DISTRIBUTED DATA MANAGEMENT FOR GRID COMPUTING

MICHAEL DI STEFANO

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC. PUBLICATION

DISTRIBUTED DATA MANAGEMENT FOR GRID COMPUTING

DISTRIBUTED DATA MANAGEMENT FOR GRID COMPUTING

MICHAEL DI STEFANO

WILEY-
INTERSCIENCE

A JOHN WILEY & SONS, INC. PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey.
Published simultaneously in Canada.

Library of Congress Cataloging-in-Publication Data:

Di Stefano, Michael, 1963–
Distributed data management for grid computing / by Michael Di Stefano.
p. cm.
Includes bibliographical references.
ISBN 0-471-68719-7 (cloth)
1. Computational grids (Computer systems) 2. Database management. I. Title.

QA76.9.C58D57 2005
004.3′6--dc22 2004031017

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

This book is dedicated to my parents, who instilled in their children the importance of hard work, honesty, education, and dedication to family and friends, for making any sacrifice, no matter how great, to ensure that all of their children succeed to their fullest potential.

CONTENTS

FOREWORD

Commercial grid computing is inevitable. As certain as the sunrise or sunset, grid computing, or the ability to abstract the business logic (application) layer from the infrastructure layer, will be a reality. As firms' technology architecture continues to become more complex and technology budgets continue to come under increasing scrutiny, firms need to rethink the way they manage and utilize technology.

The current ways of tying applications to very specific hardware just will not scale. Firms are buying new technology when other servers are sitting underutilized. Firms are acquiring more hardware when they have thousands of desktops (after work hours) and even whole data centers (across the globe) sitting dormant. And even if we continue to throw hardware at our computational challenges, sooner or later the overhead of managing this infrastructure will become overwhelming.

Besides not being able to function without grid technology to help manage our increasingly complicated technology infrastructures, our 30 years of modern computing history all point toward a need for a better way to manage a widely distributed computing architecture. Whether it is called *grid computing* or *utility computing*, the shift toward hardware and software componentization cries out for a better technology management model.

Over the entire history of computing we have consistently experienced a pronounced increase in computational power and a continual decrease in both CPU size and cost (Moore's law). In the mid-1980s, there was the mainframe; in 1990 it was the Unix server, and today there is the virtually disposable Linux or Windows-based rack-mounted cluster. Concurrently we have witnessed a continual decomposition of traditional software applications from mainline COBOL programs, with embedded program calls, to client/server, the Web, and today service-oriented architecture (SOA)–based applications. While the COBOL and client/server-based applications ran on dedicated hardware, today's SOA-based applications can be run virtually anywhere.

But what happens when firms begin to roll out these new hardware and software architectures? How will firms be able to manage every single blade server running all of these Web services? Will they know what is running on the second partition of the third blade of the twenty-fifth cluster? Will corporate data centers be able to track the utilization rate of the eighteenth blade of the fourth cluster? Will they know when the blade was underutilized, and what could have been provisioned on that platform? What if the blade is down? How will they know, who will fix it, and what will happen to its workload?

None of these issues will be resolved without a more efficient, more fully automated technology management infrastructure. This is the challenge that grid computing is tackling.

Grid computing was initially targeted at decomposing computationally challenging problems into many pieces and parceling them out to a wide array of computational resources. Today grid computing is much more than high-performance computing; it is about virtualizing and abstracting the complete technology footprint from both users and software developers. It is about having technology manage technology.

This is not an easy problem to solve. It is more than lashing together a dozen computers. It is more than breaking a large problem into smaller pieces. It is more than provisioning on the fly. Grid computing is a comprehensive technology management infrastructure that decomposes, monitors, provisions, distributes, manages, and meters virtually all technologies within the organization and sometimes outside the organization.

That is why you are reading this book. Michael's book will help you get a much better understanding of grid computing—how it works, the theory, practice, and the challenges of pulling it all together. While I firmly believe that this technology is inevitable, the real question is "When will it be practical?" With this book, and Michael's help, the answer to that question will certainly be sooner rather than later.

LARRY TABB

Founder & CEO
TABB Group

PREFACE

Grid computing technology is breaking out of its birthplace in universities and research facilities and is quickly gaining acceptance in the commercial industry. In fact, the financial industry is where my company and I were first introduced to grid computing technology. I am very active in financial firms on Wall Street as they explore the potential use of grid technology for various business applications, restructuring data centers, and operations of data centers. With more years than I care to count or even mention, I have been an integral part of architecting and building distributed computing environments (client/server topology) for the financial industry and in the past few years (at the time of writing) have been working in the grid computing topology as it extends to financial institutions. This is not to say that this is the only industry to which this technology applies. As a result, it quickly became apparent that running business applications and services in the grid computing topology was not the same as the traditional client/server and new data management techniques were needed to leverage this new topology.

The first step is the buildout of the hardware infrastructure for grid computing (compute nodes, networks, etc.). Once in place, "Bob's your Uncle"; the rest should be as simple as migrating applications over to, or better yet, converting business line applications into, "services" for their "customers" to "purchase." However, the reality is that the hardware and the operating system of a grid at the end of the day is just another computer consisting of CPUs, memory, disks, and a communication bus. Granted, the internal components appear radically different from those of the big servers that we are accustomed to seeing in data centers. The *compute grid* is a logical computer that physically consists of many networked computers (or compute nodes) that spans one data center, multiple data centers, floors of a building, and even cities. When moving even the simplest of applications onto the new computer, there is at least one critical tool that the developers must have, a database, specifically, a data grid. The initial reaction is: "Our applications already have a database, we will use those" or "Why don't we use the relational databases that we have already paid licenses for?" However, given the difference in physical topology between the client/server and grid computing, the architects and developers will immediately realize that managing data in a grid computing environment is very different. Without the proper data management tools, developers are back to writing down to the bare metal of the grid to get data in and out of the grid, distributing the data among all the nodes where work needs to be performed, and must manage some sort of data synchronization (e.g., distribution of data across the nodes of the grid, and with external data sources that include not only databases but also all the various middleware tools, file systems, etc.). The information technology staff in many organizations have already received the green light to start to deliver applications on the compute grid without the required tools for providing data management. As a result, these projects will require more time and thus cannot achieve fast time to market, low costs, and so on since large amounts of time must be spent on creating pure infrastructure code customized for each application. The reusability of such code is small or nonexistent, resulting in additional resources and time to deal with the nuts and bolts of the grid. Without the proper data management tools, the migration will be slow and expensive at the cost of total acceptance of the technology into the commercial industry. This would jeopardize the whole "grid thing" altogether.

Working with our clients and the grid computing technology vendors, it became apparent that the management of data was not sufficiently addressed through the use of traditional data management techniques. The physical topology of the grid is as different from the client/server as the client/server was from the mainframe. Data management systems that were architected for the client/server are optimized and perform best in that topology, but not necessarily perform as needed by the grid topology. To gain optimal performance from of the grid topology, various levels of analysis are required, including the analysis of data types and their behaviors. The analysis drives different data management techniques that are required as part of the core for the data management system or the "engine" that needs to be redefined. The engine's (as an integral part of data management system) responsibility is to manage the mechanics required by the data storage devices and the movement of data into and out of the physical realm of the grid.

The first set of applications to run within the grid has operated over static data sets, and large files whose contents rarely, if ever, change. Naturally, the data management techniques for these types of data and the applications associated with them within the grid are geared toward the management and distribution of large static data sets across the nodes of the grid. Examples are GridFTP (Grid File Transfer Protocol) for distributed filing systems and various research projects such as OceanStore. However, these techniques do not translate to the management of dynamic data used by many applications within the financial services sectors (as well as other vertical sectors).

Throughout the evolution of the computer from mainframe/minicomputer to client/server to middleware to distributed computing, the early adopters piloted the transitions of each, followed by books and reference materials made readily available to the armies of architects and developers involved in the mass adoption of these respective technologies. As we are now working with the early adopters of grid computing in the financial community, most, if not all, of the reference materials on grid computing are white papers and research reports. There is an obvious vacuum of printed material specifically as it relates to how to manage data in the highly distributed topology of the grid. We, at Integrasoft, began to fill this void by creating user groups where the early adopters of grid technology regularly meet to discuss their activities and present some of the latest developments in grid computing and data management within this technology: a forum of open idea exchange and discussion. This is a small attempt since there are not enough user groups globally to reach the masses needed to acquire the technology knowledge required for this next evolutionary step in computing. I started this project of authoring a book on distributed data management in grid computing to assist in the adoption of grid computing within the commercial industry, to provide an introduction to grid computing for people who are just starting to hear about it for the first time; for those who have been studying or considering and started to use grid computing, by introducing the concepts for the management of data within grid computing; and for the early adopters of this technology who are familiar with the complexities of data management in grid computing, to hopefully spark research and development of practical product in these areas in order to establish this technology as a standard.

The audience for this book is not limited to the technical purist; the topic of grid computing is presented with the main drivers for its adoption, the economic and sociological impacts on an organization. Thus, this is an introduction for people who are along the managerial paths, who are aware of and familiar with the general terms of data management, as with relational databases, and is intended to introduce grid computing in business terms so that these individuals can see the benefits of using grid technology and become advocates for the use of this technology in their projects. It is hoped that they will be armed with the tools necessary to discuss grid computing with their technical staff with a sufficient level of understanding of this technology and to explain to the upper management and corporate leaders the benefits of using grid technology. Finally, to complete the lifecycle, project managers must be able to present their rationale for using grid computing in their projects to their corporate leaders such as the CIO and CFO (chief investment and financial officers). They, too, should, having read this book, possess an understanding of the business drivers behind grid computing and the benefits it brings to an organization as a whole.

To draw in such a wide range of audience, I leverage three techniques: drawing on a common baseline of knowledge, visitation through analogy, and finally practical applications of grid computing. For the first technique, a common baseline of knowledge, the relational database and relational data management systems are used to explain and introduce data management within the grid. Readers should be able to walk away with the tools to help them promote grid technology into their respective organizations and into the community as a whole. My intention is not to provide a deep level of detail on the relational data management concepts since technical people are typically familiar with them. Project managers should already have the level of understanding of relational data management technology on a par with what is discussed within, and drilling down into the bowels of the underlying technology would not be of practical use.

The second technique, visitation through analogy, coupled with the common baseline of relational data management, completes the conceptual bridge between what is familiar to what is not. Finally, by presenting the practical business and technical use cases that people and corporations are looking for the grid technology to solve, we will see the immediate benefits and widespread impact that the grid will have on our everyday business and information technology lives.

The field of data management in the grid is a broad one; individually the topics introduced warrant more in-depth discussion than the pages of this book can provide. In fact, each aspect or topic of distributed data management merits its own book or series of books. So, for the technical readers who are intimately familiar with the details of grid computing, this book should spark further thought and work within the topics presented and contribute in the advancement of distributed data management. The technical person becoming acquainted to grid computing will acquire a firm understand of the field and the concepts of distributed data management in grid computing. I encourage them to read the white papers and reference materials listed at the end of this book. The technologist will be able to take distributed data management products (such as the one that we have developed, from the ground up for data management within grid computing), and quickly get projects up and running by assessing the various strengths and weaknesses of each product and correlating that to their project needs.

A handful of people have been generous enough to read the manuscript of this book, some being the early adapters and some are the newcomers to the field. One person described my goals for this book as being the "rosetta stone" for grid computing. As generous as he was in that description, I tend to look at is as "beauty is in the eye of the beholder," as individuals can look at a piece of work and draw from it value particular to their respective backgrounds, experience, and job responsibilities with the ultimate goal of helping them perform their jobs better and contributing to the adoption of grid computing. Achievement of this objective will also mean that I have achieved my goal.

ACKNOWLEDGMENTS

I would like to thank my loving family for their understanding, support, and further sacrificing the already few precious moments we spent together while I took on the additional responsibility of authoring this book.

Special thanks to Dave Cohen of Merrill Lynch and my partner in business, Steve Yalovitser, for their contributions on Service Oriented Network Architecture (SONA), to Andrew Delaney of A-Team Consulting for transforming my "techese" into the English language, to Larry Tabb for his contributions in the Foreword of this book, and to my editor, Val Moliere of John Wiley & Sons for her insight into the importance of data management in grid computing and guidance during the authoring process.

PART I

AN OVERVIEW OF GRID COMPUTING

1

WHAT IS GRID COMPUTING?

Grid computing has emerged as a framework for supporting complex compilations over large data sets. In general, grids enable the efficient sharing and management of computing resources for the purpose of performing large complex tasks. In particular, grids have been defined as anything from batch schedulers to peer-to-peer (P2P) platforms.

Grid computing has evolved in the scientific and defense communities since the early 1990s. As with most maturing technologies, there is debate as to exactly what grid computing is. Some make a very clear distinction between cluster computing and grid computing. *Compute clusters* are defined as a dedicated group of machines (whether they are individual machines or racks of blades) that are dedicated for a specific purpose. Grid computing uses a process known as "cycle stealing": grabbing spare compute cycles on machines across a network, when available, to get a task done.

Since both compute clusters and grids coordinate their respective resources to perform tasks, when does a compute cluster start to become a grid? Specifically, does a compute cluster become a grid when it is leveraged to perform operations other than those for which it was originally intended?

THE BASICS OF GRID COMPUTING

Grid computing is an overloaded term. Depending on whom you talk to, it takes on different meanings. Some terms may better fit your practical usage of the technology, such as clusters. For the purposes of this discussion, however, we shall define grid computing as follows:

> Grid computing is any distributed cluster of compute resources that provides an environment for the sharing and managing of the resource for the distribution of tasks based on configurable service-level policies.

A grid fundamentally consists of two distinct parts, compute and data:

- *Compute grid*—provides the core resource and task management services for grid computing: sharing, management, and distribution of tasks based on configurable service-level policies
- *Data grid*—provides the data management features to enable data access, synchronization, and distribution of a grid

If the proliferation of jargon is a measure of a technology's viability and its promise to answer key issues that businesses are facing, then transformation of jargon to standards is a measure of the longevity of the technology in its ability to answer concretely those key business issues. The evolution of *grid computing* from jargon to standard can be measured by a number of converging influences: history, business dynamics, technology evolution, and external environmental pressures.

The drivers behind grid technology are remarkably similar to those that corporations are facing today: a starving business need for powerful, inexpensive, and flexible compute power, and limited funds to supply it. In the early 1990s, research facilities and universities used increasingly complex computational programs requiring the processing power of a supercomputer without the budget to supply it. Their answer was to create a compute environment that could leverage any spare compute cycles on campus to perform the required calculations.

Today, grid technology has evolved to the point where it is no longer a theory but a proven practice. It represents a viable direction for corporations to explore grid computing as an answer to their business needs within tight financial constraints.

There are additional forces in play that will present a fundamental paradigm shift in how computing is done. As it migrates from the hands of artistry to the realm of engineering—via the application of tried-and-true engineering principles—computing becomes a fundamental utility in the same way that gas and electricity generation and delivery is a utility. The quality of the service will be measured by its ability to meet the supply-and-demand curves of the producers and consumers.

Leveling the Playing Field of Buzzword Mania

There are many analogies in the development and adoption of grid computing to those of client/server technology. Both are fundamental paradigm shifts in the way computing is performed. As client/server technology ushered in the broad acceptance of relational database technology, grid technology will usher in new data management paradigms to address the specific topology of the physical compute grid.

To see how this is happening, it is best to untangle the concepts of data management in grid form by drawing on a fundamental baseline that we are all familiar with. The people who are going to use grid technology—developers, architects, and lines of businesses—are accustomed to thinking in terms of client/server technology and the relational data management features within a client/server paradigm. Irrespective of the compute topology—client/server, computer clusters, or a computer grid—from the user perspective, these data management service levels need to be consistently maintained.

In the early days of client/server technology one would attend a seminar sponsored by a relational database vendor, promoting relational technology in general, and the supplier's product in particular. The message was that the new compute paradigm of the client/server topology required new, more flexible data management techniques than do those currently in use. As a result, relational databases became synonymous with client/server technology and the standard for data management.

People attending those seminars were used to writing their own disk controllers for data storage, so popular questions centered on disk management. How fast does your product write to and/or read from disk? How efficient are your indices? How well does your product manage physical data positioning on the disk? The bulk of the seminar was spent on addressing these questions, and the only discussion of data management centered on the use of a new language called *Structured Query Language* (SQL) for storage and querying of the data. If you were interested, there were SQL training classes to attend, where only the basics of how to form a query were taught.

Figure 1.1 illustrates the parallels of the vocabulary and fundamentals between data management within relational databases and that within grid computing. This comparison is useful in two aspects: (1) it relates to terms that most are already very familiar with and (2) more importantly, it suggests that any data management system in grid computing must provide the same levels of service quality as within relational databases.

Figure 1.1 links a baseline of data grid vocabulary to well-known relational database terms. Relational database implementations have two fundamental components: (1) the underlying engine that manages physical resources, in this case a disk and (2) a layer on top of that to provide all the data management features and functionality that architects and developers would rely on for data management, querying, arrangement of data in highly ordered structures such as tables, the ability to transact on data, leveraging stored procedures, event triggerings, and transacting in and out of the database with external systems. These are the management features and functions that today are where our true interest lies. How do I manage tables/row locking? How do I structure indices for maximum performance? Very little attention today is given to the underlying engine.

In the same way that *relational database* is a generic term, so is *data grid*. Companies will offer implementations, products of their vision of what a data grid is. To analyze the differences between the products offered, it is possible to apply a

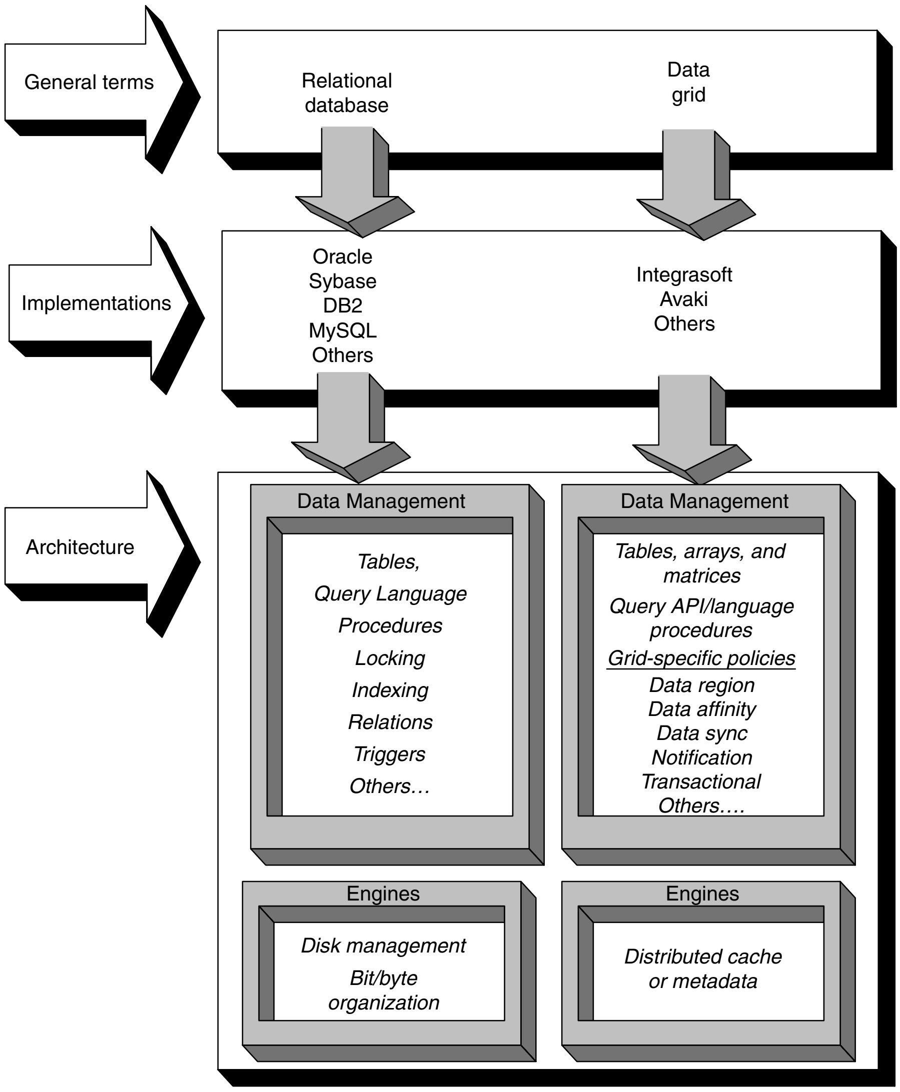

Figure 1.1. Baseline of terms and function.

baseline consisting of generic term, implementation, data management, and engine. Each implementation of a data grid will have an engine. That engine may be a metadata dictionary or a distributed cache. It will also handle the data management aspects of this data grid, defining how to structure data in tables, arrays, or matrices; how to query data; and how to transact on the data.

Depending on the exact implementation of this engine—whether it is a metadata dictionary that routes requests to the true long-term persistent stores, or a distributed cache that spans all computers in the grid to form one virtual space—there are