ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 3, Issue 2, pp: (366-368), Month: April - June 2015, Available at: www.researchpublish.com

Benefit of Processor Co-Allocation in Multi-Cluster Grid Systems

[1]Anju Kundal, [2]Aman Attri, [3]Seema Devi

Abstract: **In multi-cluster systems, parallel applications may benefit from processor co-allocation, that is, the simultaneous allocation of processors in multiple clusters. Co-allocation allows the allocation of more processors than available in a single cluster, it also increase the execution time of applications due to the relatively slow wide area communication. The aim of this paper is to discuss the benefits of co-allocation in multi-cluster systems. In addition, we compare the performance of scheduling policies that are specifically designed for co-allocation.**

Keywords: **multi-cluster systems, processor, co-allocation.**

I. INTRODUCTION

OVER the last decade, multi-cluster systems have become the mainstream execution environment for many large-scale applications. In such systems, parallel applications may benefit from using resources such as processors in multiple clusters simultaneously, that is, they may use *processor co-allocation*. This leads to higher system utilizations and lower wait times by allowing parallel jobs to run when they need more processors than are available in a single cluster. With processor co-allocation, the execution time of parallel applications may severely increase due to wide-area communication overhead and processor heterogeneity among the clusters.

II. A JOB MODEL FOR PARALLEL APPLICATIONS

A job comprises either one or multiple *components* that can be scheduled separately on different clusters, and that together execute a single parallel application. A job specifies for each component its requirements and preferences, such as its size (the number of processors or nodes it needs) and the names of its input files. A job may or may not specify the execution sites where its components should run. In addition, a job may or may not indicate how it is split up into components. Based on these distinctions, we consider three job request structures, *fixed* requests, *non-fixed* requests, and *flexible* requests. In a fixed request, a job specifies the sizes of its components and the execution site on which the processors must be allocated for each component. On the other hand, in a non-fixed request, a job also specifies the sizes of its components, but it does not specify any execution site, leaving the selection of these sites, which may be the same for multiple components, to the scheduler. In a flexible request, a job only specifies its total size and allows the scheduler to divide it into components (of the same total size) in order to fit the job on the available execution sites. With a flexible request, a user may impose restrictions on the number and sizes of the components. For instance, a user may want to specify for a job a lower bound on the component size or an upper bound on the number of components.

III. JOB PLACEMENT POLICIES

A. The Worst Fit Policy

The Worst Fit (WF) policy keeps the load across clusters balanced. It orders the components of a job with a non-fixed request type according to decreasing size and places them in this order, one by one, on the cluster with the largest (remaining) number of idle processors, as long as this cluster has a sufficient number of idle processors. WF leaves in all clusters as much room as possible for later jobs, and hence, it may result in coallocation even when all the components of the considered job would fit together on a single cluster.

B. The Flexible Cluster Minimization Policy

The Flexible Cluster Minimization (FCM) policy is designed with the motivation of minimizing the number of clusters to be combined for a given parallel job in order to reduce the number of inter-cluster messages. FCM first orders the clusters according to decreasing number of idle processors and considers component placement in this order. Then FCM places on clusters one by one a component of the job of size equal to the number of idle processors in that cluster. This process continues until the total processor requirement of the job has been satisfied or the number of idle processors in the system has been exhausted, in which case the job placement fails.

IV. THE IMPACT OF SYSTEM PROPERTIES ON CO-ALLOCATION PERFORMANCE

A. The Impact of Inter-Cluster Communication

In a multi-cluster system environment, it is likely that the inter-cluster communication is slower than the intra-cluster communication in terms of latency and bandwidth, which are the key factors that determine the communication performance of a network. This slowness, in fact, depends on various factors such as the interconnect technology that enables the inter-cluster communication among the processes of a parallel application, the distance between the clusters, the number and capabilities of the network devices, and even the network configuration. Therefore, depending on the communication requirements of a parallel application, the inter-cluster latency and bandwidth may have a big impact on its execution time performance.

B. The Impact of Heterogeneous Processor Speeds

Unless an application developer does take into account processor speed heterogeneity and optimizes his applications accordingly, the execution time of a parallel application that runs on co-allocated clusters will be limited by the speed of the slowest processor, due to the synchronization of the processes. This is a major drawback of co-allocation especially for computation intensive parallel applications which do not require intensive inter-cluster communications.

V. ADAPTIVE PROCESSOR ALLOCATION USING MOLDABILITY

When a job cannot fit in any single site in a computational grid, in addition to multisite parallel execution, adaptive processor allocation is another choice which allocates a smaller number of processors than specified upon submission to a job, allowing it to fit in a single site for immediate execution. This would improve system utilization and shorten the waiting times for user jobs at the cost of enlarged job execution time.

The major difference between the adaptive processors allocation procedures for a single-site parallel computer and for a heterogeneous grid environment is the site selection process regarding the calculation and comparison of computing power of different sites. A site's free computing power is defined as the number of free processors on it multiplied by the computing speed of a single processor. Similarly, the required computing power of a job is defined as the number of required processors specified upon job submission multiplied by the computing speed of a single processor on its home site.

VI. SITE SELECTION POLICIES FOR LOAD SHARING IN A HETEROGENEOUS GRID

A. Independent clusters

This corresponds to the situation where no grid computing technologies are involved. The computing resources at different sites are independent and have their own job queues without any load sharing activities among them. Each site's users can only submit jobs to their local site and those jobs would be executed only on that site. This architecture is used as a comparison basis to see what performance gain grid computing can bring.

B. Load-sharing computational grid

Different sites connected with an interconnection network form a computational grid. In the computational grid, there is a global job scheduler as well as a globally shared job queue. Jobs submitted by users at different sites are automatically redirected to the global queue and the jobs retain the identities of their home sites. In this section, different sites in the computational grid are viewed as different processor pools and each job must be allocated to exactly one site. No jobs can simultaneously use processors on different sites.

VII. CONCLUSIONS

For heterogeneous grid environments, no existing processor allocation methods can consistently deliver the best performance under different resource and workload conditions. Processor co-allocation lessens the waiting time for the job and improves the execution time of the job. Sometimes it takes more execution time due to communication overhead.

REFERENCES

[1] O. Beaumont, A. Legrand, Y. Robert, Optimal algorithms for scheduling divisible workloads on heterogeneous systems, in: Proceedings of the International Parallel and Distributed Processing Symposium, 2003.

[2] Huang, K.-C. and Chang, H.-Y. 2006. An Integrated Processor Allocation and Job Scheduling Approach to Workload Management on Computing Grid. In the Proceedings of the 2006 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'06), Las Vegas, USA, 703-709.

[3] Huang, K.C., Shih P.C. and Chung, Y.C. 2007. Towards feasible and effective load sharing in a heterogeneous computational grid. In the Proceedings of the second international conference on Advances in grid and pervasive computing, ed, 2007, 229-240.

[4] Huang, K.C., Shih P.C. and Chung, Y.C. 2009. Adaptive Processor Allocation for Moldable Jobs in Computational Grid. At 10th International Journal of Grid and High Performance Computing, 1(1), (March 2009), 10-21.

[5] Bucur, A.I.D. 2004. Performance analysis of processor co-allocation in multicluster systems. PhD Thesis, Delft University of Technology, Delft, The Netherlands.

[6] Huang, K.C., Shih P.C. and Chung, Y.C. 2008. Adaptive Processor Allocation with estimated job execution time in Heterogeneous Computing Grid.