I-Cluster: Reaching TOP500 Performance Using Mainstream Hardware

B. Richard, P. Augerat[1], N. Maillard[1], S. Derr[1], S. Martin[1], C. Robert[1]
HP Laboratories Grenoble

HPL-2001-206
August 22nd , 2001*

E-mail: bruno_richard@hp.com, {philippe.augerat, nicolas.maillard, simon.derr, stephane.martin, celine.robert}@imag.fr

cluster, performance, TOP500, network, switch, PC, Linpack benchmark

A common topic for PC clusters is the use of mainstream instead of dedicated hardware i.e., using standard desktop PCs and standard network connectivity, with technology to organize them so that they can be used as a single computing entity. Current work in this "off-the-shelf cluster" domain usually focuses on how to reach a high availability infrastructure, on how to efficiently balance the work between nodes of such clusters, or on how to get the most computing power for loosely-coupled (large grained) problems. **hp** Labs Grenoble, teaming with INRIA Rhone-Alpes, teamed up to build a cluster out of 225 standard **hp** e-PC interconnected by standard Ethernet, with the objective of getting the highest computational performance and scaling from the simplest desktop PC to the most powerful computers in the world. As an additional constraint, we decided to use a cluster that models a modern enterprise network, using standard machines interconnected through standard Ethernet connectivity. This paper describes the issues and challenges we had to overcome in order to reach the **385th rank** in the TOP500 list of most powerful supercomputers in the world on June 21st, 2001, being the first mainstream cluster to enter TOP500 ever. Also we provide hereafter some details about the software and middleware tuning we have done, as well as the impact of different factors on performance such as the network topology and infrastructure hardware.

[1] ID Laboratory

I-Cluster: Reaching TOP500 performance using mainstream hardware

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

P. Augerat, N. Maillard, S. Derr, S. Martin, C. Robert, ID Laboratory
{philippe.augerat, nicolas.maillard, simon.derr, stephane.martin, celine.robert}@imag.fr

June 2001

A. ABSTRACT

A common topic for PC clusters is the use of mainstream instead of dedicated hardware i.e., using standard desktop PCs and standard network connectivity, with technology to organize them so that they can be used as a single computing entity. Current work in this "off-the-shelf cluster" domain usually focuses on how to reach a high availability infrastructure, on how to efficiently balance the work between nodes of such clusters, or on how to get the most computing power for loosely-coupled (large grained) problems. ***hp*** *Labs Grenoble, teaming with INRIA Rhône-Alpes, teamed up to build a cluster out of 225 standard* ***hp*** *e-PC interconnected by standard Ethernet, with the objective of getting the highest computational performance and scaling from the simplest desktop PC to the most powerful computers in the world. As an additional constraint, we decided to use a cluster that models a modern enterprise network, using standard machines interconnected through standard Ethernet connectivity. This paper describes the issues and challenges we had to overcome in order to reach the* ***385th rank*** *in the TOP500 list of most powerful supercomputers in the world on June 21st, 2001, being the first mainstream cluster to enter TOP500 ever. Also we provide hereafter some details about the software and middleware tuning we have done, as well as the impact of different factors on performance such as the network topology and infrastructure hardware.*

B. INTRODUCTION

hp Laboratories Grenoble and the INRIA started the *I-Cluster*[1] project in June 2000. I-Cluster is a research program on communities of standard Internet Appliances (e-client) aiming at distributing virtual functions like high-performance super-computing tasks, distributed storage or network caching, using only non-dedicated hardware. In this scope, a Beowulf-class cluster of few hundred PCs would be contributed by **hp**, constituting the basis for the experimental cluster.

In October 2000, a first part of the I-Cluster (100 machines interconnected with 3 switches) was built. After fine tuning the environment and Software, we measured a performance level[2] of 36 Gflop/s on the 100 machines. This performance and scalability features were above what we had expected, and we decided to extend the cluster to 225 machines and to experiment how to get the most performance out of the cluster in order to enter the TOP500.

C. PERFORMANCE SCALE

It is tempting to develop a supercomputer using mainstream "off the shelf" hardware. As these types of products are manufactured in volume, one can expect competitive pricing and high quality. So we can imagine buying literally thousands of PCs, and interconnecting them using standard connectivity methods. One good example of such a large-scale computer is

[1] *More I-Cluster information is available from http://icluster.imag.fr/*

[2] The performance benchmark used is Linpack, in the conditions settled by http://www.top500.org/

exploited in the SETI@Home project [3], which interconnects hundreds of thousands of machines over the Internet, raising a cumulated performance of several Teraflop/s.
For supercomputers, a common way to measure performance is to use the Linpack benchmark as described by the TOP500 consortium. Linpack stresses the system through computing large-scale linear algebra problems on fully filled matrixes. This type of program requires some heavy double precision computing on each processing node, but it also requires a lot of communications between computing nodes (mainly global and synchronous communications). If it were to be executed on a *Metacomputer* such as the one built through the SETI@home project, the Linpack benchmark would give extremely bad results, as the communication between machines suffers from a very poor latency inherent to Internet. On the other hand, supercomputers (or high performance clusters) with sophisticated hardware and high performance networks will have a good Linpack scalability but a poor performance/price ratio. Our work then was to design the intermediate "Linpack and price scalable" supercomputer.

D. I-CLUSTER CONFIGURATION

a. *Computing nodes*

Our cluster consists of 225 **hp** e-PC machines. Each of these PC is equipped with a 733 MHz Pentium III ®, 256 MB of RAM, and a 15 GB hard disk. The e-PC represents the evolution of the classic PC into a smaller, simpler and more reliable device, designed as a typical corporate office desktop. In the scope of I-Cluster, this brings us high stability as well as a simplified maintenance model. The required space is much smaller than what would be required if we had installed 225 common desktop machines, and the power requirements are only in the range of 50 W per machine, which is only about 30% of what it would be for a common desktop machine. The ventilation and air conditioning requirements are easier to maintain with a cluster of e-PCs than a cluster of common desktops.
On the other hand, not being able to open the e-PCs also means that it is not possible to modify their hardware. Hence it is *not possible* to add memory, nor to add a second network adapter or to upgrade the network adapter to a low-latency network such as Myrinet [4], SCI [5] or VIA [6], which are typically used for interconnection in computational clusters.

b. *Connectivity*

The network technology being constrained by the **hp** e-PC connectivity, the nodes are connected to five switches by Ethernet 100 links. The switches are **hp** ProCurve 4000 Switches interconnected with gigabit Ethernet (note that this configuration may match the configuration of many large companies Intranet.)
As I-Cluster uses Ethernet for its interconnections, the switches need to rely on *store-and-forward* algorithms to process the Ethernet frames and to calculate their routes. This causes a delay in the processing of each network frame, hence increasing the latency and diminishing I-Cluster's performance.
Our configuration allows the mesh of switches to have a pentagram or double ring structure (see Figure 1: I-cluster Topology).
In a completely connected mesh of network switches, each switch is directly connected to the four other ones through an Ethernet 1000TX interconnect. Several alternatives have been envisioned. Among them, a tree of switches, that would eventually give a good network performance, but inter-switch communications would have required to cross 3 different switches, hence giving a poor latency.
Flat Neighborhood Networks (FNN) such as described in the KLAT2 project [8] is an interesting alternative for the network topology, but it requires several network adapters on each PC of the cluster. In I-Cluster's case, the selected PC model is **hp** e-PC, which is not extensible hence cannot support a second network adapter.
With our *pentagram* mesh, Ethernet frames traveling from one e-PC to another e-PC that is attached to a different switch have to cross the two switches, hence doubling the latency due to switches. As we will see below, connecting all the machines from a specific Linpack row to the same switch greatly increases performance (typically 14%).

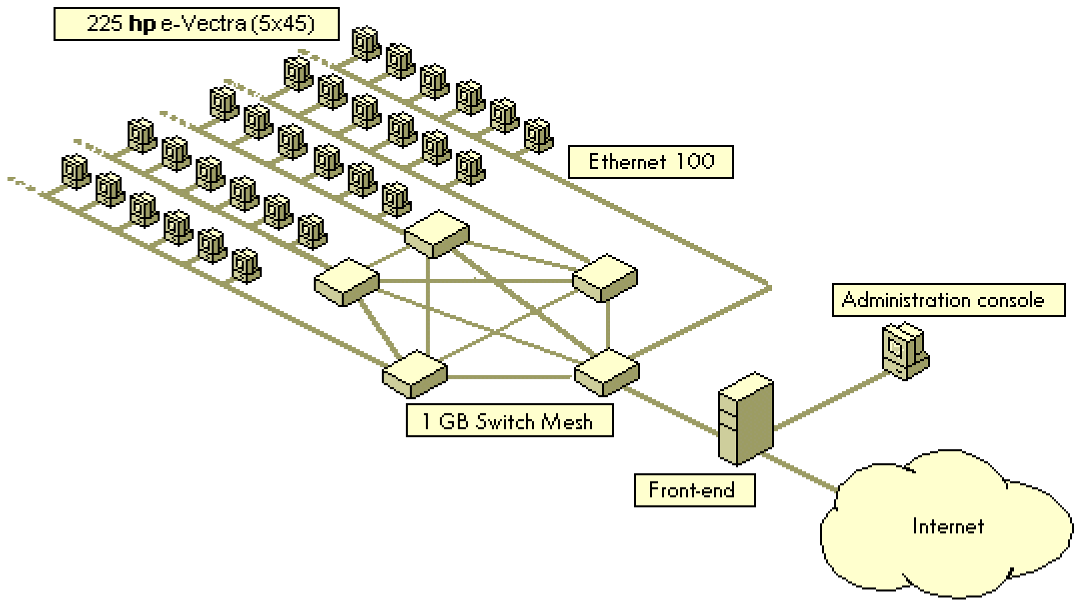

Figure 1: I-cluster Topology

c. Software

I-Cluster is based on Linux Mandrake 7.0 (different kernel versions have been tested with various results, see "Network drivers and Linux Kernel").

d. The parallel Linpack benchmark

We used the HPL[3] (High Parallel Linpack) package [1] as the basis for our experiments. It consists of the distributed computation of a LU factorization, for a dense matrix of *n x n* double precision numbers. The complexity of the algorithm is constrained to an order of:

$$ops = \frac{2.n^3}{3} + 2.n^2$$

Parallelization is made with a block decomposition of the matrix, which allows both a recursive algorithm to be applied, and a sequential granularity that is compatible with the use of (sequential) matrix-like computations, thus allowing calls to specialized and highly performing libraries (Blas).

The block algorithm requires a grid-like use of the available processors, in order to map the two-dimensional matrix on a two-dimensional topology. Besides, the numerical analysis shows that most of the communication resides in broadcasts that occur on each line of the grid. An extremely flat grid would therefore lead to a bottleneck due to extensive broadcasts.

On the other hand, one phase of the algorithm being the propagation of a block-pivot, which occurs on each column and is a synchronizing event (every node has to wait for the pivot in order to proceed to the following computation), it is not possible to get a high efficiency if the grid is too deep. The balance between the number of lines and the number of columns of the grid, linked to the broadcast algorithms, is therefore one of the key issues for the parallel algorithm's efficiency.

[10] provides an extensive analysis of communication costs in the algorithm. The conclusion is that, on a simplified machine model, the parallel efficiency (ratio between the sequential time, and the parallel time, reduced to one processor), is a decreasing function of the quantity $lp.\left(\frac{pq}{n^2}\right)$, where the grid of processors has p lines and q columns; n is the size of the matrix; and l is the latency of the network.

Each node therefore requires a memory quantity of $\frac{n^2}{pq}$, and if you keep it constant (i.e. increase

[3] In this document, "HPL" or "Linpack test" will be referring to the HPL program unless specified.

the matrix's size when adding nodes to the computation), you remain with a limiting factor due mainly to the product *latency x broadcast length*, which is coherent: the flatter the grid, the longer the broadcasts, and the more the parallel execution slows down. The latency will play a huge role in this inefficiency. This latter result was one more incentive for us to optimize the network capacities, or the software drivers, to reduce the latency.

E. REACHING TOP PERFORMANCE

The I-Cluster research program started upon receipt of the first batch of machines mid-October 2000. Because of budget constraints we only received 100 machines. This number was increased to 216 machines in March 2001, then to 225 machines in April.

a. *Optimized parameters*

We conducted many experiments to evaluate the importance of various components and parameters for the overall performance of the cluster. The idea was to optimize each parameter as independently as possible, identify optimizations that provide a performance gain, synchronize all the optimizations, then repeat the whole process until no more parameter provides a gain. We focused on the following parameters:
- Blas library choice (MKL, Atlas…), compiler used, and parameters;
- Linpack parameters (Matrix size, block size, broadcast algorithm);
- Operating System and drivers configuration;
- Network topology, including adaptation to Linpack parameters.

We therefore had to deal with each level of software optimization, from the hardware (topology) to front-end parameters (Linpack's options).

1) *Blas library and tests*

The Blas libraries are responsible for the basic matrix-like and vector-like -routines that are the core CPU operations performed by the Linpack software.
The first performance measures were done using a BLAS library optimized using the ATLAS suite [9], which performs optimizations depending on the actual hardware of each computing node. The gcc compiler was used for the first experiments. We planned to use the Intel Compiler or the Portland Group one in order to take advantage of Intel Pentium ® III Streaming SIMD Extensions (SSE), which is an extension present in the model of microprocessors we use in I-Cluster e-PC machines. Unfortunately, this technology that allows the processor to perform two double precision calculations in one single step is not activated on Pentium III, so we never measured a major advantage using either compiler.
We compared several available Blas libraries, mainly Atlas, the Intel Math Kernel Library and a library issued from the ASCII project (actually it consists in a fusion between the Atlas library for all the operations but the *dgemm*, which is directly tuned for the Pentium III by Greg Henry from Intel).

20 runs were done for each Blas library.

	Mean	S
Henry	499.38	6.52
Atlas 3.2.0	493.96	1.8
Atlas 3.2.1	478.3	1.7
MKL	439.9	2.1

Table 1: Blas Libraries Performance

As a conclusion, all our tests finally induced us to choose an Atlas based version, with Greg Henry's optimized dgemm.

2) *Linpack parameters*

There are quite a number of parameters for the HPL test itself (without counting those at compile-time!). These are: the size *N* of the matrix; the size *N_b* of the blocks it should be divided into; the number of rows (*p*) and columns (*q*) in the grid-like topology of processors; the algorithm used for a panel factorization (*pfact*); the minimal size of a block to make a recursive call to the factorization algorithm and the algorithm used for it; the type of broadcast required; the number of pivots computed before sending them to other processors; and the algorithm for swapping the pivots. Many of these parameters are dependent on each other. A good part of the job was to tune each one regarding the architecture of the cluster.
The tuning guide of HPL suggests a choice of a square-like grid, with a number of columns slightly bigger than the number of rows. Obviously, depending on the topology, the broadcast had to change. Apparently, the possibility of decomposing *N* in prime factors was predominant for the scalability of the performance.

➢ Broadcast algorithm

Linpack is a synchronous program based on MPI, hence using "*rendez-vous*" mechanisms. If one computing node is slower than the others, the whole performance will suffer, as the faster nodes will align on the slower one during *rendez-vous*. HPL provides 6 algorithms for the broadcast. Each one was extensively tested, on nearly all configurations. The optimal broadcast algorithm had to be changed between 98 nodes tests and the 210 and 225 ones. Anyway, it is easy to pick the broadcast algorithm that best suits the grid.

We also tested some "hand-made" broadcast algorithms that were variations around the pre-defined broadcasts then we tested the MPI_Bcast of MPICH, which is implemented using a binomial tree (and hence different from all other broadcast algorithms of HPL). None of these solutions outperformed the best HPL native broadcast.

- Matrix and Block sizes

These are some of the parameters that we most empirically dealt with. We reached our best performance with a matrix whose RAM size ranged from 200 Mbytes to 220 Mbytes per node:

$$M(n) = 8.\frac{n^2}{pq}$$

It seemed that for the higher matrices, a subset of the nodes swapped from time to time. This may be due to the handling of buffers from MPI or to the operating system. Some performance evaluation work should be done on this problem. On the average, a size of 212 Mbytes per node seemed good.

As to the block size, theoretically we should have:

Determined approximately the size M x N of the matrix treated on one node;

Determined sequentially the best value K for the BLAS dgemm operation on matrices M x K and K x N;

Fixed N_b = K and N accordingly.

The latter operation should be done following the rules: pN_b divides N and qN_b divides N; that is to say,

gcd(p,q) x N_b divides N.

3) Network drivers and Linux Kernel

The i-cluster is designed for a standard every day use, so we decided to keep the TCP/IP stack as provided in Linux 2.2.4 kernels. This choice badly influences HPL performance numbers because of the latency overhead of the TCP stack. Latency is caused by data extra copies between user space and the network card memory.

This phenomenum can only be avoid using specific hardware (Myrinet [4], SCI [5]) or specific low latency protocols (GAMMA [7]) or architectures (VIA [6]). Until now, none of these solutions achieved the same stability, low cost and compatibility as TCP. This latency is widely introduced by HPL with a great number of medium sized messages.

Several parameters have been tuned for HPL performance. For example the maximal size of the TCP buffer has been modified and several network card drivers have been tested. Each driver has been installed and tested on the Linux 2.2.17 and 2.4.2 kernels. Last kernel showed little improvement and was used for further testing.

4) Network architecture

Tests with more than 45 nodes use a multi-switch layout and allocating nodes to the grid scheme is getting complex.

The first step is to balance involved nodes among switches. Then we have to specify an order for the nodes used by MPI, such that it builds its grid with the right nodes at the right place and minimizes the number of inter-switch communication. For a 210 nodes computation and five switches, good mapping provided an extra 5 Gflop/s, allowing us to reach up to 76.4 Gflop/s where before we only had 71.8 Gflop/s.

As far as topologies are concerned, we could test three of them (simple ring, double ring and star) for the network switches interconnection. The star used a complete graph topology (pentagram) to interconnect the 5 switches with links at 2 Gbit/s, and a ring. Surprisingly, we did not notice any improvement with the star while a double ring improves the results by 1.5%. Here again, some tool to monitor the packet exchanges and the collisions on each of these networks should enable us to understand better what is going on.

b. Performance results

We first used a 100 node cluster from which we got very good scalability results. Using 1 up to 45 nodes is very easy since all nodes reside on the same switch. Trying to use an “almost square” grid gives the best results.

With more nodes and switches, scalability is good, but finding the best Linpack parameters is not so easy. Blind runs of Linpack would give very non-deterministic results.

Grid	**Best performance (Gflop/s)**
15 x 14	67.9
14 x 15	76.4
10 x 21	74.4

Table 2: Grid vs. Performance

Moreover, the optimal allocation of nodes to a grid is not possible for all cluster sizes.
Best runs with 210 nodes gave 76.4 gigaflops results while runs with 215 nodes gave at most 55 flops. This is because the only grids that match 215 nodes and 5 switches are 5x43 or 43x5, which are not balanced.
When the number of nodes and switches allows non-flat grids, a few runs should lead to very good performance.

A 15 x 15 grid yields theoretically quasi-optimal results. To get the best experimental results, we increased the matrix size until some swap phenomenon prevented achievement of peak performance. The best run gave then **81.6 Gflop/s**.

The experiments showed us that the increase in number of nodes involved in the computation did not damage the performance. Scalability of I-Cluster is very good (nearly linear). In "Figure 2: Performance vs. number of nodes", the gigaflops measures were made on machines on a single switch. Processors grids that do not suit the Linpack test such as (1x41) grid are not included in these results. With 45 nodes on a switch, the performance reaches 350 Mflop/s by node. Bad behaviors correspond to almost flat grids (2x17, 2x19).

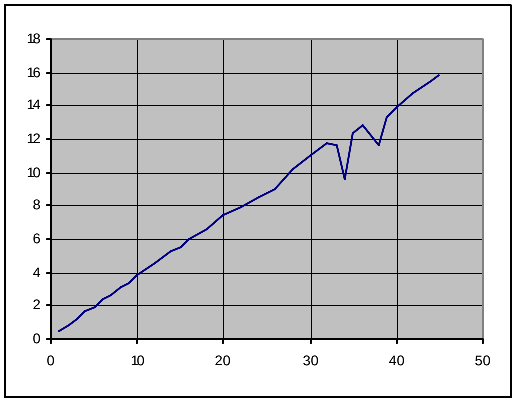

Figure 2: Performance vs. number of nodes

Also as shown in the following figure, Linpack scales with more nodes and switches:

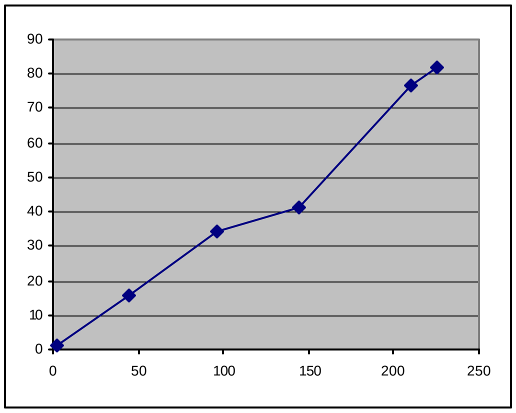

Figure 3: Multi-switch performance/nodes

c. *Performance/price*

The following table shows the costs for I-cluster. Note that the software is house-built or Open Source, hence we did not show any Software costs.

Parts	**Cost ($)**
225 ***hp*** e-Vectra @$950	213750
5 ***hp*** ProCurve switch @$3300	16500
25 ***hp*** ProCurve expansion boards @$700	17500
Total	**247750**

Table 3: I-Cluster Hardware costs

I-Cluster's performance/price index is hence around *$3000 per Gflop/s.*

This ratio may vary with the number of nodes per switch, but remains good at any cluster size, as shown in the following figure:

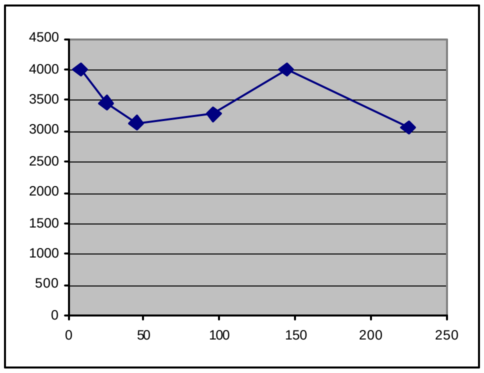

Figure 4: Price/Gflop/s vs. number of nodes

F. PERSPECTIVES

A few changes in the software architecture could improve the overall performance:

Linpack uses various operations (matrix, vector…) from the Blas library, and ATLAS and MKL Blas libraries perform differently on each of these operations. Thus, we could expect to get the optimal Blas library performance by picking the best from both worlds and merging them. Part of this work was done by plugging the "dgemm Greg Henry routine" into ATLAS.

The MPI library used is the standard one. It may be worthwhile rewriting the collective communications routines to take care of the cluster topology. Using a multicast communication framework should be a good way to improve Linpack performance

Switching from TCP/IP to a low latency protocol (such as Gamma or M-VIA) would also boost the performance.

Such research axes have not been considered during our experiments and are left as an exercise for the reader…

G. THANKS AND ACKNOWLEDGEMENTS

The I-Cluster performance experiment has been very exciting and intensive. We want to thank all the participants that helped, in particular Bruce Greer and Greg Henry from Intel, Antoine Petitet from Sun Labs.

H. CONCLUSION

Being the first ones to enter the TOP500 using only mainstream hardware (standard PCs, standard Ethernet connectivity) was quite a challenge. Reaching a **81.6 Gflop/s performance** showed that such challenge is technically achievable. At the time we installed the I-Cluster, it proved to be one of the two hundred best supercomputers ever built according to J. Dongarra performance list [2]. The performance tuning required for that was around 6 man/month of experienced people's efforts. However, this timing is mainly due to software testing rather than Linpack experiments. Starting from our experiments results, it is now much faster to implement and tune a *mainstream cluster* using clues and guidelines from this paper.

So mainstream hardware in an "intranet framework" is a viable alternative to supercomputers: We obtained a TOP500 performance level with hardware approximately ten times cheaper. Also our performance/price ratio is better than the one for a high performance cluster, since a complete e-PC is cheaper than a single low latency Myrinet or SCI network card, we also would easily beat high performance clusters with same cost. And as we showed that scalability in the context of I-Cluster was not a problem, it is easy to extend the cluster to meet a required level of performance.

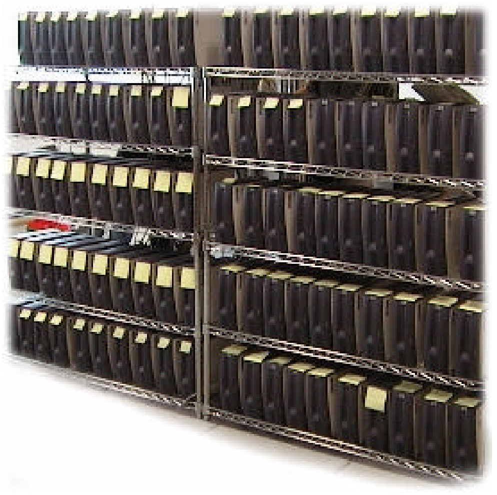

Figure 5 : I-Cluster

I. REFERENCES

[1] HPL http://www.netlib.org/benchmark/hpl/

[2] Performance of Various Computers Using Standard Linear Equations Software, J. Dongarra, Technical Report CS-89-85, University of Tennessee, 1989. (An updated version of this report can be found at http://www.netlib.org/benchmark/performance.ps).

[3] A new major SETI project based on Project Serendip data and 100,000 personal computers. W. T. Sullivan, III, D. Werthimer, S. Bowyer, J.Cobb, D. Gedye, D. Anderson. Published in: "Astronomical and Biochemical Origins and the Search for Life in the Universe", Proc. of the Fifth Intl. Conf. on Bioastronomy. 1997

[4] Nanette J. Boden, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su, "Myrinet - A Gigabit per Second Local Area Network," in IEEE-Micro, February 1995, vol. 15(1), pp. 29-36.

[5] IEEE Computer Society, IEEE Standard for Scalable Coherent Interface (SCI), IEEE, ieee std 1596-1992 edition, August 1993.

[6] D. Dunning, G. Regnier, G. McAlpine, D. Cameron, B. Shubert, F. Berry, A. M. Merrit, E. Gronke, and C. Dodd, "The Virtual Interface Architecture," *IEEE Micr*o, vol. 18, pp. 66–69, Mar.–Apr. 1998.

[7] G. Ciaccio, Optimal Communication Performance on Fast Ethernet with GAMMA, in proceedings of PC-NOW'98 (International Workshop on Personal Computers based Networks Of Workstations, in conjunction with IPPS/SPDP 1998), Orlando, Florida, March 30/April 3, 1998. LNCS 1388, Springer.

[8] H. G. Dietz and T.I.Mattox, "KLAT2's Flat Neighborhood Network," Proceedings of the Extreme Linux track of the 4th Annual Linux Showcase (ALS2000), Atlanta, GA, USA, October 12, 2000.

[9] Automated Empirical Optimizations of Software and the ATLAS project, http://www.netlib.org/atlas/, R. Clint Whaley Antoine Petitet Jack J. Dongarra, September 19, 2000

[10] HPL scalability analysis, http://www.netlib.org/benchmark/hpl/scalability.html