Available online at www.sciencedirect.com

ScienceDirect

Future Generation Computer Systems 23 (2007) 92–99

FGCS
FUTURE GENERATION COMPUTER SYSTEMS

www.elsevier.com/locate/fgcs

Mining performance data for metascheduling decision support in the Grid

Hui Li[a,*], David Groep[b], Lex Wolters[a]

[a] Leiden Institute of Advanced Computer Science (LIACS), Leiden University, PO Box 9512, 2333 CA, Leiden, The Netherlands
[b] National Institute for Nuclear and High Energy Physics (NIKHEF), PO Box 41882, 1009 DB, Amsterdam, The Netherlands

Available online 11 July 2006

Abstract

Metaschedulers in the Grid need dynamic information to support their scheduling decisions. Job response time on computing resources, for instance, is such a performance metric. In this paper, we propose an Instance Based Learning technique to predict response times by mining historical performance data. The novelty of our approach is to introduce policy attributes in representing and comparing resource states, which are defined as the pools of running and queued jobs on the resources at the time of making predictions. The policy attributes reflect the local scheduling policies and they can be automatically discovered using genetic search. An extensive empirical evaluation is conducted to validate our technique using real workload traces, which are collected from the NIKHEF production cluster on the LHC Computing Grid and Blue Horizon in the San Diego Supercomputer Center (SDSC). The experimental results show that acceptable prediction accuracy can be achieved, where the normalized average prediction errors for response times are ranging from 0.57 to 0.79.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Response time predictions; Instance based learning; Metascheduling; Grid

1. Introduction

Large scale Grids typically consist of many heterogeneous and geographically distributed resources. As an example, the LHC Computing Grid (LCG) currently has approximately 140 sites in 34 countries with a total number of 12,516 CPUs and 5 petabytes' storage. Metascheduling in such an environment raises a serious challenge and many scheduling algorithms, architectures and systems have been proposed [1,3,14]. Scheduling at the meta level differs from local scheduling in that metaschedulers do not have control over the resources. Instead, metaschedulers make decisions on behalf of users and hand jobs over to the local resource management systems. There is one common aspect in this process despite the diversity of Grid scheduling instances, namely, the goodness of scheduling decisions depends heavily on the quality of information available about the resources. There is relatively static information such as machine types, number of CPUs and storage capacity. This can be obtained by monitoring tools via Grid information services. There is also more dynamic information such as job response times. This type of information is very important to support the metascheduling decisions, but is not available by only monitoring. It must be predicted based on historical data recorded on the resources.

The main theme of this paper is about job response time predictions on computing resources, and our approach is based on mining the historical performance data. We believe that knowledge about local scheduling policies can be discovered in the data and this knowledge can be utilized in predictions. Techniques from statistical data mining can help us in getting there. Specifically, we investigate how an Instance Based Learning technique is applied in predictions, how a genetic algorithm is used for parameter optimization, and elaborate our design choices. We focus on resources such as space-shared parallel supercomputers, clusters, and study workload traces recorded on them.

The rest of the paper is organized as follows: Section 2 introduces job response time predictions and discusses the related work in this area. Section 3 defines job similarity and resource state similarity, which are the two key concepts in our technique. Section 4 elaborates the IBL-based prediction algorithm, including the distance function and the induction

* Corresponding author. Tel.: +31 71 527 7094; fax: +31 71 527 6985.
E-mail addresses: hli@liacs.nl (H. Li), davidg@nikhef.nl (D. Groep), llexx@liacs.nl (L. Wolters).

0167-739X/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.future.2006.04.009

Table 1
Representative job attributes recorded in workload traces

Abbreviation	Job attribute	Type	Abbreviation	Job attribute	Type	Abbreviation	Job attribute	Type
g	Group name	Nominal	n	#CPUs	Numeric	m	Used memory	Numeric
u	User name	Nominal	r	Req. run time	Numeric	rm	Req. memory	Numeric
q	Queue name	Nominal	tod	Arrival time	Numeric	rt	Run time	Numeric
e	Job name	Nominal	s	Exit status	Numeric	qt	Queue wait time	Numeric

models. Section 5 describes the design and construction of the genetic algorithm for parameter optimization. Section 6 presents the empirical evaluation and analysis using real workload traces. Conclusions and future work are discussed in Section 7.

2. Response time predictions

Response time of a job is defined as the time elapsed from its submission till completion on a resource. Two metrics need to be estimated for the response time: one is how long a job executes on the resource (application run time), the other is how long the job waits in the queue before starting (queue wait time). A popular approach is to derive predictions from similar observations in the past, and the historical data available in site workload traces naturally serves as the basis for such a study.

Techniques have been proposed for predicting application run times using historical information. In [11] "templates of attributes" are defined for categorizing historical jobs and statistical techniques like mean or linear regression are applied to generate predictions. In [6] Instance Based Learning (IBL) techniques are investigated for run time predictions. IBL uses historical data "near" the query point to build a local model for approximation. A proper distance metric has to be defined to measure the distances between data instances. In fact the IBL algorithm is a generalization of the template approach, in which distances are simplified to binary values (belong or not belong to a specific category).

For queue wait times, the basic idea of most techniques is based on scheduler simulation. In [13] scheduling algorithms like FCFS, LWF, and backfilling are simulated for predicting queue wait times, where application run times are estimated using the template approach. In [7] simulation is also used to predict queue wait times for a policy-based scheduler called Maui. Although relatively better prediction accuracy can be achieved, several major drawbacks remain for the simulation approach. Firstly, it cannot meet the real-time requirements in the Grid brokering process. Secondly, it is not scalable since there are different types of local scheduling systems deployed on different Grid sites. Some sites have schedulers with combinations of basic scheduling algorithms, and most sites enforce different kinds of policies in their own fashion.

As an alternative, we are trying to derive queue wait times also from historical observations. Our assumption is that "similar" jobs under "similar" resource states would most likely have similar waiting times, given that the scheduling algorithm and policies remain unchanged for a reasonable amount of time. Similar ideas have been studied in [12], in which summary statistics about the resource state (e.g. free CPUs, number of running jobs) is used as attributes for defining templates. The template approach to estimate application run times as described above can be applied similarly for waiting times. Our work distinguishes from it in two aspects: firstly, we introduce attributes that reflect scheduling policies to represent resource states in a more fine-grained level for similarity comparison, and these policies can be automatically discovered via genetic search. Secondly, we use Instance Based Learning as the common framework for both application run times and queue wait times. We elaborate our approach in the following sections.

3. Similarity definition

The key problem is how to define similarity to compare jobs. Table 1 shows the representative job attributes recorded in workload traces. For job run times, some of these attributes can be naturally used for similarity definition. For queue wait times, however, new attributes need to be defined as the waiting time of a job is typically a result of interactions among the job, other jobs on the resource, and the local scheduler. We introduce the definitions for job similarity and resource state similarity as follows.

3.1. Job similarity

Seven recorded attributes are considered to define job similarity. They are "group name" (g), "user name" (u), "queue name" (q), "job name" (e), "number of CPUs" (n), "requested run times" (r), and "arrival time of day" (tod). Depending on the availability, any potentially useful attribute such as node speed and executable arguments can be added to this list. The pre-selected attributes are mostly self-explanatory by their names and they have two main types, namely, nominal (g, u, q, e) or numeric (n, r, tod). In the Instance Based Learning algorithm described later, similarity between jobs is formulated by a distance function composed of attributes. Jobs with smaller distances are considered more similar. For instance, jobs from the same group, say "bioinfo", and with the same executable name called "proteinmatch" will have smaller distances than those who have nothing in common, and therefore have a higher chance of being used for predictions of the same kind. To make a good distance function, we employ a genetic search to weight attributes according to the metric being predicted (application run time or queue wait time).

3.2. Resource state similarity

We define a *resource state* as the pool of running and queued jobs on the resource at the time of making a prediction. More