

Generating Grid Resource Requirement Specifications

Richard Huang
Computer Science & Engineering
Department
University of California, San Diego
ryhuang@cs.ucsd.edu

Henri Casanova
Information and Computer Sciences
Department
University of Hawai'i at Manoa
henric@hawaii.edu

Andrew A. Chien
Computer Science & Engineering
Department
University of California, San Diego
achien@cs.ucsd.edu

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems – *distributed applications*.

General Terms

Performance.

Keywords

Resource requirement specification, scientific workflows.

1. INTRODUCTION

There is a clear trend of increased deployment of cluster platforms, due to decreasing hardware costs, increasing choices in cluster vendors, and increasing availability of open source cluster management tools. At the same time there is a trend of improved networking technology, with bandwidths reaching tens of Gigabits per second. This increasing availability of computing resources and increased bandwidths enable scientists to execute loosely synchronous applications such as scientific workflows, which this work targets, at a large scale.

Executing scientific workflows across wide-area resource deployments is only possible with a *middleware* infrastructure for connecting and executing application components. The middleware infrastructure needs to address challenges such as resource discovery, resource selection, resource monitoring, resource binding (including negotiation with various resource managers), and security issues. The Globus Alliance was formed to consolidate knowledge and software; the resulting software repository is the Globus Toolkit [1].

One of the key challenges of executing applications over large numbers of resources is selecting an appropriate resource subset. Many researchers have explored the resource selection question. Deployed and proposed resource selection systems encompass bilateral or multi-matching process such as matchmaking in Condor [5] or systems such as RedLine [4] where resources are selected via a constraint-solving process. In the Virtual Grid Execution System [3], a relational database is employed to organize the resources for faster querying. All of these systems provide a resource requirement specification language to express some characteristics of the needed resource collection (RC).

The problem we are addressing in this work is: How does one generate the best resource specification? Most users do not know what resource requirement specification to provide to the resource

selection service in order to obtain a resource set that optimizes application performance while staying within some resource or cost budget. The characteristics of a resource collection as well as the scheduling heuristic employed can affect the application performance and the resource cost. While the benefits of using an appropriate resource collection are clear [2], it is not clear how to compose the best resource specification. To the best of our knowledge, this problem has not been studied in the literature. In practice users typically resort to naïve approaches such as specifying a desired number of resources that matches the parallelism in their workflows.

2. EMPIRICAL PREDICTION MODEL

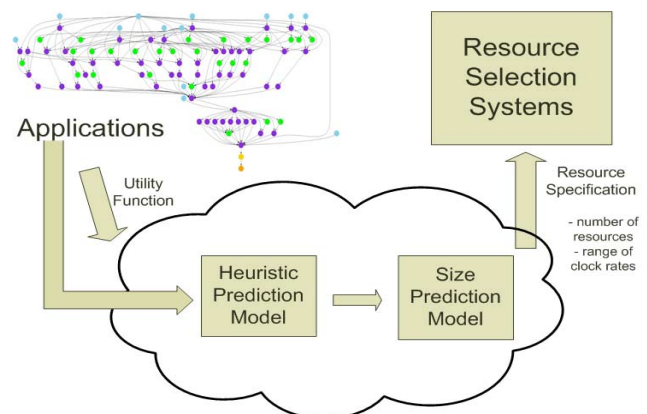


Figure 1. Resource Specification Prediction Model

Our vision for a solution to the problem of generating the best resource specification is depicted in Figure 1. The resource specification prediction model is composed of two parts: the *heuristic prediction model* predicts the best scheduling heuristic for an input DAG and an optional utility function; the *size prediction model* predicts that best RC size given the DAG, the optional utility function and the scheduling heuristic. In this paper, we focus solely on size prediction, which is accomplished via an empirical prediction model.

2.1 Relevant DAG Characteristics

The first step in constructing our empirical size prediction model is to determine the relevant DAG characteristics that affect the choice of the RC size. Through simple experiments, we found that the following DAG characteristics are instrumental in constructing our empirical model: DAG size (number of tasks in the DAG), communication-to-computation ratio (CCR), parallelism (normalized DAG width), and regularity (variance among number of tasks per level for different DAG levels). The DAG height and the average number of tasks per level are subsumed by the above

four DAG characteristics. Impacts of other DAG characteristics are not significant and thus were not used in the model construction.

2.2 Best RC Size

For illustrative purposes, Figure 2 shows the turn-around time (scheduling time + application makespan) as a function of RC sizes for a DAG with size of 1000, CCR of 0.01 and parallelism of 0.7 (results obtained in simulation over a homogeneous collection of processors). Each line represents different regularity values. We define the best RC size as the RC size where the application turn-around time is minimized for any DAG. Notice that beyond a threshold adding more resources to the RC does not improve the turn-around time and may instead degrade performance (due to added scheduling time for a bigger RC). The goal of the size prediction model is to predict the best RC size, or the *knee*, for any given DAG.

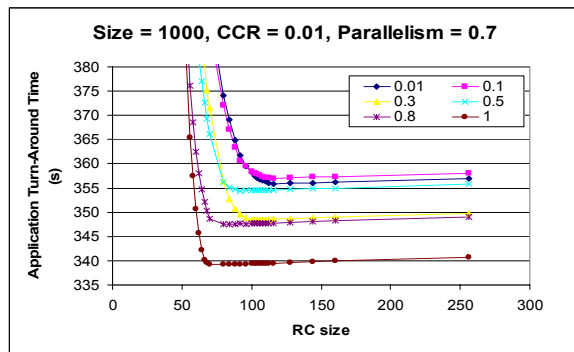


Figure 2. Determining best RC size

2.3 Sample Set of DAGs for Observation

Our strategy in formulating a size prediction model is to generate an observation set of DAG configurations with a range of the relevant DAG characteristics. We use the following: from 100 to 10,000 for DAG size; 0.01 to 1.0 for CCR; 0.3 to 0.9 for parallelism (0 is no parallelism and 1 is fully parallel tasks); and 0.01 to 1.0 for regularity. For each DAG configuration, we instantiate ten random DAGs. For each instance, we simulate application execution using a reference scheduling heuristic on RCs of increasing size and compute the compute time. From this data, we empirically determine the best RC size for all (CCR, size, parallelism, regularity) tuples.

2.4 Model Formulation

For better tractability, first we consider only parallelism and regularity for each (size, CCR) pair. From our data, we noticed approximate doubling of best RC size as the parallelism is increased and slight decrease in best RC size as regularity is increased. We hypothesize that the knee value for each (size, CCR) pair can be modeled as: $Knee = 2^{(a\alpha + b\beta + c)}$, where α is parallelism and β is regularity. Thus for each (size, CCR) pair, we need to solve for a , b , and c . When we plot the logarithm of the knee values versus parallelism and regularity for each (size, CCR) pair, we noticed very similar planar shapes, thereby suggesting that we can use a planar fit to solve for a , b , and c .

The last remaining step is to account for varying DAG sizes and CCR. We resort to empirical approximation based on linear interpolation between data points in the observation set.

2.5 Model Validation

We use two workloads to validate our model: randomly generated DAGs and DAGs from real applications. We validate our prediction model by comparing the performance achieved by using the predicted RC size and the optimal application performance (obtained by brute force). We find that when using our model application performance is only within a few percent of optimal at often much reduces resource cost. Further, we find that the use of our model provides a large improvement over the current (naïve) practice of using the maxim parallelism as the RC size.

3. CONCLUSION

In this work we have attempted to provide part of the missing link between application users and resource selection systems. We construct an empirical prediction model to generate resource specifications based on application characteristics and optional utility function trading off performance and cost. Our initial validation results were conclusive and the use of our model leads to good performance at low resource cost. As enhancements to our model, we plan to explore generating alternative resource specification when the resource selection system cannot return the desired resource collection. The next big step in this research is the development of a heuristic prediction model.

4. ACKNOWLEDGMENTS

The authors and research described here are supported in part by the National Science Foundation under awards NSF Cooperative Agreement ANI-0225642 (OptIPuter), NSF CCR-0331645 (VGrADS), NSF ACI-0305390, and NSF Research Infrastructure Grant EIA-0303622.

5. REFERENCES

- [1] Foster, I. and Kesselman, C, Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 1997. 11(2): P.115-128.
- [2] Huang, R., Casanova, H., and Chien, A.A, Using Virtual Grids to simplify Application Scheduling. In *Proceedings of the IEEE International Parallel and Distributed Processing Symposium (IPDPS 2006)*
- [3] Kee, Y.-S., et al., Efficient Resource Description and High Quality Selection for Virtual Grids. In *Proceedings of the IEEE International Symposium on Cluster Computing and the Grid (CCGrid 2005)*.
- [4] Liu C. and Foster I., A Constraint Language Approach to Grid Resource Selection. Technical Report (TR-2003-07), University of Chicago Department of Computer Science, 2003.
- [5] Raman, R., Livny, M., and Solomon, M., Matchmaking: Distributed Resource Management for High Throughput Computing. In *Proceedings of the IEEE Symposium on High Performance Distributed Computing (HPDC 1998)*.