JDOS: A Jini Based Distributed Operating System

M. Saywell and J.S. Reeve*
Department of Electronics and Computer Science
University of Southampton
Southampton SO17 1BJ, UK
email jsr@ecs.soton.ac.uk

September 2, 2002

Abstract

J-DOS provides and integrated JAVA environment for the execution of a program across an interconnected network of heterogeneous computers. The system provides a file system, shared memory and a distributed execution scheme, all of which is transparent to the user. The framework used to provide these service is sufficiently general as to allow the provision of extra services by the user. We describe the client-server execution, remote execution and the shared file system, paying particulat attention to the techniques used to distribute threads over many nodes. Distributed Mandelbrot set generation and rendering is used to benchmark and validate the the remote execution and load balancing aspects of the system.

1 Introduction

With the wide scale deployment of fast computer interconnection technology, for instance, 100Mbit and Gigabit ethernet, the clustering of computers is

*Corresponding Author

becoming more usual. The purpose of a cluster is to present a number of physically distinct machines as a single networked virtual computer[1]. However such systems are typically only distributed at the processing level, requiring independently configured third party software to provide further distributed fiunctionality. For example NFS[4] is commonly used to share file systems and X11[2] provides support for remote GUIs. Our purpose is to provide the essential distributed services of remote processing, shared memory and a common storage medium and offer a framework which is readily extensible so that supplementary services can be added with minimal programming overhead. In contrast to high performance computing environments like *Beowulf*[3] which usually provide The Message Passing Interface[6], J-DOS provides the ability to adapt and scale in an un-managed way. Additionally the system doesn't attempt to disguise its distrubted nature and instead allows the programmer to distribute processes when it is most suitable to do so.

J-DOS provides a dynamic extensible distributed computing environment in 100% pure Java. The system includes a file system, shared memory and a distributed processing environment. Provision is made for extra services to be added by the user without detailed knowledge of the system or the underlying Jini and RMI behaviour. The J-DOS environment is "zero configuratio", in the sense that the introduction of a new service is made simply by running it to trigger the automatic registration registration of the service which then becomes available to clients. Finally, J-DOS provides classes which automate the thread distribution and collection processes.

2 Description

Remote Method Invocation RMI[7] is a feature of Java which allows a JVM running on one host to invoke methods on another host, arguments and return values are passed across the network. RMI is essential as it is the only way provided that allows distinct instances of a Java Virtual Machine running on separate hostes to communicate but it requires remotely invokable methods to be pre-declared and stubs generated. The programmer must also deal know the location of the remote service. The Jini[5] set of classes, built on top of RMI, however does provide a client server architecture in which the client can locate and utilise service by name reference. When a client connects to a Jini enabled network it will typically send out a multicast request asking for all look-up servers to reply with their network address.

A look-up server maintains a registry of all currently available services. As part of the registration process the server must provide a *proxy*. This is held on the look-up server and downloaded to clients when they request the corresponding service. The client uses the proxy to talk directly to the server which provides the Java byte code necessary for the client to control and use the service. The client needs to know the method signatures of the proxy in advance as it uses them to filter the available services, as reported by the look-up server.

3 Conclusions and Future Delelopments

The system is currently being developed by adding a remote GUI interface so that graphical applications could be displayed on one node and the J-DOS interface on another. Inter-thread communication is also being developed so that programmers are not restricted to the client server model. This would allow parallel programs with any logical communication patterns to be run under the system. The file system, as so far developed, is not distributed and woul benefit from being so by allowing more overall storage capacity and allowing data searches to be distributed over all available nodes. Redundancy could also be built into such a file system making it more robust.

Overall J-DOS provides a dynamic extensible distributed computing environment in Java, with well defined interfaces by which programmers can write and distribute there own programs, without detailed knowledge of the cluster and using either client server or shared memory communication or both.

References

[1] C.Catlett and L.Smarr. Metacomputing. *Communications of the ACM*, 35:44–52, 1992.

[2] D.Young. *X Window system programming and applications with XT*. Prentice Hall, 1990.

[3] T.Sterling nad G.Bell and J.Kowalik. *Beowulf Cluster Computing with Linux*. MIT Press, 2001.

[4] R.Sandberg. The SUN network file system: Design, implementation and experience. Technical report, SUN Microsystems, Inc., 1985.

[5] S.Oaks and H.Wong. *Jini in a nutshell*. O'reilly, 2000.

[6] W.Gropp, M.Snir, B.Nitzberg, and E.Lusk. *MPI: The Complete Reference*. MIT Press, 1998.

[7] W.Grosso. *Java RMI*. O'reilly, 2002.