

NETWORK OF BROWSERS – A MULTI-PROCESSOR COMPUTER

Luke Fletcher and Vishv Malhotra
School of Computing, Private Box 100
University of Tasmania, Hobart Tas 7001 Australia
{luke.fletcher, vishv.malhotra} @utas.edu.au

Abstract

The paper describes an experimental system in which we linked together a number of computers over the Internet to form a multi-processor computer system. The arrangement uses Java-enabled web-browsers as the tool for adding available computers in the multi-processor system. The number of processors in this system can grow and shrink dynamically as the computers join or leave the system.

Key Words

Multi-processor computer, Internet, Java, Web server, HTTP.

1 Introduction

As computational power becomes cheaper and more pervasive, we continue to seek even more computational power at even lower costs for a variety of new and innovative applications ranging from important economical, scientific and life-saving medical experiments to recreational computation [1]. Notwithstanding the unfulfilled demand for computation, at any point in time we have hundreds of thousands of computers around the world that are under-utilised. A typical computer is used for only a small fraction of the time duration over which it could be used.

The under-utilisation of the available computational power results from many reasons. Foremost of these reasons is our inability to locate free computers. Security concerns are also significant deterrence in this regard. Owners of the computers are reluctant to hand controls of their computers over to strangers who need these computational resources. At the same time clients/users of these facilities, if they can lease them, need to feel secure before they transfer their sensitive data and programs to a computer environment that an anonymous benefactor may have made available.

The paper describes a network of browsers (NoB) that combines the available free computers into a huge multi-processor computer system. In deed, it allows the

individual computers to join the multi-processor computer system when they are available. At the same time when the owners of these computers want to use their computers for their own work, they withdraw their computers from the arrangement. Thus, the multi-processor computer system continues to grow and shrink as free computers join in to become its processing units and the busy computers leave and cease to be its processing units. The owners of the participating computers decide if, when and for how long they donate time on their computers to the multi-processor system.

The network of browsers (NoB) uses the ubiquitous internet technology – the humble web browser – to match the demand for computation power against the available computational resources. Further, the existing and future security measures in the Internet domain provide and will continue to improve and enhance the trust environment for those who make available their computational resources to the others. At the same time, those who make use of these resources will be assured about the protection of their data and software.

An experimental prototype of a network of browsers (NoB) was developed in the School of Computing, University of Tasmania. The paper describes the system and presents initial experiences and analysis from this experiment in running a simple concurrent program to solve a crossword problem.

In section 2, we briefly summarise the aspects of Internet technology that have bearing on the network of browsers (NoB) developed in this university. Section 3 describes the system. Section 4 provides a brief description of the crossword problem that we used to test the system. Experimental performance results are also presented in section 4. We conclude the paper in section 5 by making some concluding remarks and listing the directions we wish to work on to make the system more usable.

2 Background: Web Browsers, Web Servers and Java

The Internet is comprised of three kinds of components: browsers, servers and the network.

The network provides the communication media that allows any browser to interact with any server. The predominant protocol for interaction between a browser and server is through Hypertext Transfer Protocol (http) [10]. The protocol is memory-less and comprises of two steps. A browser sends a request for an html document to a specific server. The server responds to the request by sending the requested document. The response concludes the interaction and the browser and the server are free for their next intersection with the other players on the World-wide Web (WWW).

The browsers are usually enhanced, with a Java plug-in, to run Java byte-codes called applets [2, 3, 4]. The plug-in executes Java applets received from the servers. At the same time, modern servers too can be extended by means of Java servlets and other server-side extensions, for example, CGI. A servlet is Java byte-code running alongside a server. The server passes certain http requests to a designated servlet when it receives them from a browser. Thus, these requests are handled by a specially tailored program (servlet). The servlet-browser interaction may use a protocol that maintains state over a sequence of request-response interactions (transaction) between them.

Another ingredient of interest in the construction of the networks of the browsers (NoB) is the Java programming language and associated Java virtual machine. Java programs are translated into Java byte-code that can be run on interpreters called Java Virtual Machines (JVM). This enables the java programs to run independent of the underlying hardware and virtually on any computer. Java plug-in for a browser is a JVM capable of running with the browser. Thus a Java enabled browser can run an applet when it receives it from a server in response to a request. Traditionally, these applets are intended to provide sophisticated and powerful application-specific interactive interfaces to the browser users.

In addition to the three components that provide obvious functionality to our system, we are also interested in the Java2 security model [5]. It provides fine-grained security guarantees for distributed applications in an environment where objects and code migrate over the network. Though the model is well integrated into the Standard Java Development Kit (SDK), we are not yet aware of a plug-in that implements the fine-grained security model of Java2. For the present we can only rely on traditional sand-pit security model and its extension to signed applets. In this model the imported code (applets) run in the browser host with a small set of rights considered safe. Browsers such as Internet Explorer and Netscape allow for the users to vary some rights of the external code to suit their needs. As opposed to this scheme, also known as Java 1.0 security model, Java 1.1 uses a model in which

appropriately signed applets are fully trusted. We are confident that future developments in Java will implement Java2 security model. Java2 security model allows for rights of the classes to be defined independent of other classes. A fine grained security policy will be useful to individually set the rights of the applets and also of the other objects on the applets. Thus, enabling the network of browsers (NoB) to be more secure and providing greater confidence and security assurances to those loaning their computers as processing units of the multi-processor computer system and to those using it to run their applications.

3 Network of Browsers (NoB)

A prototype network of browsers (NoB) was built as an honours project [6] and preliminary performance results were obtained using a simple crossword problem. As shown in Figure 1, the three main actors in the prototype system are:

- i. *Client*: A client is a user (or a computer) who is interested in execution of a suitably devised program on the prototypical network of browsers (NoB) computer.
- ii. *Server*: The server is a well advertised site that acts as a broker between the clients and the donors of computer times.
- iii. *Donors*: The donors are owners of computers on the internet who volunteer to make their computers available for executing clients' programs. In what follows we often use the word donor to mean the donor's computer.

A client contacts a broking server to load their program components, called tasks, on the server system. The client also provides data and establishes execution dependencies between the tasks. A sever receives the tasks from a client and then waits for donor computers to contact it to volunteer to execute the tasks.

A donor volunteers to execute tasks by starting a Java-enabled web-browser and accessing a specified web page on the server. The server returns an html page with suitably coded applet to the browser. All further interaction with the server for executing the tasks on the browser's Java plug-in is a responsibility of this applet.

A donor will be able to reclaim the computer by simply closing the browser. The same effect is achieved if the browser is moved to a different web page as it stops the controlling applet.

