Fault-tolerant data sharing for high-level grid programming: a hierarchical storage architecture

Marco Aldinucci, Marco Danelutto
Computer Science Department, University of Pisa
Largo Bruno Pontecorvo 3, 56127 Pisa, Italy
aldinuc@di.unipi.it

Gabriel Antoniu, Mathieu Jan
INRIA Rennes
Campus de Beaulieu, 35042 Rennes, France
gabriel.antoniu@irisa.fr, mathieu.jan@irisa.fr

Abstract
Enabling high-level programming models on grids is today a major challenge. A way to achieve this goal relies on the use of environments able to transparently and automatically provide adequate support for low-level, grid-specific issues (fault-tolerance, scalability, etc.). This paper discusses the above approach when applied to grid data management. As a case study, we propose a 2-tier software architecture that supports transparent, fault-tolerant, grid-level data sharing in the ASSIST programming environment (University of Pisa), based on the JuxMem grid data sharing service (INRIA Rennes).

Keywords: Grid, shared memory, High-level programming, memory hierarchy, P2P.
1. Introduction

Grid computing has emerged as an attempt to provide users with the illusion of an infinitely powerful, easy-to-use computer, which can solve very complex problems. This very appealing illusion is to be provided (1) by relying on the aggregated power of standard (so, inexpensive), geographically distributed resources owned by multiple organizations; (2) by hiding as much as possible the complexity of the distributed infrastructure to users. However, the current status in most software grid infrastructures available today is rather far away from this vision. When designing programs able to run on such large-scale platforms, programmers often need to explicitly take into account resource heterogeneity, as well as the unreliability of the distributed infrastructure. In this context, the grid community converges towards a consensus about the need for a high-level programming model, whereas most of the grid-specific efforts are moved away from programmers to grid tools and run-time systems. This direction is currently pursued by several research initiatives and programming environments, such as ASSIST [17], eSkel [9], GrADS [14], ProActive [8], Ibis [16], Higher Order Components [11], etc., along the lines of CoreGRID’s “invisible grid” approach to next generation grid programming models.

In this work, we explore the applicability of the above approach to data management for high-level grid programming. We consider three main aspects that need to be automatically handled by the data storage infrastructure: data access transparency, data persistence and storage fault-tolerance.

Transparent access to data across the grid. One of the major goals of the grid concept is to provide an easy access the underlying resources, in a transparent way. The user should not need to be aware of the localization of the resources allocated to the application submitted. When applied to the management of the data used and produced by applications, this principle means that the grid infrastructure should automatically handle data storage and data transfer among clients, computing servers and storage servers as needed. However, most projects currently still rely on the explicit data access model, where clients have to move data to computing servers. In this context, grid-enabled file transfer tools have been proposed, such as GridFTP [5], DiskRouter [12], etc. In order to achieve a real virtualization of the management of large-scale distributed data, a step forward has been made by proposing a transparent data access model, as a part of the concept of grid data-sharing service, illustrated by the JuxMem software experimental platform [6].

Persistent storage of data on the grid. Since grid applications typically handle large masses of data, data transfer among sites can be costly, in terms of both latency and bandwidth. Therefore, the data-sharing service has to provide persistent data storage. Data produced by one computation can be