Scheduling BoT Applications in Grids Using a Slave Oriented Adaptive Algorithm*

Tiago Ferreto[1], César De Rose[1], and Caio Northfleet[2]

[1] Faculty of Informatics - PUCRS, Brazil
{ferreto, derose}@inf.pucrs.br
[2] HP-Brazil
caio.northfleet@hp.com

Abstract. Efficient scheduling of Bag-of-Tasks (BoT) applications in a computational grid environment reveals several challenges due to its high heterogeneity, dynamic behavior, and space shared utilization. Currently, most of the scheduling algorithms proposed in the literature use a master-oriented algorithm, in which the master is the only responsible for choosing the best task size to send to each slave. We present in this paper a different approach whose main originality is to be slave-oriented, *i.e.* each slave locally determines, from a set of initial runs, which workload size is more adapted to its capacities and notifies the master of it. Finally, we show some measurements comparing our algorithm with other three well-known scheduling algorithms using the SimGrid toolkit.

1 Introduction

Computational grids as a platform to execute parallel applications is a promising research area. The possibility to allocate unprecedent amounts of resources to a parallel application and to make it with lower cost than traditional alternatives (based in parallel supercomputers) is one of the main attractives in grid computing. On the other hand, the grid characteristics, such as high heterogeneity, complexity and wide distribution (traversing multiple administrative domains), create many new technical challenges. In particular, the area of scheduling faces entirely new challenges in grid computing. Traditional schedulers (such as the operating system scheduler) control all resources of interest. In a grid, such a central control is not possible. First, the grid is just too big for a single entity to control. In a grid, a scheduler must strive for its traditional goals, improving system and application performance [1].

Bag-of-Tasks (BoT) applications are parallel master/slave applications whose tasks are independent to each other. A vast amount of work has been done in order to schedule efficiently Bag-of-Tasks applications improving the load balancing in distributed heterogeneous systems. Most of the algorithms focus on the adaptation of the workload during the execution, using either a fixed increment or decrement (*e.g.* based on an arithmetical or geometrical ratio) or a

* This research was done in cooperation with HP-Brazil.

more sophisticated function to adapt the workload. Yet the solutions presented are all based on some evaluation by the master of the slaves' capacities and of the tasks workload. This implies a significant overhead since the master has to maintain some kind of information about its slaves.

We propose in this paper the scheduling of BoT applications in Grids with a different approach whose main originality is to be slave-oriented, *i.e.* each slave locally determines, from a set of initial runs, which workload is more adapted to its capacities and informs the master of it. In turn, the master can compare the workload demanded by the slave to the network penalty paid and make the proper adjustments to adapt the workload. We have thus a workload adaptive algorithm.

2 Related Work

In this section we focus in self-scheduling algorithms [2]. These algorithms divide the total workload based on a specific distribution, providing a natural load balancing to the application during its execution. This class of algorithms is well suited for dynamic and heterogeneous environments, such as grids, and for divisible workload applications.

The Pure Self-scheduling [2] or Work Queue scheduling algorithm divides equally the workload in several chunks. A processor obtains a new chunk whenever it becomes idle. Due to the scheduling overhead and communication latency incurred in each scheduling operation, the overall finishing time may be greater than optimal [3].

The Guided Self-scheduling algorithm [4] (GSS), proposed by Polychronopoulos and Kuck, and Factoring [3], proposed by Flynn and Hummel, are based on a decreasing-size chunking scheme. GSS schedules large chunks initially, implying reduced communication/scheduling overheads in the beginning, but at the last steps too many small chunks are assigned generating more overhead [2]. Factoring was specifically designed to handle iterations with execution-time variance. Iterations are scheduled in batches of equal-sized chunks. The total size of the chunk per batch is a fixed ratio of the remaining workload.

In all algorithms shown above, the amount of workload sent to each slave is defined by the master. We propose in the following section another approach, where the evaluation of the load to be assigned to each slave is done by the slave itself.

3 Local Decision Scheduling Algorithm

The local decision scheduling algorithm (LDS) addresses BoT applications using divisible workloads, *i.e.* all independent tasks demands the same amount of computational resources. The algorithm focus on a heterogeneous, dynamic and shared environment, characterizing a typical computational grid. It is based on a distributed decision mechanism, building in each slave a performance model, which represents the application behavior based on resources utilization. Each