Exploiting Extreme Processor Counts on Cray XT Systems with High-Resolution Seismic Wave Propagation Experiments

Mike Ashworth, *STFC Daresbury Laboratory, Daresbury Science and Innovation Campus, Warrington WA4 4AD, UK,* **Mario Chavez,** *Institute of Engineering, Universidad Nacional Autónoma de México, UNAM, C.U., 04510, México, DF, México,* **Eduardo Cabrera,** *DGSCA, UNAM, C.U., 04510, México, DF, México*

ABSTRACT: *We present simulation results from a parallel 3D seismic wave propagation code that uses finite differences on a staggered grid with 2nd order operators in time and 4th order in space. We describe optimisations and developments to the code for the exploitation of extreme processor counts. The ultra-high resolution that we are able to achieve enables simulations with unprecedented accuracy as demonstrated by comparisons with seismographic observations from the Sichuan earthquake in May 2008.*

KEYWORDS: earthquakes, seismology, wave propagation, Cray XT, parallel computing, parallel performance

1. Introduction

On 19th September 1985 a large Ms 8.1 subduction earthquake occurred on the Mexican Pacific coast with an epicentre at about 340 km from Mexico City. The losses were of about 30,000 deaths and 7 billion US dollars. On 12th May 2008 the Ms 7.9 Sichuan, China, earthquake resulted in about 80,000 deaths and losses of 120 billion US dollars. As the recurrence time estimated for these highly destructive type of events is only a few decades, or hundreds of years, there is considerable seismological, engineering and socio-economical interest in modelling these types of events, particularly, due to the scarcity of observational instrumental data for them.

Realistic three-dimensional modelling of the propagation of large subduction earthquakes poses both a numerical and a computational challenge, particularly because it requires enormous amounts of memory and storage, as well as intensive use of computing resources [1]. Realistic modeling of the seismic wave propagation for these types of earthquakes should include volumes of the earth crust of hundreds of kilometers. Three-dimensional seismic wave propagation problems of realistic-earth size can be successfully modelled using finite difference modeling. This method is highly suitable for parallel execution on today's distributed memory parallel computers using explicit message passing parallelization.

1.1 Wave propagation modelling

The 3D velocity-stress form of the elastic wave equation consists of nine coupled, first order partial differential hyperbolic equations for the three particle velocity vector components and the six independent stress tensor components. The equations are discretised on a regular structured staggered grid and solved with an explicit scheme which is second-order accurate in time and fourth-order accurate in space. Staggered grid storage allows the partial derivatives to be approximated by centred finite differences without doubling the spatial extent of the operators, thus providing more accuracy.

1.2 Parallel implementation

The model domain is partitioned in all three dimensions into individual sub-domains and allocated to the available processors, using simple partitioning to give an equal number of grid points to each processor. MPI point-to-point message passing is used to implement halo-exchange between neighbouring sub-domains to update the velocities and stresses calculated at each time step

A hybrid procedure has been used combining long period simulations from the finite difference code and high frequency synthetics to obtain 3D synthetic seismograms for the above-mentioned Sichuan earthquake [2]. The comparisons between the observed and the synthetic seismograms are satisfactory, both in time and frequency domains. Maximum synthetic accelerations, velocities, displacements and permanent displacements obtained from the model partially explain the modification of the topography, and the extensive damage observed on the infrastructure and towns located on top of the Sichuan earthquake rupture zone, as well as the slight damage observed at Chengdu, located at an epicentral distance of 90 km.

In this paper, we present the results from benchmark measurements, optimisation and performance profiling studies performed on the parallel 3D wave propagation staggered-grid finite difference code. The code was run on three different Cray XT platforms allowing us to explore the performance of the code at extreme processor counts. This work extends that performed elsewhere [3].

2. The Benchmark Runs

2.1 The Cray XT Systems

The Cray XT massively parallel computers combine commodity and open source components with custom-designed components. The architecture is based on the Red Storm technology that was developed jointly by Cray Inc. and the U.S. Department of Energy's Sandia National Laboratories. The XT system is based around AMD Opteron 64-bit processors, with each processor being directly connected via the chip's HyperTransport to a dedicated Cray SeaStar chip (based on the IBM POWER architecture). Each SeaStar contains a 6-Port router and communications engine.

We have run on the Cray XT4 HECToR[1] system in the UK and on both Jaguar[2] systems at ORNL; the older XT4 system and the new 'Petaflop' XT5. The HECToR system has 2.8 GHz dual-core AMD parts with a total of 11328 cores and 6 GB memory per node. The 6 GB per node is implemented as two memory banks of 4 GB and 2 GB and performance results reported elsewhere [4] have revealed that the asymmetric nature of this configuration can result in a measurable performance degradation for memory intensive codes. The Jaguar XT4 system has 30976 cores composed of 2.1 GHz quad-core chips and the 'Petaflop' system 2.3 GHz quad-core parts with a total of 150152 cores.

There are some significant differences between dual-core and quad-core AMD chips apart from the number of cores per chip. The key features are presented in Table 1. Performance on the 2.3 GHz quad-core Opterons is of particular interest in the UK as the HECToR system will be upgraded to use these parts in the summer of 2009.

Dual-core	Quad-core
Core	
2.8 Ghz clock frequency	2.3 Ghz clock frequency
SSE SIMD FPU 2 flops/cycle 5.6 GFlop/s peak	SSE SIMD FPU 4 flops/cycle 9.2 GFlop/s peak
Cache hierarchy	
L1 Dcache/Icache 64 kB/core	L1 Dcache/Icache 64 kB/core
L2 Dcache/Icache 1 MB/core	L2 Dcache/Icache 512 kB/core
No L3 cache	L3 Shared cache 2 MB/Socket
Software Prefetch and loads to L1	Software Prefetch and loads to L1, L2, L3
Evictions and Hardware prefetch to L2	Evictions and Hardware prefetch to L1,L2,L3
Memory	
Dual Channel DDR2	Dual Channel DDR2
10 GB/s peak @ 667MHz	12 GB/s peak @ 800MHz
8 GB/s nom. STREAM	10 GB/s nom. STREAM

Table 1. Key differences between Cray XT4 dual-core and quad-core nodes (after [5])

The code was run with the PGI Fortran compiler version 8 using "-O3 –fastsse" compiler options.

2.2 The Benchmark Case

The actual size of the problem is 500 x 260 x 124 km. In order to test the scalability out to extreme processor counts, model grids were generated at decreasing resolutions of 500m, 250m, 125m, 62.5m and 31.25m. The associated timesteps were 0.02s, 0.01s, 0.005s, 0.0025s and 0.00125s, respectively to comply with the Courant-Friedrich-Lewy condition. At 500m resolution this leads to a grid size of 1000 x 520 x 248 grid points. The 31.25m resolution model has a grid size of 16000 x 8320 x 3968 points. Distributions were designed for different numbers of MPI tasks such that the number of tasks in each dimension divides exactly into

[1] http://www.hector.ac.uk/
[2] http://www.nccs.gov/computing-resources/jaguar/

the model grid dimension. For example the 31.25m resolution model could be run on 20480 tasks with the tasks distributed in a grid of 40 x 32 x 16 yielding sub-domains of size 400 x 260 x 248.

3. Optimization

3.1 Vectorization

One of the key success factors in achieving high performance from the AMD Opteron is the vectorization of key computationally intensive loop nests. This is especially important for the quad-core parts as the Streaming SIMD Extensions (SSE) execution width doubles to 128 bits. The most compute intensive parts of the 3D wave propagation code are triply nested loops calculating terms in the finite difference equations. We have found that all important loops are reported as being vectorized.

3.2 Halo Exchange

The original code [1] uses MPI_SendRecv for the halo exchange. Whereas this is elegant it may be inefficient for two reasons:

a) It may not make the best use of the underlying message passing hardware.

b) It is implemented by passing array sections into the MPI routine e.g. a(0,:,:). This non-sequential memory access may not be best handled by the compiler, e.g. if it results in an additional memory-to-memory copy.

For a halo exchange communication pattern on the Cray XT series it is advisable to post receives first as early as possible using MPI_IRecv and then issue the sends. Pre-posting the MPI receives means that data can be copied direct into the application-space buffer. Otherwise an additional copy into unexpected buffer space may be required [5]. This leads to a strategy for the halo exchange as follows:

```
post receive into buffer 2
copy data to be sent into buffer1
post send from buffer1
wait for receive
copy received data from buffer2
wait for send
```

The probability that the buffer is available on the receiving processor could be improved by overlapping with calculations, but this has not been tried in the current implementation

3.3 Boundary Conditions

The code contains three subroutines which provide exponential damping in each of the three dimensions at the edges of the model domain. They are only called for sub-domains which lie at the edges of the domain, so this is a source of load imbalance across the processors. The boundary condition code contained a single DO-loop for the dimension perpendicular to the boundary and used Fortran 90 array syntax for the other two dimensions. This introduced some non-sequential memory accesses in one out of the three dimensions. This was rectified by replacing the loops by a Fortran 77 triply-nested loop so that the order of memory accesses could be made explicit.

3.4 Excessive subroutine calling

Performance profiling using the tau[3] profiling tool gave a profile of the time spent in each subprogram including the number of times each subprogram was called. Two subroutines were being called over 320 million times each. Calling a Fortran subroutine this many times can lead to a large overhead, due to the prolog and epilog code added by the compiler to handle passing arguments, creation and destruction of local variables etc. This overhead increases with the number of arguments in the subroutine call. The calls occurred in a doubly-nested loop within the time-stepping loop. Automatic in-lining of these tools using PGI compiler options did not help so the code was modified by pushing the loop nest inside one of the subroutines and manually in-lining the other, which was much smaller. Some modification of the argument lists was required in order to achieve this.

4. Performance Benchmarks

Performance benchmark runs were carried out using the model domain and resolutions described in section 2.2. Run lengths were 500 timesteps for the coarser resolutions reduced to 100 timesteps for the finest 31.25m resolution, in order to reduce the total run-time. Note that there is limited I/O in these benchmarks, with limited statistics being output sufficient to validate the runs. Realistic production jobs would require check-pointing and much larger quantities of output.

Total execution time was measured using MPI_WTime() and a performance metric was calculated. This is the product of the total number of gridpoints and the number of timesteps, this representing the total work involved, divided by the execution time. Scaling the performance metric with the number of gridpoints and timesteps is a valid means for comparing the performance of the different benchmarks, assuming that the amount of work (number of floating point operations) scales linearly with the number of gridpoints and timesteps.

The cumulative effectiveness of the performance optimizations described in the previous section is shown in Figure 1, which shows performance of the 62.5m resolution model on up to 8192 processor cores on HECToR. The original code is compared with three optimized versions. Opt 1 contains just the improved halo exchange. Opt 2 is as Opt 1 combined with the improved boundary conditions code. Opt 3 is as Opt 2 together with the optimization to reduce the number of subroutine calls.

[3] http://www.cs.uoregon.edu/research/tau/home.php

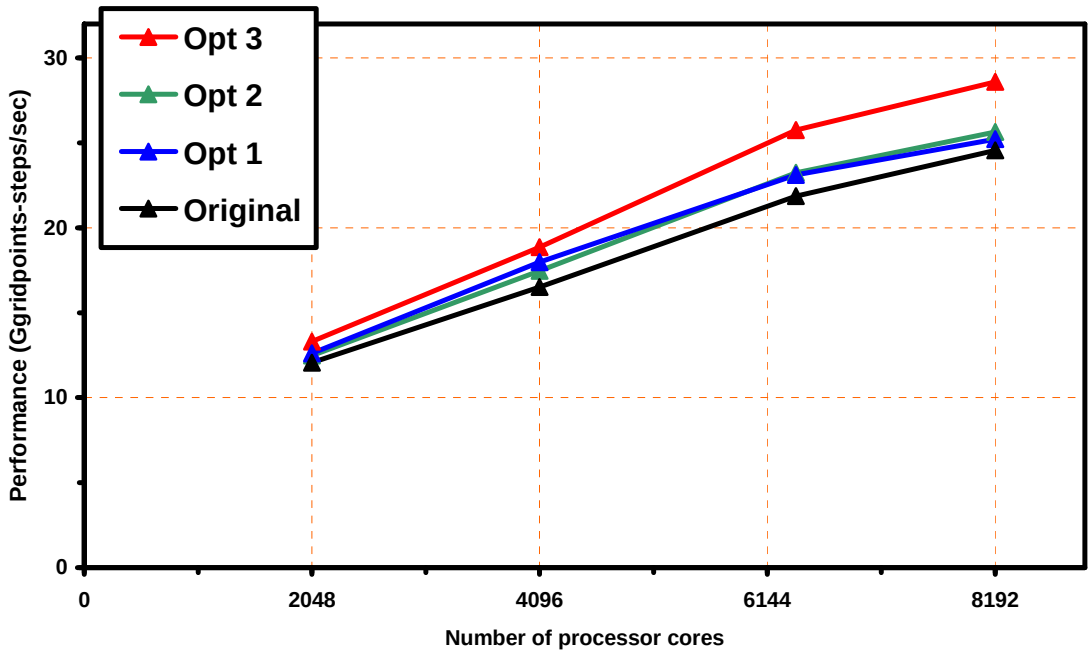

Figure 1. Performance of the wave propagation code at 62.5m resolution on HECToR for different code optimizations (see text)

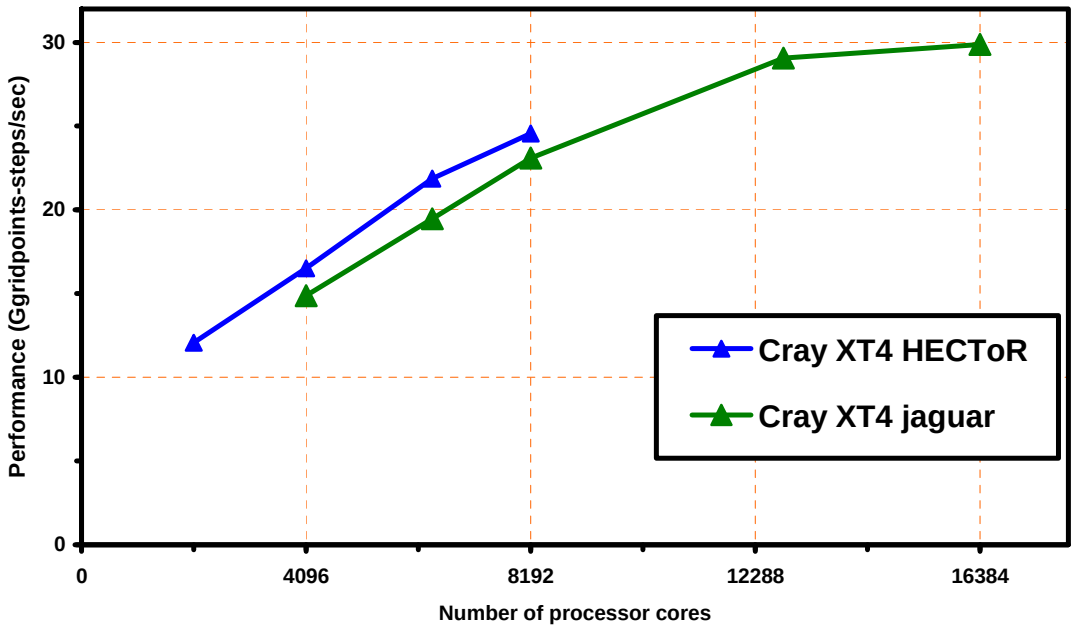

Figure 2. Performance of the wave propagation code at 62.5m resolution on the HECToR and Jaguar systems

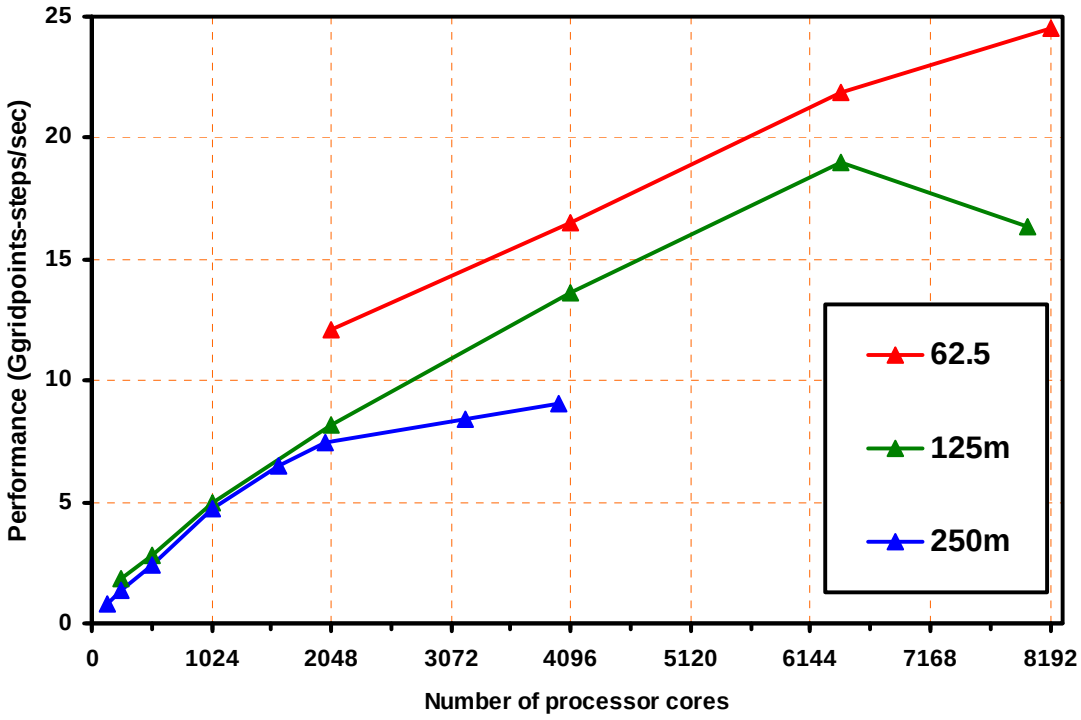

Figure 3. Performance of the wave propagation code at different resolutions on the HECToR system

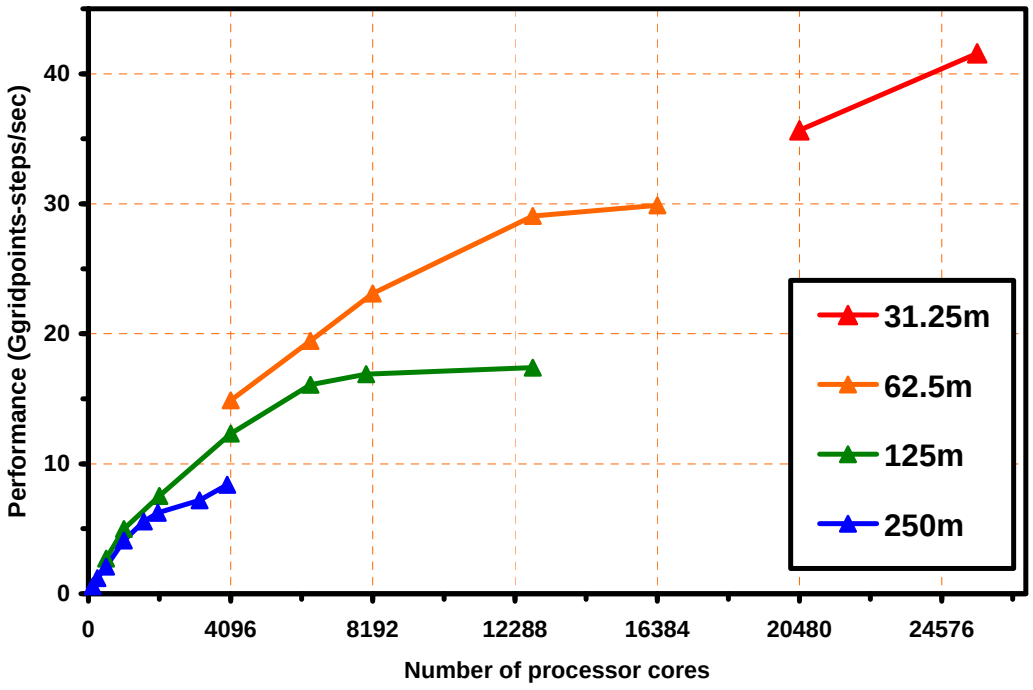

Figure 4. Performance of the wave propagation code at different resolutions on the Jaguar system

The cumulative effect of these optimisations is around a 10% improvement at 1024 cores rising to 15% at 8192.

Figure 2 shows the performance of the 62.5m resolution model on both the HECToR and Jaguar systems. The model does not fit into the 2GB/core limit on the quad-core system and 8192 is the largest batch job size on HECToR, so they two systems can only be compared between 4096 and 8192 cores. HECToR is faster by 11% at 4096, falling to 6% at 8192 cores. This is despite the Linpack performance (as obtained from the TOP500[4] list) being 37% faster on the Jaguar system.

The comparison with Linpack is more significant than might at first appear. The value of an Allocation Unit on the HECToR system is tied to its Linpack performance. So when the system is upgraded from dual-core to quad-core, unless the performance improvement of a user's application follows that of Linpack the user's allocation will effectively be devalued.

Recalling that HECToR has the faster clock speed, in order to obtain higher performance on the quad-core system one must a) vectorize all important loops to take advantage of the doubling of flops/cycle; b) be unaffected by the reduced memory bandwidth per core; and c) be unaffected by contention among the cores for network connections. As the wave propagation code vectorizes well we conclude that the performance per core is probably limited by memory traffic; a known issue for many finite difference structured grid codes.

We have used Cray's Craypat performance analysis tool to obtain figures from the hardware counters for the 62.5m resolution model on 2048 cores. The cores execute at around 550 Mflop/s or 9.9% of peak performance. The Level 1 cache hit rate is 96% showing that sequential vectorization is successful, but the computational intensity is only 0.2 ops/cycle and the instructions executed per cycle is only 0.45 confirming that the execution rate is probably limited by the rate of memory accesses.

The performance of the code at different resolutions is shown for the HECToR system in Figure 3 and for Jaguar in Figure 4. As expected, the smaller grid sizes run out of scalability at lower core counts. For a fixed problem size as the number of tasks increases the local grid size at each task reduces. For a three-dimensional grid the amount of work is proportional to the volume whereas the communications load depends on the surface area. As the local grid size reduces the ratio of communication to computation increases and the execution time is eventually dominated by communications costs.

5. Conclusions

We have run a series of benchmarks for a wave propagation code used to simulate seismic waves from large subduction earthquakes. This has allowed us to explore the performance of the code from hundreds of processor cores up to a maximum of 25,600 cores. Further tests being carried out on the Jaguar Petaflop system could not be included in the paper but may make it into the conference presentation.

We have found that, in common with other structured grid finite difference codes, provided the problem size can be scaled to maintain the size of the local problem for each task, the code can scale to large numbers of cores.

Although the finite difference code vectorises well the performance of individual cores is limited by the rate of memory accesses and the intrinsically low computational intensity of the method. This is especially apparent when moving from dual-core to quad-core chips, where the improvement of performance for this code is much less than the headline Linpack figure.

In the future we will be able to run at increased resolution on Petascale systems and large-scale calculations will enable more realistic simulations of earthquake events to be carried out.

Acknowledgments

This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-ASC05-00OR22725. The authors also acknowledge support from the Scientific Computing Advanced Training (SCAT[5]) project through Europe Aid contract II-0537-FC-FA.

We are grateful to John Levesque of Cray Inc. for performing benchmark runs on the Jaguar Petaflop system.

References

[1] *3D Parallel Elastodynamic Modeling of Large Subduction Earthquakes*, E. Cabrera, M. Chavez, R. Madariaga, N. Perea and M. Frisenda, in Proceedings of Euro PVM/MPI 2007, LNCS 4757, F. Capello et al. (eds), pp. 373-380, 2007, Springer-Verlag Berlin Heidelberg 2007

[2] *3D Wave Propagation Modeling Of The 12 05 2008 Sichuan Ms 7.9 Earthquake*, M. Chavez, E. Cabrera, H. Chen, N. Perea1, A. Salazar, D. Emerson, M. Ashworth, Ch. Moulinec, M. Wu and G. Zhao, American

[4] http://www.top500.org/

[5] http://www.scat-alfa.eu

Geophysical Union, Fall Meeting 2008, abstract #U23B-0051

[3] *Benchmark Study Of A 3d Parallel Code For The Propagation Of Large Subduction Earthquakes*, M. Chavez, E. Cabrera, R. Madariaga, N. Perea1, Ch. Moulinec, D.R. Emerson, M. Ashworth and A. Salazar, in Proceedings of Euro PVM/MPI 2008, … ***complete***

[4] *Application Performance on the UK's New HECToR Service*, F. Reid, M. Ashworth, T. Edwards, A. Gray, J. Hein, P. Knight, A.D. Simpson, K. Stratford and M. Weiland, Proceedings of the Cray User Group, Helsinki, 5-8 May 2008

[5] *Cray and the Quad-core Experience*, Jason Beech-Brandt, HECToR User Meeting, 22nd April 2009, London

[6] *Optimization for the Cray XT4™ MPP Supercomputer*, John M. Levesque, Cray Center of Excellence Training Materials

About the Authors

Mike Ashworth has led the Advanced Research Computing Group in the Computational Science & Engineering Department at STFC's Daresbury Laboratory since 2002. The Group is engaged in the development and optimization of large-scale applications for high-performance systems across a wide range of scientific disciplines. His own work focuses on the development and optimization of environmental modelling and CFD codes, including performance engineering and application of Grid technologies. He also serves on the CUG Board of Directors as Director-at-Large. E-Mail: mike.ashworth@stfc.ac.uk.

Mario Chavez is leading a group at the Institute of Engineering of UNAM, engaged in the development and application of hybrid computational modelling of the 3d wave propagation of extreme earthquakes for risk estimation purposes. E-mail: chavez@servidor.unam.mx

Eduardo Cabrera works at the Supercomputing Department in DGSCA of UNAM. He optimizes high-performance scientific applications. E-mail: eccf@super.unam.mx