Challenges in Designing Grid Marketplaces

Kartik Hosanagar

June 2007

WSRC Paper 2007/6-1

Wharton-SMU Research Centre

ANY OPINIONS EXPRESSED ARE THOSE OF THE AUTHOR(S) AND NOT NECESSARILY THOSE OF THE OFFICE OF RESEARCH, SINGAPORE MANAGEMENT UNIVERSITY
Challenges in Designing Grid Marketplaces

Kartik Hosanagar

This project is funded by the Wharton-SMU Research Centre, Singapore Management University
Challenges in Designing Grid Marketplaces

Abstract

Recent years have seen several proposals for the use of economic mechanisms to allocate grid resources with a special emphasis on scheduling. While this is an important operational issue, the focus of this literature has assumed that capacity (of CPU, bandwidth, storage and software licenses) on the grid exists and that resource requirements for jobs being submitted can be estimated quite accurately a priori. For example, a number of scheduling techniques have been proposed in order to allocate resources to incoming jobs in a grid. All these techniques rely on ex ante estimates of resource requirements in order to assign jobs to specific resources. Computing such estimates is inherently a hard task but is critical to driving the efficiency of these scheduling algorithms. In this paper, we present two alternative models for predicting the running time of jobs submitted to a grid. We apply these models to a large dataset from a real-world grid. We find that our models do well in predicting resource requirements and can serve as the underlying prediction engine for schedulers in grids.

1 This paper is funded by a research grant at Singapore Management University (grant number: C216/MP06W003).
1 Introduction

A grid is a type of parallel and distributed system that enables the sharing, selection, and aggregation of geographically distributed "autonomous" resources dynamically at runtime depending on their availability, capability, performance, cost, and user preferences.\(^2\) A well known example of a grid computing project is UC Berkeley’s SETI@Home screensaver which allowed Berkeley’s SETI project to tap into spare computing power available on millions of PCs that had the screensaver installed. Similarly, Intel’s Grid project allows the firm to process engineering simulation jobs by using spare computing power available in Intel’s various offices. Intel estimates that its grid has saved the firm $500M over 10 years and increased computer utilization from 35% to 80% over the same period.\(^3\) Grid computing offers great benefits in solving computationally complex problems related to financial modeling, weather prediction and molecular modeling for biotech applications. As a result, there has been considerable focus recently on the design of grid marketplaces. Several firms including Sun and IBM have expended considerable resources in trying to establish themselves as primary resource providers in grid markets. Initiatives by these large firms are making grid marketplaces a reality.

A key driver of efficiency in grids is the ability to schedule jobs. A number of scheduling algorithms have been proposed in the literature (see for e.g., Kumar et al 2005). Almost all these scheduling approaches rely on using an estimate of the resource requirements in order to schedule a job. Such estimates are either provided by users or computed by the scheduler. However, as we note below, several studies have shown that users struggle to provide good estimates even when they have the incentives to do so. Furthermore, as we will demonstrate,

\(^2\) Source: Gridcomputing.com
\(^3\) Source: http://news.zdnet.com/2100-9595_22-523296.html
resource consumption by jobs follows a heavy-tailed distribution. Thus, it is non-trivial for the scheduler to estimate the resource requirements.

Poor estimates of resource requirements can significantly undermine the efficiency of these scheduling algorithms. Further, since job submitters estimate their costs based on these estimates and known pricing policies, there can be considerable ex-post regret when costs exceed expectations. Solutions are needed to address inefficiencies caused by poor estimates of resource requirements so that buyers can better estimate costs, schedulers can better assign jobs to resources and resource providers can better plan their capacity decisions. There has been very little work on addressing these issues. In this paper, we propose an empirical model to predict resource requirements in a grid. The model is calibrated to data from a real-world setting and provides reasonably high accuracy. The use of decision support models to estimate resource requirements can improve user experience, increase efficiency of scheduling algorithms and simplify capacity sizing decisions for resource providers.

The rest of the paper is organized as follows. In Section 2, we review the related literature on scheduling in grids. In Section 3, we describe our data and present an empirical model to predict the running time of a job. We find that the empirical models are quite accurate. We discuss the implications of these findings and propose extensions in Section 4.

2 Literature Review

A primary area of focus in grid economics is related to resource allocation. This stream of work studies the use of market mechanisms to allocate computing cycles to incoming jobs. For example, Feldman et al (2005) formulate a resource allocation game and study the efficiency and fairness of the Nash equilibria that results. Wellman et al (2001) propose an auction mechanism to allocate distributed resources to users. Other work on market-based resource allocation

3 Models to Estimate Running Time of a Job

In this section, we explore a few different models for estimating resource consumption by jobs. The model parameters are calibrated to real-world data from Intel’s grid. This grid system is heavily used for design and verification tasks. It consists of about 60,000 work stations with about 120,000 CPUs and serves thousands of engineers worldwide.

3.1 Data Description

The data used in this study consists of jobs submitted during four days in May 01-05, 2006. Users submit tasks such as engineering design simulations to the grid. These tasks may be composed of several jobs. Each job is assigned to a workstation (resource provider) by the scheduler. There are 53 unique users, 101 unique tasks, 130,822 jobs and 962 unique workstations in our dataset. Each observation in our dataset corresponds to a job. We ignore jobs that had a processing time of under 0.1 seconds (grid error) and jobs which terminated with an exit code that indicates a failure. The resulting data consists of 118,978 unique observations. We know the user ID, task ID, workstation ID and workstation configuration (CPU, physical memory, swap memory, Operating system) for every job. In addition, we also know the running time of each job along with the maximum resident set size (memory) used by the job.

3.2 Empirical Model

In Figure 1, we plot the distribution of Rtime, the running time of jobs in our dataset. The mean running time for jobs is 2293 seconds and the standard deviation is 8577 seconds. The maximum Rtime in the dataset is 316,156 seconds. The variance is large and the distribution
seems heavy tailed. Thus, it is extremely important to be able to distinguish jobs from a scheduling perspective. It matters a lot whether the job takes 500 seconds or 30,000 seconds. In table 1, we present the mean and standard deviation of Rtime by task and by user for the top 10 tasks and users respectively. It is worth noting that the variance at the user or task level is considerably lower. Thus, it is worth accounting for the task and user information in predicting Rtime.

![Distribution of running time of jobs](image)

**Figure 1: Distribution of running time of jobs**

<table>
<thead>
<tr>
<th>Task ID</th>
<th>Mean</th>
<th>StdDev</th>
<th>User ID</th>
<th>Mean</th>
<th>StdDev</th>
</tr>
</thead>
<tbody>
<tr>
<td>11149</td>
<td>267.87</td>
<td>79.53</td>
<td>11288</td>
<td>9046.86</td>
<td>3813.22</td>
</tr>
<tr>
<td>8228</td>
<td>575.32</td>
<td>234.58</td>
<td>29548</td>
<td>362.08</td>
<td>444.49</td>
</tr>
<tr>
<td>11085</td>
<td>447.32</td>
<td>153.62</td>
<td>4675</td>
<td>8822.11</td>
<td>4251.56</td>
</tr>
<tr>
<td>11124</td>
<td>323.45</td>
<td>94.08</td>
<td>1045</td>
<td>37871.88</td>
<td>23011.18</td>
</tr>
<tr>
<td>11087</td>
<td>400.89</td>
<td>133.79</td>
<td>5860</td>
<td>686.31</td>
<td>2657.47</td>
</tr>
<tr>
<td>11128</td>
<td>286.37</td>
<td>64.32</td>
<td>1651</td>
<td>13226.12</td>
<td>19633.32</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>1117</td>
<td>336.55</td>
<td>96.68</td>
<td>2660</td>
<td>7311.00</td>
<td>2011.56</td>
</tr>
<tr>
<td>11154</td>
<td>323.21</td>
<td>94.02</td>
<td>2726</td>
<td>15899.88</td>
<td>14839.00</td>
</tr>
<tr>
<td>2506</td>
<td>883.77</td>
<td>261.68</td>
<td>21151</td>
<td>637.82</td>
<td>328.74</td>
</tr>
<tr>
<td>1002</td>
<td>599.63</td>
<td>477.48</td>
<td>1019</td>
<td>7055.05</td>
<td>10711.79</td>
</tr>
</tbody>
</table>

Table 1: Mean and Standard Deviation of Rtime by task and by user

We begin with an analysis of Rtime by user-task-workstation group. Then, we reduce the dimensionality of the model by building a model at the user-task level.

3.2.1 Grouping by User-Task-Workstation

In this section, we segment the data into groups based on unique combinations of user-task-workstation IDs. There are 11852 user-task-workstation groups in our dataset. We only consider the 647 groups with at least 50 observations. In Figure 2, we plot Rtime and maximum resident set size (MaxRSS) for one of the groups. MaxRSS is the largest amount of physical memory that the job used during its run. Based on similar plots for other groups, we hypothesize the following quadratic relationship between Rtime and MaxRSS:

\[ Rtime = const + aMaxRSS + bMaxRSS^2 \]  

(1)
The parameter estimates and a plot of the predicted Rtime for the group in Figure 2 are presented in Table 2 and Figure 3 respectively. The model fits the data quite well. The distribution of $R^2$ for the 647 models is in Figure 4. More than 40% of the groups have an $R^2 > 0.8$; 70% have $R^2 > 0.5$ and 90% have $R^2 > 0.2$. Thus, these models go a long way in predicting the runtime of each job.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
</tr>
</thead>
<tbody>
<tr>
<td>Const</td>
<td>69060.14</td>
</tr>
<tr>
<td>MaxRSS</td>
<td>-343.83</td>
</tr>
<tr>
<td>MaxRSS$^2$</td>
<td>0.43</td>
</tr>
</tbody>
</table>

Table 2 Estimated regression coefficients for one utw group

It is worth noting that the models above predict Rtime based on MaxRSS which is itself a variable observed after the job is completed. Thus, they do not necessarily allow us to directly predict the running time of the job before the job completes execution. However, it is nonetheless useful to model the relationship between Rtime and MaxRSS for at least two reasons. First, since the distribution of MaxRSS is known for each group, the above models will yield a probability distribution of Rtime that has far lower variance than the unconditional distribution in Figure 1. Predicting Rtime within a +/-10% range may be sufficient for most scheduling algorithms. More importantly, the scheduler can measure the maximum memory used early on in job execution to predict Rtime while the job is being processed. Such online predictions can be
used to reassign jobs to a new machine or to detect runaway jobs early in their execution. We discuss this issue further in Section 4.

Information on the user, task and the selected workstation can be very useful in reducing much of the variance in Rtime. However, there are two major drawbacks of using the above approach of grouping by user-task-workstation. First, this creates a large number of groups, with several groups having limited observations. For example, among the 11852 user-task-workstation groups, only 647 had more than 50 jobs. Thus, it is hard to build predictive models for every group. Secondly, if jobs from a particular user or task have not previously been sent to a workstation, there is no way of estimating the running time. Similarly, if a new resource provider enters the market, it is again hard to estimate the running time if the job were forwarded to this provider. Thus, we reduce the dimensionality in the model by grouping jobs based on user and task IDs alone. Further, we introduce workstation characteristics as regressors. This allows us to predict the running time for a job irrespective of the choice of workstation.

![Figure 4: Distribution of $R^2$ for the 647 user-task-workstation groups](image1)

![Figure 5: Distribution of $R^2$ for the 42 user-task groups](image2)

3.2.2 Grouping by User-Task
There are 156 user-task groups in our dataset (note the significant reduction in number of groups). We will account for workstation characteristics as regressors in our model as follows:

\[
\ln \text{Rtime} = \alpha_0 + \alpha_1 \text{MaxRSS} + \alpha_2 (\text{MaxRSS})^2 + \alpha_3 \text{osver} + \alpha_4 \ln(1 + \text{CPUspeed}) + \\
\alpha_4 \ln(1 + \text{CPU2speed}) + \alpha_4 \text{memory} + \alpha_4 \text{memory}^2 + \alpha_8 \ln \text{swap}
\]  

(2)

In the model above, \(\text{osver}\) is the version of the Operating system in the workstation (binary variable labeled 0 or 1 based on the flavor of Linux used), \(\text{CPUspeed}\) is the clock frequency of the CPU (in MHz), \(\text{CPU2speed}\) is the clock frequency of the 2\(^{nd}\) CPU for those machines that have two CPUs (no workstation had more than 2 CPUs), \(\text{memory}\) is the amount of physical memory (Mbytes) and \(\text{swap}\) is the amount swap memory (Mbytes) in the workstation. The above model assumes that the impact of each of the parameters on \(\text{Rtime}\) is multiplicative.

The regression result for one of the user-task groups is in Table 3. In Figure 5, we present the distribution of \(R^2\) across 42 groups that have at least 50 observations and demonstrate sufficient heterogeneity in workstation parameters (such as CPUspeed, memory, etc) to allow estimation of corresponding coefficients. 34\% of the groups have an \(R^2 > 0.8\); 50\% have \(R^2 > 0.5\) and 83\% have \(R^2 > 0.2\). Thus, equation 2 also demonstrates good predictive power, although somewhat lower than the model in Section 3.2.1. However, as pointed out earlier, accounting for workstation characteristics as regressors in the model and grouping jobs by user and task IDs, but not workstation ID, provides considerable flexibility.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Coefficient</th>
<th>Std. Error</th>
<th>t-Statistic</th>
<th>Prob.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Const</td>
<td>-120.59</td>
<td>7.48</td>
<td>-16.13</td>
<td>0.000</td>
</tr>
<tr>
<td>MAXRSS</td>
<td>3.16</td>
<td>0.16</td>
<td>19.19</td>
<td>0.000</td>
</tr>
<tr>
<td>MAXRSS_SQ</td>
<td>-0.02</td>
<td>0.00</td>
<td>-18.75</td>
<td>0.000</td>
</tr>
<tr>
<td></td>
<td>OS_Version</td>
<td>Log(CPUSPEED)</td>
<td>Log(CPU2SPEED)</td>
<td>MEMORY</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>---------------</td>
<td>----------------</td>
<td>----------</td>
</tr>
<tr>
<td>1st user</td>
<td>20.53</td>
<td>-1.42</td>
<td>-1.03</td>
<td>-25.39</td>
</tr>
<tr>
<td>2nd user</td>
<td>6.20</td>
<td>0.05</td>
<td>0.06</td>
<td>7.45</td>
</tr>
<tr>
<td>3rd user</td>
<td>3.31</td>
<td>-28.52</td>
<td>-17.05</td>
<td>-3.41</td>
</tr>
<tr>
<td>4th user</td>
<td>0.001</td>
<td>0.000</td>
<td>0.000</td>
<td>0.001</td>
</tr>
</tbody>
</table>

Table 3: Regression coefficients for one user-task group

4 Conclusions

The efficiency of grids from a scheduling and capacity planning standpoint rely on accurate estimation of resource requirements for submitted jobs. Furthermore, user experience can be considerably enhanced by providing estimates on processing times and estimated cost. We present a model to estimate running times of jobs and find that several parameters such as user ID, task ID and workstation characteristics can be used to provide reasonable bounds on the running time of jobs. We showed that in the typical case good estimates of the running time of a job can be produced regardless of the fact that in the worst case such estimates are hard to obtain. The study opens a number of interesting directions worth pursuing as we extend this work. We have hinted at an online prediction algorithm above. Given that MaxRSS, the maximum memory used by the job during its run, is the strongest predictor of the job’s running time, it may be possible to monitor parameters available during job execution to predict the running time. These parameters include current MaxRSS, I/O time used so far, etc. The trivial application of this is to give the user a better estimate of running time as the job is being processed. A more interesting application would be to use these estimates to detect jobs that will not be able to complete
execution on time and reassigning them to other workstations. This is especially relevant for systems which allow migration of jobs, for example Mosix.\(^4\)

Our work on predicting run time and other resource requirements have applications beyond the resource allocation context used in the paper. As grids truly become marketplaces, data and decision support tools such as that described in the paper will become relevant for problems such as capacity planning and optimal pricing as well.

References


\(^4\) Source: www.mosix.org
