for utility-driven cluster computing. Published in Journal Software—Practice & Experience archive VoL. 36, No.13, November 2006, 1381 - 1419.

[4] E. Carsten, et al. 2002 On advantages of grid computing for parallel job scheduling.in: Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGrid'02, 2002, pp. 39–39.

[5] Ernemann, C., Hamscher, V., Streit, A., Yahyapour, R. 2002 Enhanced Algorithms for Multi-Site Scheduling. In: Parashar, M. (ed.) GRID 2002. LNCS, vol. 2536, pp. 219–231. Springer, Heidelberg (2002).

[6] http://www.adaptivecomputing.com/products/open-source/maui/.

[7] J. Ramírez-Alcaraz, et al. 2011 Job allocation strategies with user run time estimates for online scheduling in hierarchical grids. Journal of Grid Computing (2011), Vol.9, 95–116.

[8] K.-C. Huang, H.-Y. Chang. 2006 An integrated processor allocation and job scheduling approach to workload management on computing grid. In Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications, PDPTA'06, 2006, pp. 703–709, Las Vegas, USA.

[9] K.-C. Huang, et al. 2007 Towards feasible and effective load sharing in a heterogeneous computational grid. In Proceedings of the 2nd International Conference on Advances in Grid and Pervasive Computing, GPC'07, 2007, pp. 229–240.

[10] Marco Pasquali, Ranieri Baraglia, Gabriele Capannini, Laura Ricci, Domenico Laforenza. 2011 A multi-level scheduler for batch jobs on grids. The Journal of Supercomputing, Vol. 57, No. 1, 81-98, July 2011.

[11] M. Hovestadt, O. Kao, A. Keller, and A. 2003 Streit. Scheduling in HPC Resource Management Systems: Queuing vs. Planning. In D. G. Feitelson and L.Rudolph, editor, Proc. of the 9th Workshop on Job-Scheduling Strategies for Parallel Processing, vol. 2862 of Lecture Notes in Computer Science,1– 20, Springer, 2003.

[12] P.-C. Shih, et al.. 2010 Improving grid performance through processor allocation considering both speed heterogeneity and resource fragmentation. The Journal of Supercomputing (2010), 1–27.