

Moving the data to the computation: multi-site distributed parallel computation

Bruce M. Boghosian, Lucas I. Finn

Department of Mathematics, Tufts University, Medford, Massachusetts, USA

`bruce.boghosian@tufts.edu, lucas.finn@tufts.edu`

Peter V. Coveney

Centre for Computational Science & Department of Chemistry, University College London, London, UK

`p.v.coveney@ucl.ac.uk`

January 6, 2006

Abstract

We discuss criteria for the performance of distributed computations and show that an approach based on *distributing the data* is more efficient than moving the computation to the data in well defined circumstances. We summarise recent results from a US/UK Joint High End Computing Project in which so-called “meta-computing” applications are being run across several sites in the US and UK. Our findings demonstrate the viability of utilising a high-performance computing grid to run a single large-scale application over multiple, geographically remote sites, and open the way to addressing “grand challenge” problems that will always remain beyond the scope of a single supercomputer. We also expect this work to be of strategic significance for the future provision of high-end computing resources on both national and international bases.

1 Introduction

It is often suggested in “e-Science” circles that scientific research is now dominated by the sheer size of the data sets involved, to such an extent that one should avoid moving data around wherever possible, in favour of shipping codes to those sites where the data is located and performing the computations *in situ*. An interesting recent paper by Gordon Bell, Jim Gray and Alex Szalay [1] states that it is worth moving data to remote computing facilities only if there are more than 10^5 compute operations per byte, otherwise the cost of shifting the data negates any advantage thereby obtained. At first sight, as the authors argue, this seems to favour constructing a distributed computing infrastructure predicated on the idea of keeping computations local to the data. Here, however, we show that one may readily encounter important situations where that condition is exceeded, and it is more efficient to distribute the data to locations where the computations may be performed concurrently.

For present purposes, we consider one specific application to illustrate these points. We plan to return in the near future with a more detailed technical paper which reports the performance of other codes in similar circumstances. The application we report on here is called VORTONICS. This is a software suite containing, *inter alia*, a three-dimensional lattice Boltzmann solver for the Navier-Stokes equations. It employs a regular computational lattice, is parallelized with MPI, outfitted with parallel visualization using VTK, and is grid enabled using MPICH-G2. It has been extensively tested, both in single-site and cross-site runs on the TeraGrid in the US, and also on the Newton machine at CSAR on the UK National Grid Service. The software allows the user to dynamically remap between different cross-processor data layouts, and to resize the computational lattice using Fourier resizing. It also includes sophisticated routines for visualizing the vortical structure of the fluid, for producing graphical output, and for computational steering.

When VORTONICS is run in parallel and/or on a computational Grid, it is possible for the user to specify that the spatial lattice be decomposed, and indeed to specify the precise manner of that decomposition. We

# Sites	Locations Employed			
1	NCSA			
2	NCSA	TACC		
3	NCSA	TACC	ANL	
4	NCSA	TACC	ANL	PSC

Figure 1: **TeraGrid Sites Employed in Cross-Site Timings**

refer to the decomposition of the spatial lattice across different computer centres on a computational Grid as *Geographically Distributed Domain Decomposition* (GD^3). While it would be preferable to fit the state of the code in the memory of a single multiprocessor in a single computer centre, this may not be possible for large problems involving terabytes of state. If no single computational resource is sufficiently large to hold the entire problem in memory, one must choose between swapping the state to media with higher volume but lower bandwidth, such as disk, or employing GD^3 . Our results indicate that, for the important case of the communication pattern associated with a computation on a regular lattice, GD^3 is clearly the preferable of these two alternatives and, furthermore, will proceed with what must, at first sight, be described as remarkable efficiency.

The ultimate scientific motivation for our work is the following: The Direct Numerical Simulation (DNS) of fully developed turbulence at very high Reynolds numbers (100,000 or higher) is the grand challenge problem in fluid dynamics at which we are aiming; because of the computational complexity of the algorithm, and the need to resolve short length scale turbulent eddies, one needs to employ very large lattices. These rapidly become too big to fit on any single supercomputer. However, using long simulations on a grid of high-performance machines, it is in principle possible to distribute the calculation and thereby to simulate Reynolds number higher than any realisable on a single machine.

2 Cross-site performance of the VORTONICS code

We conducted timings to assess the performance of the multiple relaxation time lattice Boltzmann (MRTLb) module of the VORTONICS software package using MPICH-G2 for a number of cross-site runs on the TeraGrid. The runs employed between one and four sites, distributed as described in Fig. 1. The timings used between 64 and 416 CPU's across these sites. The CPU's were not necessarily distributed evenly across the sites; for example, the two-site 256-CPU run employed 192 CPU's at NCSA and 64 CPU's at TACC. In all cases, one CPU per node was used, and no attempt was made to find the optimal cross-site layout for the data. Twelve Access Gateway (AGW) nodes were used for the cross-site run involving the Pittsburgh Supercomputing Center (PSC). Results including UK sites will be reported later.

The global lattice dimensions for each run were chosen so that every CPU had exactly 128^3 sites, decomposed as a $128 \times 128 \times 128$ block; for example, the global lattice dimensions for the 4-site 416 CPU simulation were $1664 \times 1024 \times 512$. This local lattice size was kept fixed in all timings reported here. It is large enough that any latency associated with looping over the sites is not an issue, and small enough to fit in the memory of all of the processors that we use. The cube decomposition ensured minimal surface-to-volume ratio; for near-neighbor communication on a regular lattice, this leads to concomitantly minimal communication bandwidth requirements.

Fig. 2 tabulates the total site updates per second (SUPS) of the MRTLb algorithm, while Fig. 3 tabulates the *effective* site updates per second per processor (SUPSPP), for all of the timing runs conducted. We point out two features of this data:

- The number of lattice site updates per second for a fixed number of computing centre sites increases approximately linearly with the number of CPU's employed. This is illustrated in Fig. 4.
- The factor by which the effective SUPSPP is reduced for cross site runs employing N sites is substantially less than N ; of course, any attempt to conduct the same large run at a single site by swapping between different storage media would clearly result in a slowdown factor of *at least* N . In going from one to two sites, our slowdown factor is about 1.5; interestingly, virtually no additional penalty is

CPU's	Number of Sites			
	1	2	3	4
64	28,416,000	17,024,000		
128	53,760,000	34,432,000	34,304,000	23,424,000
192	73,344,000	46,848,000		
256	96,512,000	65,536,000	66,560,000	40,960,000
320		69,120,000	59,840,000	58,560,000
352			77,088,000	
416				90,272,000

Figure 2: **Total Site Updates per Second**

CPU's	Number of Sites			
	1	2	3	4
64	444,000	266,000		
128	420,000	269,000	268,000	183,000
192	382,000	244,000		
256	377,000	256,000	260,000	160,000
320		216,000	187,000	183,000
352			219,000	
416				217,000

Figure 3: **Effective Site Updates per Second per Processor**

incurred in going to three sites. In going from one to four sites, the factor is about 2.3. Again, this can be seen by comparing the various curves in Fig. 4.

Moreover, recent modifications to MPICH-G2 should further improve these performance figures. For example, all of these timings use TCP packets for data communication; very soon, MPICH-G2 will switch to UDT, and preliminary studies indicate that this change will improve the code's performance by at least a factor of two. In addition, future version of Globus should allow us to employ more than one processor per node, also leading to a substantial improvement in efficiency.

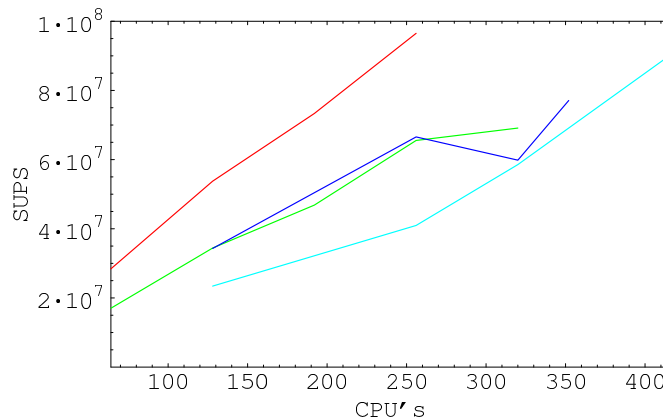


Figure 4: **Site Updates per Second for Varying Number of Sites (1, 2, 3, 4)**

3 Conclusions

Our results from the VORTONICS code (obtained before, during and after SC05) show that a problem of given size, split equally across two geographically remote sites, runs only 1.5 times more slowly than when it resides on a single site; when distributed across up to six sites, the performance degradation is only a factor of four.

The implications are clear and dramatic. For problems of this sort that are too big to fit into the memory of a single machine, it is considerably more efficient to solve them on a geographically distributed grid than on a local machine, where one would have to cut the system up into pieces which would be simulated in turn; such a distributed “metacomputing” activity allows us to effectively solve problems that cannot be fitted entirely onto a single (super)computer. These findings are of considerable strategic importance for the use of grid computing to address grand challenge problems in high performance computing and can be expected to play an influential role in defining future policy and planning for UK, European and wider international supercomputing.

Such an approach should prove to be a game-changer in future HPC strategies and usage by facilitating the simulation of much larger systems than would otherwise be possible. To be sure, there is an economic dimension to this approach as well: Users should no longer have to wait until a sufficiently large supercomputer becomes affordable, hence available, to fit a new size of problem [3]. Obviously, every machine, however large and expensive, will be of finite size; a grid of supercomputers will always enable bigger problems to be tackled than can be accommodated on a single machine.

As with any “disruptive” technology, grid computing requires new methods and working practices and the HPC domain is no exception. In particular, to realise the step jump afforded by cross-site parallel computation requires the implementation of advanced reservation and co-scheduling of grid resources. Indeed, we have repeatedly argued for this in the past [2]. The US TeraGrid is moving toward providing such a facility within the near future (probably in 2006); the UK National Grid Service and its high performance facilities are, relatively speaking, quite some way behind in the provisioning of equivalent facilities.

We recommend that a co-scheduling service is made a high priority, and that users of both the US TeraGrid and UK NGS (including the UK’s HPC resources) are incentivised to exploit it through appropriately structured resource allocations policies. In particular, it is critically important that users be able to co-schedule resources without the intervention of a human at the various computer sites involved. This means that each site must allow for on-line, automatic placement and confirmation of reservations.

Acknowledgments

We thank the US National Science Foundation and the UK Engineering and Physical Sciences Research Council for funding this work under a Joint US/UK High End Computing Project 2005-06. BMB and LIF were partially supported by subcontract to the University of Chicago under Award Nos. CA SCI-0525308 and CSA SCI-0438712 from the National Science Foundation.

References

- [1] G Bell, J Gray & A Szalay, “Petascale Computational Systems: Balanced CyberInfrastructure in a Data-Centric World”, October 2005, available from <http://research.microsoft.com/users/GBell/Supercomputing&CyberInfrastructures.htm>
- [2] P V Coveney (ed.), “Scientific Grid Computing”, *Phil. Trans. R. Soc. London A*, **363**, 15 August 2005.
- [3] C. Gottbrath, J. Bailin, C. Meakin, T. Thompson, J.J. Charfman, “The Effects of Moore’s Law and Slacking on Large Computations,” available from <http://www.gil-barad.net/~chrisg/mooreslaw/Paper.html>