

Developing a PHP-based Legacy Application Grid System

*Lorenzo Campanelli
Stetson University*

Abstract

A big problem with the emerging distributed Grid technologies is that each proposed implementation carries unique design and interface features that do not necessarily conform to an accepted standard. This lack of a presiding standardized model for useful Grid systems has created a level of uncertainty and complexity with respect to practical deployment of systems outside of academia. While implementation of emerging systems within their respective research domains may seem successful, it still leaves questions open about how practical these competing designs are to existing legacy applications. This project proposes a PHP-based Grid system with the goal of providing a practical and rapidly deployable system which maintains compatibility with existing legacy applications.

1. Introduction

Neither Rome nor Linux were built in a day, and the same is the case of a typical Grid System. Just as the development of an Operating System, like Linux, was built one “brick” at a time to manage emerging local hardware, Grid Systems are being developed by gradually exploiting networked resources and properly managing them within the requirements of applications. The purpose of an Operating System is to properly manage resources as applications demand them, and so does a typical Grid System. However, what makes a Grid system unique from other resource managers is that its responsibility consists of managing independent subsystems that are networked through separately administered domains. This very idea of having a management system govern distant components in an efficient and reliable manner is exactly what makes the design and implementation of a Grid challenging [1].

In essence, a Grid System is the platform which drives a large scale virtual computer. Instead of booting local hardware and allocating local memory like an Operating System would, the Grid System must boot resources from various heterogeneous systems that belong to privately administered domains. All of this must be done in an autonomous manner, so that a user of the system will feel as though his or her interface drives a single, virtual system. The challenges that emerge from these Grid platforms usually involve reliability and security. For example, since the fundamental requirement of a Grid System is to be based on an internet structure, there is a high possibility for dropouts, high latency, and exposure to unauthorized users. Although these challenges can hinder the development of a useful Grid, the extra efforts to defeat such threats are rewarded with greater efficiency over alternate systems for distributive applications.

With the increased risk of constructing inter-network Grids, one can expect a higher return of performance when designed and managed properly. For example, although the Grid depends on unreliable internet communication to utilize components, it compensates for failures by being highly scaleable and recoverable if enough component

alternatives are available. This is exactly why the design of a Grid system is just as significant as the design of a local Operating System, in which the overall performance, availability, and security of the system to the user depends on its ability to manage resources.

2. Background

Grid computing derived its name from its ideal similarity to the way power companies and their users share the modern electrical grid. Just as the power grid is readily accessible and collectively organizes many power plants to provide shared service to users, a Computational Grid is an infrastructure that provides shared access to high-end computational capabilities. It provides this access based on the fundamental requirement of a large scale pooling of computer resources. This is how a Grid System, in essence, acts as a computational power grid. The idea for a Computational Grid has appeared relatively recently with respect to earlier related ideas in parallel and distributed computing research [2].

2.1 Traditional Distributed Computing vs the Grid System

Both Traditionally Distributed Systems and Grid Systems have the same purpose of effectively dividing large-scale applications into small-scale processes to be distributed on a group of subsystems. Although both systems serve a similar purpose, a Grid System approaches the distribution of an application with a global mindset. For example, while traditional Distributed Systems involve closely coupled and homogeneous components, Grid Systems attempt to interconnect loosely coupled, heterogeneous components on a much larger scale. Also, a Grid System focuses more on joining foreign subsystems from separately administered domains, while traditional Distributed Systems combine locally available components from a single domain. In this regard, a Grid System provides a cross-institutional infrastructure of computationally powerful resources to be shared by users at the global level.

Because of this global approach, a Grid System becomes the superset of traditional distributed implementations. In other words, all sets of traditional distributed systems become subsets of the global inter-set Grid System. When a Grid System provides global access to this collaborative resource, it allows for some very powerful advantages with access to potentially infinite resources [3].

2.2 Advantages of a Grid System

The infinite potential of a Grid System is supported by many advantages over previously implemented supercomputing architectures. One advantage is that a Grid system provides very inexpensive access to supercomputer capabilities. This allows for a Grid to have superior power over a standalone supercomputer for a fraction of the cost. For example, the [SETI@home](#) [4] Grid System Project has calculated that one of the most powerful supercomputers, the IBM ASCI White, is rated at about 12 TeraFLOPs for \$110 million, while the [SETI@home](#) Grid is rated at about 15 TeraFLOPs and has so far cost around \$500,000. Further savings come from lower operating costs, such as less direct electricity consumption, environmental conditioning costs, and other centralized costs associated with maintaining a single supercomputer.

An additional advantage that comes with Grid Systems is their greater scalability over local supercomputers. For example, because the Grid System is a grouping of loosely coupled components, supplying resources dynamically to application demands is as simple as adding or removing components from the Grid network. Theoretically, a Grid is infinitely scalable, allowing as many computing resources needed to join the Grid as the requirements change. However, a traditional centralized supercomputer is either

too expensive to scale in response to changing demand, or completely limited to the tolerable expansion that was implemented in its initial design.

Furthermore, a byproduct of the distributed nature of a Grid System is the exploitation of existing system resources. Subsystems of the Grid are typically not dedicated to the Grid at all, and only contribute resources when those resources are not in use by their respective administrative domains. Estimates of computer resource utilization by the average enterprise have indicated that up to 90% of CPU cycles remain idle. In many cases, these resources are not properly managed within a traditional network, so applications that require such resources are unable to access them. Even systems that become obsolete to an institution and would otherwise be disposed of may instead be recycled into beneficial components of a larger Grid System. These reasons are what allows a Grid System to efficiently use otherwise wasted or idle computer resources, and instead contribute them to applications that need them.

2.3 Requirements of Grid Systems

The usefulness of the Computational Grid infrastructure is dependent of three fundamental qualities. First, the infrastructure must be dependable in terms of performance, with regards to security, bandwidth, and overall processing power. Second, system consistency must be maintained by following presiding standards so that application development becomes compatible with interfaces and services. A major challenge of maintaining Computational Grid consistency is encapsulating heterogeneity without compromising processing performance [5]. If these three overall conditions can be met by the distributed infrastructure, then the system can be considered a functional Grid System.

3. Related Work

The most promising Grid systems proposal is that of Globus, which has been heavily researched and developed into a highly scaleable toolkit. With this toolkit, Grid systems may be deployed with a layered approach by building required applications above the fundamental middleware interface. As a result, these layered systems become services, which can further be managed by other components from the Globus toolkit. For example, the toolkit's Unified Resource Information Service provides real-time information about the underlying metasytem structure and status. Also, the Globus Heartbeat Monitor module provides system fault detection and component recovery. This specific module has two sets of API to provide *registration* and *notification* interfaces with client applications [7]. It is Globus' intention to provide such a *metacomputing abstract machine* by providing these many interfaces and layered design for developers to build upon [6]. However, this is an example of how a surplus of interfaces and complex design can prove impractical for legacy applications outside the efforts of the Globus research requirements. In addition to these proprietary interfaces, there is an adopted interface, the Metacomputing Directory Service, which uses the standardized LDAP directory protocol. However, even the LDAP API in Globus M.D.S. is altered to satisfy unique requirements of these proposed Grids [8]. This project will therefore refrain from excessive interface and design complexity, and instead integrate such features to serve the specific requirements of legacy applications.

4. Implementation

Implementation of this project's Grid System will be completely done using the PHP scripting language. The major development will be focused on a fundamental middleware system, in which arbitrary existing applications can be deployed and executed amongst the components of the Grid. Each component of the Grid will exist as an http daemon running the respective PHP scripts to handle the component's tasks.

User authentication will be implemented using a public/private key system, so that passwords will not be required when accessing the Grid. The user interface will allow for various functions, such as software deployment, execution, and status. The PHP middleware's primary purpose will be to properly manage the programs distributed on the Grid and efficiently distribute user input and collect program output for delivery to the user interface.

Because of the powerful functions available in the PHP scripting language, development of a fundamental middleware should occur rapidly. However, there are many issues related to the management of the Grid, which the project will explore various solutions for. For example, deploying software on the Grid will have to adapt to the heterogeneous environment that exists by releasing the correct binaries to their appropriate platforms. Also, the distribution of user input to software must remain efficient and robust, so that if a component fails to execute, the input is properly recovered and sent to an alternate component. Software output must be collected and delivered accurately to the user interface as it becomes available from each component. Additionally, the status of components and the overall Grid must be reported to alert the user of system changes and progress. These requirements will be implemented with different designs, both researched and original, until the best performing system is completed.

4. Conclusion

The research into Grid system design has so far only proposed many competing interfaces from which to deploy Grids. As system proposals emerge, they only serve the requirements of research-related applications. Also, many of these research-related applications have very complex deployment procedures, which make them less practical for existing distributive applications. The way to apply the advantages of Grid System Computing to legacy applications is to cut down on the complexity of interface design, and avoid conforming to excessively broad requirements. Therefore, this project will attempt developing a more deployable, fundamental Grid System designed to serve the requirements of existing legacy applications.

References

1. Arnold Bragg and Harry Perros, An Architectural Framework and Tool for Modeling Grids. MCNC Research and Development Institute.
2. Ian Foster and Carl Kesselman, "Computational Grids". Morgan-Kaufman, 1999.
3. Leon Erlanger, "Distributed Computing: An Introduction". PC Magazine, April 4, 2002.
4. SETI@Home <http://setiathome.ssl.berkeley.edu/>
5. Ian Foster, "What is the Grid? A Three Point Checklist". GRIDToday, July 20, 2002.
6. Ian Foster and Carl Kesselman, "Globus: A Metacomputing Infrastructure Toolkit". Intl J. Supercomputer Applications, 11(2):115-128, 1997.
7. Paul Stelling and Craig Lee, "A Fault Detection Service for Wide Area Distributed Computations". Proc. 7th IEEE Symposium on High Performance Distributed Computing, pp. 268-278, 1998.
8. Steven Fitzgerald et al. "A Directory Service for Configuring High-Performance Distributed Computations". Proc. 6th IEEE Symposium on High-Performance Distributed Computing, pp. 365-375, 1997.