IGI PUBLISHING **ITJ 5099**
701 E. Chocolate Avenue, Hershey PA 17033-1240, USA
Tel: 717/533-8845; Fax 717/533-8661; URL-http://www.igi-global.com

This paper appears in the publication, International Journal of Grid and High Performance Computing,Volume 1, Issue 3
edited by **Emmanuel Udoh** and **Frank Zhigang Wang** **© 2009, IGI Global**

Grid-Enabling Applications with JGRIM

Cristian Mateos, ISISTAN - UNCPBA, Argentina
Alejandro Zunino, ISISTAN - UNCPBA, Argentina
Marcelo Campo, ISISTAN - UNCPBA, Argentina

ABSTRACT

The development of massively distributed applications with enormous demands for computing power, memory, storage and bandwidth is now possible with the Grid. Despite these advances, building Grid applications is still very difficult. We present JGRIM, an approach to easily gridify Java applications by separating functional and Grid concerns in the application code, and report evaluations of its benefits with respect to related approaches. The results indicate that JGRIM simplifies the process of porting applications to the Grid, and the Grid code obtained from this process performs in a very competitive way compared to the code resulting from using similar tools. [Article copies are available for purchase from InfoSci-on-Demand.com]

Keywords: *Dependency Injection; Grid Computing; Gridification; Grid Programming Models; Grid Resources; JGRIM*

INTRODUCTION

The Grid (Foster and Kesselman, 2003) is a distributed computing environment in which resources from dispersed sites are virtualized through specialized services to provide applications with vast execution capabilities. Just like an electrical infrastructure, which spreads over cities to convey and deliver electricity, the Grid offers a computing infrastructure to which applications can be easily "plugged" and efficiently executed by leveraging resources of different administrative domains. Precisely, "Grid" comes from an analogy with the electrical grid, since applications will take advantage of Grid resources as easily as electricity is now consumed.

Unfortunately, this analogy does not completely hold yet since it is difficult to "gridify" an application without rewriting or modifying it. A major problem is that most Grid toolkits provide APIs for merely implementing applications from scratch

(Mateos et al., 2008a). Examples of such toolkits are JavaSymphony (Fahringer and Jugravu, 2005), Java CoG Kit (von Laszewski et al., 2003), GSBL (Bazinet et al., 2007), GAT (Allen et al., 2005) and MyCoG.NET (Paventhan et al., 2006). Hence, the application logic results mixed up with code for using Grid services, making maintainability, testing and portability to different Grid libraries and platforms somewhat hard. Furthermore, gridifying existing code requires to rewrite significant portions of it to use those APIs. These problems are partially addressed by tools that take an executable, along with user parameters (e.g. input arguments, CPU and memory requirements, etc.), and wrap the executable with a component that isolates the details of the Grid. Some tools falling in this category are GEMLCA (Delaittre et al., 2005), LGF (Baliś & Wegiel, 2008) and GridSAM (McGough et al., 2008). However, the output of these tools are coarse grained applications whose execution cannot be configured to make better use of Grid resources (e.g. parallelize and/or distribute individual application components). Overall, this represents a trade-off between ease of gridification versus flexibility to configure the runtime aspects of gridified applications (Mateos et al., 2008a).

To address these issues, we propose JGRIM, a novel method for porting Java applications onto service-oriented Grids, this is, based on Web Services. JGRIM minimizes the requirement of source code modification when gridifying Java applications, and provides simple mechanisms to effectively tune transformed applications. JGRIM follows a *two-step* gridification methodology, in which developers first implement and test the logic of their applications, and then Grid-enable them by undemandingly and non invasively injecting Grid services. Therefore, we conceive gridification as shaping the source code of an ordinary application according to few coding conventions, and then adding Grid concerns to it. In a previous paper (Mateos et al., 2008b), we reported preliminary comparisons between JGRIM and other approaches for gridifying software in terms of source code metrics. In this article we also report JGRIM execution performance on an Internet-based Grid, measuring execution time and network usage of two resource-intensive applications. The rest of the article analyzes the most relevant related works, describes JGRIM, and presents the experimental evaluations.

RELATED WORK

Motivated by the complex and challenging nature of porting conventional applications to the Grid (Gentzsch, 2009), research in tools and methods to easily gridify ordinary software is growing at an astonishingly rate. Besides providing APIs for developing and executing Grid applications, many of these tools actually materialize alternative approaches to support easy gridification of existing applications. For an exhaustive survey on technologies to port applications to the Grid, see (Mateos et al., 2008a). Below we describe a representative subset of such tools.

ProActive (Baduel et al., 2006) is a platform for parallel distributed computing that provides *technical services*, a support which allows users to address non-functional concerns (e.g. load balancing and fault tolerance) by plugging certain external configuration to the application code at deployment time. ProActive applications comprise one or more mobile