ABSTRACT

The problem of load balancing parallel applications is particularly challenging on computational grids, since the characteristics of both the application and the platform must be taken into account. This chapter reviews the wide range of solutions that have been proposed. It considers tightly coupled parallel applications that can be described by an undirected graph representing concurrent execution of tasks and communication of tasks, executing on computational grids with static and dynamic network and processor performance. While a rich set of solution techniques have been proposed, there has not been of yet any performance comparisons between them. Such comparisons will require parallel benchmarks and computational grid emulators and simulators.

INTRODUCTION

Distributed high performance computing (HPC) applications have formed an important class of grid applications from the early days of the I-Way (DeFanti et al., 1996; Foster and Kesselman, 1999) to the TeraGrid (http://www.teragrid.org) of today. The main reason is that the aggregation of multiple parallel computers permits problem solutions that require more resources than are available in a single system. Many of these applications, such as partial differential equation (PDE) solvers, can be described by an undirected graph representing concurrent execution of tasks and communication between tasks, as in Foster’s PCAM design methodology (Foster, 1995). Parallel execution requires partitioning of the application’s graph in such a way that communication between the resulting
Resource-Aware Load Balancing of Parallel Applications

subgraphs is minimized and the load is roughly balanced. These subgraphs must be mapped to the processors of a parallel computer. In many cases partitioning is only required once before execution of the application begins. However, in situations where the computational requirement of tasks varies with time, as in adaptive mesh refinement methods, the graph may need to be repartitioned to re-balance the computational load of processors (Schloegel, Karypis and Kumar, 2003; Teresco, Devine, and Flaherty, 2005). Here we use the term “load balancing” to refer to both static partitioning and dynamic repartitioning. This load balancing problem becomes particularly challenging when a homogeneous parallel computer is replaced by a heterogeneous computational grid.

Load balancing using graph partitioning is a well-established research area that has resulted in a number of popular software tools used by computational scientists (Schloegel, Karypis and Kumar, 2003). Several approaches have been proposed for this NP-hard problem. Many employ a multilevel approach (Schloegel, Karypis and Kumar, 2003), which collapses (coarsens) the graph recursively, partitions the smallest graph, and refines the partition as the graph is un-coarsened. One critique of classical graph partitioners is that minimizing the number of edges cut by a partition misrepresents the actual communication volume (Hendrickson, 1998), but in many cases this metric still provides a reasonable estimate of the total communication volume of the parallel application. In reality, however, it is the time (due to computation and communication) spent by the slowest processor that determines the execution time of a parallel application. This becomes particularly important when the computational platform is heterogeneous in processor and network performance, in which case the number of edges cut by a partition is a poor measure of parallel overhead. We call resource-aware load balancing the partitioning and mapping of an application’s graph to a heterogeneous computational platform, either before execution begins or periodically during execution.

Recent work has begun to address the problem of resource-aware load balancing. This includes DRUM, JOSTLE, MinEX, PaGrid, PART, and SCOTCH, among others. Our goal is to review and contrast these recent efforts and discuss the many avenues for future work. A comprehensive review of this area has not been published to date (Li and Lan, 2004; Devine et al., 2005).

BACKGROUND

Since heterogeneous computing is a broad research area and there are many definitions of computational grids, we define the characteristics of the applications and platforms that we are considering.

Applications

We consider tightly-coupled parallel applications that can be described by an undirected graph representing concurrent execution of tasks and communication between tasks. The most important class concerns the numerical solution of PDEs (Schloegel, Karypis and Kumar, 2003), but such applications also include molecular dynamics problems (Koenig and Kalé, 2007) and cellular automata (Cappuccio et al., 2001). Parallel execution requires partitioning of the application’s graph in such a way that communication between the resulting subgraphs is minimized and the load is roughly balanced. These applications are typically iterative, with alternating communication and computation phases (Botadra et al., 2007). Although partial overlap is possible, the entire communication overhead cannot be overlapped with computation because of the dependency between subsequent iterations. Scheduling of operations on dense matrices on heterogeneous systems has been studied (Dongarra and Lastovetsky, 2006), but is not considered here. We are also not concerned here with scheduling of independent tasks or scheduling of workflows represented by directed...
Related Content

Exploring Video Sharing Websites Content with Machine Learning
[www.igi-global.com/article/exploring-video-sharing-websites-content-with-machine-learning/119192?camid=4v1a](www.igi-global.com/article/exploring-video-sharing-websites-content-with-machine-learning/119192?camid=4v1a)

Runtime Service Discovery for Grid Applications
[www.igi-global.com/chapter/runtime-service-discovery-grid-applications/64515?camid=4v1a](www.igi-global.com/chapter/runtime-service-discovery-grid-applications/64515?camid=4v1a)

A Computational Grid Scheduling Model to Maximize Reliability Using Modified GA
Zahid Raza and Deo P. Vidyarthi (2013). *Applications and Developments in Grid, Cloud, and High Performance Computing* (pp. 16-35).
[www.igi-global.com/chapter/computational-grid-scheduling-model-maximize/69025?camid=4v1a](www.igi-global.com/chapter/computational-grid-scheduling-model-maximize/69025?camid=4v1a)

Green Energy Model for Grid Resource Allocation: A Graph Theoretic Approach
[www.igi-global.com/article/green-energy-model-for-grid-resource-allocation/115242?camid=4v1a](www.igi-global.com/article/green-energy-model-for-grid-resource-allocation/115242?camid=4v1a)