Performance Modeling Based Scheduling and Rescheduling of Parallel Applications on Computational Grids

H.A. Sanjay Sathish Vadhiyar
Supercomputer Education and Research Centre
Indian Institute of Science
Bangalore - 560012, India
{sanjay@rishi.,vss@}serc.iisc.ernet.in

Abstract

Computational Grids consist of both dedicated and non-dedicated clusters. For effective mapping of parallel applications on grid resources, a grid metascheduler has to evaluate different sets of resources from different clusters. In order to aid the metascheduler to evaluate a set of resources for the execution of a parallel application, we have developed performance modeling strategies to predict execution times of the application on the set of resources. Our strategies adapt to changing network and CPU loads on the grid resources. Our strategies give less than 30% average percentage prediction errors in all cases, which, to our knowledge, is the best reported for non-dedicated environments. We have also developed scheduling strategies that use these predicted execution times in order to determine best schedules for a parallel application. For applications consisting of multiple phases, we plan to use phase detection techniques to automatically divide the application into multiple phases, and to use our performance modeling strategies to derive per-phase predicted execution times. These per-phase predicted times will be used by a rescheduler framework to migrate an executing application to a different set of resources as the application reaches a new phase.

1. Introduction

Many computational grid frameworks are composed of multiple distributed sites. Each site has one or more clusters or Massively Parallel Processors (MPPs) with each cluster consisting of a set of homogeneous machines. Some of these clusters are dedicated batch systems while others are time-shared non-dedicated systems. Jobs submitted to a grid are handled by a matascheduler which interacts with the local schedulers of the clusters for scheduling jobs to the individual clusters.

Grids have been found to be powerful research-beds for execution of various kinds of parallel applications. Tightly-coupled applications exhibit poor performance when executed across multiple clusters due to low-speed network links between the clusters. Hence tightly-coupled applications are typically executed within a single cluster. When a tightly-coupled parallel application is submitted to a grid, the metascheduler has to choose a set of resources from a cluster for application execution. The metascheduler, with the help of local schedulers, has to evaluate different candidate resource sets, with each candidate set consisting of resources from a cluster, and select the most suitable resources for application execution. The candidate resource sets are mostly evaluated in terms of predicted execution times of the application on the candidate resources and the resource set with the minimum predicted execution time is chosen for application execution. Thus, models that predict execution times of the parallel applications on a set of resources and a search procedure (scheduling strategy) which selects the best set of machines within a cluster for application execution are of importance for enabling tightly coupled applications on grids.

In this work, we have developed a comprehensive set of performance modeling strategies [13, 17] to predict execution times of tightly-coupled parallel applications on a set of resources in a dedicated or non-dedicated cluster with the purpose of aiding grid metascheduler in making scheduling decisions. We have also implemented a set of scheduling strategies to select the best set of resources for application execution. We plan to extend this work for large scientific applications where the computation and communication complexities can vary in different phases of application execution and where the amount of computations and communications within a phase can be non-uniform among different processors. We describe the plans for performance modeling multi-phase applications and rescheduling the applications to different sets of resources during phase transitions.

2. Related Work

Most of the existing modeling strategies assume uniform loading conditions on the systems when the experiments for modeling are conducted and use the models to predict execution times for large problem sizes and/or larger number of processors for the same loading conditions [1, 3, 10, 11, 15]. This assumption is unrealistic in non-dedicated environments. Some modeling methods also require analytical models expressing the computation and communication characteristics of the applications in order to predict the execution times of the applications [1, 11, 14, 15]. Building robust analytical models require detailed knowledge of the applications and such knowledge is available only with the application developers. Some of the existing efforts for non-dedicated environments can deal with different loading conditions during training the models and predictions, but require the loads to be constant during an application execution [2, 3]. The work by Schopf and Berman [14] predicts execution times of the applications when the external loads can change during the application executions, but require detailed analytical models. Our modeling strategies work for non-dedicated environments where the loads on the machines can vary during the execution of the application. Our models work with the executable binaries for the applications. Except for the minimum problem size, our models do not need any other knowledge about the applications.

Most of the existing search procedures will address scheduling of multiple jobs, and on dedicated environment. But our goal is to schedule single job on a non-dedicated multi-cluster environment. The work by Nudd et. al. [8] considers scheduling on multi-cluster environment, but their technique addresses scheduling of multiple jobs on uniform loading conditions. There are few efforts that schedule tightly coupled parallel applications using performance models [7, 18]. Unlike these efforts, our scheduling strategies are tightly integrated with the performance models. Our strategies not only use the performance models for evaluations of schedules, but also for guiding the search of the schedules.

Existing work on detecting and predicting phases of a multi-phase application are mostly for sequential applications and parallel applications with simple models [5, 12]. The existing rescheduling efforts for tightly coupled parallel applications were built for specific applications [4, 9].

3. Performance Modeling Based Metascheduler Framework

Our metascheduer framework is illustrated in Figure 1. When a tightly-coupled parallel application is integrated into the grid by the application developer, the minimum problem size and the minimum number of processors for the application are specified by the developer. A small set of non-dedicated resources in a cluster is then chosen for training an initial list of performance model functions for the application. At the end of the modeling phase, a filtered list of combinations of performance model functions with small standard error values is formed. The list is sorted in the ascending order of standard error values. The functions in the list can predict execution times of the application for a given set of problem and resource characteristics on the

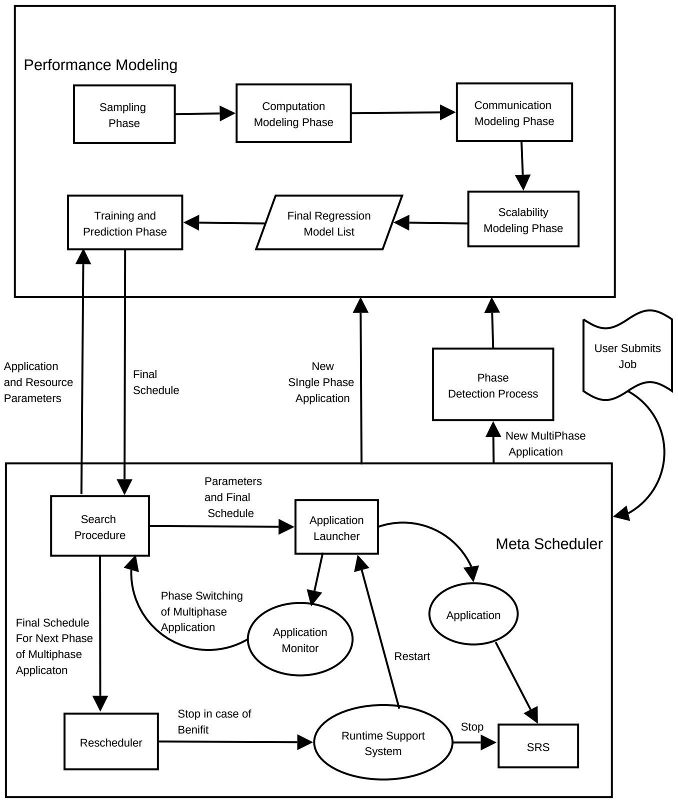
Performance Modeling
Sampling Phase
Computation Modeling Phase
Communication Modeling Phase
Scalability Modeling Phase
Final Regression Model List
Training and Prediction Phase
Application and Resource Parameters
Final Schedule
New SIngle Phase Application
Phase Detection Process
User Submits Job
New MultiPhase Application
Parameters and Final Schedule
Search Procedure
Application Launcher
Meta Scheduler
Phase Switching of Multiphase Application
Application Monitor
Application
Final Schedule For Next Phase of Multiphase Applicaton
Restart
Stop in case of Benifit
Rescheduler
Runtime Support System
Stop
SRS

Figure 1. Metascheduler Framework

cluster where the model functions were trained. We refer to this cluster as a trained cluster. The results of the training phase, namely, the ordered list of performance model functions and the training data, are then ported from one of the trained clusters to the untrained clusters by scaling the coefficients of the functions.

When a user submits a problem to the metascheduler corresponding to the application with a given problem size, the search procedure evaluates different sets of resources from different clusters. For a given candidate set of resources in a cluster with a given set of resource characteristics, the grid scheduler uses the top performance model function in the ordered list, for the cluster to predict the execution time of the problem. The grid scheduler then chooses an optimal set of resources in one of the clusters for application execution.

In the case of multiphase applications, a phase detection process identifies the different phases of the application. The same modeling procedure is followed to derive performance models for different phases. When a user submits a multiphase application to the metascheduler, the application is scheduled on a set of resources corresponding to the first phase. An application monitor tracks the phase changes in the application. When the application changes to a new phase, the monitor invokes the search procedure to determine the best schedule for the next phase. A rescheduler component calculates the performance gain due to migrating the application to the new schedule. If there is a performance gain, the rescheduler will migrate the application to the new schedule. Application migration is enabled by instrumenting the application with a checkpointing library, called SRS [16]. The rescheduler coordinates with a runtime support system that interfaces with the application for migration.

4. Performance Modeling

In our modeling method, we calculate the time taken for the execution of parallel application as:

$$T(N, P, minAvgAvailCPU, minAvgAvailBW) = \frac{f_{comp}(N)}{f_{cpu}(minAvgAvailCPU) \cdot f_{Pcomp}(P)} + \frac{f_{comm}(N)}{f_{bw}(minAvgAvailBW) \cdot f_{Pcomm}(P)} \quad (1)$$

where N is the problem size; P is the number of processors; minAvgAvailCPU and minAvgAvailBW represent the transient CPU and network characteristics, respectively; f$_{comp}$ and f$_{comm}$ indicate the computational and communication complexity, respectively, of the application in terms of problem size; f$_{cpu}$ is the function to indicate the effect of processor loads on computations; f$_{Pcomp}$ is used along with computational complexity to indicate the computational speedup or the amount of parallelism in computations ; f$_{bw}$ is the function to indicate the effect of network loads on communications. f$_{Pcomm}$ is used along with communication complexity to indicate the communication speedup or the amount of parallelism in communications.

Our performance modeling strategies based on regression, predict execution times of applications in non-dedicated systems where the external CPU and network loads on the systems can change during application execution. The different functions of Equation 1 are derived using various steps illustrated in Figure 1. Our strategy is adaptive to grid load dynamics since it can use different functions at different times based on load changes. We have also developed cross-platform performance modeling techniques whereby the performance modeling results of an application on one cluster can be used for predicting execution times of the application on another cluster.

5. Scheduler

Our scheduler framework consists of a search procedure that is used with an application-specific performance model. The application specific performance model, shown in Equation 1, is used as the objective function for the search procedure. We have implemented different existing optimization techniques, namely, *ant colony optimization (ACO)*, *genetic algorithm (GA)*, *simulated annealing (SA)*, *branch and bound (BB)* technique, and *dynamic programming (DP)* for our search procedure. Our search procedure is tightly coupled with application specific performance model since it uses the model for both evaluation of schedules and guiding the search paths. This is illustrated in the pseudo code shown in algorithm 1.

Algorithm 1 Pseudocode for Performance Modeling Based Genetic Algorithm

```
1: Set alarm to make algorithm time tunable
2: Randomly generate initial population
3: Evaluate the fitness of each individual in the population
   using Equation 1.
4: while terminating condition do
5:    Breed new generation through crossover
6:    Randomly choose a machine X in a individual to
      mutate
7:    for Each machine Y, which is not a part of individual
      do
8:       Using Equation 1 calculate the rank of the machine
         Y, if it replaces machine X in the individual
9:    end for
10:   Pick the top ranked machine to replace the machine
      X
11:   Evaluate the individual fitness of the offspring using
      Equation 1.
12:   Select the best ranking individuals using elitism
13: end while
```

6. Performance Modeling Large Scientific Applications

For predicting the execution times of large applications with multiple phases of computation and communication complexities, we plan to investigate the appropriateness of various available techniques for phase detection and prediction [5]. These techniques use various application parameters including working sets, conditional branches and basic blocks to identify phases in the application. The primary challenge will be to study the usefulness of the techniques for various kinds of non-dedicatedness of the systems. Another challenge is to determine the appropriate thresholds in variation of performance metrics that can be used to define phase boundaries. For each of the detected phases, we can then use the CPU and network load measurements and execution times within the phase boundaries to derive per-phase execution models.

7. Rescheduler

The predictions by the per-phase execution time models can be used by rescheduling strategies to dynamically migrate the application to different sets of resources suitable for a phase as the application enters the phase. The application is dynamically instrumented with checkpointing calls [16] at the phase boundaries. When the application enters the phase, the appropriate data at the phase boundary is checkpointed.

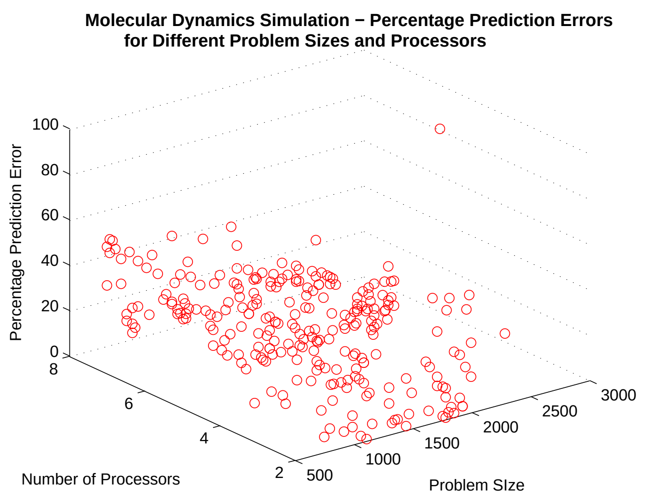

Figure 2. Percentage Prediction Errors

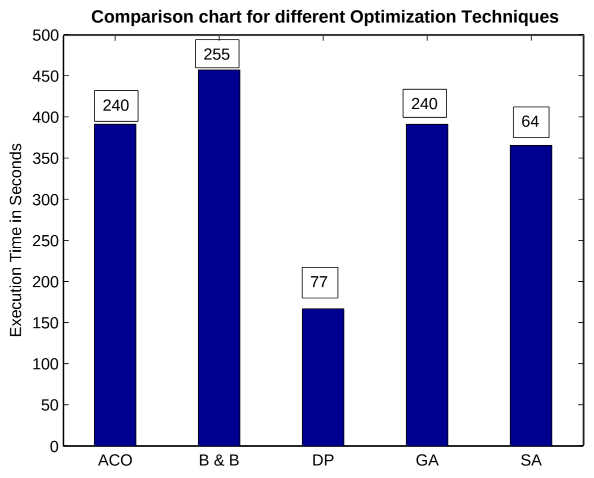

Figure 3. Comparison chart

The rescheduler invokes the search procedure with the performance model for the new phase, stops the executing application and continues on the new schedule of resources.

8. Experimental Results

We have evaluated our strategies on 4 different clusters (8, 16, 24, 32 processors) for 7 different applications. We tested our strategies with both random CPU and network loads and also with load traces obtained from machines in the GrADS testbed [6]. We verified our performance models both in terms of average percentage performance prediction errors and also in terms of its usefulness to a metascheduler to arrive at the correct scheduling decisions. We obtained less than 30% average percentage prediction errors in all cases which to our knowledge, is the highest reported for predicting application execution times for any number of processors on non-dedicated systems. Figure 2 plots the percentage prediction errors for different problem sizes and number of processors for the Molecular Dynamics Simulation. We find that average percentage prediction error is 18.86%. Our cross-platform performance modeling also resulted in less than 30% average percentage prediction errors.

We have evaluated our scheduling strategies by simulating multi-cluster setups. We find that strategies which tightly coupled with application specific performance model gives better schedules than strategies which use performance models only for evaluating schedules. Figure 3 plots the execution time of the best schedule determined by the scheduling strategies on a 256 machine simulation setup.The text box above each bar indicates the number of machines chosen for that best solution.

9. Conclusions

In this work, we had proposed performance modeling based metascheduler framework for computational grids. Our performance modeling strategies are adaptive to grid dynamics since we deal with load changes during application execution and also use different model functions at different times, for predicting execution times. We have also developed cross-platform modeling techniques for porting the results of performance modeling on one platform or cluster to other clusters in the grid. Our performance modeling strategies gave less than 30% average percentage prediction errors in all cases, which is reasonable for non-dedicated systems. Our scheduling strategies which are tightly coupled with application specific performance model gives efficient schedules for executing parallel applications. We have also described our plans for performance modeling multi-phase applications and rescheduler framework.

10. Future Work

We also plan to augment our techniques for predicting execution times for complex multi-component applications. We also plan to extend our modeling techniques to model the I/O costs in the applications.

References

[1] V. Adve and M. Vernon. Parallel Program Performance Prediction using Deterministic Task Graph Analysis. *ACM Transactions on Computer Systems*, 22(1):94–136, 2004.

[2] C. Anglano. Predicting Parallel Applications Performance on Non-Dedicated Cluster Platforms. In *ICS '98: Proceedings of the 12th international conference on Supercomputing*, pages 172–179, 1998.

[3] R. Badia, J. Labarta, J. Gimenez, and F. Escale. DIMEMAS: Predicting MPI Applications Behavior in Grid Enviornaments. In *In Workshop on Grid Applications and Programming Tools (GGF8)*, Seattle York , U.S.A, June 2003.

[4] Liang Chen, Qian Zhu, and Gagan Agrawal. Supporting Dynamic Migration in Tightly Coupled Grid Applications. In

SC '06: Proceedings of the 2006 ACM/IEEE conference on Supercomputing, 2006.

[5] A. Dhodapkar and J. Smith. Comparing Program Phase Detection Techniques. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 217–227, December 2003.

[6] GrADS Traces. `http://pompone.cs.ucsb.edu/~rich/data`.

[7] Casanova. H., Obertelli. G., Berman. F., and Wolski. R. The Apples Parameter Sweep Template: User-level middleware for the Grid. In *Proceedings of the Supercomputing*, November 2000.

[8] Ligang He, Stephen A. Jarvis, Daniel P. Spooner, Xinuo Chen, and R. Nudd. Hybrid Performance-oriented Scheduling of Moldable Jobs with QoS Demands in Multiclusters and Grids. In *In IEE Proceedings Software*, October 2004.

[9] Gregory Koenig and Laxmikant Kale. Optimizing Distributed Application Performance Using Dynamic Grid Topology-Aware Load Balancing. In *IPDPS '07: Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium*, 2007.

[10] B. Lee, D. Brooks, B. de Supinski, M. Schulz, K. Singh, and S. McKee. Methods of Inference and Learning for Performance Modeling of Parallel Applications. In *Proceedings of Symposium on Principles and Practice of Parallel Programming (PPoPP'07)*, San Jose , California, U.S.A, March 2007.

[11] G. Nudd, D. Kerbysin, E. Papaefstathiou, S. Perry, J. Harper, and D. Wilcox. PACE - A Toolset for the Performance Prediction of Parallel and Distributed Systems. *The International Journal of High Performance Computing Applications*, 14(3):228–251, 2000.

[12] E. Perelman, M. Polito, J.-Y. Bouguet, J. Sampson, B. Calder, and C. Dulong. Detecting Phases in Parallel Applications on Shared Memory Architectures. In *20th International Parallel and Distributed Processing Symposium*, 2006.

[13] H.A. Sanjay and Sathish Vadhiyar. Performance Modeling of Parallel Applications for Grid Scheduling. *Submitted after first revision to Journal of Parallel and Distributed Computing - Elsevier*, sep 2007.

[14] J. Schopf and F. Berman. Performance Prediction in Production Environments. In *Proceedings of 12th International Parallel Processing Symposium*, Orlando , USA, March 1998.

[15] V. Taylor, X. Wu, and Rick Stevens. Prophesy: An Infrastructure for Performance Analysis and Modeling of Parallel and Grid Applications. *ACM SIGMETRICS Performance Evaluation Review*, 30(4):13–18, March 2003.

[16] S. Vadhiyar and J. Dongarra. SRS - A Framework for Developing Malleable and Migratable Parallel Applications for Distributed Systems. *Parallel Processing Letters*, 13(2):291–312, june 2003.

[17] J. Yagnik, H. A. Sanjay, and S. Vadhiyar. Performance Modeling based on Multidimensional Surface Learning for Performance Predictions of Parallel Applications in Non-Dedicated Environments. In *ICPP '06: Proceedings of the 2006 International Conference on Parallel Processing*, pages 513–522, 2006.

[18] A. YarKhan and J. Dongarra. Experiments with Scheduling using Simulated Annealing in a Grid Environment. *Lecture notes in computer science - Grid 2002*, November 2002.