Building Parallel Applications Using Remote Method Invocation

Mubarak Saif Mohsen, Abdussamad Haza'a
Sana'a Community College
Sana'a, Yemen
{mubarak_saih.hazaia}@yahoo.com

Zurinahni Zainol, Rosalina Abdul Salam
School of Computer Sciences
Universiti Sains Malaysia
11800 Penang, Malaysia
{zuri,rosalina}@cs.usm.my

Abstract - The Java programming language and environment is stimulating new research activities in many areas of computing, not the least of which is parallel computing. Java is an attractive environment for writing portable message passing parallel programs. While there is growing interest in using Java for high performance applications, this paper discusses several known efforts towards the design of Message Passing Interface (MPI) interfaces for Java. Efficient handling of these issues allows Java applications to obtain performance which rivals that of traditional native message passing systems. Finally, we develop our parallel application using Remote Method Invocation to illustrate how these concepts, design and performance of a RMI technique can be used to build parallel applications.

Keywords: Java, Remote Method Invocation, Message Passing.

1 Introduction

One way of handling large-scale problems effectively is by using parallel computers. A parallel computer consists of several processors that work on different parts of a common problem. Today, parallel computers consisting of hundreds of processors are commercially available. The basic idea behind parallel computation is to carry out several tasks simultaneously and thereby reduce execution time. A successful application of this idea requires, among other things, developing parallel algorithms that handle the task division, coordination, and communication among the processors. Parallel application design and development is a major area of interest in the domain of high performance scientific and industrial computing. In fact, parallel computing is becoming an integral part in several major application domains. With the advent of fast interconnecting networks of workstations and PCs, it is now becoming increasingly possible to develop high-performance parallel applications using the combined computing powers of these networked-resources, at no extra cost. Contrast this to the situation until the early 90s, where parallel computing was largely confined only to special-purpose parallel computers, each priced high enough to be affordable only by major research/academic institutions. Consequently, high-speed networks and fast general-purpose computers are aiding towards the mainstream adoption of parallel computing at a much more affordable cost.

The main point of the message-passing paradigm is that the processes communicate by sending messages to each other. Thus the message-passing model has no concept of a shared memory space or of processors accessing each other's memory directly. The message-passing paradigm has become increasingly popular in recent times. One reason for this is the wide number of platforms which can support a message-passing model. Programs written in a message-passing style can run on distributed or shared-memory multi-processors, networks of workstations, or even uni-processor systems. The acronym "SPMD" stands for single-program-multiple-data and refers to a restriction of the message-passing model which is:

- SPMD is actually a "high level" programming model that can be built upon any combination of the previously mentioned parallel programming models.
- A single program is executed by all tasks simultaneously.
- At any moment in time, tasks can be executing the same or different instructions within the same program.
- SPMD programs usually have the necessary logic programmed into them to allow different tasks to branch or conditionally execute only those parts of the program they are designed to execute. That is, tasks do not necessarily have to execute the entire program - perhaps only a portion of it.
- All tasks may use different data

Java Remote Method Invocation (RMI) mechanism [1] provides integrated and reliable communication between objects in a distributed system via remote method calls. It allows programmers to develop distributed Java programs with the same syntax and semantics used for non-distributed programs. The RMI prime goals are to provide source-code portability, to allow efficient implementation across a range of architectures, to a great deal of functionality and to support for heterogeneous parallel architectures.

The paper is organized as follows. Section 2 reviews previous work on message passing. Section 3 discusses RMI design for Message-passing programming model. Section 4 presents the implementation. Section 5 presents