GRID COMPUTATION VIA JAVASPACES 
OF THE FIRST ORDER LEAST SQUARES FEM 

Andrew Hellicar, John Kot, Francis Cooray, Howard Lovatt

CSIRO ICT Centre
Cnr Vimiera & Pembroke Rds, Marsfield, NSW 2122, Australia.
andrew.hellicar@csiro.au

Abstract

A distributed finite element method (FEM) solver using Sun’s JavaSpaces technology was implemented and tested on a heterogeneous mix of computers including laptops, desktops and SMPs running MS Windows, Linux and Unix operating systems. Test problems in 2 and 3 dimensions were solved using a distributed iterative conjugate gradient method.

I. INTRODUCTION

In recent times the usage of Grid Computing has exploded, from being a niche activity pursued mainly within the computationally intensive life sciences area, to enjoying widespread implementation across all fields requiring high performance computation, such as computational electromagnetism [1, 2].

The future direction of Grid Computing, including the hardware and application structure, is an open question. The structure of grid applications range across many flavours including implementations of MPI, space based, data driven [3] and batch job [2] techniques. The choices of hardware setup range from specialised clusters running optimised code with web service interfaces, to heterogeneous mixes of platforms and operating systems offering compute power to any user. Heterogeneity itself introduces problems of code portability. One of largest grid projects, Globus [4], has in recent times moved completely to a Java framework to enable robust installation on the wide range of platforms in existence today and tomorrow.

Here we are interested in portable code, distributed upon a heterogeneous dynamic multiuser environment. We have implemented a grid application structure based upon the space based flavour, specifically JavaSpaces [5], with all code written in Java. We discuss its implementation, benefits, and application to a trial problem.

II. JAVASPACES

JavaSpaces is a space based technique. Space based techniques involve a central space where tasks may be deposited. A range of ‘worker’ processes check the space whenever they are idle for tasks requiring evaluation. Any available task is taken and computed, the generated results are deposited back into the space. This ensures load balancing, as the faster machines complete more tasks.

Once the described setup is implemented, all that is required for a ‘master’ to compute a problem is to split the problem into smaller tasks, deposit the tasks into the space and wait for the results to appear (see Figure 1).

One benefit of Java is that the tasks are objects and contain both data and the code for processing the data; therefore, masters are able to generate new applications without preinstalling code on every machine on the grid. When this is coupled with Java’s bytecode machine independence, the simplicity of JavaSpaces is realised. Workers may join and leave the grid while the system is running, and new applications may easily be developed and rolled out.
III. TEST PROBLEM

A test problem was implemented to examine the ability of JavaSpaces to solve problems in computational electromagnetism. The system of first-order PDEs:

$$ \nabla \vec{F} = 0 \quad (1) $$

was implemented in both 2D (Cauchy-Riemann system) and 3D (Clifford Monogenic system) [6] and solved by the least squares FEM.

Serendipity quadrilateral and hexahedral elements [7] were implemented to mesh the region of interest and a least-squares FEM was implemented using a Jacobi-preconditioned element-by-element conjugate gradient (CG) solver. Boundary conditions specified by the master were enforced within each CG iteration.
IV. IMPLEMENTATION

The user interface was implemented via a webpage, whereby the master runs within an applet on the user’s computer. The applet displayed the field at each iteration through a slice of the problem domain, see Figure 2. The CG routine consists of matrix-vector products and other operations. Only the matrix-vector products were distributed onto the JavaSpace, as the computation work, relative to the data to be communicated, was negligible for the other CG operations. Results for the size of CG residual are plotted in Figure 3.

![Figure 3: Convergence of the CG method against iterations.](image)

V. CONCLUSIONS

We implemented and tested the JavaSpaces paradigm for a distributed FEM solver in 2 and 3 dimensions. The solver was successfully distributed across a heterogeneous test platform comprising SMPs, laptops and desktops running Unix, MS Windows and Linux.

References


