

Adaptive Load Balancing for Scientific Applications on Computational Grids

Sumanth J.V, David R. Swanson, Hong Jiang
Department of Computer Science and Engineering
University of Nebraska-Lincoln, U.S.A
{sumanth, dswanson, jiang}@cse.unl.edu

“As computer networks become cheaper and more powerful, a new computing paradigm is poised to transform the practice of science and engineering.”

-Ian Foster

I. INTRODUCTION

Grid Computing, an emerging computing model, allows for the solving of massive computational problems by making use of unused resources of a largely heterogeneous collection of computers treated as a virtual cluster embedded in a distributed networking infrastructure. It involves sharing heterogeneous resources spanning multiple administrative domains.

To maximize the performance of an existing parallel application, an adaptive load balancing mechanism is essential to ensure maximum resource utilization. However, the highly dynamic nature of the grid (with respect to computation and communication) poses a significant challenge to the load balancing algorithm.

This poster presents a load balancing technique that is particularly well suited for iterative algorithms that operate on divisible loads. A system model that can make reasonably accurate predictions about the execution time is developed. The load balancer uses the performance (computation and communication) observed during the previous iteration to improve the load distribution. The need for a more general feedback control system is motivated so that load balancing can be performed optimally for a larger class of algorithms.

II. ARCHITECTURE

In a typical grid environment, the number of compute nodes available to a particular application can vary continually. Nodes can be revoked by their owners at any point of time during the computation. Background processes of the client node's owner such as operating system updates, anti-virus software, email clients etc. can also interfere with the available compute power available to the application.

We have currently deployed our applications on top of Condor, a cycle scavenging batch scheduler. We utilize a client/server architecture where computations are performed on as many clients as available. The server is responsible for the initial preparations for the computations such as reading data sets from a file, communicating with the available clients and load balancing the amount of work assigned to each client.

The client is responsible for initializing communication with the server and performing the assigned computations.

III. SYSTEM MODEL AND LOAD BALANCER

Our load balancing algorithm begins by constructing a system model that is capable of predicting the execution time of the application given the data set size. We model the execution time of a $\Theta(n^k)$ application with an k^{th} -degree polynomial whose co-efficients are initially determined by a benchmark before joining in the computation. This is applicable to applications where the number of operations performed by the innermost loop of the computation is a function of only the data set size.

Iterative algorithms such as PDE solvers and MD simulations are characterized by repeating identical operations at every iteration. This allows us to update the co-efficients of the prediction polynomial based on the execution times observed at every iteration. Updating the polynomials at every iteration allows the load balancer to adapt to changes in the available computational power, network latency and bandwidth. Since the execution times are measured from the server side, these polynomials also reflect the communication performance of the underlying networks.

The load balancer then tries to determine the optimal partitioning of the data set such that all the clients can complete execution within the same amount of time. Since the prediction polynomials monotonically increase with their independent variable in the considered range, a binary search can be performed to determine the execution time at which all the nodes can complete their assigned computations.

We have successfully applied this technique to long range MD simulations and obtained very encouraging results. This work can be extended to non-iterative algorithms if a multi-round approach is taken.

IV. CONCLUSION

This poster describes a novel approach to load balancing a certain class of scientific applications. The load balancer works by using a system model to predict the performance of the distributed system. The system model is constantly updated based on previously observed performance. We have implemented MD simulations on this framework with encouraging results. We are currently looking into feedback control systems to generalize this approach.