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

ITJ 5097

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**

Persistence and Communication State Transfer in an Asynchronous Pipe Mechanism

Philip Chan, Monash University, Australia

David Abramson, Monash University, Australia

ABSTRACT

Wide-area distributed systems offer new opportunities for executing large-scale scientific applications. On these systems, communication mechanisms have to deal with dynamic resource availability and the potential for resource and network failures. Connectivity losses can affect the execution of workflow applications, which require reliable data transport between components. We present the design and implementation of π-channels, an asynchronous and fault-tolerant pipe mechanism suitable for coupling workflow components. Fault-tolerant communication is made possible by persistence, through adaptive caching of pipe segments while providing direct data streaming. We present the distributed algorithm for implementing: (a) caching of pipe data segments; (b) asynchronous read operation; and (c) communication state transfer to handle dynamic process joins and leaves.[Article copies are available for purchase from InfoSci-on-Demand.com]

Keywords: *Asynchronous Operation; Decoupled Communication; Fault-Tolerant Communication; Persistence; Pipe Mechanism*

INTRODUCTION

Heterogeneous distributed systems are the emergent infrastructures for scientific computing. From peer-to-peer, volunteer computing systems to the more structured ensembles of scientific instruments, data repositories, clusters and supercomputers such as computational grids (Foster and Kesselman, 1999), these systems are heterogeneous and dynamic in availability. Furthermore, the wide-area links that interconnect these resources are prone to transient or permanent failures. These dynamic characteristics introduce unique challenges for executing large-scale scientific applications.

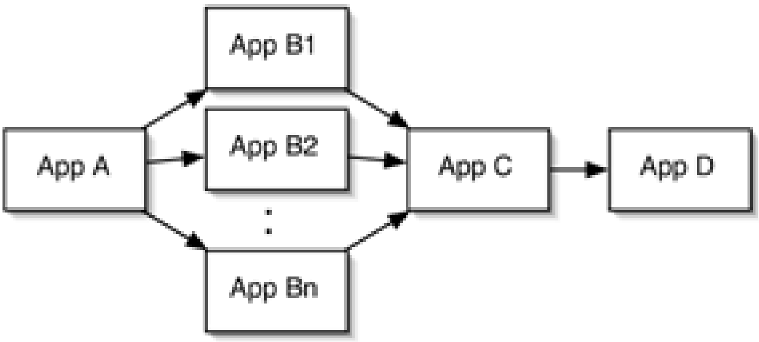
This research is motivated by the need to support fault-tolerant communication within scientific workflows. A workflow consists of multiple processing stages, where intermediate data generated in one stage are processed in subsequent stages. A workflow component can be a device or an application, which is often modified to enable communication. Thus, a scientific workflow is a computational/data-processing pipeline; with data being captured, processed and manipulated as it pass through various stages (Figure 1). Currently, the data transfers between component applications are realised by: (a) file transfers (e.g. GridFTP); (b) remote procedure calls (e.g. RPC-V, GridRPC, OmniRPC); and (c) custom mechanisms (e.g. Web Services).

For coupling workflow components, we propose the π-channel, an asynchronous and persistent pipe mechanism. It is part of the π-Spaces/π-channels programming model which features:

1. Simplified application coupling using string channel names through π-Spaces. A π-Space is a name space for π-channels, enabling dynamic binding of channel endpoints between processes.
2. π-channel data are adaptively cached to achieve persistence. This allows π-channels to be created and written to, even in the absence of the reader. Persistence also makes π-channels accessible even after the writer has terminated.
3. Asynchronous receives are made possible through a communication thread; thus, an application is able to accept pipe segments even when it is busy in computation.

This article focuses on how π-channel persistence relates to fault-tolerant communication in scientific workflows. The extended API and semantics for π-Space/π-channels are presented. We describe the

Figure 1. A simple four-stage workflow application. Arrows indicate data flow between component applications. Application B is an n-process parallel application.


App B1
App A
App B2
App C
App D
:
App Bn