Grid Resource Selection by Application Benchmarking for Computational Haemodynamics Applications

Alfredo Tirado-Ramos[1], George Tsouloupas[2], Marios Dikaiakos[2], and Peter Sloot[1]

[1] Faculty of Sciences, Section Computational Science,
University of Amsterdam,
Kruislaan 403, 1098 SJ Amsterdam, The Netherlands
{alfredo, sloot}@science.uva.nl
[2] Department of Computer Science,
University of Cyprus,
75 Kallipoleos St., P.O. Box 20537, 1678, Nicosia, Cyprus
{georget, mdd}@cs.ucy.ac.cy

Abstract. Grid benchmarking for improved computational resource selection can shed a light for improving the performance of computationally intensive applications. In this paper we report on a number of experiments with a biomedical parallel application to investigate the levels of performance offered by hardware resources distributed across a pan-European computational Grid network. We provide a number of performance measurements based on the iteration time per processor and communication delay between processors, for a blood flow simulation benchmark based on the lattice Boltzmann method. We have found that the performance results obtained from real application benchmarking are much more useful for running our biomedical application on a highly distributed grid infrastructure than the regular resource information provided by standard Grid information services to resource brokers.

1 Introduction

Resource selection in Grid environments is a crucial problem. Regardless of who performs the resource selection, be it users or automated systems (i.e. schedulers or resource brokers), the decision makers are faced with the difficult task of matching/mapping jobs to resources. Previous work on the specification of resources and services in complex heterogeneous computing systems and metacomputing environments in general [1] and, particularly, in grid environments, [2], has led to a better understanding of the issues. Nevertheless, the evolution of Grid architectures has underlined the need for addressing application-specific characterization of the resources available. Grid benchmarking, or the characterization of Grid computational resources for improving resource selection, can be used to help improving the performance of computationally intensive parallel

V.S. Sunderam et al. (Eds.): ICCS 2005, LNCS 3514, pp. 534–543, 2005.

applications by enhacing the resource selection process. Our application for pre-operative support, a blood-flow simulation solver, is an implementation of the lattice Boltzmann method, a mesoscopic approach for simulating fluid flow based on the kinetic Boltzmann equation. The problem statement is straight-forward: how can we find the best resources to run the application at hand? Additionally, since the application is often run in multiple instaces using parameterised runs, it would be desirable to have acces to information that would help better schedule these jobs. In this article we discuss our results after performing a number of experiments to investigate the levels of performance offered by hardware resources distributed across the CrossGrid European computational Grid. We show how we can rank resources based on a benchmark derived from the blood-flow simulation kernel. In the remainder of this article, Section 2 describes the parallel solver we use as a benchmark in more detail. Section 3 lays out some specifications of the experimentation testbed used, the CrosGrid tesbed. In Section 4 we provide a short description of the GridBench framework which we used to perform our benchmarking experiments. In Section 5 we discuss some of the issues related to benchmarking on the Grid, particularly in light of resource characterization and ranking based on kernel performance, and offer our results. Finally, in Section 6 we briefly discuss our conclusions and relevant future work.

2 Non-invasive Vascular Reconstruction

For our benchmarking experiments we use a parallel solver from the Virtual Radiology Explorer (VRE) Grid-based Problem Solving Environment (PSE), a type of integrative collaborative environment [3] that includes simulation, interaction, and visualization components for pre-treatment planning in vascular interventional and surgical procedures. This PSE was developed by the University of Amsterdam and deployed within the European Crossgrid project [4]. The deployment of the interactive VRE system within Crossgrid resulted in a pan-European experimental PSE using resources from across Europe, exploiting available achievements from other European Grid projects such as European DataGrid[1] and the Large Hadron Collider Computing Grid[2]. We lay out a base architecture for PSEs using the Grid as a medium, with a validated case study in vascular reconstruction. For additional background, motivation, and the latest Grid-based results, see [6]. The VRE contains an efficient parallel computational hemodynamics solver [7] that computes pressure, velocities, and shear stresses during a full systolic cycle. The simulator is based on the Lattice-Boltzmann method (LBM), a mesoscopic approach for simulating fluid flow based on the kinetic Boltzmann equation [8]. The data used as input for the VRE can be obtained from several imaging techniques used to detect vascular disorders. For instance, 3D data acquired by Computed Tomography or Magnetic Resonance

[1] http://www.eu-datagrid.org
[2] http://lcg.web.cern.ch/LCG/Documents/default.htm

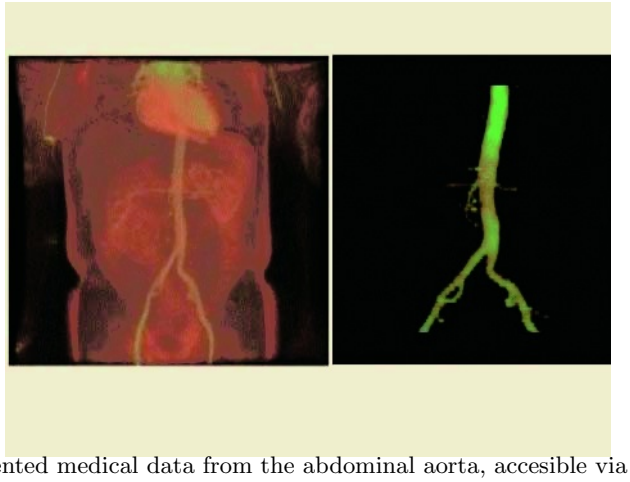


Fig. 1. Segmented medical data from the abdominal aorta, accesible via Grid Storage Elements functioning as medical repositories

Imaging, or particularly Magnetic Resonance Angiography for imaging blood vessels that contain flowing blood. To convert the medical scans into meshes our solver can work with, the raw medical data is first segmented so that only the arterial structures of interest remain in the data set (Figure 1).

Measurements are important for diagnoses. Clinical decision-making relies on evaluation of the vessels in terms of the degree of narrowing for stenosis and dilatation (increase over normal arterial diameter) for aneurysm. The selection of a bypass (its shape, length, and diameter) depends on sizes and geometry of an artery.

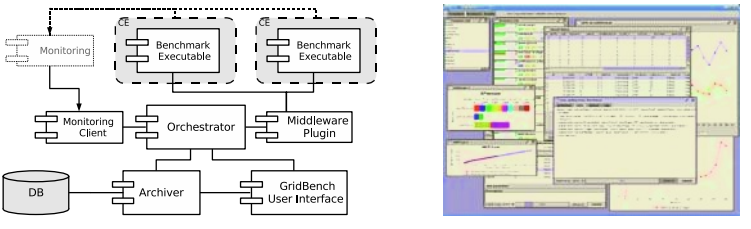
3 The CrossGrid Experimental Testbed

The CrossGrid distributed testbed[3] shares resources across 16 European sites. The sites range from relatively small computing facilities in universities, to large computing centers, offering an ideal mixture to test the possibilities of an experimental Grid framework. National research networks and the high-performance European network, Geant[4], assure interconnectivity between all sites. The network includes a local step, typically inside a University or Research Center, via Fast or Gigabit Ethernet, a jump via a national network provider at speeds that will range from 34 Mbit/s to 622 Mbit/s or even Gigabit, to the national node, and a link to the Geant network at 155 Mbit/s to 2.5 Gbit/s. Our experiments were conducted on the CrossGrid testbed, following the basic LCG-2 architecture. In this architecture, a Grid VO is made up of a set of geographically distributed sites (computer clusters) containing computational or storage resources. Each site contains a Computing Element, which manages a set of Worker Nodes. A site may also contain a Storage Element, which is an interface to mass storage.

[3] http://www.eu-crossgrid.org
[4] http://www.dante.net/geant/

4 The GridBench Tool

GridBench [9] is a tool for benchmarking Grids. It consists of a framework containing a set of tools that aim to facilitate the characterization of Grid nodes or collections of Grid resources. The framework has two main objectives: to generate metrics that characterize the performance capacity of resources belonging to a Virtual Organization (VO), and to provide a tool for researchers that wish to investigate various aspects of Grid performance using well-understood kernels that are representative of more complex applications deployed on the Grid. In order to perform benchmarking measurements in an organized and flexible way, GridBench provides a means for running benchmarks on Grid environments as well as collecting, archiving, and publishing the results. The framework allows for convenient integration of new and existing benchmarks into the suite, as well as the customization of existing benchmarks through parameters. We have used the tool to perform our biomedical application benchmarking experiments (Figure 2).



(a) (b)

Fig. 2. GridBench: 2(a) shows the main components and services of the GridBench software architecture. 2(b) shows the GridBench GUI, used for defining and executing benchmarks, and browsing and analyzing results

5 Results and Discussion

Our application benchmark, the BStream kernel, is part of an interactive Grid application that involves processing of 3D data, which makes it computationally expensive. Shown here is the computationally intensive part of the application, which uses the Message Passing Interface (MPI) for parallelization. This code was instrumented to measure elapsed time for each iteration as well as the time spent on MPI communication, and integrated into GridBench[5]. As a dataset we used different sample files that represent our normal workload, from simple tube-like artery structures to aorta segments containing bifurcartions.

[5] The charts presented in this section were automatically generated using the GridBench Graphic User Interface.

5.1 Resource Comparison

Figure 3(a) shows the measured iteration times of the BStream kernel on 13 sites available in our testbed. In each case, the same workload was applied by using identical input data and parameters. Figure 3(a) shows the results obtained by using 2 CPUs in each measurement. For the 2 CPU measurements, using 2 CPUs on the same Worker Node was preferred over using two CPUs on two different Worker-Nodes. This is important, since it was found that this would seriously impact performance of this kernel (Figure 5). Resources *cluster.ui.sav.sk* and *loki01.ific.uv.es* employ single-CPU nodes while the majority of site employ dual-CPU Worker Nodes. In Figure 3(a) (as well as the rest of the charts in Figure 3) we observe that iteration times remain fairly constant throughout the duration of the computation. For our experiments, we have set the application kernel to run for 800 iterations, so it can be seen that right before the end of each run (at around 760 to 780 iterations) a jump in performance of about 30% larger time per iteration values is experienced in all nodes. This is mainly due to the design of the current version of the kernel, where the first processor that

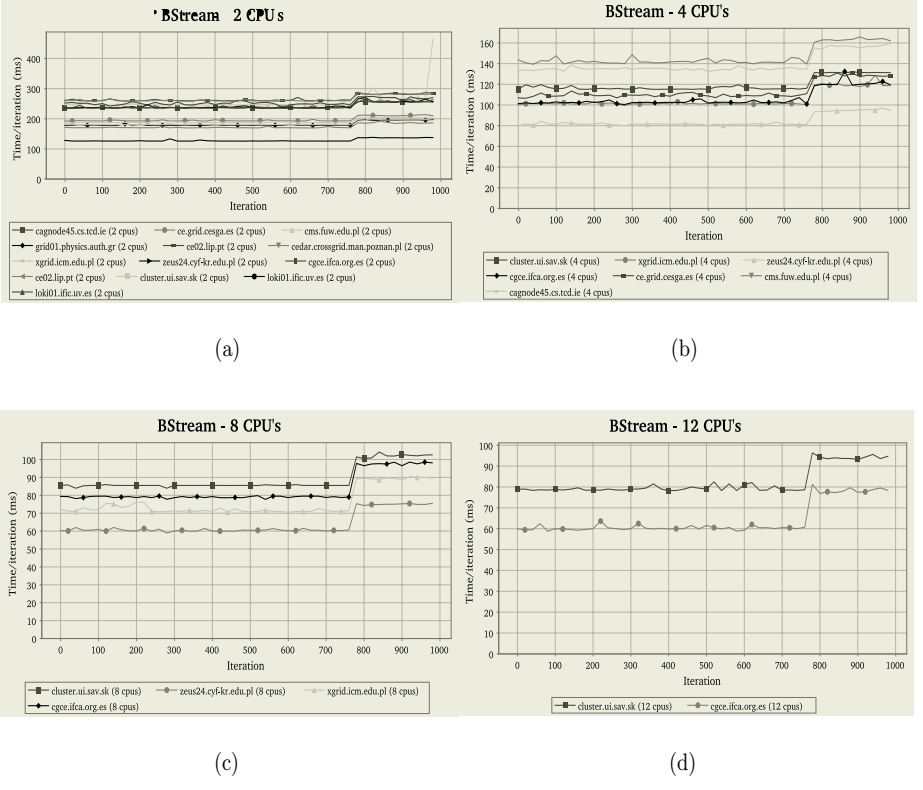


Fig. 3. The performance of the kernel at a set of sites using 2, 4, 8 and 12 CPU's

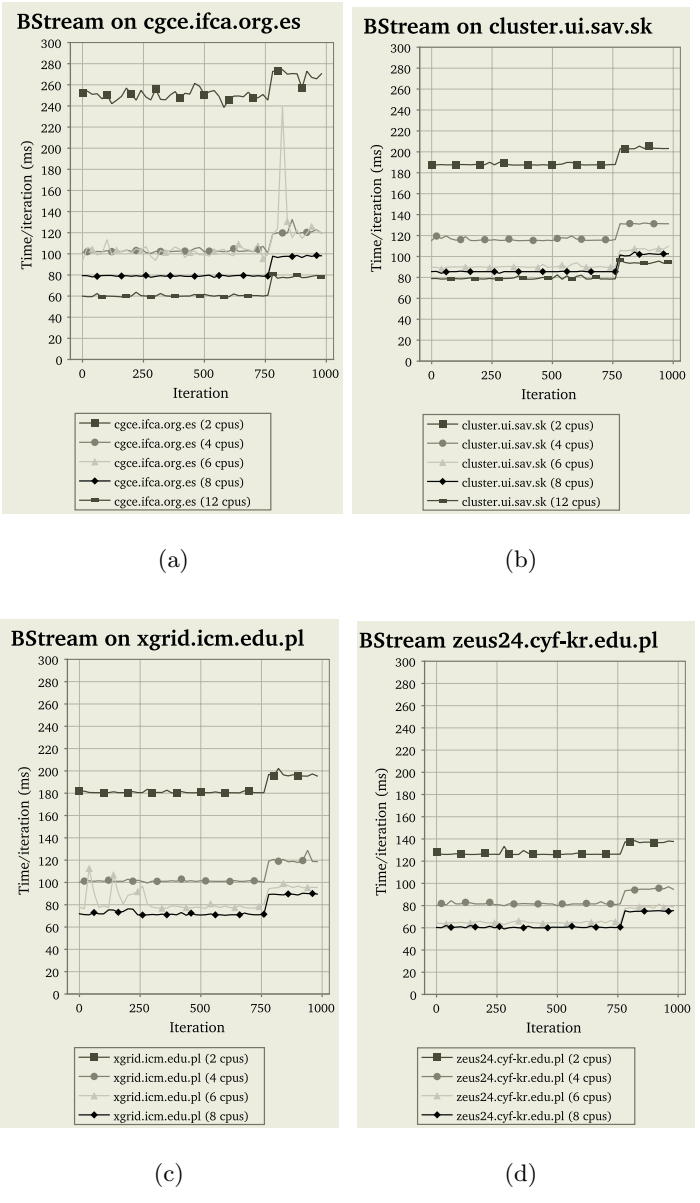


Fig. 4. Scalability as it is measured at four sites. Lower iteration times are better

started running gathers data from all other processors before producing the final output[6]. Nevertheless, iteration times remain relatively invariant regardless of

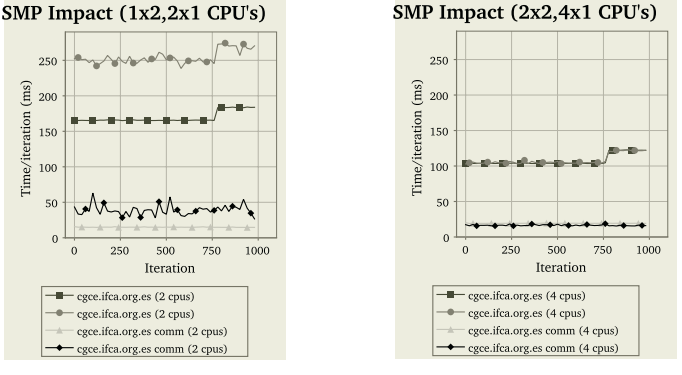
[6] The new version of the BStream kernel, which is work in progress at the time of writing this paper, has a different design that addresses this issue.

the number of iterations. For this reason it is reasonable to assume that short run-time experiments (using a small number of iterations) are representative of our real-life experiments, in which we use larger iteration counts.

Figures 3(b), 3(c) and 3(d) show the performance of the kernel at a set of sites using 4, 8 and 12 CPUs respectively. Generally we observe a "downward" trend indicating that the code is somewhat scalable, i.e. using a larger number of cpus at a given site will yield a faster run-time, but more on scalability will be given in the next sub-section.

5.2 Scalability

Figure 4 shows the scalability of the kernel as it is measured at four sites: *cgce.ifca.org.es* (up to 12 CPUs), *cluster.ui.sav.sk* (up to 12 CPUs), *xgrid.icm.edu.pl* (up to 8 CPUs) and *zeus24.cyf-kr.edu.pl* (up to 8 CPUs). It is quite interesting to observe that the different sites display a different scalability. For example, in Figure 4(a) the runtime is reduced to less than 30% when going from 2 CPUs to 8 CPUs, while in Figure 4(b) the improvement is only just under 50%. Similarly, while in 4(a) there is approximately a 25% improvement in runtime when going from 8 CPUs to 12 CPUs, in 4(b) there is only marginal improvement. Scalability *at each resource* needs to be taken into considaration for efficient resource selection.



(a) (b)

Fig. 5. Impact of MPI communication on runtime. 5(a) Iteration and communication times using 2 CPUs on the same (dual) Worker Node (1x2), and 1 CPU on each of 2 Worker Nodes. 5(b) Iteration and communication times using 2 CPUs on each of 2 (dual) Worker Nodes (2x2), and 1 CPU on each of 4 Worker Nodes (4x1)

5.3 Communication Measurements

The BStream kernel uses MPI for inter-proces communication, which we compiled using the MPICH4 device. The code is highly coupled and it is expected that the performance of the interconnect, i.e., the LAN connecting the cluster nodes will

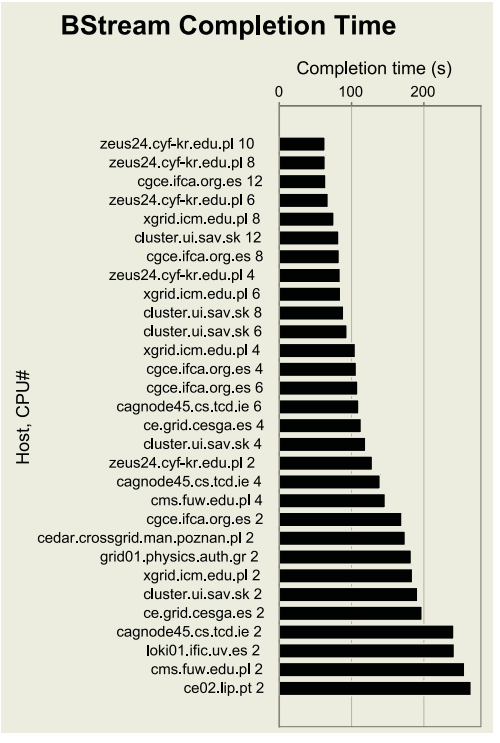


Fig. 6. Completion times of the BStream kernel using different numbers of CPUs on several resources

have a considerable impact on the performance of the kernel. To investigate this, the BStream code was instrumented to measure the time spent in communication[7]. To isolate the effect of the network we ran the code using just two CPUs on a dual-CPU Worker Node (1x2)[8], using two CPUs on two different (identical) Worker Nodes (2x1). This is shown in Figure 5(a). Figure 5(b) shows a similar experiment using 4 CPUs. In 5(a) we observe that there is considerable difference in communication performance which also impacts the time per iteration. On the other hand, in 5(b) we observe no significant difference when running in either mode, since the network is used is both cases (both in 2x2 and in 4x1).

5.4 Decision-Making

Figure 6 conveys a lot of usefull information since it provides a ranking of runtimes on all of the resources available. This ranking could be used directly in

[7] The impact of the instrumentation was measured and was found to be insignificant.
[8] The MPI library used was not optimized for SMP, and communication still went through the TCP/IP stack.

resource selection *especially* in cases where the relative CPU, Memory and network speeds at each resource (site) are not known. For example, it appears that it is better to run the code at *zeus24.cyf-kr.edu.pl* using 4 CPU's than at *cluster.ui.sav.sk* using 8 CPU's.

6 Conclusions and Future Work

We have presented a concise set of benchmarking results for the characterization of computational Grid resources in terms of the performance of CPU, main memory and interconnects, for a biomedical application for the simulation of blood flow. We have presented a set of benchmarking experiments to experiment with specific computation versus communication metrics. This small set of biomedical application benchmarking results are run on the highly distributed Grid resources offered by the European CrossGrid testbed with small overhead and with minimal effort by the user. These benchmarks can be invoked in a periodic and on demand manner using the GridBench framework, with the resulting measurements archived and made available via Grid services based on a web services framework. Furthermore, we found that ranking resources based on the performance of a stripped-down and instrumented version of an application can give us realistic resource rankings that reflect the performance of the application itself. We have shown how the results obtained using GridBench can be used for ranking of resources and how they can help in resource selection. In the future we plan to further investigate the relation between full-blown application performance and micro-benchmark performance in the context of highly distributed Grid environments.

Acknowledgments

We would like to acknowledge the meaningful contributions of R. Belleman, A. M. Artoli, and L. Abrahamyan to this work.

References

1. Matthias Brune, Jorn Gehring, Axel Keller, Burkhard Monien, Alexander Reinefeld, Specifying Resources and Services in Metacomputing Environments, Parallel Computing, 1998, vol. 24, pp. 1751–1776, Elsevier Science.
2. Greg Chun, Holly Dail, Henri Casanova, and Allan Snavely. Benchmark probes for grid assessment. Technical report, UCSD, 2003.
3. Houstis E.N., Rice J.R., Weerwarna S., Papachio P., Wang K. Yang and Gaitatzes M., Enabling Technologies for Computational Science Frameworks, Middleware and Environments, Chapter 14, pages 171-185. Kluwer Academic Publishers, 2000.
4. A. Tirado-Ramos; P.M.A. Sloot; A.G. Hoekstra and M. Bubak: An Integrative Approach to High-Performance Biomedical Problem Solving Environments on the Grid, Parallel Computing, (special issue on High-Performance Parallel Bio-computing) vol. 30, nr 9-10 pp. 1037-1055. Chun-Hsi Huang and Sanguthevar Rajasekaran editors, 2004.

5. W. Hoschek, J. Jaen-Martinez, A. Samar, H. Stockinger, K. Stockinger, Data Management in an International Data Grid Project, IEEE/ACM International Workshop on Grid Computing Grid'2000, 17-20 December 2000 Bangalore, India "Distinguished Paper" Award.
6. http://www.science.uva.nl/research/scs/HotResults/
7. A.M. Artoli, A.G. Hoekstra, P.M.A. Sloot, Simulation of a systolic cycle in a realistic artery with the Lattice Boltzmann BGK method., Int. J. Mod. Phys. B, Vol. 17, Nos. 1-2 (2003) 95-98.
8. S. Succi, The Lattice Boltzmann Equation for fluid dynamics and beyond, Oxford Science Publications, Clarendon Press, 2001.
9. George Tsouloupas, Marios D. Dikaiakos, GridBench: A Tool for Benchmarking Grids, Proceedings of the 4th International Workshop on Grid Computing (GRID2003), Phoenix, AZ, November 2003, pp. 60-67, IEEE.