A JavaApplet to Visualize Algorithms on Reconfigurable Mesh

Kensuke Miyashita and Reiji Hashimoto

Okayama University of Science, Ridaicho, Okayama 700-0005, Japan
{miyasita, reiji}@ees.ouc.ac.jp

Abstract. Recently, many efficient parallel algorithms on the reconfigurable mesh have been developed. However, it is not easy to understand the behavior of a reconfigurable mesh. This is mainly because the bus topology can change dynamically during the execution of an algorithm. In this work, we have developed JRMS, a Java applet for visualizing parallel algorithm on the reconfigurable mesh to help in understanding and evaluating it. This applet accepts an algorithm written in C-like language and displays the behavior of it graphically. It also displays much information to evaluate the exact performance of an algorithm. Because this software has been developed as a Java applet, it can run on any operating system with standard web browser.

1 Introduction

A reconfigurable mesh (RM) is one of models of parallel computation based on reconfigurable bus system and mesh of processors. Many results and efficient algorithms on the RM are known, such as sorting, graph related problem, computational geometry and arithmetic computation[3].

We make use of the channel of data flow (bus topology) to solve problems on an RM, but it is not easy to follow data since the bus topology changes dynamically during the execution of algorithm on RM. For same reason, it is also hard to find collisions of data on bus. Therefore it is very hard to design, analysis or understand the algorithms on an RM by hand.

Algorithm visualization is very useful to understand the behavior of computational model. It also assists the user to design, analysis and debug the algorithms. Normally, the user can specify the data and control the execution of algorithm via some user interfaces.

As far as we know, there are three tools to visualize algorithms on RM, the Simulator presented by Steckel et al.[6], the Visualization System presented by us[3] and the VMesh by Bordim et al.[1].

The first one displays the behavior of algorithm graphically and the results of simulation are stored in a file. However the algorithm to be visualize in this simulator must be written in assembler language. Hence a sophisticated algorithm is implemented as a large file and there is more probability of existence of miss-codings or bugs. In addition, it cannot display the data that each processor holds neither statistic information.

© Springer-Verlag Berlin Heidelberg 2000
The second one is written in Java and accepts an algorithm written in C-like language. It displays the behavior of algorithm graphically with the source code of algorithm in another window. However it can visualize only a constant-time algorithm (the algorithm without loops), and it runs very slowly.

The third one is written in C language and use X Window System as graphical interface so that it can run on the operating system with X Window System. It accepts an algorithm written in C-like language and displays the behavior of algorithm graphically with the source code of it in another window. An algorithm written in C-like language is translated into a real C source file and is compiled and linked with the VMesh objects. Thus it provides flexible programming interface. However the user has to maintain all source files of the VMesh, libraries used by it, C compiler and linker to recompile it.

In this paper, we present the JRM, a Java applet to visualize the behavior of algorithm on an RM. An applet can run in a standard web browser with Java Virtual Machine (JVM) without recompiling. Thus the JRM can run on Windows, Mac OS and many other UNIX-like operating systems. The JRM can visualize the behavior of algorithm and the data that each processor holds. It can also report bus length, the number of inner-bus configurations and some statistic information. Another important feature is that the JRM accepts an algorithm file specified by the URL (Uniform Resource Locator). Thus it can read an algorithm from anywhere in the Internet.

2 Reconfigurable Mesh

A reconfigurable mesh (RM) is formalized as follows. An RM of size $m \times n$ has $m$ rows and $n$ columns, and there are $mn$ SMD processors located in a 2-dimensional grid. A processor located at the intersection between $i$th row and $j$th column is denoted by $\text{PE}(i, j)$ $(0 \leq i < m, 0 \leq j < n)$. Each processor has its own local memory, while no shared memory is available.

A static (outer) bus connects 2 adjacent processors. An outer bus is attached to a processor via a port. Each processor has 4 ports denoted by N (North), S (South), E (East) and W (West). Each port can be connected by reconfigurable (inner) buses with any possible configurations. A connected component made of outer buses and inner buses is called a sub-bus.

All processors work synchronously and a single step of an RM consists of the following 4 phases: **Phase 1:** change the configuration of inner buses, **Phase 2:** send data to a port (the data which is sent at this phase is transferred through a sub-bus), **Phase 3:** receive data from a port (all processors connected to same sub-bus can receive same data sent at the previous phase), **Phase 4:** execute constant-time local computations. The configuration of inner buses is fixed from phase 2 till phase 4.

In this paper, we employ the CREW (Concurrent Read, Exclusive Write) model of bus. That is, at most one processor connected to a sub-bus is allowed to send data at any given time. When more than one processor send data along same sub-bus, a collision occurs and the data is discarded.