Co-operating Mobile Agents for Distributed Parallel Processing

R Ghanea-Hercock
BT Laboratories, Intelligent Systems Research, Martlesham Heath, Ipswich, IP5 3RE, UK.
ghanear@info.bt.co.uk

J C Collis
BT Laboratories, Intelligent Systems Research, Martlesham Heath, Ipswich, IP5 3RE, UK.
jaron@info.bt.co.uk

D T Ndumu
BT Laboratories, Intelligent Systems Research, Martlesham Heath, Ipswich, IP5 3RE, UK.
ndumudt@info.bt.co.uk

ABSTRACT

This paper describes a novel approach to dynamic distributed parallel processing using a mobile agent-based infrastructure. Our goal is to extend the concept of the Parallel Virtual Machine architecture [8] by using a combination of collaborative and mobile software agents to enable automatic and dynamic configuration of distributed processes.

Our approach to distributed processing is regulated by a two-tier management system. At the strategic level, an anchored centralised agent is responsible for managing user interaction and determining how tasks are to be distributed. Whilst the mobile agents who deliver code to remote machines, manage local processing at an operational level. We therefore developed specialised mobile agents, each performing particular roles, which co-operate as a *team* to achieve user defined goals. The resulting system provides users with a much simpler means of utilising the power of distributed parallel computing.

Keywords: Mobile Agents, Parallel Processing

1. INTRODUCTION

The potential power of networked computation resources has become apparent with the emergence of very large Intranets with high bandwidth links. With fast Pentium class desktop machines now ubiquitous, the total memory and process potential available within a single Intranet can be immense. PVM and similar systems [3,8] attempt to harness this potential, providing a means of combining distributed computing resources into a single virtual computer, usually termed a distributed virtual machine (DVM). Ideally the complexities of how, where and why the processes are running should be hidden behind a high-level interface.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
Autonomous Agents '99 Seattle WA USA
Copyright ACM 1999 1-58113-066-x/99/05...$5.00

1.1 The Team Agent Approach

The remainder of this paper describes a mobile agent based DVM that involves several agent roles, with different functions and skills.

As these agents behave co-operatively as a team our system is called **MATS**: the Mobile Agent Team system. One of the design objectives of MATS was to minimise the code that would need to be moved about the network. This was achieved by localising the most complex functionality in an anchored (non-mobile) agent, which can then communicate with the simpler specialised mobile agents through a message passing mechanism. In effect, this is a marriage of the autonomous and social features of 'collaborative' agents with the facilities offered by mobile agent technology.

The resulting mobile agents are lightweight, meaning they require less processor time to be serialised, and are quicker to transmit. This has an important implication for the viability of parallel processing. If the agents are large and cost a significant amount of machine resources to transmit, then the benefits of solving the problem in parallel may be lost.

MATS is implemented in the Java programming language, using the Voyager mobile agent platform from ObjectSpace Inc. [7]. Voyager was chosen because it provides a well-developed mobile agent class library, that is also stable and well documented. The Voyager package uses the 1.1 release of the Java language, as it supports object serialisation.

2.0 MOBILE AGENT TEAMS

To minimise the complexity of our mobile agents, we have differentiated them according to particular roles. An analogy can be drawn with social insects, which often contain multiple forms within one species, each performing a particular supporting role in the society

2.1The Hive Agent

The Hive is a fixed agent and to begin a distributed computation session the user sends the task code to be processed to the Hive, and specifies the parameter ranges that will be used. The Hive must then decompose the problem into the optimum level of task granularity. Each component process is then 'wrapped'

i.e. associated with the necessary code for mobility, messaging, and distribution. This code is termed a Queen agent.

2.2 Resource Assessment and Modelling

Scout agents are periodically created by the Hive and dispatched to hosts around the network. On arriving at a remote host Scouts perform tests on the local resources and analyse what hardware and software is present.

Users interact with the MATS system through the Hive's visual front end. A key feature of this interface is its ability to visualise the network and geographic location of the mobile agents. Visualising mobile agents is inherently difficult as they move between sites and so must be continuously tracked. Furthermore, unlike conventional code, mobile agents are not always available for analysis and interaction, and may even have moved to a site that is not continuously connected to the network. Once a distributed computation has been started it is the Hive's duty to monitor its progress, collect and collate results, and if necessary send new instructions to the mobile agents.

2.3 Queen Agents

A Queen is a mobile agent that is responsible for solving part of the larger problem. The Queen will then become active and move to the first site on its itinerary. On arrival the Queen analyses the local conditions and launches as many 'worker' threads as the local machine can comfortably support, the worker threads then begin to solve the task in question.

2.4 Worker Threads

The worker threads are not mobile agents, but processes created by the Queen to perform part of some larger computation. Each thread is an instantiation of the user-supplied problem task code given to the Hive at the start of the process.

3.0 RESULTS

MATS has been tested by using a standard benchmark Genetic Programming (GP) problem, i.e. symbolic regression of a curve [4]. Copies of the GP problem were assigned to several Queen agents, which were distributed across the computing resources of a local area network, (comprising several UltraSparcs and dual processor PCs running Windows NT). The system therefore facilitated the construction of a complex demetic exchange process, in which fit chromosomes can migrate between several sub-populations. The effect being to provide a far larger genetic search space than available on a single machine, while allowing for asynchronous connection between each computer. The preliminary results of using MATS to distribute and solve an intensive processing task, (such as this genetic programming computation), indicate that a significant percentage of the total distributed CPU and memory resources can be utilised.

4.0 ANALYSIS

We can now consider which specific attributes of mobile agents are most beneficial to the DVM application. Of the advantages of mobile agents advocated by Lange in [5], several would seem to offer the facilities needed by DVM systems. The asynchronous and autonomous nature of mobile agents provides a fault tolerant and robust system. Tests demonstrated that failure of a single host or even temporary failure of the Hive agent did not seriously disrupt the system, as each remaining Queen agent continues to operate autonomously until contact with the Hive agent is re-established.

5. CONCLUSIONS

This work offers a novel system for distributed parallel processing, operating over heterogeneous environments, with dynamic reconfiguration of the management system. This has been motivated by our desire to simplify the configuration and usage of a distributed virtual machine and to use the principle of agent co-operation to improve a mobile agent application. We hope that this work has demonstrated that an agent-based solution provides a viable, and often better, alternative to current static, non-mobile DVMs.

Whilst mobile agents [1,2] would seem to be an obvious means for distributing and performing parallel computations, the extra layer of abstraction imposes many overheads. Consequently we developed specialised entities, each performing particular roles, which act together as a *team* to achieve user defined goals. By using a two stage management system we have incorporated strategic central planning and operational management, and there is still considerable scope for making the system even smarter and even safer.

6.0 REFERENCES

[1] Clements P. E., Papaioannou T., & Edwards J.M., "Aglets: Enabling the Virtual Enterprise", pub. in MESELA '97.

[2] Harrison G. C., Chess D. M., & Kershenbaum A., "Mobile Agents: Are they a good idea?", IBM Internal Research Report, T. J. Watson Research Center, 1995.

[3] Geist A., et al. "PVM: Parallel Virtual Machine A Users' Guide and Tutorial for Networked Parallel Computing", MIT Press Scientific and Engineering Computation, Janusz Kowalik, Editor, Massachusetts Institute of Technology, 1994.

[4] Koza J., "Genetic Programming: On the Programming of Computers by Means of Natural Selection." (book) Publisher MIT Press, 1992.

[5] Lange D., "Mobile Agents: Environments, Technologies, and Applications", Proceedings of PAAM'98, March 1998, pp11-14.

[7] "ObjectSpace Voyager and Agent Platforms Comparison", public white-paper from http://www.objectspace.com/Voyager/voyager.html.

[8] Sunderam V.S., "PVM: A Framework for Parallel Distributed Computing", V. S. Concurrency: Practice and Experience, 2, 4, pp 315—339, December 1990. Available from: http://www.netlib.org/ncwn/pvmsystem.ps