A Survey of Load Balancing in Grid Computing

Yawei Li, and Zhiling Lan

Department of Computer Science
Illinois Institute of Technology, Chicago, IL 60616
{liyawei, lan}@iit.edu

Abstract. Although intensive work has been done in the area of load balancing, the grid computing environment is different from the traditional parallel systems, which prevents existing load balancing schemes from benefiting large-scale parallel applications. This paper provides a survey of the existing solutions and new efforts in load balancing to address the new challenges in grid computing. We classify the surveyed approaches into three categories: resource-aware repartition, divisible load theory and prediction based schemes.

1 Introduction

Due to its critical role in high-performance computing, the load balancing issue has been studied extensively in recent years and a number of load balancing toolkits are available for parallel computing [4, 9, 11]. Most of these works focus on traditional distributed systems and are no longer suitable for the emerging Grid [6, 8]. The Computation Grid evolves high performance computing platforms to heterogeneous, dynamic and shared environments, which present obstacles for applications to harness conventional load balancing techniques directly. Thus how to migrate or adapt load balancing schemes to Grid becomes the focus of this research area.

The rest of this paper is organized as follows. Section 2 briefly describes the load balancing problem and gives taxonomy of traditional load balancing work. Section 3 elaborates and compares the current load balancing approaches in Grid environment. Section4 draws a concise conclusion.

2 Background of Load balancing

For parallel applications, load balancing attempts to distribute the computation load across multiple processors or machines as evenly as possible with objective to improve performance. Generally, a load balancing scheme consists of three phases: information collection, decision making and data migration. During the information collection phase, load balancer gathers the information of workload distribution and the state of computing environment and detects whether there is a load imbalance. The

decision making phase focuses on calculating an optimal data distribution, while the data migration phase transfers the excess amount of workload from overloaded processors to underloaded ones.

Based on different policies used in the three phases, load balancing schemes can be classified in three dimensions as follows:

1. *Static Schemes vs. Dynamic Schemes*: Static schemes [7] assume *priori* knowledge of both application and system state. Static load balancing is advantageous in terms of low overhead as the decision is only made once before computation; however, it cannot adapt to fluctuation of application computation requirement and system state. Dynamic load balancing (DLB) is proposed to make decision based on the changes of application and system environment.
2. *Centralized Schemes vs. Distributed Schemes:* In centralized schemes, a central manager takes charge of collecting load information and making decision based on global knowledge. Centralized schemes usually entail global synchronization to obtain global information at the cost of high synchronization cost. Distributed load balancing allows every processor maintains its local view of workload distribution and makes decision based on the partial knowledge. Typical distributed load balancing schemes are neighbor-based, such as diffusion method [2]. The lack of global knowledge slows down the convergence rate of global balancing, e.g., diffusion algorithm requires quadratic time complexity.
3. *Application-level Schemes vs. System-level Schemes:* Application-level load balancing focuses on minimizing the makespan of a parallel application. Here, makespan is defined as the completion time of the entire job. System-level load balancing, also known as distributed scheduling, is to maximize process throughput or the overall utilization rate of the machines.

.3 Load Balancing Schemes for Grid Computing

Recently, there are some efforts in load balancing attempting to cope with one or more of the challenges brought by Grid: heterogeneity, resource sharing, high latency and dynamic system state. We classify these schemes into three categories as listed below.

3.1 Resource-aware Repartition Based Schemes

Graph repartition is the most dominant dynamic load balancing approach in scientific computing area [11]. The repartition schemes model the problem domain as a graph consisting of weighted vertices and communication edges. Consequently, the load balancing problem is transformed into one with goal to reduce the imbalance among subdomains as well as minimize the communication edge-cut. Figure 1 illustrates the typical interactions between partition-bases algorithms and parallel applications.

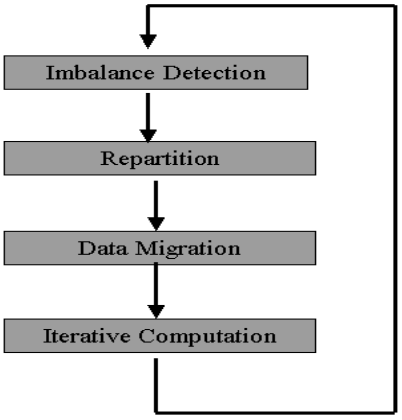

Fig. 1. Interactions between repartition algorithms and applications

Unfortunately, traditional partitioners share the drawback that stems from the *priori* knowledge of the workload and system state, and cannot adapt to system changes. Meanwhile, they mainly focus on tightly-coupled systems consisting of homogenous processors thus cannot handle the heterogeneity in Grid. Furthermore, these algorithms only consider the imbalance of subdomains and the edge-cut communication, while neglect the data movement cost that can be a crucial performance bottleneck due to the high communication latency in Grid.

To address the resource heterogeneity, the DRUM [5] project proposes a resource-aware partition model cooperating with the Zoltan toolkit [4] to perform dynamic load balancing operation. As indicated in formula (1), DRUM uses a linear model to calculate a scalar sum of node power, which uses both static benchmark data and dynamic performance data collected by monitor agents.

$$power_n = w_n^{comm} c_n + w_n^{cpu} p_n, w_n^{comm} + w_n^{cpu} = 1 \quad (1)$$

Here, p_n is the processing power on node n determined by CPU utilization rate, idle time and benchmark rating; c_n is the communication power on node n that reflects the dynamic infomation on each communication channel; w_n^{cpu} and w_n^{comm} are the weight factors for processing and communication power respectively; $power_n$ denotes the synthetic node power. Based on the node power above, the partition algorithm calculates a non-uniform workload distribution in heterogeneous machines. However, in this scheme, only CPU contention is considered, the impacts of bandwidth and memory contention are not addressed yet.

Similar to DRUM, GRACE [12] utilizes the NWS [13] package, a Grid resource performance monitoring toolkit, to obtain runtime state, e.g. the CPU available time, end-to-end TCP network bandwidth and free memory. Then the proposed system-sensitive partitioner calculates a new workload partition in proportional to the combined capacity metric synthesizing all the above runtime information. Although this approach can

harnesses heterogeneous resources, currently it simply assumes that each resource has equal weight, which cannot be adjusted according to the application characteristics. Thus this partitioner cannot reflect the adaptive requirements of applications.

Different from the above two partitioners, the MinEX [3] partitioner proposes a latency-tolerant algorithm that emphasizes minimizing the data movement cost incurred by load balancing operation. The rationale behind MinEX is to initiate the communication as early as possible. Therefore it can take advantage of overlapping the computation of internal data with the communication of incoming data as much as possible. The inter-cluster experiments show that this approach can efficiently hide the communication latency in Grid.

3.2 Divisible Load Theory (DLT) [10] Based Schemes

In DLT, both the computation and communication load of applications can be arbitrarily divided. Furthermore, there is no special relation among data, which allows workload to be assigned to multiple processors or machines in a linear way.

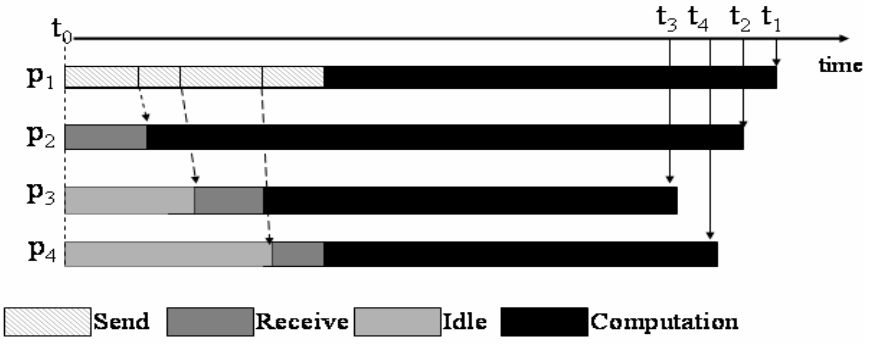

Fig. 2. A load balancing scattering operation

In [7], the MPI_Scatterv primitive in MPI is enhanced to support static load balancing data scattering in Grid environment. The authors attempt to find an optimal data distribution based on the DLT theory. For example, as indicated in figure 2, a Gantt-char-like timing diagram illustrates the communication and computation phases of divisible load. The application is running on four processors denoted by Pi. Root processor P1 scatters data to other processors at the initial time T0 then begins to process its own workload. Other processors start computation after receiving workload from root processor. The computation on each processor stops at the time Ti respectively. According to the DLT, the objective is to find out an optimized data distribution that minimizes the makespan of the application so that all processors can stop at the same time. The problem is modeled as a linear optimization problem under the subject to the communication latency and computation time, and a guaranteed heuristic solution is proposed to ensure all the sub-tasks end simultaneously. This method takes full consideration of the communication latency in Grid and makes an optimized load distribution decision. However, it suffers from the inability to dynamic

changes of the processor utilization rate and fluctuations of network channel, which limits this approach to static load balancing.

Cyril Banino and Olivier Beaumont [1] recently propose a master-slave scheme for divisible load in heterogeneous platform. This scheme assumes parallel applications will always enter a steady state in which they spend a fix time fraction on computation and communication respectively. Therefore, instead of minimizing the makespan of the application, this scheme targets on maximizing the throughput of steady-state applications in unit time so as to avoid NP-complete complexity of makespan optimization. The objective of steady state optimization is to find out an optimal solution of time fractions application spent in computing, sending and receiving tasks. Another advantage of their work is that this model is very flexible and can extend to arbitrary platform topologies or application domains with different computation and communication requirement. But the validity of the steady state assumption still needs further verification, especially for adaptive applications.

3.3 Prediction Based Schemes

Load balancing schemes require accurate estimation of the future computation time and communication cost to establish the performance evaluation model. Simply using linear expression to calculate those values may be insufficient in Grid as unexpected system state changes and resource contention make the future system state quite different from the current.

For the Cactus application [14], Yang et al. propose a conservative time balancing scheme to calculate the load distribution of the Cactus application by utilizing the prediction value of host load in Grid. They use an aggregated time series prediction technique and give a simple but effective way to estimate the processor computation power during the period when the application will run. The work is allocated in a way so that the execute time of application is minimized. The experiments show that this scheme works well for short-time task. However, the performance model only focuses on the Cactus application, thus cannot directly apply to other applications, especially long-term running jobs.

Charm++ [9] also integrates prediction strategies into the load balancing framework. Charm++ uses multiple build-in prediction strategies to forecast the future system load so that the load balancer can make partition decision more accurately. It also exposes a callback interface that allows user implements prediction functions and gives load prediction hint to load balancing framework, which makes the predicted value more adaptive to the application at the cost of transparence.

4 Conclusions

These works make progress in addressing one or some challenging issues in load balancing for grid computing, but no solution is fully adaptive to the characteristics of Grid. Repartition methods focus on calculating data distribution in a heterogeneous

way, but do not address the issue related to the cost incurred by data movement in Grid. DLT-based schemes can effectively model both computation and communication; however, the validation of these schemes for adaptive applications is needed. Prediction based schemes need further investigation in case of long-term applications.

References

1. Banino, C., Beaumont, O., Carter, L.: Scheduling Strategies for Master-Slave Tasking on Heterogeneous Processor Platforms, IEEE Transactions on Parallel and Distributed Systems, Vol. 15, No. 4, (2004) 319-330
2. Cybenko, G.: Dynamic load balancing for distributed memory multiprocessors, Journal of Parallel and Distributed Computing, Vol. 7, Issue 2. (1989) 279 - 301
3. Dasa, S.K., Harvey D.J., Biswas, R.: MinEX: A latency-tolerant dynamic partitioner for grid computing applications, Future Generation Computer Systems, Vol. 18, No. 4,(2002) 477--489
4. Devine, K., Hendrickson, B., Boman, E., John, M.S., Vaughan, C.: Design of Dynamic Load-Balancing Tools for Parallel Applications, Proceedings of the International Conference on Supercomputing, Santa Fe, (2000) 110 - 118
5. Faik, J., Gervasio, L. G., Flaherty, J. E., Chang, J. , Teresco,: A model for resource-aware load balancing on heterogeneous clusters, Tech. Rep. CS-03-03, Williams College Department of Computer Science, http://www.cs.williams.edu/drum/
6. Foster, I., Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann Publishers, San Francisco, California, (1999).
7. Genaud, S., Giersch, A., Vivien, F.: Load-Balancing Scatter Operations for Grid Computing, International Parallel and Distributed Processing Symposium, Nice, France, (2003)
8. Johnston, W., Gannon, D., Nitzberg, B.: Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. IEEE Computer Society Press, (1999).
9. Laxmikant V. K., Krishnan, S.: CHARM++: a portable concurrent object oriented system based on C++, Proceedings of the eighth annual conference on Object-oriented programming systems, languages, and applications, Washington, D.C., (1993) 91-108
10. Thomas G. R.: Ten Reasons to Use Divisible Load Theory. IEEE Computer Vol. 36, No. 5, (2003) 63-68
11. Schloegel, K., Karypis, G..: Multilevel Diffusion Schemes for Repartitioning of Adaptive Meshes. Journal of Parallel and Distributed Computing, Vol.47, Issue 2, (1997) 109-124
12. Sinha, S., Parashar, M.: Adaptive System-Sensitive Partitioning of AMR Applications on Heterogeneous Clusters, Cluster Computing: The Journal of Networks, Software Tools, and Applications, Kluwer Academic Publishers, Vol. 5, Issue 4, (2002) 343-352
13. Wolskiy, R.: Dynamically Forecasting Network Performance Using the Network Weather Service, UCSD Technical Report TR-CS96-494, (1998)
14. Yang, L., Schopf, J. M., Foster, I.: Conservative Scheduling: Using Predicted Variance to Improve Scheduling Decisions in Dynamic Environments, Supercomputing, Phoenix, Arizona, (2003)