I-Cluster: Intense computing with untapped resources

Bruno Richard, Philippe Augerat[1]
HP Laboratories Grenoble
HPL-2002-81
April 4th , 2002*

E-mail: bruno_richard@hp.com, philippe.augerat@imag.fr

cluster, scientific computing, peer-to-peer, supercomputing

I-Cluster is a research initiative from HP Labs Grenoble in partnership with INRIA (ID-IMAG Laboratory). It provides a distributed Peer-to-Peer framework of tools that transparently take advantage of unused network resources and federate them to crystallize into specific virtual functions. These functions are compute-intensive services, which are statically decomposed to fit the I-Cluster framework, then dynamically recomposed upon users needs. Typical I-Cluster functions are supercomputing, content rendering, content distribution. The I-Cluster experimentation platform, based on 225 typical desktop machines has been benchmarked as the 385th most powerful supercomputer in the world using Linpack, making it the first cluster based on mainstream technologies to enter the TOP500 in May 2001.

This paper introduces the I-Cluster initiative and its overall architecture, and describes the prospects addressed by the research team. These include scaling conventional clustering tools to large number of machines, solving Peer-to-Peer computing security issues using OS sandboxing, self-organization and resilience to unanticipated disconnections of a large and heterogeneous community of computers, and automatic resource collection.

[1] INPG, ID Laboratory, Grenoble, France

I-Cluster: Intense computing with untapped resources

Bruno Richard
hp *Laboratories Grenoble*
bruno_richard@hp.com

Philippe Augerat
ID Laboratory
philippe.augerat@imag.fr

Abstract

I-Cluster is a research initiative from HP Labs Grenoble in partnership with INRIA (ID-IMAG Laboratory). It provides a distributed Peer-to-Peer framework of tools that transparently take advantage of unused network resources and federate them to crystallize into specific virtual functions. These functions are compute-intensive services, which are statically decomposed to fit the I-Cluster framework, then dynamically recomposed upon users needs. Typical I-Cluster functions are supercomputing, content rendering, content distribution. The I-Cluster experimentation platform, based on 225 typical desktop machines has been benchmarked as the 385th most powerful supercomputer in the world using Linpack, making it the first cluster based on mainstream technologies to enter the TOP500 in May 2001.
This paper introduces the I-Cluster initiative and its overall architecture, and describes the prospects addressed by the research team. These include scaling conventional clustering tools to large number of machines, solving Peer-to-Peer computing security issues using OS sandboxing, self-organization and resilience to unanticipated disconnections of a large and heterogeneous community of computers, and automatic resource collection.

1. Problem statement

A lot of computer resources are available on enterprise networks nowadays, often used as commodity tools for user productivity. As nearly each worker has his own individual computer, there are lots of machines connected on the network. Users typically spend just few hours a day using their machines, so there are idle periods where the resources such as processing power, memory, and connectivity remain unused. The I-Cluster research project aims at building a framework that gathers such idle resources from an intranet and federates them into a system able to perform compute-intensive tasks.

Typical I-Cluster functions are supercomputing, content rendering, content distribution. Additional requirements for the I-Cluster project are that the framework smoothly adapts to the existing infrastructure; it transparently takes advantage of each existing machine of the network without requiring user actions or system changes; it adapts to machine and network heterogeneity and tries to make an efficient overall use of the community resources available at execution time; it is self-organized hence does not require administrator intervention; it has some resilience to environment changes such as unexpected connections and disconnections; and it leverages standard cluster applications (such as PovRay or Linpack) without code modifications required.

2. Earlier work

Distributed computing can be defined as "a computer system in which several interconnected computers share the computing tasks assigned to the system" [1]. Such systems include computing clusters, Grids and global computing systems gathering computing resources from individual PCs on the network. Several distributed computing frameworks already exist that fit in this picture.

Cluster computing is a type of infrastructure that has been popularized by NASA's project *Beowulf* [2]. The main goal of the project was to build a scientific computing system based on COTS (*Components off the shelf*) i.e., based on standard PCs interconnected by Ethernet. This project has had an instant success, and paved the way to lots of other academic or commercial efforts. However, a Beowulf system remains centralized and managed, which makes it fall apart from the I-Cluster requirements.

Mosix [3] is a project that started decades ago in the Hebrew University of Jerusalem. It builds a computing farm based on machines of the network, and then offers some load balancing using migration of tasks between computing nodes. Mosix is taking advantage of the Unix operating system and adds some kernel-level components that closely work with the system, particularly by handling the OS objects such as open files or sockets. Mosix can be used on a network so that the computing resources are globally shared, so it fits well our needs on this side. Moreover, the system is able to dynamically adapt to dynamic changes in the availability and load of resources. On the other hand, Gillen *et al.* [4] have shown that the proportion of user systems based on a Microsoft Windows operating system is between 91 and 96% of the number of installed computers. As Mosix is limited to the Unix environment, its acceptance by common users is therefore low. So it is important for I-Cluster to be able to use resources from idle Windows machines without limiting to Unix systems support. For the same reason, systems such as AMOEBA [5] or Glunix [6] do not fit the I-Cluster vision.

The Berkeley NOW project is building system support for using a network of workstations (NOW) to act as a distributed supercomputer on a building-wide scale. This vision, introduced by Anderson *et al.* [7], has been implemented by several NOW systems. This family of systems offers a single system image to a collection of individual computers from a network. The interconnection between the workstations is a high bandwidth communication link, and each of the individual computers is a powerful machine. As NOW systems are based on COTS (commercial off-the-shelf) components produced in large volumes, they offer an excellent price/performance ratio. However, I-Cluster needs to adapt to *already* available resources and network, so we cannot assume high performance from the computers nor from the interconnection links. For this reason, NOW is not a good candidate for I-Cluster's infrastructure.

The NetSolve project [8] aims at harnessing disparate computational resources and aggregate their power into a scientific computing machine. A NetSolve server is responsible for dialog with NetSolve agents installed on each participating computer from the network. User can submit jobs to the server, which will be distributed to available computers according to their capabilities and load. Although the system is able to adapt to job requirements, it is not possible to aggregate the power of separate machines for a single job (although an actual cluster of machines may be declared as a single NetSolve computing resource). As I-Cluster required being able to aggregate separate computers and use them to compute fine-grained supercomputing applications we did not base our research on it.

Condor [9] is a system that fits well the I-Cluster requirements. Condor is a HTC (*High Throughput Computing*) environment that is able to distribute computations on idle remote workstations, making an efficient use of the available network resources. Condor uses a publishing system where each participating computer will expose its computing capabilities and available resources to the server. When a user starts a job, Condor will transparently find the Computer that most tightly fits the job, allocate it to the job and delegate the execution. However, Condor is oriented towards environments where Computers are powerful machines, and allocates jobs onto discrete machines. It cannot aggregate resources from several distinct computers and use them as a single resource for performing a computation, except for the embarrassingly parallel class of applications, and for instance for the master-slave class. One of the goals of I-Cluster is to be able to perform scientific i.e., *fine-grained*, jobs, which requires being able to aggregate resources from several computers, and make an overall use of them so that the network latencies and load are minimal between aggregated machines. This is a limitation of Condor regarding its use for I-Cluster. Another limitation is that Condor runs applications in the user environment i.e., typically windows, although traditional computing applications are natively developed for Unix systems. Hence applications have to be ported to Windows to execute in such cases.

Global Computing, also called *Metacomputing* or *Desktop Computing*, brings processing scalability to the picture through the aggregation of resources of large number of individual Internet PCs. The typical use of Distributed Computing requires applications which are run in a proprietary way by a central controller. Such applications are usually targeting massive multi-parameters systems, with long running jobs (months or years) using peer-to-peer foundations. One of the first widely visible Distributed Computing events occurred in 1994, where Atkins, Graff, Lenstra and Leyland [10], with the help of about 1600 Internet computers owned by 600 participants, broke the *RSA challenge* between August 1993 and April 2nd 1994 using a Global Computing approach. This made people realize how much power can be available from idle Internet PCs. SETI@home [11] is a scientific research project aiming at building a huge virtual computer based on the aggregation of power offered from internet-connected computer during their idle periods. The project has been widely accepted and rose a huge raw processing power (several dozens of Tflop/s) from more than 3 million internet-connected computers. Although collection of network resources and aggregation of them to solve a global problem is a goal that SETI@home shares with I-Cluster, we did not want to limit I-Cluster to a given limited set of jobs, and wanted to keep it possible for users to write their own specific jobs and execute them on an I-Cluster platform. The architecture of SETI@home involves execution of jobs in a master-slave mode, hence requires a very specific adaptation of the problem to the platform. In particular, inter-nodes communication is not possible on SETI@home, as all the communication goes to or from the central server. Other systems such as Avaki, Entropia, Parabon, United Devices, or distributed.net are other examples of successful Global Computing systems. However, as scientific computing requires inter-node communication, Global Computing is not a good candidate for I-Cluster.

Grid computing is another concept that has been first explored in the 1995 I-WAY experiment [12], in which high-speed networks were used to connect high-end resources at 17 sites across North America. Out of this activity grew a number of Grid research projects that developed the core technologies for “production” Grids in various communities and scientific disciplines. The efforts on Grid technology are now concentrated around the *Globus* project [13]. A Computing Grid can be seen and used as a single, transparent Computer. The user logs in, starts jobs, moves files and receives results in a standard way. However, Grid computing requires heavy coordination and federation of resource providers and consumers.

Self-organization and self-sustaining required by the I-Cluster system hence make Grid or Globus systems too heavy.

3. Approach

The approach we developed for I-Cluster was to build our own system, based on an architecture derived from our knowledge of existing systems.

The basic idea behind I-Cluster is that it dynamically keeps track of available network resources, and is able to aggregate some of them into a virtual cluster when a computing service is invoked, in a way that the allocated virtual cluster nicely matches the resource set required by the service for its execution. It then proceeds into instantiation of the service onto the allocated computers. For instance, I-Cluster will transparently keep track of one thousand machines connected to a network, their individual capabilities, availability and connectivity. Then upon request I-Cluster can use this information and find a set of 8 machines with given individual capabilities and given interconnection requirements, which allows for fine-grained services to be executed.

I-Cluster cloud

The heart of the system is the *I-Cluster cloud*, which is a Peer-to-Peer community of devices with the capability to communicate and share information in a transparent way. The cloud adapts to changing conditions such as network partitions, disconnections, or relocations of mobile devices onto the network. The cloud is able to integrate new devices added to it, forget old/unused devices, and dynamically keep a model of the network interconnection between devices, as well as a directory of available machines and their resources. The cloud is based on decentralized algorithms, where each device from the cloud has its own view of the cloud; no device has complete information about the system state; and devices make decisions based only on local information, so that the failure of one machine does not impact the algorithms and the community is resilient. Horizontal scalability i.e., adaptability of the I-Cluster cloud to a large number of devices (we target 10,000 devices clouds), requires that the algorithms order remains logarithmic or linear at worst. Also the algorithms must adapt to the capabilities of each device; a PDA with limited CPU, RAM, storage, connectivity which is a member of an I-Cluster cloud will take less responsibility in the community and consequently will route less messages.

The I-Cluster cloud is built on a pure Peer-to-Peer class of algorithms i.e., without requiring any centralized server. The information is held at the community level, in a collaborative way. It uses *gossiping* mechanisms is to keep information available about the cloud devices and network, and expose this information when a user requests a computing service in order to take advantage of available resource for the actual processing. The cloud is able to hold a dynamic directory of available resources, considering the resource cards made available by each device member of the cloud. The resource discovery is done locally by each device participating in the community, which will expose information such as number of CPUs, speed, RAM size, and storage capacity.

However, as each device of a cloud has its own view of the cloud, it has only partial information about the cloud resources. The cloud also holds information about the status of each member device, which indicates whether the device is connected to the cloud, whether it is performing a job or if it is available. This information will be necessary upon job submission for allocation of resources necessary to a given job. The cloud also stores the network topology that interconnects its devices.

The gossiping algorithms used in the I-Cluster cloud are enhanced derivatives of Mosix's *adaptive resource sharing* algorithms described by Barak and La'adan [14]. Here is a basic description of our algorithm: Each device operates on its own, and keeps a local database containing a limited list of peers within the cloud. Each device is identified by its network address or name. This list is dynamic i.e., it is possible that it is extended to include new devices joining the community. The database also contains the resource card available for each of the known peers. With each object is attached a time stamp [1] of its last modification date. Each individual device of the cloud, at regular times (typically 5 seconds), will connect to a random peer and they will exchange their database contents. If a fresher copy of a database item exists in the other device's database, the local item is updated with the newer information. After the exchange, each device may choose to drop some of the exchanged items so that its database size remains small (typically from 200 items on a PDA to 1000 items on a powerful machine). The dropped items will be the least relevant ones i.e., the ones which have the smallest chance to get useful for an I-Cluster computing service execution. Typically items corresponding to lower performance computers will be dropped, as well as the ones corresponding to computers which are not in the network neighborhood (expressed in number of hops).

The Mosix team has shown [14] that the time for mean information is of order $\log_2(N)$ i.e., when a fresh information is published to a device database, there is a 50% probability that the information will be replicated on another device's database after $\log_2(N)$ steps. For example, in a 1024 nodes cloud, if a machine becomes available, 50% of the other machines of the cloud will get the in-

[1] Note that distributed and synchronized time stamps may be used for the sake of consistency of the time reference between the cloud devices. A well-known example of such distributed time stamps is the *Lamport clock*.

formation after only $\log_2(1024)=10$ steps of the algorithm, hence after 50 seconds in our case. Of course this applies well in Mosix's case, where the considered machines are stable computing servers and where each machine has a complete knowledge about the participants list. In our case disconnections of machines make the convergence slower. Moreover, we had to enhance the algorithm so that it adapts to the capacity of each device. In Mosix's case, each server holds information about all other servers, which is inapplicable in our case, as this would generate huge memory requirements on each device of the cloud, and very large messages would have to be exchanged between devices, hence generating important network traffic.

Match finding

Once each device is able to get its own view of the cloud, it then knows a list of available machines of the network that it can use for submitting jobs. The user selects the computing service to invoke, and sets some additional information such as the entry parameters for the job, data file locations and the location where the output data should be exported (NFS, FTP, X display). This information is handled to a component called the *match finder*, which will be responsible for finding a virtual cluster which has the best capabilities for execution of the job. The I-Cluster Framework uses the available network resources and topology very tightly. This makes it possible for a given Service to require a set of resources with critical network requirements. It is then possible to execute a very finely grained job –supercomputing- on the allocated resources.

Each I-Cluster computing service is associated to a description of how the service can be composed (instantiated) onto a cluster. The service composition takes into account the complexity of the computational job, expressed in computational power required by the job, number of machines, type and speed of processor required, minimal memory size, bandwidth and latency per node, as well as the required interconnect capacity between allocated nodes expressed in terms of network topology, maximum network latency between nodes and network bisection bandwidth.

The match finder will check the local database for a set of computers that match the service requirements, and attempt to allocate them to the job. This is done using a 2-phase commit algorithm, which first attempts the allocation of each required computer, and commits each of them if all the computers are available for the computation. This very simplistic allocation scheme may be very inefficient for a common cluster configuration, but in our case we assume that lots of idle machines will be available at the time of service invocation, hence the chances that two different attempts occur at the same time on a single machine are low. I-Cluster shares the same allocation paradigm than MPI: The machines allocated to a given service are fully allocated for the duration of the service, and the number of machines cannot change during the execution.

Once the allocation of the virtual cluster is done by the match finder, we need to replicate the input data on each of the machines. Replication can be done using tools [15] that have been built by the ID-IMAG team in order to efficiently distribute files to a number of machines in a scalable way. As our testbed is 225 machines large, raw file copies would quickly overload the network. Once the data is replicated on the machines, the system proceeds into starting the execution of the job on the given machines.

Execution sandbox

I-Cluster aims at using existing enterprise PCs and combine them into virtual functions and services. This means that "I-Cluster devices" are basically non-dedicated network machines on which we need to add specific software that enable them as being part of I-Cluster's cloud. Today's machines are usually running Microsoft Windows as their basic operating system. We developed the components necessary for each machine to be able to switch to an I-Cluster mode and come back from this mode to the standard user mode.

The *sandbox model* allows the description of a code execution model as well as a set of rules used to develop code that will execute in the sandbox. A sandbox can be used to execute untrusted code on a machine, while having access to a set of system functionality such as network connectivity, while protecting the executing machine as well as the data attached to it from any security attack that could occur from malicious or deficient code loaded by the sandbox. Hawblitzel et al [16] proposed a classification of protection mechanisms in several classes. The first class of protection is the *hardware virtual memory*, which allows code execution isolation based on hardware support. The second class defines *capacity systems*, used in micro-kernel Operating systems such as Amoeba [17] or Chorus [18], and based on an instrumentation of source code which builds the traces necessary for building the *capacity lists*, which will be checked at run-time. The Java Runtime Environment [19] belongs to this category, as well as Internet C++ [20], a POSIX-compliant runtime execution mode. The last class of protection mechanisms is *Software-based Fault Isolation* (SFI), such as the one used by the VMWare system [21]. The I-Cluster sandbox belongs to this class, and makes a temporal sharing of the PC resources. At a given time, a computer will be either fully dedicated to its user, and running as a standard Windows machine; or the computer may switch to *I-Cluster mode*, where the resources are fully dedicated to I-Cluster.

The I-Cluster mode of a PC can basically be seen as a specific partition where a specific Linux distribution is taking place. This partition is created in a user-transparent way upon installation of I-Cluster on a machine. As we required adapting to real world conditions, we had to devise a way to operate and install this mode so that it does not impact the user. We developed a set of tools that create an unfragmented 512-MB file in the user's file system, then copy the image of the I-Cluster partition into the file. This way, we get a cluster partition *included* in the user's partition. Our tools are based on Extensible Firmware Interface [22] extensions available on recent PCs to hide the I-Cluster partition; more precisely, we extended the *GUID Partition Tables* (GPT) defined for IA64 architectures hard disk partition, and added some support for it in the Linux kernel used within our cluster mode. We developed tools to automatically control the boot sequence of the PC, selecting whether the PC should be started in user mode or in cluster mode upon each start. We coupled some tools to Windows 2000/XP's hibernation mode, screen saver and to remote wake up PC capability defined by the OnNow consortium [23]. We can then easily use *launch windows* i.e., lengthy periods of inactivity such as off-work periods of computer users, nights, holidays and week-ends. In such periods our tools will automatically switch the PC from user to cluster mode, where the resources will eventually get used. Then the PC will automatically switch back to user mode some time before the expected return time of the user. This way, the user never notices transitions of his PC to cluster mode. However, in case the user comes back unexpectedly, at night for instance, I-Cluster can abort any ongoing computation and restore the PC to user mode in less than 2 minutes (restoring from Windows hibernation).

The main advantage of our execution sandbox is that each system fully switches to the cluster mode when allocated. This means that all the PC resources are dedicated to the computation. There is no need to have part of the memory and other system resources used for handling the Windows Operating System in the background, as our sandbox includes a fully operational Linux Operating System.

4. Results

The system described here is still under ongoing research. Most of the components are not functional yet, and several months of work will be necessary until a stable version of the whole framework is available. We plan to have a working demonstration of the I-Cluster cloud deployed on a large set of machines by the end of 2003.

However, we have already installed an I-Cluster experimentation platform, based on 225 HP e-PC based on a 733 MHz Pentium III architecture, with 256 MB of RAM and 15 GB of hard disk. They are interconnected by Ethernet 100 using switches (see Figure 1). This platform models a typical small company's network with mainstream computers that you could expect finding on anyone's desk. The goal was to experiment whether getting some good performance with scientific applications would be possible with I-Cluster running over infrastructures available in medium or large companies, or on a University campus.

Using our experimentation platform, we have been able to reach 81.6 Gflop/s using Linpack. We reached the 385th place on May 2001's TOP500 [24], showing that world-class supercomputing is possible only using mainstream computers and interconnection. We also showed in [25] that, under certain circumstances, the system is scalable in performance i.e., its performance increases linearly with the number of machines used in the computation.

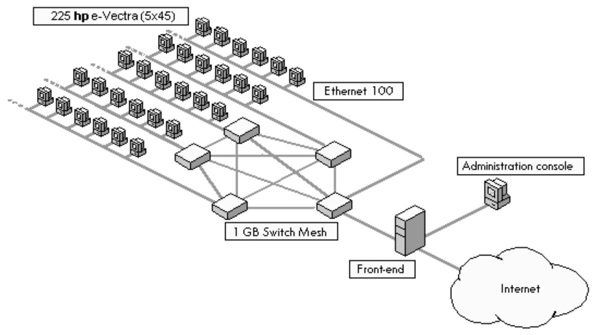

Figure 1: I-Cluster experimentation platform topology

5. Next steps

I-Cluster is a research program, whose vision is to build a full framework of tools that makes exploitation of untapped network resources possible in the scope of scientific/cluster computing. Large areas of research are still under work, and the project components described in paragraph 3 are not yet available and still require few months of development.

Here are some areas that we will investigate in a short term:

Simulation, validation and proof of our algorithms will be an interesting area of research. The currently available tools such as Network Simulator 2 (NS2) barely support emulation of large communities and are typically limited to few hundred nodes simulations. As we need to evaluate to convergence of algorithms, we will need to investigate how we can scale current tools. This should help us assess the convergence of the I-Cluster cloud, measure the quality of our algorithms and search the parameters that provide the best results.

We will begin a research effort about job resilience in I-Cluster: Although the community and the cloud are resilient, once a job is started we cannot prevent a fault on one of the allocated machines to stop the whole job. We will be looking at ways to enhance this resilience using either checkpointing and restart mechanisms or using redundancy of execution nodes.

Security will probably be a future area of research. We currently have a very secure sandbox for execution of cluster jobs, but we need to work on extended analysis of I-Cluster users security, including distributed authentication and accounting.

Also lots of other research areas will be investigated in the longer term, depending on the project needs and opportunities.

6. Conclusion

We introduced the I-Cluster research project from HP Labs Grenoble in partnership with INRIA (ID-IMAG Laboratory), which aims at building a framework of tools for performing scientific and cluster computation using untapped resources from a network. I-Cluster introduces lots of new concepts, particularly concerning how it handles the transition from *federated computing model* to *community computing model*. Based on a Peer-to-Peer heart, I-Cluster will massively scale and gracefully adapt to real world conditions. It allocates a virtual cluster amongst machines on a network, based on the computing service requirements in terms of number of nodes, processing power, interconnection topology. It uses an execution sandbox that makes a temporal sharing of each computer between a user mode and a cluster mode, in which the resources are made available to the community for performing common cluster computations.

The description of our experimentation platform helped understanding how reaching a TOP500 level of performance is possible using only mainstream resources, such as the ones commonly available from an enterprise network or from a university campus.

7. References

[1] Institute of Electrical and Electronics Engineers. "IEEE Standard Computer Dictionary", A Compilation of IEEE Standard Computer Glossaries. New York, NY: 1990.

[2] Becker D.J., Sterling T., Savarese D., Dorband J.E., Ranawak U.A., Packer C.V., "Beowulf: A Parallel Workstation for Scientific Computation", Proceedings of the International Conference on Parallel Processing, 1995.

[3] Barak A., Wheeler R., "MOSIX: An Integrated Multiprocessor UNIX", Proceedings of Winter 89 USENIX Conference, pp.101-112, San Diego, CA., 1989

[4] Gillen A., Kusnetzky D., Sayama A., Dayal H., Hingley M., "Windows Operating Environment Forecast and Analysis", IDC publisher, document 24827, 2001.

[5] A. S. Tanenbaum, M. F. Kaashoek, R. van Renesse, H. Bal, "The Amoeba Distributed Operating System - A Status Report", Computer communications, Vol. 14 , pp. 324-335, Jul 1991.

[6] Remzi H. Arpaci, Andrea Dusseau, Amin M. Vahdat, Lok T. Liu, Thomas E. Anderson, David A. Patterson, "The Interaction of Parallel and Sequential Workloads on a Network of Workstations", UC Berkeley Technical Report CS-94-838, November, Also Submitted to Sigmetrics '95, 1994.

[7] T. E. Anderson et. al., "A case for NOW (Networks of Workstations)", IEEE Micro, Feb 1995.

[8] Casanova H., Dongarra J.J., "NetSolve: A network server for solving computational science problems", Tech. Report CS-95-313, University of Tennessee, 1995.

[9] Litzkow M.J., Livny M., Mutka M.W., "Condor - A hunter of idle workstations", Proceedings of the 8th International Conference on Distributed Computer Systems, 1988.

[10] D. Atkins, M. Graff, A. K. Lenstra and P. C. Leyland., "The Magic Words are Squeamish Ossifrage", Advances in Cryptology - ASIACRYPT '94, J. Pieprzyk and R. Safavi-Naini, eds., Lecture Notes in Computer Science 917, (1995), 263-277.

[11] Anderson D., et al., "SETI@home: The Search for Extraterrestrial Intelligence", Technical report, Space Sciences Laboratory, University of California at Berkeley. http://setiathome.ssl.berkeley.edu/, 1999

[12] I. Foster, C Kesselman, "The Grid: Blueprint for a New Computing Infrastructure". Morgan Kaufman Publishers, Inc. San Francisco, California, 1999.

[13] Foster I., Kesselman C., "Globus: A Metacomputing Infrastructure Toolkit", Proceedings of the Workshop on Environments and Tools for Parallel Scientific Computing, SIAM, Lyon, France, 1996

[14] A. Barak and O. La'adan, "The MOSIX Multicomputer Operating System for High Performance Cluster Computing", Journal of Future Generation Computer Systems, 13(4--5):361--372, March 1998. 3.4.1 138

[15] Martin C., Richard O.,"Parallel launcher for clusters of PCs", Proceedings of Parco 2001.

[16] Hawblitzel C., Von Eicken T., "A case for language-based protection", Tech. Rep. 98-1670, Department of Computer Science, Cornell University, 1998.

[17] Tannenbaum A.S., Van Renesse R., Van Staveren H., Sharp G.J., Mullender S.J., Jansen J., Van Rossum G., "Experiences with the Amoeba distributed operating system", CACM, 33(12):46--63, 1990.

[18] Abrossimov V., Rozier M., Gien M., "Virtual Memory Management in Chorus", Proceedings of Progress in Distributed Operating Systems and Distributed Systems management, 1989.

[19] Arnold K., Gosling J., Holmes D., "The Java™ Programming Language", The Java™ Series, Addison Wesley, 1994.

[20] Abis M., Dayley B., "An Open Alternative to Java and C-Sharp", http://ivm.sourceforge.net/, 2000.

[21] VMware, Inc. "Virtual Platform" http://www.vmware.com/products/virtualplatform.html, 2000.

[22] Intel et al., "Extensible Firmware Interface", Version 1.02, http://developer.intel.com/technology/efi/, 2001.

[23] Microsoft Corporation, "OnNow: the evolution of the PC platform", http://www.microsoft.com/hwdev/onnow.htm, 1997.

[24] Meuer H.W., Strohmaier E., Dongarra J.J., Simon H.D., "TOP500 Supercomputer Sites", Proceedings of the 16th International Supercomputer Conference, June 2001.

[25] Richard B., Augerat P., Maillard N., Derr S., Martin S., Robert C, "I-Cluster: Reaching TOP500 performance using mainstream hardware", HP Labs technical report HPL-2001-206, http://www.hpl.hp.com/techreports/2001/HPL-2001-206.html, 2001.