**JavaPorts**: A Framework for the Rapid Prototyping of Distributed Applications on Clusters of Workstations

Demetris G. Galatopoullos (NU) and Elias S. Manolakos (NU)

**Goals**

The JavaPorts system is a programming environment, which promotes the development of modular, reusable parallel, and distributed component based applications for cluster computing. It aims at providing application developers with:

- The capability to easily configure reusable software components for the concurrent tasks of an application
- Anonymous message passing hiding the details of the communication and coordination
- Tools for the definition, assembly and reconfiguration of concurrent applications using pre-existing and/or new software components
- Seamless porting of single-node single-thread programs to multiple-node multiple-thread applications

**Significance**

The JavaPorts system exploits the advantages of workstation clusters which in conjunction with recent advances in networking emerged as cost-efficient alternatives to expensive supercomputers for compute-intensive problems. Among other important features the JavaPorts system underlines:

- Cluster computing with independence from the memory model (shared vs. distributed)
- Platform-independence by using the Java technology
- Object-oriented parallelism with modularity and reusability
- Separation of the coordination details from the computational aspect of the application.

The problems, which are suitable for the named system, are coarse-grain compute-intensive problems that exhibit low amount of task communications. One example of such problems is large-scale image processing algorithms.

**Technical Approach**

The software developer may define the mapping of a concurrent application to a cluster as a task graph (Figure 1).

![Figure 1: Task Graph](image1.png)

1. Define a task graph using the JavaPorts application configuration
2. User Controlled
3. Tasks assigned to Machines
4. "Plug-in" different algorithms
5. Dynamic Cluster Node Assignment
6. Visualization of Target Algorithm

A four-phase application design flow (Figure 2) allows the developer to transition from the task graph representation of the application to the execution of the software components on the specified cluster nodes.
The JavaPorts Application Configuration Language is composed of a set of simple and self-explanatory commands, which are sufficient to describe the mapping of the application onto the cluster. The parameters defined by the user in the configuration file will be used to generate or update a set of task templates. A sample of a configuration file is depicted in Figure 3 and it corresponds to the task graph shown in Figure 1.

![Figure 2: Application Design Flow](image)

**Figure 2: Application Design Flow**

The JavaPorts Application Configuration Language is composed of a set of simple and self-explanatory commands, which are sufficient to describe the mapping of the application onto the cluster. The parameters defined by the user in the configuration file will be used to generate or update a set of task templates. A sample of a configuration file is depicted in Figure 3 and it corresponds to the task graph shown in Figure 1.

![Figure 3: Configuration File](image)

**Figure 3: Configuration File**

The anonymous message passing is achieved through the PORT object, a software abstraction that serves as a mailbox to a task which may write to (or read from) it in order to exchange messages with other peer tasks. The Port interface supports:

- Asynchronous and Synchronous anonymous communication operations
- Execution monitoring and control
- Tasks and nodes location transparency

A task (component) developer can use the Port Interface to allow the task she codes for to communicate and exchange information with other tasks, without having to know whether the task and the other tasks connected to it are allocated to the same or to different machines.
Relation to ERC

- The JavaPorts system is designed to make painless the transition of an application from a single machine environment to a cluster of workstations platform. As outlined in R3c (Scalable Computation) section of the proposal, our system will be able to undertake the role of the sufficient programming tool for effectively migrating image-based applications from stand-alone PCs to a cluster machine, without the need of re-implementing the algorithm every time the cluster architecture changes. Moreover, the original application developer with minimal guidance will be able to perform this migration.

Current Status

- Currently the JavaPorts system version 1.0 has been released and it is being used internally for application development on a variety of cluster platforms (Sun/Solaris, PC/Linux, PC/Ultrix).

- The JavaPorts system version 1.1 is at its final stages of testing. With this version we are improving the ability to service large volumes of asynchronous message transfers, while providing the user with insight on the status of communication requests. We are also providing an improved Port interface with added commands for synchronous message passing.

Plans and Project Evolution

- Within year 2000 we are planning on releasing version 2.0 of the JavaPorts system also providing:
  - Transparent distributed termination of system and user initiated threads.
  - Thread scheduler for a variety of cluster platforms
  - Dynamic port creation and deletion
  - Graphical User Interface for the designing of the Task Graph
  - Tasks library (Java components) for sensing and imaging applications development

- As a longer-term vision within the next three years, the JavaPorts project will be directed towards addressing application level Quality Of Service (QoS) and code mobility issues

References


PI Contact Information

Elias S. Manolakos
Associate Professor
Northeastern University
ECE Dept. 422 Dana Build.
Phone: 617-3733021
Fax: 617-3734189
e-mail: elias@ece.neu.edu
URL: www.cdsp.neu.edu/info/faculty/manolakos/manolakos.html