A Grid Supercomputing Enviroment for High Demand Computational Applications

Ingrid Barcena[*], José Antonio Becerra[ς], Joan Cambras[*], Richard Duro[ς], Carlos Fernández[♦], Javier Fontán[♦], Andrés Gómez[♦], Ignacio López[♦], Caterina Parals[*], José Carlos Pérez[♦], Juan Villasuso[♦]

Abstract

Despite the huge increase in processor and interprocessor network performace, many computational problems remain unsolved due to lack of some critical resources such as floating point sustained performance, memory bandwidth, etc... Examples of these problems are found in areas of climate research, biology, astrophysics, high energy physics (montecarlo simulations) and artificial intelligence, among others. For some of these problems, computing resources of a single supercomputing facility can be 1 or 2 orders of magnitude apart from the resources needed to solve some them. Supercomputer centers have to face an increasing demand on processing performance, with the direct consequence of an increasing number of processors and systems, resulting in a more difficult administration of HPC resources and the need for more physical space, higher electrical power consumption and improved air conditioning, among other problems. Some of the previous problems can´t be easily solved, so grid computing, intended as a technology enabling the addition and consolidation of computing power, can help in solving large scale supercomputing problems. In this document, we describe how 2 supercomputing facilities in Spain joined their resources to solve a problem of this kind. The objectives of this experience were, among others, to demonstrate that such a cooperation can enable the solution of bigger dimension problems and to measure the efficiency that could be achieved. In this document we show some preliminary results of this experience and to what extend these objectives were achieved.

Introduction

Supercomputing centers offer high performance computing resources focused to solve the highest demanding computational problems (the so called grand challenge simulations). However, and spite of the continue exponential increase in processor performance following Moore´s law, and to the development of new massive parallel computing architectures, many computational problems in different areas remain unsolved in a realistic way due to extremely long computational times or even can not be solved in any way primarly due to the lack of computational resources like available memory or permanent storage.

As in many other aspects, cooperation and resources aggregation between different institutions can be the only real solution to provide enough capabilities to solve these problems actually, offering increased computation resources. Grid computing[1] establishes some mecanisms needed to achieve this kind of computing power agregation. One of the most know projects in grid computing is the Globus project[2], which has been developed a set of software components becoming a de-facto standard during the last years.

Many different testbeds have appeared in different grid projects aimed to experiment and exploit this emerging technology. However, most of these projects in grid computing are focused in the execution of

[*] Centre de Supercomputació de Catalunya (CESCA)
[ς] Autonomous Systems Group (GSA), University of A Coruña (UDC)
[♦] Centro de Supercomputación de Galicia (CESGA)

sequential tasks, or what has been called High Throughput Computing (like the Europena Datagrid project[3]), and there are few projects focused in the execution of parallel tasks within the Grid. Examples of these experiencies include the Cactus simulation framework [13, 14] and the European Crossgrid project[4]. These parallel applications can specially benefit from the lastest developments in wide area networks in terms of bandwith and realibility, as well as the recent introduction of Quality of Service.

As a first step in this way, two supercomputing centers in Spain, namely Centro de Supercomputacion de Galicia (CESGA) and Centre de Supercomputació de Catalunya (CESCA), have initiated a series of experiences in order to explore how suitable are these technologies in a real case, parallel application, as well as the issues and benefits of employing Globus toolkit[8].

Objectives

The main objectives of this experience are:

1. Establish a grid computing environment between two independent supercomputing facilities.
2. Demonstrate that real world sequential problems can be scheduled in this grid environment and load balancing can be achieved, without any code change or system configuration.
3. Demonstrate that parallel codes can be executed in this grid environment without any code change or system configuration.
4. Check how systems heterogeneity can affect the efficiency of parallel computing.
5. Analyze how network heterogeneity and communication equipment can influence in this kind of problems and determine the Reliability, Availability and Serviciability (RAS) capabilities of the new generation wide area research networks.

What lies behind all these important objectives is a common effort to provide dedicated and high performance permanent computational resources to help solving scientific and real world problems demanding higher computational capacities.

Problem and application

For this experience, we focused on a real, non synthetic, problem application requiring high sustained floating point performance and large amounts of memory. These requirements made impossible to face such a task in any of the two supercomputing centers alone, with their own available resources. Moreover, it was also required that the code were already parallelized with the MPI message passing library, and thoroughly tested.

As this is the first experience running a distributed code involving both centers, we selected a problem with not many communication processes or, at least, in which network latency wasn't a real problem (as far as network latency can't be reduced, but rather "hidden" with different software techniques). We must take in account that even though the increasing capacities of recent wide area networks in terms of bandwidth and quality of service, latency remains as a bottleneck as far as optical links have, by now, light speed limits. For the testbed of the two supercomputer centers, there are more than 1200 kilometres of optical fiber between CESGA and CESCA and light signals need more than 8 milliseconds for a full round trip. Due to the electro-optical conversion overheads and packet routing, the measured round-trip time is almost twice bigger, in the range of 20 to 30 ms. This is more than 1000 times bigger than typical high-speed, low latency interconnect networks like Quadrics, Myrinet, or Memory-Channel interconnecting low distance computers.

It was also desirable that the problem could be easily re-sized, ideally without recompiling the code. This could result in better resource provisioning, specially taking into account that production systems were used for the final tests (we will demonstrate later that small test-systems results can't simulate the results of the big case problem in these enviroments). With this characteristic, we could go through a 3 phase testing: first check for a small and simple case, so that the message passing mechanisms and the network performance could be tested for many hours. Second, a mid-size test, in which production systems were tested and finally a full-size test for the real problem.

After inquiring users of both supercomputing facilities, a self deployed Artificial Intelligence application[5, 6], SEVEN, was selected as the most adequate code according to the characteristics needed for this experiment.

This code has all the characteristics needed to run smoothly in our configuration:

- It is a well tested parallel application and the final users have a deep knowledge of its internal structure.
- The problem solved by this application demands HPC resources in terms of processor performance and memory, and can be easily resized to solve different size problems.
- Even though in each iteration dozens of megabytes of data have to be transmitted, these transmissions only happen after a significant amount of computing time, i.e., the computing to communications time ratio is aproximately one hundred of times bigger.

Objectives of the simulation

The objective of the simulation is to obtain a controller for an autonomous robot Pioneer 2-DX. The purpose of this controller is to provide the robot with a given behaviour: this robot (R) is located in a square room with other two robots. One of these two robots is a pursuer and the other one is a prey. Robot R must escape from the pursuer and follow the prey. The pursuer and the prey don't have any special difference apart from their movement speeds, so that temporal information is necessary in order to distinguish both robots because they can only be differentiated by checking the way in which they move. The controller must implement this autonomous behaviour with the information provided by the sonar sensors of Robot R.

Introduction to evolutionary algorithms

Evolutionary algorithms employ an analogy with the evolutionary process and the search of a solution in a given space. Basically, there is an initial population of individuals. Each individual is a possible solution of the problem (a point in the solutions space) and is characterized by one or more chromosomes (each chromose codes the solutions space). This population evolves with the time: the individuals combine with each other and suffer mutations in the genes of their chromosomes, so that the possibilities to survey in the next generation are proportional to how near they are to the solution of the problem. The way in which the individuals to be reproduced are selected and how they are combined leads to different kinds of evolutionary algorithms.

We have used a Macroevolutionary Algorithm[7], which is a new type of evolutionary algorithm where, instead of using the word “individual” to refer to a candidate solution, we talk about “species”. So, a survival coefficient is assigned to each specie and the species with the worst survival coefficient become extinct and new species, derived from those that become extinct and from survivals, take their place. There are two

modifications with respect to the original algorithm. First, as the fitness measure is not determinist (due to the stochastic behaviour of some parameters in the simulation), survival species are evaluated once more in each generation. Second, if the fitness for a given specie changes more than a given percentage after doing that extra evaluation, the number of evaluations for new species is incremented.

For our problem, the individuals are the candidate controllers, implemented using Artificial Neural Networks (ANNs), for a given behaviour of the robot. The genes of the chromosome are the parameters that describe those ANNs. To know how far away an individual is to the solution of the problem, the behaviour of the robot with the controller corresponding to that individual genes is checked in a simulated environment, and a quality parameter is given depending on how good that robot implements the behaviour that the designer wanted to obtain.

Problem description

In this case, the size of the global population is 64000 individuals, grouped in 64 races, 1000 individuals each one. The size of the population depends on the chromosome size and the complexity of the search space. Each individual has a chromosome with 17301 genes. Each chromosome codes an artificial neural network with an input layer of 17 neurons, 2 hidden layers of 64 neurons each one and an output layer of 2 neurons. The ANN has gaussian synapsis and, there are delays between the neurons of the input and the first hidden layers. All the parameters in the ANN (Gaussian synapsis parameters, delay values, neurons biases and the slope of the neurons activation) are constants during the evolutionary process. Each individual is evaluated 8 times in a simulated environment for a period of time of 300 steps (equivalent to one minute of real time).

Code description: parallel implementation

To implement the asynchronous transfer of data, in every generation and just before checking if there must be a migration of the individuals, the next code is executed in order to check if any race has sent data (best individual, the number of the working generation, etc.). Basically, if there is any data, it is read and placed where needed:

```
for (int i = 0; i < evolAlg->races; i++)
  if (i != race)
    while(commMaster.Iprobe(i, 1000))
    {
      commMaster.Recv(data, dataElements, MPI::DOUBLE, i, 1000);
      evolAlg->lastWriteGeneration = (evolAlg->lastWriteGeneration >
       (int)data[0])? evolAlg->lastWriteGeneration : (int)data[0];
      dataGen = (int)data[1];
      evolAlg->currentGeneration[i] = dataGen + 1;
      pos =  i * evolAlg->generations + dataGen ;
      evolAlg->totalFitness[pos] = data[2];
      evolAlg->bestFitness[pos] = data[3];
      bestInd = evolAlg->bestIndividual + (pos) * evolAlg->genes;
      for (int j = 0; j < evolAlg->genes; j++)
```

```
            bestInd[j] = data[j + 4];
        }
```

The next part of the code runs in every generation after checking if is necessary to migrate (and do the migration if needed) and after updating the information about the actual state of the evolution and after writing to file all the relevant data. The data is sent using buffers to continue the work immediately.

```
for (int i = 0; i < evolAlg->races; i++)
      if (i != race)
      {
        data[0] = evolAlg->lastWriteGeneration;
        dataGen = evolAlg->currentGeneration[race] - 1;
        data[1] = dataGen;
        pos =  race * evolAlg->generations + dataGen;
        data[2] = evolAlg->totalFitness[pos];
        data[3] = evolAlg->bestFitness[pos];
        bestInd = evolAlg->bestIndividual + (pos) * evolAlg->genes;
        for (int j = 0; j< evolAlg->genes; j++)
          data[j + 4] = bestInd[j];
        commMaster.Bsend(data, dataElements, MPI::DOUBLE, i, 1000);
      }
```

Grid configuration: systems and networks

Both centers have many different supercomputing servers: from vector parallel computers to symmetric multiprocessing systems. For this test, the most powerful supercomputer of each site was used: 2 Hewlett Packard HPC320 supercomputers.

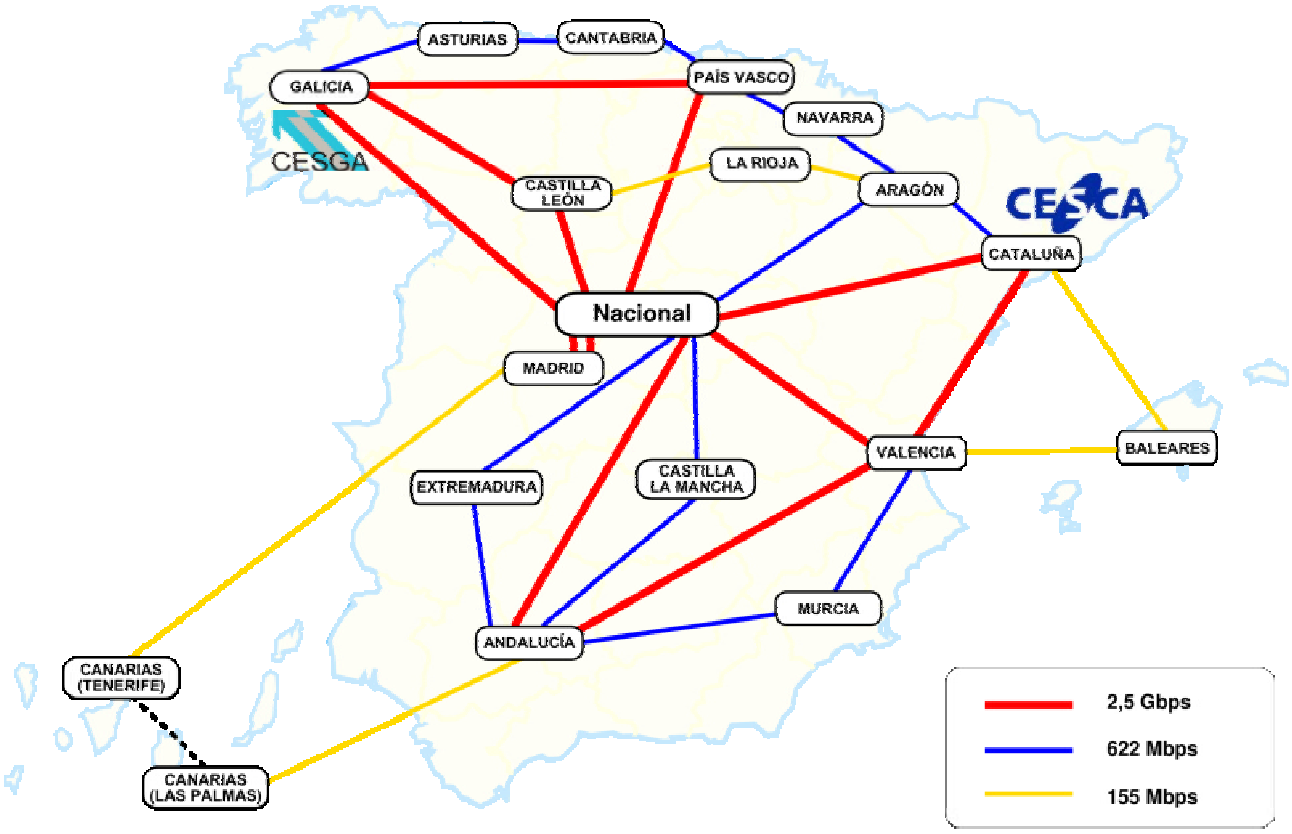

Figure 1. RedIris-2 network, showing the links speed and the posible routes between two points. CESGA is located at the northwest of Spain and CESCA at the northeast, with more than 1200 kilometers from each other. The shortest path between them has 3 hops with 2,5 Gbps links.

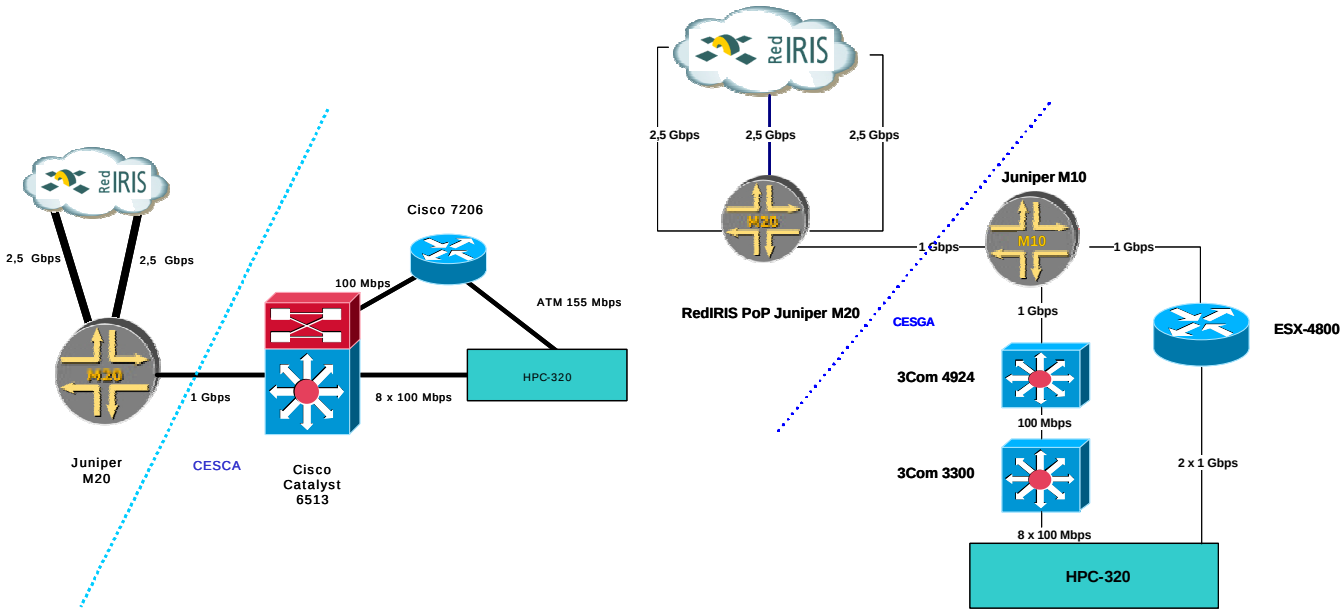

Figure 2. Diagram showing the internal LAN between the HPC320 at CESCA (left) at CESGA (right side) from the local area to the RedIris network.

CESGA´s HPC320 system runs Tru64 v5.1A and has 8 computing nodes ES45 with 4 Alpha EV68 CPUs at 1GHz in each node. 6 nodes have 8GB of memory and the other 2 have 16GB per node. Globally, the system hosts 32 CPUs, 24 with 2GB per CPU and 8 with 4GB per CPU. The nodes are internally connected with a high speed and low latency Memory-Channel II with 2 rails and 100MB/s bandwidth per rail, configured in fail over mode. This supercomputer has a peak performance of 64 GFlops and achieved 45 GFlops in the linpack benchmark. Two nodes have gigabit-ethernet and six nodes have fast-ethernet links for the public internet connection.

CESCA´s HPC320 system is quite similar to CESGA´s one: runs Tru64 v5.1A and has 8 computing nodes ES40 with 4 Alpha EV68 CPUs at 833MHz in each node. Six nodes have 2GB of memory and the other 2 have 4GB per node. Globally, the system hosts 32 CPUs, 24 with 512 MB per CPU and 8 with 1 GB per CPU. The nodes are internally connected with a high speed and low latency Memory-Channel II with 1 rail having 100MB/s bandwidth. The peak performance of this supercomputer is 53,31 GFlops and achieved 40 GFlops in the Linpack Benchmark. The 8 nodes are linked to the public network with one Fast-ethernet link in each node.

These two supercomputers together offer a peak performance of 117,31 GFlops, 85 GFlops sustained, 64 processors, 100 Gbytes of memory and 4 Terabytes of disk space. Building a grid with both systems is simplified because they share the same CPU architecture and operating system.

CESGA is located in the northwest of Spain and CESCA in the northeast of Spain. The National Research Network, Rediris2, has recently improved its infrastructure introducing multigigabit and fail-over mechanims in all their point of presence in the country (Figure 1). As shown in this figure, there are more than 10 different routes from CESGA to CESCA: directly through Madrid or going through Castilla Leon, or through Aragon, etc... Depending on the load and availability of the links, network traffic can go through different hops. However, given the actual load and the performance required for this test, it was determined that going directly through Madrid involving the less number of hops resulted in lower connection latency. For this test the network connection was forced to go directly through Madrid. This line is over 1200 kms long and the mean measured RTT is 20ms. The high bandwidth links could result in a theoretical peak performance between both supercomputing centers of more than 5 Gbps or more than 600MBytes/s, equivalent to the typical high bandwidth local interconnections, which could be exploited in next, more network oriented, experiments.

Software components

Two major components were used for this simulation: Globus (version 2.2.4) and MPICH-G2[9, 10, 11] (version 1.2.5-1a). Besides, taking into account that both systems were in production while performing the initial tests, it was needed to install the corresponding Globus job managers for each site: the "Pro" version of the Portable Batch System (PBS Pro v. 5.2) in the case of CESGA, and the Load Sharing Facility (LSF Base version 5.0) in CESCA.

Standard user accounts were used for the test, and Redegrid[1] Certification Authority provided the user and host certificates needed by Globus software.

[1] Redegrid stands for Galicia´s network on parallel and distributed computing and Grid technologies. Seven research groups and CESGA collaborate on this network.

The initial experiences involved running simple calculations like the standard MPI test codes and the Pallas MPI Benchmarks[12]. For these initial tests, test systems provided by HP were used in order to achieve the necesary knowledge and experience to implement this configuration in the production servers.

Code Execution

Minor changes had to be incorporated in the configuration of the systems and network topologies in order to efficiently run the application (obviously this was required to work with production and stable platforms).

One of the changes to be applied involved the network configuration of CESGA´s HPC320. In this system only 2 nodes had public IP addresses. The other 6 nodes had to be connected to the public internet network establishing a link between the internal LAN switch 3Com 3300 with an external switch 3Com 4300, and public IP addresses were configured in these interfaces. No changes were needed in the rest of the network topology, neither in terms of routing configuration or quality of service, as the available bandwidth was far enough for the needs of this particular case. Only firewall configuration in both supercomputing facilities was relaxed to open completely the connection between both supercomputers. Regarding the TCP stack configuration, no change was applied as the available bandwidth with the standard parameters was enough. However, this point and other parameters related to the Quality of Service (QoS) will have to be reviewed in order to execute intensive message passing applications in this grid.

No changes were required in the code. Each race evolved independently and the only communications between them were the migrations, where the best individual of each race is copied to the other races. To do so, a high capacity buffer for the transmission and reception of messages was implemented in order to deal with the high latencies of the network and especially the different processor speeds. With this mechanism the computations in every processor evolved in a decoupled way, asynchronously, so that each process checks the buffer in every step to get new data from the rest of the processes and continues computing. . Using this implementation, nearly 100% of user CPU was consumed by the application during most of the run. Only the first step (or generation) was synchronous, forcing that all processors had to wait for the slowest to finish, and a lower percentage of CPU efficiency. This behavior must be related with the implementation of the used mpi software, as there is no such synchronization requirement in the program. After this initial step the rest of the simulation runs asynchronously.

One of the first observed problems was that the application could run without problems with up to 16 processors in each site, but failed to pass the DUROC startup phase when the number of processors involved was increased. After some debugging and having checked the increasing number of sockets in use with the number of processors, we suspected about filtering and firewalling rules within the RedIris network. We solved to change the port range of the MPICH-G2 protocol to use port numbers from 51000 to 52000, as RedIris doesn´t actually block these ports. With this strategy this problem disappeared completely.

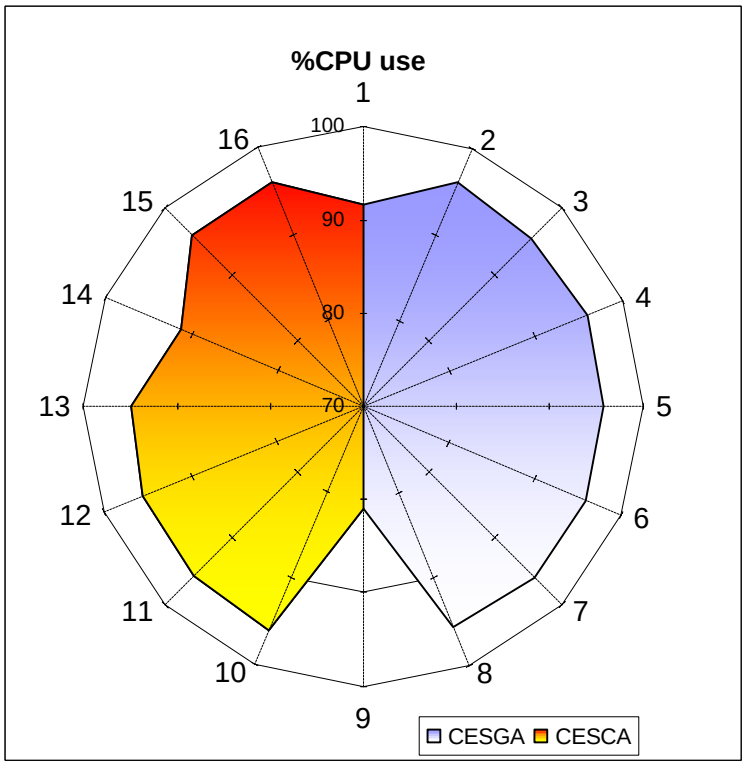

Figure 3. Kiviatt diagram showing the mean CPU usage of the SEVEN code, for the whole execution of the simulation on each node. Nodes 1 to 8 belong to CESGA´s HPC320 (blue color), meanwhile nodes 9 to 16 belong to CESCA´s HPC320 (red color). Most of the nodes have 95% CPU use, with the exception of the front-end nodes, primarly because they had to deal with extra computation managing the jobs.

The code was submitted with an RSL from CESGA´s system, distributing 32 jobs in each site. Once all the processes were started by the corresponding queue system, the simulation started and run for more than 9 hours. In the next paragraphs we describe and represent some of the measurements collected during the execution.

Observed Performance

We define the efficiency as the ratio of computing time spent by the application and the total time. The global efficiency of the simulation achieved on both systems during the whole run was 94.3%. The kiviatt diagram in figure 3 represents the efficiency achieved by each individual node of the HPC320 systems. Nodes 1 to 8 belong to CESGA´s HPC320 and nodes 9 to 16 belong to CESCA´s HPC320. The diagram shows that CESGA´s nodes have better CPU efficiency (95.3% vs. 93.2%). CESCA´s lower efficiencies are mainly due to the lack of memory and possible memory pages throttling. Specially node number 9 is the one which runs the Globus job-managers processes at CESCA and is the node with less CPU efficiency. Running the job-managers has two direct consequences: First, the processors have an extra load and consequently less CPU cycles available for the simulation, and second, there is less memory free for the application (aproximately 224 Mbytes of memory are consumed by the job-managers). This node also hosts the LSF manager and other cluster services, consuming extra CPU cycles and memory available for the simulation code.

Every step or generation took on average, 17 minutes at CESGA and 22 minutes at CESCA. This ratio of CESCA nodes being 1.29 times slower that CESGA´s is equivalent to the ratio in processor speeds (833MHz vs. 1000MHz) and different efficiencies achieved in each system. The first step needs exactly twice as much time (34 and 44 minutes) because the whole population is evaluated before entering the loop, composed by the selection, reproduction, evaluation and migration (if needed) steps.

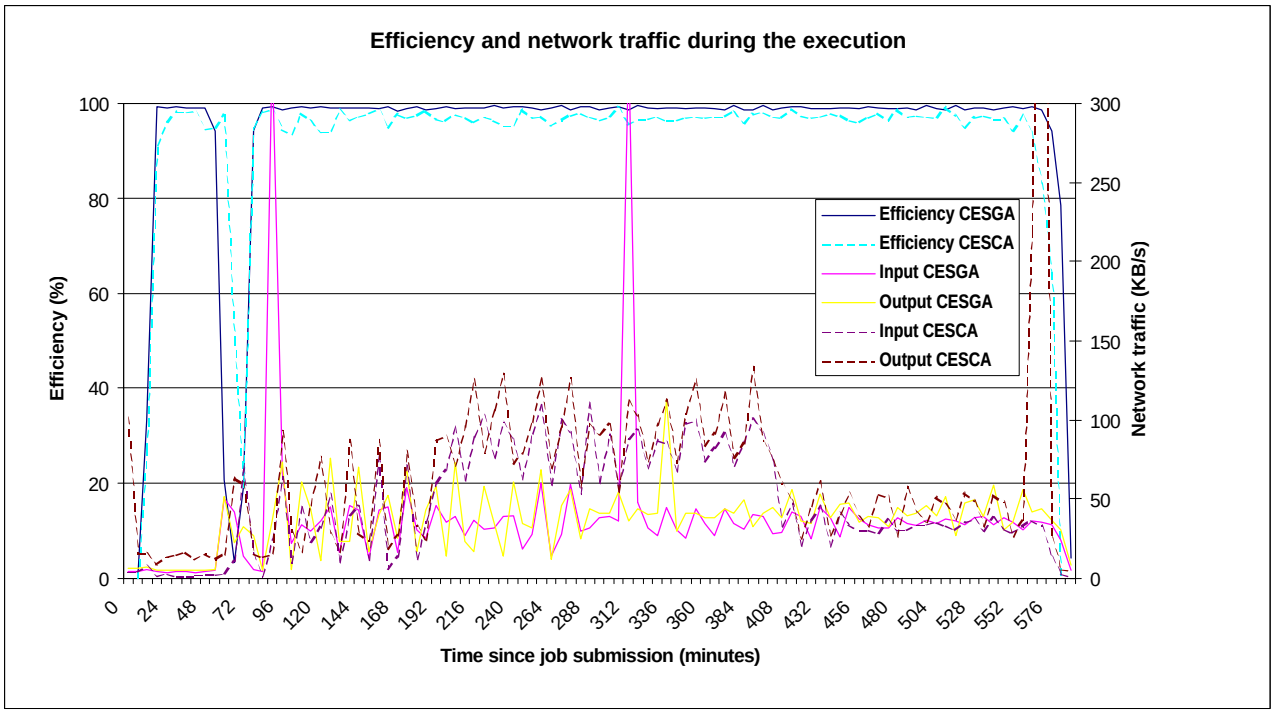

Figure 4. Evolution of the efficiency (left scale) and network input/output (right scale) measured in 6 minutes intervals.

In figure 4 we represent the evolution of the efficiency measured in 6 minutes intervals. In the horizontal axis we represent the time in minutes since the jobs were submitted to the queue systems and the vertical axis represents the efficiency achieved on this time interval. We must note some important points:

1. After job submission, jobs are queued by the respective job management software (PBSpro & LSF), and after a short delay are submitted for execution in the corresponding node.
2. When every job starts executing, the MPI_Init waits until all the processes participating in the simulation have reached that point.
3. Once all the processes have reached that point, the execution continues and all processes consume almost a 100% CPU, starting the simulation. This occurs at minute 18 of the execution.
4. The first step needs twice as much time than the rest of steps, that means 34 minutes on CESGA and 44 minutes on CESCA, so after 52 (18+34) minutes there is a low in the efficiency at CESGA (shown at point 54), and 10 minutes later appears the one at CESCA (at point 62: 44+18), shown in the graph at point 66.
5. When all the nodes reach the end of the first step or generation, the first node of CESGA´s system still has processing on it: there is one CPU in this node still working on the code.

In the graph of figure 4 we also show the network traffic of the individual nodes. This traffic in divided in 4 categories. Input CESGA and Output CESGA represent input and output network traffic on CESGA´s nodes.

Input CESCA and Output CESCA represent input and output network traffic on CESCA´s nodes. We must note that this network traffic includes intranode communications of a single HPC320 system. The total data transfer measured at CESGA´s router between both supercomputers was 8.3 Gbytes, divided in 4.6 Gbytes in the way from CESGA to CESCA and 3.7 Gbytes in the way from CESCA to CESGA (the previous graph represents ALL the data exchanged, including HPC320 TCP intercommunication).

Each process needs 963 Mbytes of virtual memory, and the observed resident size (RSS) was 300Mbytes at CESCA and 600Mbytes at CESGA (we must note how the lack of memory could adversely affect performance at CESCA´s nodes).

Application Results

In the next paragraph we describe some preliminary results from the simulation. In figure 5, a graph showing the evolution of the quality of the individuals in every generation is shown, which represents how the evolutionary algorithms progress in the right direction, with the peaks representing migration of the individuals among different populations. As we have said before, each race evolves separately and when a migration happens the better individual of each race is copied to every other race replacing the worst individuals. The fact that the fitness of the best individual raises when migration has taken place, means that the slowest race (the one that writes the fitness information in a file) receives better individuals from other races thanks to migration. The fact that fitness goes down afterwards means that the initial number of evaluations for each individual is not enough to obtain a reliable fitness measure, which is not a very big problem because of the explained autoincrement mechanism for the number of evaluations.

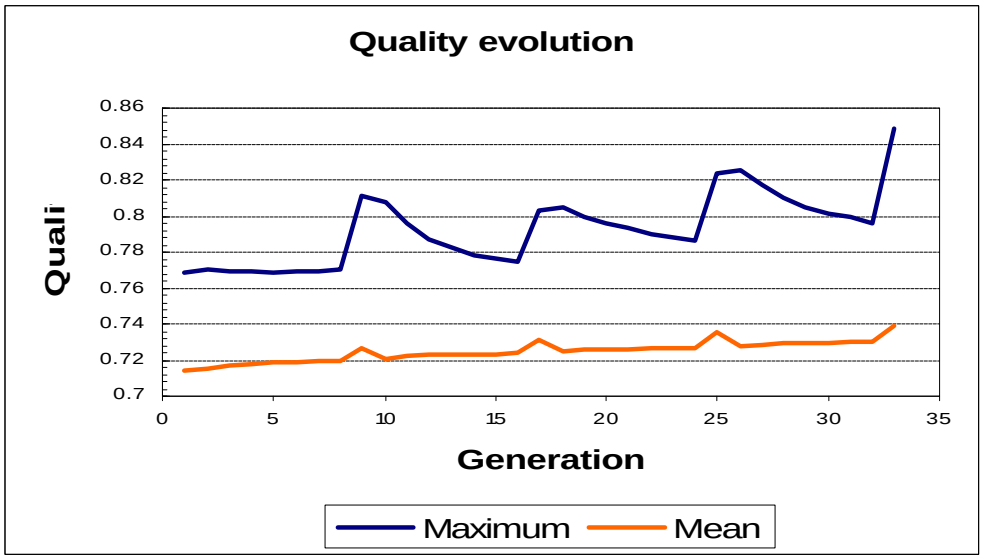

Figure 5. Quality evolution of the application, during the whole simulation (35 generations in total). The mean quality shows a continuos increase, while the maximum has peaks caused by the migration of the individuals every 5 generations.

Summary and Future Plans

This work demonstrates that collaboration between supercomputing facilities can enable more demanding problems to be solved. Scalability has been demonstrated under the premise that computing and network traffic can be decoupled. A real problem can be solved using this infrastructure without major changes in the

code, which results in a more attractive solution for the software programmer. Software infrastructure (mainly Globus components) has proved its maturity and only minor changes had to be configured. From the systems and networks administrator's point of view, minor changes had to be addressed and for many problems the actual infrastructure and configuration can provide the resources needed with minimal changes. New generation wide area network infrastructures have proven their adequacy in terms of reliability and bandwidth to safely and optimally interconnect spread computing resources, and in the future will be the catalyst to execute new applications in a distributed and grid computing environment.

However, some points remain to be solved. A global scheduler for load balancing and optimal allocation of resources, optimal tuning of network parameters, better quality of service implementation and a hierarchical mechanism to communicate within separate clusters will be needed when an increasing number of systems and processors are available in each site, and message passing intensive applications need to be executed using all these disperse distributed resources. Also, the heterogeneity in terms of processors architectures and speeds, available memory and network topology, latency and bandwidth will keep on being a handicap for parallel applications in a grid computing framework. Meanwhile, only applications with the characteristics previously mentioned, can be grid-enabled without major efforts.

With the knowledge learnt in this experience, in the future similar runs would be performed with some changes: first, the node with the jobmanagers should be out of the computing part, specially if memory is a problem. Also, a minor number of jobmanagers should be used. Also, TCP stack parameters tuning should be carefully adapted to speed up network concerned problems.

Acknowledgements

We would like to thank Hewlett Packard for providing us with the required test computing platforms and their continuous support during this experience. We would like to thank also to Spain´s National Research Network, RedIris, for their cooperation and efforts in providing the high performance and reliable network infraestructure to develop this work.

About the Supercomputing Center of Galicia (CESGA)

Created in 1993, CESGA is an Information Technology company providing services to the research and development community. CESGA's primary aims are: to provide high performance computing and communications resources and services to the scientific community of Galicia and CSIC and to promote the use of state-of-the-art, high performance information and communications technologies applied to research within the scientific community of Galicia. CESGA provides high performance computing, storage and networking services to the scientific community. CESGA runs Galicia's high capacity research network and the Neutral Internet Exchange of Galicia (GALNIX). CESGA also hosts the RedIRIS node in Galicia (NW Spain) *www.cesga.es*

About the Centre de Supercomputació de Catalunya (CESCA)

CESCA was created in 1991 and is a joint effort of the Generalitat de Catalunya, the Fundació Catalana per a la Recerca, Catalonian public universities and CSIC. Is aims are centered in three areas of activity: high performance computing services, the management of the network Anella Cientifica and the Catalonian Neutral Internet Exchange (CATNIX) and the node of RedIRIS in Catalunya, as well as the promotion of the use and benefits of these technologies. *www.cesca.es, www.catnix.net*

References

[1] I. Foster and C. Kesselman, editors. The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann Publishers, 1999.

[2] http://www.globus.org

[3] http://www.eu-datagrid.org

[4] http://www.eu-crossgrid.org

[5] R. J. Duro, J. Santos, J. A. Becerra, "Evolving ANN Controllers for Smart Mobile Robots", Future Directions for Intelligent Systems and Information Sciences, pp. 34-64, Springer-Verlag, 2000.

[6] J. Santos, R. J. Duro, J. A. Becerra, J. L. Crespo, F. Bellas, "Considerations in the Application of Evolution to the Generation of Robot Controllers", Information Sciences, vol. 133, pp. 127-148, Elsevier, 2001.

[7] Marín, J. and Solé, R. V., "Macroevolutionary Algorithms: A New Optimization Method on Fitness Landscapes", IEEE Transactions on Evolutionary Computation 3, 4, pp. 272-286, 1999

[8] I. Foster and C. Kesselman. Globus: A toolkit-based grid architecture. In [1], pages 259-278.

[9] MPICH-G: I. Foster and N. Karonis. A grid-enabled MPI: Message passing in heterogeneous distributed computing systems. In Proceedings of SC'98. ACM Press, 1998.

[10] Wide-Area Implementation of the Message Passing Interface, I. Foster, J. Geisler, W. Gropp, N. Karonis, E. Lusk, G. Thiruvathukal, and S. Tuecke, Parallel Computing,,24(12):1735-1749, 1998

[11] MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, N. Karonis, B. Toonen, and I. Foster, Journal of Parallel and Distributed Computing (JPDC), Vol. 63, No. 5, pp. 551-563, May 2003

[12] http://www.pallas.com/e/products/pmb/

[13] G. Allen, T. Goodale, and E. Seidel. The cactus computational collaboratory: Enabling technologies for relativistic astrophysics, and a toolkit for solving pdes by communities in science and engineering. In 7th Symposium on the Frontiers of Massively Parallel Computation-Frontiers 99, New York, 1999. IEEE.

[14] Supporting Efficient Execution in Heterogeneous Distributed Computing Environments with Catus and Globus, G. Allen, T. Dramlitsch, I. Foster, N.T. Karonis, M. Ripeanu, E. Seidel, and B. Toonen, Proc. SC01 (SC2001), no page numbers available, Denver, CO, November 10-16, 2001.