Failure-Aware Resource Selection for Grid Computing

Zhiling Lan and Yawei Li
Department of Computer Science
Illinois Institute of Technology, Chicago, IL 60616
{lan, liyawei}@iit.edu

1. Introduction

Over the past decade, grid computing has been rapidly becoming a promising model for high-throughput computing, distirbuted supercomputing, and data-intensive computing [11]. Grid is defined as a hardware and software infrastructure that enables coordinated resource sharing within dynamic organizations. While Grids bring unprecedented computing power for users, how to efficiently utilize their enormous power remains a challenging problem. In particular, job scheduling has been actively studied in the field of grid computing. Generally, job scheduling in Grids is handled in a hierarchical manner: global scheduling to select suitable machines or sites (denoted as *resource selection or site selection*), and local scheduling to choose appropriate nodes at a site. Therefore, how to select an optimal resource for a given job among heterogeneous and dynamical sharing resources is of critical importance for grid computing.

There are many research efforts on addressing the resource selection issue for grid computing. Globus is a well-known grid infrastructure, which provides a number of services for remote job invocation and management. However, it does not specify any policies for optimal resource selection and users are supposed to specify resources through the RSL language [1]. Broadly speaking, the optimal resource selection problem is addressed by either embedding application-specific information in the selection module, or targeting particular classes of applications, or utilizing load- or performance-based selection policies [8,9,10].

Grids are more pront to failures than traditional parallel machines as there are potentially thousands of resources that are heterogeneous and sharing among various applications [4]. Several research works have been proposed on fault management for grid computing [4,5,10], which mainly deal with failures through job migration or rescheduling after the job allocation phase. To date, little work has been done on addressing the reliability issue during resource selection.

In this paper, we propose a resource selection framework that is intended to identify an optimal resource for a given application by considering the reliability characteristics of available resources. In contrast to the existing fault tolerance practice in grid computing, the proposed work emphasizes on choosing an optimal resource by considering reliability, fault tolerant mechanism, and processor performance of available resources during the phase of resource selection. The rationale behind is that an intelligent resource selection could save tramendous overhead that may be introduced by job migration or rescheduling after job allocation. The proposed framwork can be easily integrated with existing grid infrastructure and failure-aware resource management systems.

2. Failure-Aware Resource Selection

2.1. System Architecture

A block diagram of the proposed failure-aware resource selection framework is illustrated in Figure 1. We assume that each site contains a cluster that is homogeneous, but clusters at different sites may be heterogeneous.

- A user submits an application with specified requirements to the resource selector. The requirements include the number of processors required and the estimated execution time on a base machine.
- The resource discovery service, e.g. the Globus MDS (Monitoring and Discovery Service) provides the basic mechanism for discovering and disseminating information about the structure and state of Grid resources.
- The performance service and the reliability service maintain historical information of performance and reliability data of resources. The information includes relative processor speed per site, average queue wait time per site, failure history, and fault tolerant mechanism adopted per site.
- The resource selector is responsible for choosing the optimal resource that can provide the minimal completion time for a given application, considering the reliability characteristics of available resources. The result will then be forwarded to the resource allocation service.
- The resource allocation service, e.g. the Globus GRAM (Grid Resource Allocation and Management), supports remote submission of the application to remote resources, and subsequently monitoring and control the resulting computation.

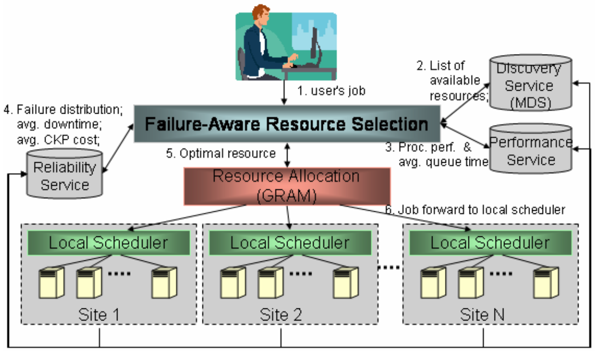

Figure 1. Failure-Aware Resource Selection

2.2. Selection Policy

When a user request arrives, the failure-aware resource selector is responsible for choosing a site that can provide the minimum expected job completion time: $\min_{1\le i\le N}(E[T_i^{completion}])$ where N is the number of available sites.

Job completion time consist of two parts: *queue-wait time* T_i^{queue} *and execution time* T_i^{exec}. T_i^{queue} is estimated as the average queue wait time at a site, while T_i^{exec} depends on many factors, including processor speed, reliability characteristics, and associated fault tolerance mechanism at a site. Historical data maintained by the performance and reliability service is used to calculate these values. Based on the analytical models proposed in [7], we calculate expected execution time at a site as below:

- If site i does not provide any fault tolerance support, then the expected job execution time at site i is calculated by the following equation:

$$E[T_i^{exec}] = T_i^{free} + \frac{C_i^d F_i(T_i^{free})}{\bar{F}_i(T_i^{free})} + \frac{\int_0^{T_i^{free}} t\,dF_i(t)}{\bar{F}_i(T_i^{free})} \qquad (1)$$

where

T is the estimated job exec. time on a base machine;

ρ_i is the node performance of site i, relatived to the base machine;

T_i^{free} is the fault - free job execution time at site i $(= \frac{T}{\rho_i})$;

$F_i(\cdot)$ is the CDF of failure at site i;

C_i^d is the average downtime at site i;

- If site i provides periodic checkpointing mechanism, then the expected job execution time can be calculated as below:

$$E[T_i^{exec}(N_i)]\bar{F}_i(\tau_i) = T_i^{free}\bar{F}_i(T_i^{free}) + C_i^d F_i(T_i^{free}) + \int_0^{T_i^{free}} t\,dF_i(t) + \sum_{k=1}^{N_i-1} E[T_i^{exec}(N_i-k)][F_i((k+1)\tau_i) - F_i(k\tau_i)] \qquad (2)$$

where

C_i^{cp} is the checkpointing cost at site i;

N_i is the number of checkpoints for the job at site i;

T_i^{free} is the fault - free job exec. time at site i $(= \frac{T}{\rho_i} + N_i C_i^{cp})$;

τ_i is the priodic interval at site i $(= \frac{T_i^{free}}{N_i})$;

3. On-going and Future Work

Currently, we are conducting trace-based simulations to evaluate the proposed work. Failure logs and job logs collected from various production systems are used.

Our future work includes integrating the proposed failure-aware resource selection with existing resource management systems and test on production Grid systems, such as the TeraGrid.

Reference

[1] The Globus Project. http://www.globus.org

[2] C. Liu, L. Yang et al., "Design and Evaluation of a Resource Selection Framework for Grid Applications", Proc. of HPDC'02, 2002.

[3] B. Lee and J. Weissman, "Adaptive Resource Selection for Grid-Enabled Network Services", Proc. of NCA'03, 2003.

[4] R. Medeiros, W. Cirne, et al., "Faults in Grids: Why are they so bad and What can be done about it", Proc. of Intl. Workshop on Grid Computing, 2003.

[5] L. Burchard, C. De Rose, et al., "VRM: A Failure-Aware Grid Resource Management System", Proc. of International Symposium on Computer Architecture and High Performance Computing, 2005.

[6] J. Frey, T. Tannenbaum, et al., "Condor-G: A Computation Management Agent for Multi-Institutional Grids", Proc. of HPDC'01, 2001.

[7] Sachin Garg, Yennun Huang, et al., "Minimizing Completion Time of a Program by Checkpointing and Rejuvenation", SIGMETRICS 1996.

[8] Seung-Hye Jang, Valerie Taylor, et al., "Performance Prediction-based versus Load-based Site Selection: Quantifying the Difference", *Proc. of PDCS-2005*, Las Vegas, Nevada, 2005.

[9] F. Berman and R. Wolski, "The AppLeS Project: A Status Report", Proc. of the 8th NEC Research Symposium, Germany, 1997.

[10] H. Casanova and J. Dongarra, "NetSolve: A Network-Enabled Server for Solving Computational Science Problems", The International Journal of Supercomputer Applications and High Performance Computing, 1193: 212-223, 1997.

[11] I. Foster and C. Kesselman, "The Grid: Blueprint for a New Computing Infrastructure", Morgan Kaufmann, 2004.