OPTIMIZING DATA LOCALITY 
FOR THE EFFICIENT SOLUTION OF MULTIPHYSICS PROBLEMS ON SYSTEMS WITH THOUSANDS OF MULTICORE PROCESSORS

Dr Lee Margetts, University of Manchester, United Kingdom  
Professor Mariano Vazquez, Barcelona Supercomputing Center, Spain  
lee.margetts@manchester.ac.uk  
mariano.vazquez@bsc.es

Overview

This paper presents a simple optimisation strategy for solving coupled finite element problems efficiently on systems with thousands of multicore processors. It is a strategy that perhaps would not be obvious to a software engineer unfamiliar with the finite element method as it involves taking advantage of the mathematical structure of the finite element stiffness matrix. The paper also contrasts how data is localized in the authors’ tightly coupled approach with data locality for loosely coupled multiphysics simulations. In the conference presentation, performance figures for optimized and unoptimized code will be compared. The study has been carried out using a Cray XT4 system with 5668 quad core processors. Future work will involve evaluating this strategy on “many-core” processors with higher core counts.

A Brief History of Engineering Software

In the 1960s, numerical techniques such as the finite element method were developed to analyse engineering structures such as aircraft frames. The constitutive models used were for predicting simple elastic stress-strain behaviour. In the following decades, the finite element method followed a developmental roadmap that roughly matched the increasing power of computers, with analyses becoming more physically realistic. Various software applications, originally developed in the US laboratories and European universities, were commercialized in the 1970s and 1980s. These were typically domain specific, with separate programs for implicit structural analysis (stress), explicit structural analysis (crash, impact) and fluid dynamics [1]. In the 1990s, computing had reached a stage where the next natural step was for engineers to be able to model the interaction between different physical processes.

Early Attempts at Multiphysics

The most well known example of coupled physical processes is fluid-structure interaction. Attempts to perform this type of analysis triggered considerable effort in getting the prevailing commercial packages, separate ones for fluids and structures, to talk to one another. A coupled fluid-structure interaction problem would involve the following iterative process [2]:

1. Solve the structural problem on a finite element mesh  
2. Store the results of the structural problem in a database  
3. Convert the results into a format compatible with the fluid solver  
4. Solve the fluid problem on a finite volume grid  
5. Store the results of the fluid problem in a database  
6. Convert the results into a format compatible with the structural solver  
7. Return to (1) until converged
As the activities of engineers changed, the largest of the structural engineering software vendors each acquired a fluid mechanics firm in order to provide their customers with an integrated solution. In the last ten years, computer scientists have developed the concept of “code coupling”, where different applications are wrapped and put together as a meta-program [3].

**Data Locality on Single Processor Workstations**

In the previous example, the data that couples together the two physical processes resides on disk. When the computation switches (or ping-pongs) from the fluid to the solid part, the processor has to wait while data is written to disk, interpolated into a different representation of the problem domain and read from disk again. For the majority of engineering problems, this strategy is obviously not very efficient. That said, if one iteration of the coupled process takes days to complete (as it might do on a single processor workstation), then the overhead of writing to disk is perhaps insignificant and the strategy reasonable.

**Data Locality on Supercomputers**

Now imagine that the two software applications have been parallelized to run on thousands of single core processors (we’ll come to multi-core at the end of the paper). If the disk based strategy was adopted, data transfer times to and from disk would probably dominate the computation time. Year on year, processor clock speeds have increased at a faster rate than data transfer rates (disk to memory to register), making the problem worse. Indeed, even if the data was transferred in memory, taking into account modern cache memory architectures, this code coupling strategy would still be in efficient.

**Element by Element Solution Algorithms**

In the 1990s, various groups implemented iterative solution techniques for parallel finite element analysis using an element by element strategy [4,5]. At a basic level, these methods involve processing a sequence of matrix-vector multiplications (or a matrix-matrix multiplication in special circumstances). These BLAS2 and BLAS3 type computations can be processed very efficiently by most modern processors. With an element by element approach on a multiprocessor system, load balancing is fairly trivial too, with each processor working on its own set of elements. Speed up curves often exhibit superlinear behaviour as the set of elements assigned to a particular processor gets smaller, eventually fitting entirely in cache.

**Navier Stokes Equations**

The solution of the Navier Stokes equations often involves decoupling the system into separate velocity and pressure computations. This suffers from the same problem as the fluid-structure interaction example discussed earlier. When solved in a fully coupled sense, using an element by element strategy and the BiCGSTAB(l) solver, processor utilization in a multiprocessor system is very high, almost as good as could be achieved using only BLAS2 operations in the whole code (10% to 40% peak according to architecture). It has been reported [5] that a solver written by one of the authors [6] ran faster than an uncoupled implementation provided in the Earth Simulator GeoFEM package [7] and used an order of magnitude fewer processors.
**Data Locality on Multicore Processors**

So far, we have considered data locality for the fully coupled Navier Stokes problem on single core processors. Coupling systems of equations like this has sometimes been called the “monolithic” approach. It leads to finite element stiffness matrices that are larger than those of typical single physics problems. For example, a three dimensional 20 noded brick for stress analysis has a square element stiffness matrix of \(O(60)\). An equivalent element for Navier Stokes in \(O(68)\). For an electrically conducting fluid flowing under the influence of an externally applied magnetic field, the matrices are \(O(128)\).

Cache was mentioned previously. When the elements get bigger, fewer fit into cache. Increasing the number of computing cores that share the cache worsens the problem. In essence, the data is pushed away from the processor, degrading the overall efficiency of the computations, in terms of the percentage of peak performance obtained. Fortunately, when the structure of the element matrices is examined, it becomes apparent that it is possible to overcome this difficulty. This is better explained with an example.

The steady state magnetohydrodynamic equations for an incompressible fluid are shown below:

\[
\begin{align*}
\frac{\partial u_i}{\partial x_j} + \frac{1}{\rho} \frac{\partial p}{\partial x_j} - \frac{1}{\mu \rho} B_i \left( \frac{\partial B_j}{\partial x_k} - \frac{\partial B_k}{\partial x_j} \right) - \frac{\nu}{\rho} \nabla^2 u_i &= 0 \\
\left(1\right) \\
\frac{\partial u_i}{\partial x_j} + \frac{1}{\rho} \frac{\partial p}{\partial x_j} - \frac{1}{\mu \rho} B_i \left( \frac{\partial B_j}{\partial x_k} - \frac{\partial B_k}{\partial x_j} \right) - \frac{\nu}{\rho} \nabla^2 u_i &= 0 \\
\left(2\right)
\end{align*}
\]

\[
\begin{align*}
\frac{\partial u_i}{\partial x_j} &= 0 \\
\left(3\right) \\
\frac{\partial B_i}{\partial x_j} &= 0 \\
\left(4\right)
\end{align*}
\]

The equation relating to the fluid flow (1) and the equation relating to the magnetic field (2) both contain terms in velocity \(u_i\) and magnetic field \(B_i\), i.e. they are coupled. Equations (3) and (4) relate to the continuity conditions for the \(u_i\) and \(B_i\) respectively. In addition to the usual fluid properties, permittivity and conductivity are required.

After discretisation using the finite element method, the equations have the following matrix form:
The symbols $C_{12}$ and so on represent submatrices of different sizes (8 by 60 or 60 by 60). Some of these submatrices are repeated within a single element stiffness matrix (and contain identical data). Others are the same for families of elements (across groups of stiffness matrices). Element families have the same shape and material properties (and are therefore easily identified). With this knowledge, it is possible to reorder the computations to (a) keep data cache resident where the submatrices would otherwise be continually reloaded, element by element and to (b) work at an element sub-matrix level, thus increasing the number of element related computations that are in cache at any one time.

During the conference, the authors will provide comparative performance figures for optimized and unoptimized cases, using the Navier Stokes equations and Magnetohydrodynamics equations as case studies.

**Acknowledgements**

The authors would like to acknowledge the HPC-Europa2 Fellowship programme that made this international collaboration possible. Access to HECToR was provided through projects e107 and e72. Funding to attend the conference was provided by the University of Manchester.

**References**