Performance of a Parallel Astrophysical N-Body Solver on Pan-European Computational Grids

Alfredo Tirado-Ramos[1], Alessia Gualandris[1,2], and Simon Portegies Zwart[1,2]

[1] Faculty of Science, Section Computational Science,
University of Amsterdam Kruislaan,
403, 1098 SJ Amsterdam, The Netherlands
[2] Astronomical Institute Anton Pannekoek,
University of Amsterdam Kruislaan,
403, 1098 SJ Amsterdam, The Netherlands

Abstract. We present performance results obtained by running a direct gravitational N-body code for astrophysical simulations across the Dutch DAS-2 and the pan-European CrossGrid computational grids. We find that the performance on large grids improves as the size of the N-body system increases because the computation to communication ratio becomes higher and a better load balance can be achieved. Communication among nodes residing in different locations across Europe becomes more evident as the number of locations increases. Nevertheless, contrary to our expectations, we find that the performance decreases only by about a factor three for a large simulation. We conclude that highly distributed computational Grid infrastructures can be used efficiently for simulating large gravitational N-body systems.

1 Introduction

Direct summation methods to model the dynamics and evolution of collisional systems allow scientists to follow the global evolution of large stellar systems along their lifetime. As a counterpart to their high numerical accuracy, they present $O(N^2)$ complexity, which translates in massive computational requirements for complete sets of inter-particle forces.

Significant improvement in the performance of direct codes used in the numerical simulation of astrophysical stellar systems can be obtained by means of general purpose parallel computers (Dorband, Hemsendorf and Merrit, [1]; Gualandris, Portegies Zwart and Tirado-Ramos, [4]), but the use of highly distributed clusters within computational grids has not yet been explored. Grid technology is rapidly becoming a major component of computational science. It offers a unified means of access to different and distant computational resources, with the possibility to securely access highly distributed resources that scientists do not necessarily own or have an account on. Connectivity between distant locations and interoperability between different kinds of systems and resources are some of the most promising characteristics of the Grid.

P.M.A. Sloot et al. (Eds.): EGC 2005, LNCS 3470, pp. 237–244, 2005.

In this paper we present the results of the first experiments conducted on computational Grids using a direct gravitational N-body code. The code is parallelized with MPI using a systolic algorithm. We explore the effects of network latency on the performance on the DAS-2 Grid testbed [1], distributed within the Netherlands and running the Globus toolkit (Foster & Kesselman, [2]), as well as on the 18-node CrossGrid testbed [2], distributed across Europe (see Fig. 1) and running the LCG2 [3] infrastructure software.

In Sec. 2 we discuss the numerical integrator and the parallel algorithm used for our experiments. Section 3 briefly describes our experimental grid testbed setup, Sec. 4 presents the obtained timing results, and Sec. 5 contains our summary and conclusions.

2 Numerical Method

For the work described in this paper we consider a direct N-body code applied to the study of astrophysical stellar systems. An N-body code solves the equations of motion of N point particles interacting gravitationally with each other. In a direct method the gravitational force acting on each particle is computed by summing up the contributions from all the other particles according to Newton's law:

$$\mathbf{F}_i = m_i \mathbf{a}_i = -G m_i \sum_{j=1, j\neq i}^{j=N} \frac{m_j(\mathbf{r}_i - \mathbf{r}_j)}{|\mathbf{r}_i - \mathbf{r}_j|^3}. \quad (1)$$

Starting from initial values of the positions and velocities of all the stars and using the values of the forces and their first derivatives, new positions and velocities at successive times can be computed. The code uses a fourth-order Hermite integrator (Makino & Aarseth, [9]) for the determination of the trajectories. This method results in an accurate integration of the equations of motion of all the stars and allows us to study the dynamical evolution of different stellar systems in great detail. On the other hand, the calculation has a $O(N^2)$ computational complexity and is therefore very demanding.

An important feature in our code is the use of the *hierarchical* or *block time-step* scheme (Makino [8]). The value of the step is computed for every particle after each force calculation, depending on the time-scale on which its orbital parameters change, and is quantized to a power of two. In this way, groups of particles are forced to share the same time-step and can be advanced at the same time. The particles sharing the same time-step are said to form a *block*.

We implemented the *ring* or *systolic algorithm* for a Hermite scheme with block time-steps using the standard MPI library package. The particles are evenly distributed among the processors during the initialization phase and the ones which need to be updated circulate among the nodes according to a virtual

[1] `http://www.cs.vu.nl/das2`
[2] `http://www.eu-Crossgrid.org`
[3] `http://lcg.web.cern.ch/LCG/Documents/default.htm`

ring topology. Partial forces are computed by the different nodes at each step and are summed up to obtain the total forces. The number of communication steps needed to compute the total forces is equal to the number of available processors. This algorithm has the advantage of minimizing memory requirements on each node.

3 Experimental Grid Setup

3.1 The DAS-2 Testbed

For our first set of experiments, we have used the Distributed ASCI Supercomputer (DAS-2), a wide-area computer which consists of clusters of workstations distributed across the Netherlands.

The DAS-2 machine is used for research on parallel and distributed computing by five Dutch universities: University of Amsterdam, Vrije Universiteit Amsterdam, Delft University of Technology, Leiden University, and University of Utrecht. The cluster at the Vrije Universiteit contains 72 nodes while the other four clusters have 32 nodes. Each node contains two 1-GHz Pentium-IIIs, at least 1 GB RAM, a 20 GB local IDE disk (80 GB for Leiden and UvA), a Myrinet interface card, a Fast Ethernet interface. The nodes within a local cluster are connected by a Myrinet-2000 network, which is used as high-speed interconnect. In addition, Fast Ethernet is used as OS network. The five local clusters are connected by Surfnet, the Dutch university Internet backbone for wide-area communication.

The MPI implementation that is used in a Globus environment is MPICH-G2, which is a grid-enabled implementation of the MPI v1.1 standard. That is, using services from the Globus Toolkit, MPICH-G2 allows you to couple multiple machines, potentially of different architectures, to run MPI applications. MPICH-G2 automatically converts data in messages sent between machines of different architectures and supports multi-protocol communication by automatically selecting TCP for inter-machine messaging and vendor-supplied MPI for intra-machine messaging. The version available on DAS-2 is MPICH-GM, which uses Myricom's GM as its message passing layer on Myrinet. MPICH-GM is based on the MPICH package from Argonne/MSU. The current version is now able to use the fast local DAS-2 interconnect (Myrinet) on the local clusters; only communication between clusters goes over TCP/IP sockets.

3.2 The CrossGrid Testbed

For our more widely distributed set of experiments, we have used the CrossGrid pan-European distributed testbed. This infrastructure combines resources across 16 European sites (Fig. 1) into a large Grid Virtual Organization. The sites range from relatively small computing facilities in universities to large research computing centers, offering a heterogeneous set of resources to test the possibilities of a widely distributed experimental Grid framework. National research

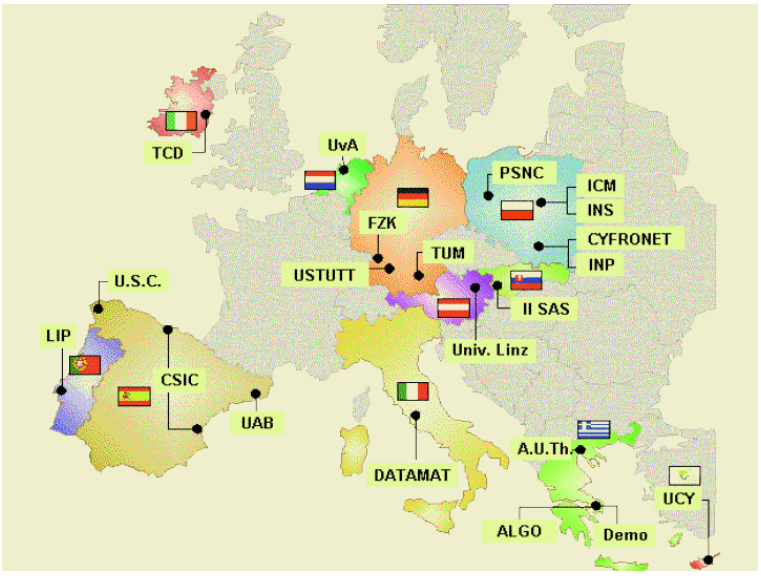

Fig. 1. Different locations in Europe participating in the CrossGrid network

networks and the high-performance European network, Geant, assure interconnectivity between all sites. The network includes a local step, typically inside a research center or university via Fast or Gigabit Ethernet, a jump via a national network provider at speeds that will range from 34 Mbits/s to 622 Mbits/s or even Gigabit, and a link to the Geant European network at 155 Mbits/s to 2.5 Gbits/s.

The CrossGrid team focuses on the development of Grid middleware components, tools and applications with a special focus on parallel and interactive computing, deployed across 11 countries. The added value of this project consists in the extension of the Grid to interactive applications. Interaction, in this context, refers to the presence of a human in a processing loop, and a requirement for near real-time response from the computer system. The CrossGrid testbed largely benefits from the EDG (Foster, Kesselman & Tuecke, [3]) experience on testbed setup and Globus (Karonis, Toonen & Foster, [6]) middleware distributions. The efforts to establish an integrated CrossGrid testbed started with the release of EDG 1.2.0; currently LCG2 is deployed in the testbed.

The CrossGrid testbed architecture and minimum hardware requirements are modeled after the LCG2 specification, with each site offering at least five system components:

- a gatekeeper that provides the gateway through which jobs are submitted to local farm nodes.
- a set of worker nodes or local farm computing nodes where jobs are actually executed; the combination of a gatekeeper with its worker nodes is usually called a computing element.

- a storage element or storage resource that includes a Grid interface ranging from large hierarchical storage management systems to disk pools.
- a user interface machine, used by end-users to submit jobs to the Grid computing elements.
- and a local configuration server, used to install, configure and maintain the above systems from a single management system.

Table 1. Sample Globus Resource Specification Language script for submitting MPI (MPICH-G2 device) jobs via Globus standard libraries, and performing file transfers and file access by Globus access to secondary storage. In the first line of the script we submit a job to a cluster in Spain requesting one processor. In the next 7 lines we set up the local environment and invoke the code called `nbodygrid` to run with N=65536 particles for one N-body time-unit. In the subsequent blocks of lines we simultaneously request one processor per cluster in Germany, Portugal and Cyprus, respectively

```
(&(resourceManagerContact="ce.grid.cesga.es:2119/jobmanager-pbs")
      (count=1)
      (label="subjob 0")(environment=(GLOBUS_DUROC_SUBJOB_INDEX 0)
      (POWER 1200)
      (DATADIR /home/cg013)
      (LD_LIBRARY_PATH /opt/globus/lib:/opt/edg/lib:/usr/local/lib)
      (GLOBUS_GRAM_JOB_CONTACT ce.grid.cesga.es:2119/jobmanager-pbs))
      (directory="/home/cg013/nbody_code/crossgrid/65536")
      (executable="/home/cg013/nbody_code/crossgrid/65536/nbodygrid")
      (arguments= "-n" "65536" "-t" "1")
)
(&(resourceManagerContact="ce010.fzk.de:2119/jobmanager-pbs")
      (count=1)
      .
      .
      .
      (arguments= "-n" "65536" "-t" "1")
      )
(&(resourceManagerContact="ce02.lip.pt:2119/jobmanager-pbs")
      (count=1)
      .
      .
      .
      (arguments= "-n" "65536" "-t" "1")
)
(&(resourceManagerContact="ce001.grid.ucy.ac.cy:2119/jobmanager-pbs")
      (count=1)
      .
      .
      .
      (arguments= "-n" "65536" "-t" "1")
)
```

The CrossGrid testbed includes a set of tools and services such as monitoring tools, development tools, a remote access server, portals and a prototype of the parallel resource broker.

Since the support of the CrossGrid resource broker for parallel applications using the MPICHG2 device was still being deployed at the time of our experiments, our job submissions were performed using the Globus job submission capabilities directly. As was the case for our experiments in the DAS2 network, the MPI package used for the tests was MPICH-G2, to allow for submission of a simulation job to a number of different sites, using different distributed processor topologies per run. Table 1 reports a sample job submission script for one run requiring 4 processors distributed among 4 different clusters.

4 Performance Results

The performance of a parallel code depends on the properties of the code itself, like the parallelization scheme and the intrinsic degree of parallelism, and on the properties of the parallel computer used for the computation. The main factors determining the general performance are the calculation speed of each node, the bandwidth of the inter-processor communication, and the network latency. In the case of a computational grid, the latency between different clusters may sensibly affect the execution times. To measure the effect of latency we performed test runs on the DAS-2 supercomputer and the CrossGrid testbed using the direct N-body code described in Sec. 2. We evolved the same initial configuration for one N-body time-unit[4] (Heggie & Mathieu, [5]), using 4 processors.

The total execution time is plotted in Fig. 2 as a function of the number of different locations hosting computing nodes. The low latency network on the DAS-2 generally results in good performance even if the nodes are allocated in different clusters. Only in the case of a very small number of particles, like for the $N = 4096$ run, the execution time increases steadily with the number of locations. This is due to an unfavorable computation to communication ratio for small N. The effects of inter-process communication are more evident for the CrossGrid runs, where the execution time generally increases with the number of locations. For large systems, however, the total time is dominated by the computation and the performance on the CrossGrid is comparable to that on DAS-2.

5 Summary and Conclusions

We have performed tests on two computational grids, the Dutch DAS-2 and the pan-European CrossGrid infrastructure. The application under consideration is a direct gravitational N-body code for simulating astrophysical stellar systems, like planetary systems, star clusters or dwarf galaxies. N-body methods are not

[4] A quantity proportional to the time needed for a typical star to cross the system.

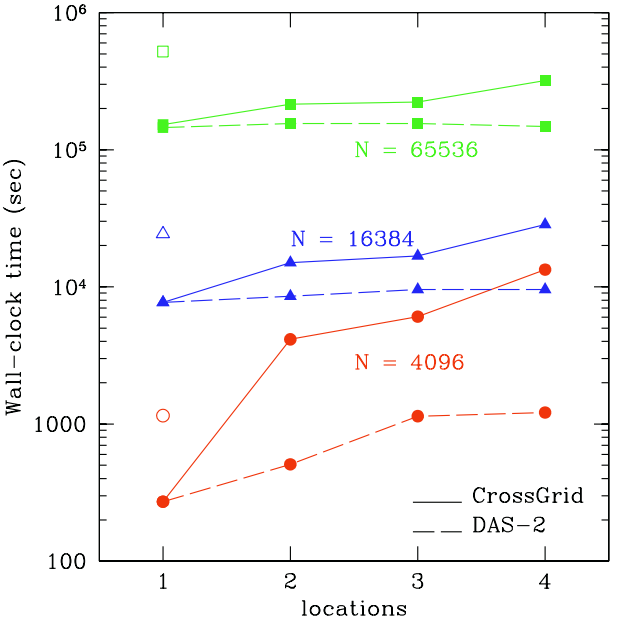

Fig. 2. Performance comparison of a direct gravitational N-body code on the DAS-2 wide-area supercomputer (dashed lines) and the CrossGrid distributed testbed (solid lines). The full dots refer to runs with N=4096, the full triangles to runs with N=16384 and the full squares to runs with N=65536. As a comparison, the empty symbols indicate the timing results in the case of a single processor on DAS-2. For all the runs we allocated 4 processors, distributed in 1 to 4 different locations. For example, in the case of one location all the 4 processors were selected in the same cluster, for two locations the processors were selected either two per location or three in one location and one in another, for four locations one processor per location was selected. The data related to one location were obtained on the DAS-2 only, which is effectively part of the CrossGrid

only applied to gravitational systems but can be used to efficiently solve a wide range of scientific problems ranging from the atomic to the cosmological scale. Applications include the study of equilibrium and non-equilibrium phenomena of microscopic and macroscopic molecular systems, equations of state and fluid dynamics. The algorithm is parallelized using a systolic scheme by means of the MPI library. For both grids we have allocated computing nodes in different locations, that is among different clusters participating in the grid network. The timing results indicate that the effects of latency are more prominent on the CrossGrid than on the DAS-2, as, for the latter, the clusters are interconnected with a faster and lower latency network. For both grids the communication effects on the performance decrease as the number of simulated particles increase. For large systems the total execution time is dominated by the computation rather than the communication and the load balance among the nodes is higher.

Acknowledgments

This work was supported by the Netherlands Organization for Scientific Research (NWO), the Royal Netherlands Academy of Arts and Sciences (KNAW) and the Netherlands Research School for Astronomy (NOVA). The authors acknowledge the DAS-2 and CrossGrid projects, for their help and support.

References

Dorband, E.N., Hemsendorf M., Merritt D.: Systolic and hyper-systolic algorithms for the gravitational N-body problem, with an application to Brownian motion. Journal of Computational Physics (2003), 185, 484-511

Foster, I., Kesselman, C.: Globus: A Metacomputing Infrastructure Toolkit. International J. Supercomputer Applications, (1997), 11(2):115-128

Foster, I., Kesselman, C., Tuecke, S.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International J. Supercomputer Applications, (2001), 15(3)

Gualandris, A., Portegies Zwart, S., Tirado-Ramos, A.: Performance analysis of parallel N-body algorithms on highly distributed systems. Submitted to IEEE Transactions on Parallel and Distributed Systems

Heggie, D.C. and Mathieu, R.D.: Standardised Units and Time Scales The Use of Supercomputers in Stellar Dynamics, (1986), 267

Karonis, N., Toonen, B., Foster, I.: MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface. Journal of Parallel and Distributed Computing (2003)

Makino, J., Hut, P.: Performance analysis of direct N-body calculations. ApJS (1988) 833–856

Makino, J.: A Modified Aarseth Code for GRAPE and Vector Processors. PASJ (1991) 859-876

Makino, J., Aarseth, S.J.: On a Hermite integrator with Ahmad-Cohen scheme for gravitational many-body problems. PASJ (1992), 44, 141-151