JavaSeis: Web delivery of seismic processing services

*Siamak Hassanzadeh *, Sun Microsystems*
Charles C. Mosher; ARCO Exploration and Production Technology

Summary

The past year has seen a phenomenal rise in interest in Java. The Java programming environment and World Wide Web browsers with Java capability have added a new dimension to the ever-increasing popularity of the Internet and the Web. In this workstation presentation we briefly describe the main principles underlying the Java programming language, followed by an outline of the JavaSeis object model for parallel seismic processing. We then demonstrate the use of a Java-based browser interface to the ARCO Seismic Benchmark Suite, which demonstrates parallel algorithms for seismic data processing.

Introduction

The Java programming language and environment are designed to solve a number of problems in modern programming practice (Hassanzadeh and Mosher, 1996). Java is a "simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded, and dynamic" language (The Java Language: http://java.sun.com).

Java is object-oriented, providing facilities to define and manipulate objects. Objects are self-contained entities which have a state (a collection of data associated with an object) and to which messages can be sent. A message is a mechanism by which the state of an object can be accessed or altered. The object-oriented facilities of Java are essentially those of C++, with extensions from Objective C for more dynamic method resolution.

Java is designed to support applications that run on heterogeneous networked computer systems. To enable a Java applications to execute anywhere on the network, the compiler generates *bytecodes*- an architecture neutral object file format designed to transport code to multiple hardware and software platforms.

The *Java interpreter* can then execute bytecodes directly on any system that the interpreter has been ported. This neutrality also renders most software portable as applications written in the Java language can run on any system. The architecture-neutral and portability of Java has led to the development of the *Java Platform,* a "Write Once, Run Anywhere" capability for delivering and running applications on heterogeneous computing environments.

Java achieves high performance by translating bytecodes on the fly (at runtime) into machine codes. For applications requiring large amounts of compute power, the compute-intensive segments can be rewritten in native machine codes and interfaced with the Java environment. Java also supports multithreading; the capability to execute multiple concurrent sequences of instructions. Multithreading enables development of applications that take advantage of parallel computing capabilities, again in a platform independent fashion.

Finally, Java applications (or applets) can be fully integrated into Hypertext Markup Language (HTML) pages, which are then executed by Java-capable World Wide Web browsers.

Parallel Seismic Processing

A conventional seismic processing sequence operates as a

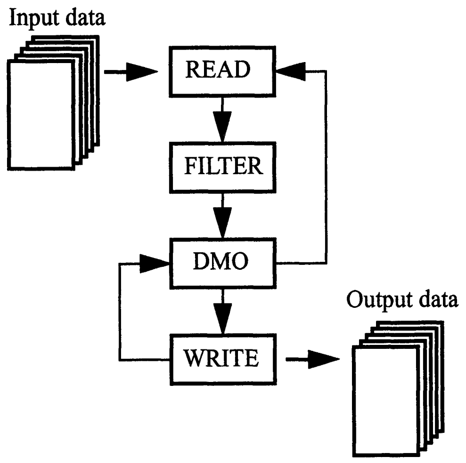

Figure 1: Example seismic processing data flow

pipeline (Figure 1). A block of seismic data is read one at a time from secondary storage devices, which is then passed through a chain of data processing routines. A final process writes the processed data back to secondary storage devices. Loops are used to create more complex flows. There are several ways to parallelize this processing sequence.

One common approach is the *master/slave* model, where one processor collects data for all other processors and assigns tasks to the slave processors as well. In this approach, the host or master controls the sequence of operations on the slaves, with all data transfers and interprocessor communications handled by the master. Another commonly used approach is the *fan-in/fan-out* method whereby parallel tasks are spawned by a single processor to other processors. Synchronizations and data transfers are handled by the control processor, as illustrated in Figure 2a.

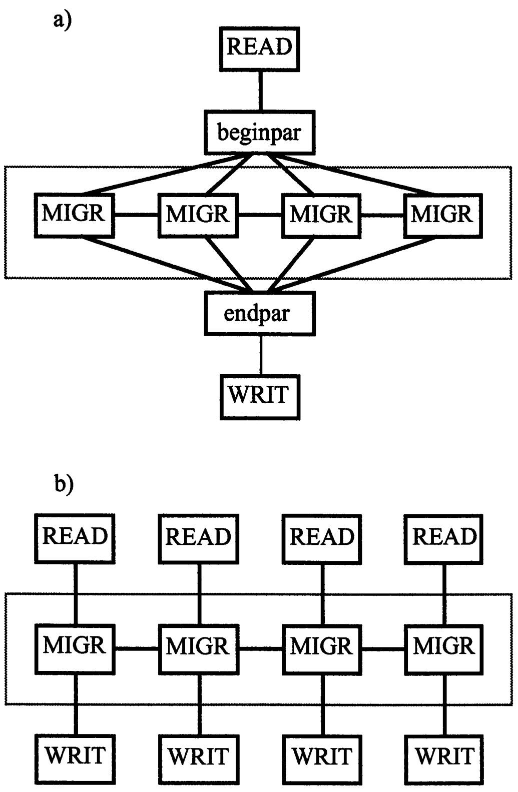

Figure 2: Parallel seismic processing models

The master/slave and fan-in/fan-out approaches, often referred to as examples of *control* parallelism, both introduce I/O bottlenecks that can limit scalability to a handful of processors, although the underlying model is very simple and puts few demands on the programmer. This approach is also limited to those cases where little or no communication is required between processors.

Seismic data, which can have as many as 5 dimensions in prestack form (two source coordinates, two receiver coordinates, and one time ordinate) offers may opportunities for *data* parallelism. In this model, parallelism is described in terms of the number of independent (or nearly independent) pieces that the data can be divided into.

Simple (embarrassingly) parallel behavior is characterized by completely independent operations that can be applied to different blocks of data. For example, if the entire data set needs to be filtered, each processor can filter a group of traces (i.e. a shot or CMP gather) independently with no communication between processors. This model of computation is also referred to as "owner computes", since each processor performs computations only on the data stored on that processor. If parallel I/O is also available, data can be read and written by all processors at once, as illustrated in Figure 2b. Given I/O resources that scale with computational resources, great speedup can be achieved using this model.

For more complicated processing algorithms, data parallelism can be extended by message passing or shared memory, which allows critical data to be accessed by large numbers of parallel tasks.

JavaSeis: An Object Model for Parallel Seismic Processing

Clearly, seismic data processing is ideally suited for parallel distributed computing. However, thus far there has not been a simple programming environment to provide a portable, distributed and parallel framework for scientific computation in general, and seismic data processing in particular. Previous attempts at developing such a framework have been based on C++ and message passing libraries.

As the basis for the JavaSeis efforts, we plan to use the ARCO Seismic Benchmark Suite (ASBS), which has recently been incorporated into the SPEChpc96 benchmark suite as SPECseis96 (Mosher, et al, 1992; Mosher, Joyner, and Hassanzadeh, 1996). An executive manages seismic data as parallel distributed objects using traditional procedural languages (C and Fortran). The environment is designed to be flexible and extendable, and is used extensively at ARCO as a prototyping environment. New processes are developed using "inheritance by copying templates". As a result, system changes often require extensive modifications to all copies of a few basic system templates. Parallelism is managed through an API defined by the services required for implementing common geophysical data processing algorithms.

The existing object-based structure of ASBS and the well-defined parallel behavior of many geophysical algorithms makes the this application suite a natural fit with the constructs defined in the Java language. Currently, maintenance and portability of applications in ASBS are made more difficult by the lack of a coherent object model and a consistent parallel programming model. Implementation of ASBS in Java would significantly improve the portability and maintainability of seismic computing applications, and would also provide a pathway to use of network based parallel computing for production seismic processing.

Demonstration

For this demonstration, we show the use of Java applets operating in standard World Wide Web browsers for setting seismic processing parameters, managing seismic processing jobs on distributed computing networks, and viewing the results. In addition, we show how Java application code can be used to efficiently implement the seismic executive from ASBS on parallel computing systems.

Acknowledgments

The authors would like to thank ARCO and Sun for their support of joint, cooperative projects in this area.

References

Mosher, C.C., S. Hassanzadeh, J. J. Chen, M.A. Archin, G.F. Lou, and D. Schneider, 1992, A benchmark suite for parallel seismic processing, SEG Extended Abstracts, pp. 360-362.

Mosher, C.C., Joyner, C.L., and Hassanzadeh, S., 1996, Scalable parallel seismic processing, Leading Edge of Geophysics, Dec. 1996.

Hassanzadeh, S., and Mosher, C.C., 1996, Java: An object oriented programming language for the cyber age, Leading Edge of Geophysics, Dec. 1996.