Exploiting Replication and Data Reuse to Efficiently Schedule Data-Intensive Applications on Grids

Elizeu Santos-Neto, Walfredo Cirne, Francisco Brasileiro, and Aliandro Lima

Universidade Federal de Campina Grande
http://www.ourgrid.org
{elizeu,walfredo,fubica,aliandro}@dsc.ufcg.edu.br

Abstract. Data-intensive applications executing over a computational grid demand large data transfers. These are costly operations. Therefore, taking them into account is mandatory to achieve efficient scheduling of data-intensive applications on grids. Further, within a heterogeneous and ever changing environment such as a grid, better schedules are typically attained by heuristics that use dynamic information about the grid and the applications. However, this information is often difficult to be accurately obtained. On the other hand, although there are schedulers that attain good performance without requiring dynamic information, they were not designed to take data transfer into account. This paper presents *Storage Affinity*, a novel scheduling heuristic for *bag-of-tasks* data-intensive applications running on grid environments. Storage Affinity exploits a data reuse pattern, common on many data-intensive applications, that allows it to take data transfer delays into account and reduce the makespan of the application. Further, it uses a replication strategy that yields efficient schedules without relying upon dynamic information that is difficult to obtain. Our results show that Storage Affinity may attain better performance than the state-of-the-art knowledge-dependent schedulers. This is achieved at the expense of consuming more CPU cycles and network bandwidth.

1 Introduction

Each year more data are generated and need to be processed [1]. Currently, there are many scientific and enterprise applications that deal with a huge amount of data [2][3][4]. These applications are called *data-intensive*. In order to process large datasets, these applications typically need a high performance computing infrastructure. Fortunately, since the data splitting procedure is easy and each data element can often be processed independently, a solution based on data parallelism can often be employed.

Task independence is the main characteristic of parallel *Bag-of-Tasks* (BoT) applications [5][6]. A BoT application is composed of tasks that do not need to communicate to proceed with their computation. In this work, we are interested

D. Feitelson, L. Rudolph, and U. Schwiegelshohn (Eds.): JSSPP 2004, LNCS 3277, pp. 210–232, 2005.

in the class of applications which has both BoT and *data-intensive* characteristics. We have named it *processors of huge data* (pHd). Shortly, pHd = *BoT + data-intensive*. There are innumerous important applications that fall in this category. This is the case, for instance, of data mining, image processing, and genomics.

Due to the independence of their tasks, BoT applications are normally suitable to be executed on grids [7][5]. However, since resources in the grid are connected by wide area network links (wan), the bandwidth limitation is an issue that must be considered when running pHd applications on such environments. This is particularly relevant for those pHd applications that present a data reutilization pattern. For these applications, the data reuse pattern can be exploited to achieve better performance. Data reutilization can be either among tasks of a particular application or among a succession of applications executions. For instance, in the visualization process of quantum optics simulations results [4] it is common to perform a sequence of executions of the same parallel visualization application, simply sweeping some arguments (e.g. zoom, view angle) and preserving a huge portion of the data input from the previous executions.

There exists some algorithms that are able to take data transfer into account when scheduling pHd applications on grid environments [8][9][10][11]. However, they require knowledge that is not trivial to be accurately obtained in practice, especially on a widely dispersed environment such as a computational grid [7]. For example, XSufferage [9] uses information about the CPU and network loads, as well as the execution time of each task on each machine, all of which must be known a priori, to perform the scheduling.

On the other hand, for *CPU-intensive* BoT applications, there are schedulers that do not use dynamic information, yet achieve good performance (e.g. Workqueue with Replication - WQR [12][13]). They use replication to tolerate inefficient scheduling decisions taken due to the lack of accurate information about both the environment and the application. However, these schedulers were conceived to target CPU-intensive applications and thus data transfers are not taken into account by them.

In this paper we introduce *Storage Affinity*, a new heuristic for scheduling pHd applications on grids. Storage Affinity takes into account the fact that input data is frequently reused either by multiple tasks of a pHd application or by successive executions of the application. It tracks the location of data to produce schedules that avoid, as much as possible, large data transfers. Further, it reduces the effect of inefficient task-processor assignments via the judicious use of task replication.

The rest of the paper is organized in the following way. In the next section we present the system model that is considered in this work. In Section 3, we present the Storage Affinity heuristic as well as other heuristics used for comparative purposes. In Section 4, we evaluate the performance of the discussed schedulers. Section 5 concludes the paper with our final remarks and a brief discussion on future perspectives.