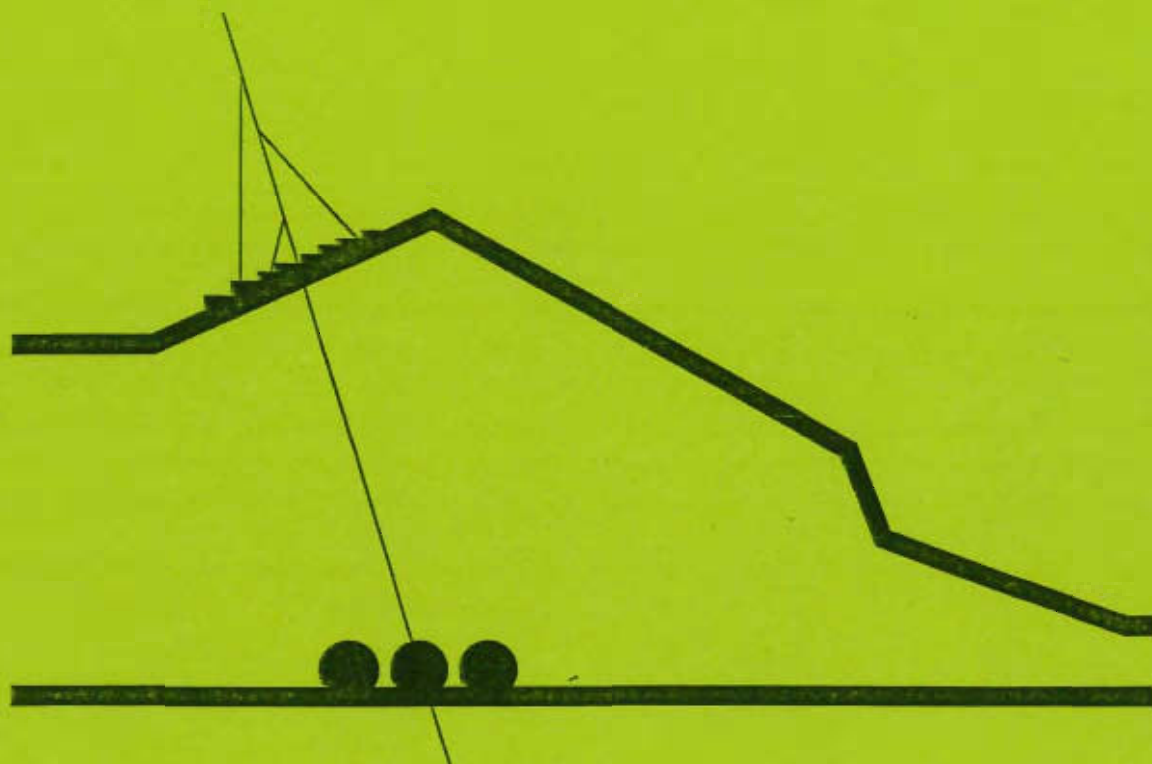


INFN/TC-99/09

23 Aprile 1999



Proceedings of the Workshop  
PC-NETS

*Trends in Computer Architectures*

Laboratori Nazionali del Gran Sasso  
24-25 March, 1999

Edited by *Enzo Fantozzi* and *Ersilia Giusti*

INFN - Laboratori Nazionali del Gran Sasso

Published by SIS-Pubblicazioni  
dei Laboratori Nazionali di Frascati

# PC - NETS

## Trends in Computers' Architecture

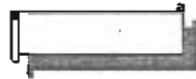
March 24-25, 1999

Laboratori Nazionali del Gran Sasso

L'Aquila (Italy)

*Organizers:* Giuseppe di Carlo  
Maria-Paola Lombardo  
Pietro Rossi

*Coordinator:* Vincenzo Fantozzi



*Participants include:*

Roberto Alfieri  
Massimo Bernaschi  
Gianluca Betello  
Alessandro Bettini  
Sven Bilke  
Alessandro Chessa  
Giovanni Chiola  
Giuseppe Ciaccio  
Roberto Covati  
Alberto D' Ambrosio  
Vincenzo Di Martino  
Andrea Donati  
Walter Errico

Aurelio Grillo  
Mario Guarracino  
Roberto Innocente  
Alessandro Lonardo  
Luigi Mancini  
Enzo Marinari  
Agostino Mathis  
Andrea Michelotti  
Melchiorre Monaca  
Emanuele Panizzi  
Sandra Parlati  
Nicoletta Pucello  
Federico Rapuano

Andrea Rodolico  
Vittorio Rosato  
Davide Rossetti  
Alan Scheinine  
Hubert Simma  
Rainer Sommer  
Nazzareno Taborgna  
Mario Torelli  
Raffaele Tripiccone  
Giuseppe Ugolotti  
Leonardo Valcamonici  
Carlo Vittoli  
Peter Wegner

## Foreword

The workshop "PC-NETS—*Trends in computer architectures*" was held at Laboratori Nazionali del Gran Sasso (INFN) on March 24–25, 1999.

The workshop addressed a wide range of topics, including hardware and software aspects, and applications. Ongoing projects have been presented. The role of PC-clusters in future supercomputer architectures has been discussed. Conclusions and perspectives were discussed by the panel : Agostino Mathis (Enea), Gianluca Betello (Telespazio), Giovanni Chiola (DISI, Genova) and Raffaele Tripiccione (INFN). We aimed at fostering the communications between computer designers and scientific users, and we feel that this goal has been at least partially achieved.

We warmly thank all of the participants for making this workshop a success. We wish to thank the coordinator Enzo Fantozzi and Ersilia Giusti for their precious help in the organization of the workshop, and preparation of these proceedings. We gratefully acknowledge the hospitality and the support of the Laboratori Nazionali del Gran Sasso, which included a guided tour of the Underground Laboratories led by Aurelio Grillo.

Giuseppe di Carlo, Maria-Paola Lombardo, Pietro Rossi

Assergi, 18 April 1999

PC-NETS  
Trends in Computers' Architecture

Laboratori Nazionali del Gran Sasso dell'INFN  
Assergi (AQ), Italy

March 24–25 1999

Organizers: Giuseppe di Carlo, Maria-Paola Lombardo, Pietro Rossi  
Coordinator: Vincenzo Fantozzi

Day 1 -- 24 March

11:00 Opening

Alessandro Bettini, LNGS Director

Chair: Pietro Rossi (Enea-Bologna)

11:15–12:00 **Vittorio Rosato** (Enea-Roma): PQE1 and PQE2000

12:00–13:00 **Walter Errico** (INFN-Pisa): Status and short term plans of  
the Apemille project.

**Davide Rossetti** (INFN-Roma): Apemille OS

13:00–14:30

*Break*

-- Lunch served at 13:30 in the LNGS 'Mensa'



Chair: Rainer Sommer (DESY–Zeuthen)

14:30–15:15 **Andrea Rodolico** (NICE s.r.l.): Load Sharing Facility,  
the computing factory: architecture, projects and  
experiences in parallel computing.

15:15–16:15 **Giuseppe Ciaccio** (DISI-Genova): The Gamma Project

**Vincenzo Di Martino** (Caspur-Roma): Using active  
messages to port parallel applications on PC clusters

16:15–17:00 **Sandra Parlati** (INFN-LNGS): The Condor INFN experience

17:00–17:30

*Coffee Break*

17:30–18:15 **Sven Bilke** (University of Amsterdam-WINS): A farmer's life–  
Simulations of fluctuating geometries on a PC-Farm.

18:15–18:45 **Mario Guarracino** (CNR-Napoli): Altair Project at CPS-CNR

18:45–19:45 **Alessandro Chessa** (Physics Department, Cagliari):

Beowulf @ Cagliari: Kalix2 after Kalix1

**Roberto Innocente** (Abdus Salam ICTP/SISSA):

MPI performance of a PC cluster at the ICTP

**Carlo Vittoli** (CRS4, Cagliari): Parallel performances:  
the IBM Sp2 vs a PC's cluster."

21:00

*Dinner*

Day 2 -- 25 March

Chair: **Andrea Donati** (INFN-LNGS)

9:00-9:45 **Giovanni Chiola** (DISI-Genova): DISCO - a status report.  
9:45-10:15 **Giuseppe di Carlo** (INFN-LNF): Octopus-the LNGS PC's cluster  
10:15-10:45 **Leonardo Valcamonici** (Caspur-Roma): Performance evaluation of  
the network subsystem of the CASPUR testing/training facility

10:45-11:15 *Coffee Break*

11:15-12:00 **Pietro Rossi** (Enea-Bologna): DISCO : a distribute data base on  
a cluster of PC's

12:00-12:45 **Hubert Simma** (DESY-Zeuthen): Pc's for multiTflops LGT  
compute engines?

12:45-14:15 *Break*  
-- Lunch served at 13:30 in the LNGS 'Mensa'



Chair: **Agostino Mathis** (Enea-Roma)

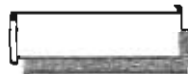
14:15-15:00 Concluding remarks

*Panel:*

**Gianluca Betello** (Telespazio-Roma)  
**Giovanni Chiola** (DISI-Genova)  
**Raffaele Tripiccione** (INFN-Pisa)

15:00 *Official program ends.*

15:00-20:00 Informal discussions and workshopping.



Vittorio Rosato

## AN APPLICATION-ORIENTED PLATFORM: THE MIMD-SIMD PLATFORM PQE1

S.Adda<sup>1</sup>, R.Cannata<sup>2</sup>, M.Celino<sup>1</sup>, M.Coletta<sup>1</sup>,  
R.Guadagni<sup>2</sup>, C.Lazzari<sup>1</sup>, C.Leoni<sup>2</sup>  
S.Nicastro<sup>3</sup>, P.Novelli<sup>1</sup>, P.Palazzari<sup>1</sup>, S.Pecoraro<sup>2</sup>,  
N.Pucello<sup>4</sup>, V.Rosato<sup>1\*</sup>, F.Valentinotti<sup>3</sup>

ENEA - Casaccia Research Centre  
P.O.Box 2400 - 00100 Roma A.D. (Italy)

<sup>1</sup>*HPCN Project*, <sup>2</sup>*INFO - Computing Centre Casaccia*  
<sup>3</sup>*Dept. for Environmental Studies*, <sup>4</sup>*INFN/PQE2000 Grant*

The computational activities currently performed in ENEA (computational chemistry and materials science, fluid-dynamics, robotics, weather forecast, oceanographic and global climate modeling, artificial vision) have prompted the assembly of an high performance platform able to efficiently perform in a large spectrum of computational applications. With this aim, a hybrid MIMD-SIMD architecture has been realized at the Computing Centre of ENEA Casaccia (Rome), in the frame of a joint industrial project ENEA-QSW (a Finmeccanica Group company) valued  $\sim 5$  millions euro. This Project, called PQE1, is a first concrete outcome of the project PQE2000 which groups, since 1995, the main italian scientific institutions (ENEA, CNR, INFN) and the industrial partner QSW. The PQE1 Project has provided an industrial spin-off to the parts of the PQE2000 Project which are in a mature state to be deployed for scientific and technological purposes.

The PQE1 machine consists, from the HW point of view, of the integration of a general purpose MIMD platform with distributed memory (Meiko CS2 with 8 dual HyperSparc 100 MHz nodes, with a proprietary "fat tree" HW-assisted network characterized by a I/O bandwidth of 50 MB/sec and a  $9\mu\text{sec}$  latency) with 7 SISAMD (Single Instruction Single Address Multiple Data) platforms (Quadrics/APE100) each of them with a different number of floating point units (see fig.1). The HW in-

---

\*Email: [rosato@casaccia.enea.it](mailto:rosato@casaccia.enea.it)

tegration of the MIMD with the SISAMD platforms has been realized by a two-ways communication line: the first, based on Transputer technology, capable to sustain a I/O bandwidth of 600 kB/sec: the second, based on a Hippi channel sustaining a 20 MB/sec. bandwidth. The computational power of the whole platform is  $\sim$  86 Gflops (83 Gflops in the SISAMD part and 3 Gflops in the MIMD part) and an addressable memory of 7.6 GB (6.6 in the SISAMD part, 1 GB in the MIMD part).

Several SW tools have been also realized in the frame of the PQE1 Project: (a) the QAPI libraries which allows communications between the MIMD and the SISAMD components [1]; (b) a suitable modification of the HPF compiler "Adaptor" to enable its interactions with QAPI calls [2]; (c) a Distributed Virtual Shared Memory (DVSM) software layer (based on MPI-CH primitives) to emulate a virtual shared memory, thus allowing the direct programming of the MIMD platform in shared-memory mode [3]; (c) a new Fortran77-compliant language (called Qfor) for the programming of the SISAMD part [4]; (d) a coordination environment called SKIE (SKEleton Integrated Environment) which provides a structured parallel programming environment [5, 6]. This tool, with a graphic interface, allows to build an application code by producing and/or assembling skeletons of the different computational parts. The produced graph of the application is firstly optimized and then mapped onto the physical machine. The latter task is performed upon a further optimization made on the basis of analytical models of performances (*templates*).

The PQE1 configuration at the ENEA Casaccia Computing Centre and all the computational resources related to the PQE1 project (development tools) shares the same file-system and are driven by the Load Sharing Facility (LSF).

The porting (or the generation) of computational codes on the PQE1 platform is performed by allotting to the SISAMD part only the specific parts of the computation which require the use of a fine-grain data parallel paradigm. These parts, in fact, can receive a significant power boost from the SISAMD machine architecture. The porting activity of application codes is, thus, usually restrained to a few sub-routines or a few parts of the code which are computationally intensive and which can be expressed by using a data parallel strategy. These parts must be partially rewritten in the native Tao language or implemented by using the new language Qfor.

The PQE1 platform has been used, so far, in a number of applications in different scientific and technological areas: (1) computational electromagnetics [7], materials science [8], wheater forecast [9], astrophysics [10], molecular modeling [11]. The PQE1 platform is going to be used, as a part of a metacomputing environment, together with high performance computational resources at the ENEA Computing Centre of Casaccia (Cray SV1) to perform specific computational tasks of vectorized codes.

The PQE1 platform will host, for the next 4-5 years, the data production in the computational domains which have received the most relevant benefits by the use of the platform. The ENEA-HPCN strategy for the next three years foresees the

set-up of a new hybrid platform based on the new generation SISAMD platform developed by INFN (called APEmille). The new platform will allow to re-use most of the SW tools developed for the PQE1 Project (SKIE, Qfor, QAPI) [12].

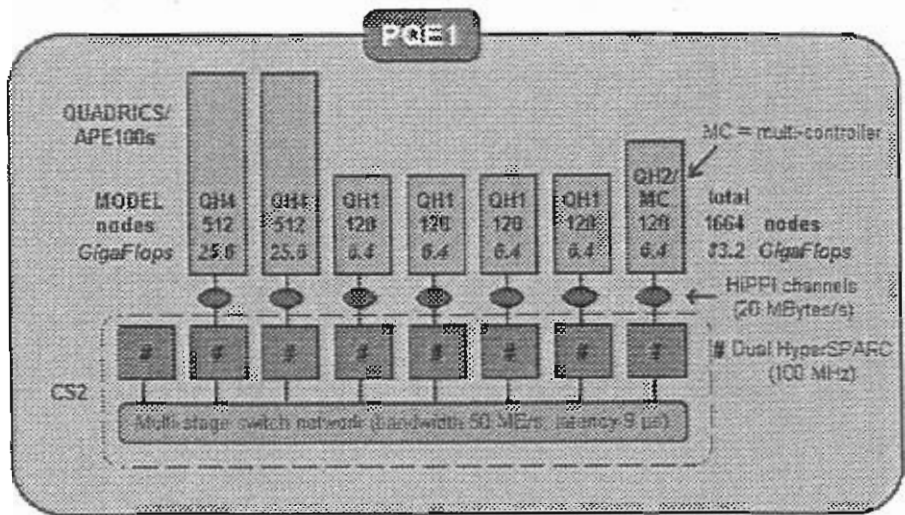
## References

- [1] PQE1 Technical Report QAPI, doc.PQE1-B61-01, N.Pucello, V.Rosato, S.Adda, "Benchmark della libreria QAPI".
- [2] Technical Report HPF, in preparation.
- [3] Technical Report DVSM, in preparation.
- [4] Technical Report Qfor, in preparation.
- [5] Technical Report SKIE, in preparation.
- [6] S.Pelagatti, "*Structured development of parallel programs*", Taylor and Francis, London, 1997.
- [7] P.Palazzari, S.Adda, P.D'Atanasio, "*Finite difference - time domain (FD-TD) simulation of electromagnetic field in complex environments through massively parallel systems*". To be published.
- [8] V.Rosato, N.Pucello, F.Cardellini, "*Simulation of x-ray diffraction patterns using a massively parallel MIMD-SIMD platform*". Proc. 7th Euromicro Workshop on Parallel and Distributed Processing - University of Madeira, Funchal, Portugal, February 3rd-5th 1999. IEEE Computer Society Ed. pag. 315. Special session on SIMD as computational engine for heterogeneous computing and large scale simulations.
- [9] S.Nicastro, F.Valentinotti, "*An Atmosphere-Ocean Forecast System on a Hybrid Architecture*" Proc. 7th Euromicro Workshop on Parallel and Distributed Processing - University of Madeira, Funchal, Portugal, February 3rd-5th 1999. IEEE Computer Society Ed. pag. 309. Special session on SIMD as computational engine for heterogeneous computing and large scale simulations
- [10] F.Saraceni, M.Coletta, V.Rosato, in preparation.
- [11] N.Pucello, M.Rosati, M.Celino, G.D'Agostino, F.Pisacane, V.ROSATO, "*New parallel Genetic Algorithm based on Molecular Dynamics approach for energy minimization of atomic systems: implementation on a SIMD-MIMD platform*", Proceedings of the Conference "Multi-scale Phenomena and their simulation", Centre for Interdisciplinary Research (ZIF), University of Bielefeld, Germany (Sept. 30 - Oct. 4 1996); N.Pucello, M.Rosati, M.Celino, G.D'Agostino, F.Fisacane and V.ROSATO, Int.J.Mod.Phys.C, 8 239 (1997).



[12] More details on the PQE1 platform can be accessed at the web page: [www.enea.it/hpcn](http://www.enea.it/hpcn).

Figure 1: Scheme of the hybrid MIMD-SIMD PQE1 platform.



## ABSTRACT

We report on the status and short term pain of the APEmille project. APEmille the last generation of APE machines is a massively parallel QCD engine.

## SUMMARY

Apemille is organized as a 3-D array of nodes, each consisting of a processing unit and its local data memory. There are direct link among each node and its six first neighbour. The edge of the array are closed like a toroid. All the nodes synchronously execute the same instruction and access their memories following a SIMD (Single Instruction Multiple Data) control scheme.

The processing units can directly access the data memory of remote nodes by means of the Apemille Communication Network. This is a synchronous network that works without contentions and hand-shakes. It is optimized for a low latency and large bandwidth communication among nearest neighbour processors (the bandwidth for remote memory accesses is 1/4 than the local one). The network is able however to manage more communication patterns, including automatic routing to arbitrary nodes for homogeneous transfer and several varieties of broadcast communications.

The P.U. of Apemille is mainly an arithmetic processor able to execute 500Mflops. It supports several data types: floating-point 'normal' operation ( $a \times b + c$ ) are implemented for 32-and 64-bit IEEE format, as well as for single precision complex and vector (pair of 32-bit) operands. Arithmetic and bit-wise operations are available for integer data types. Operands can be converted between the various formats.

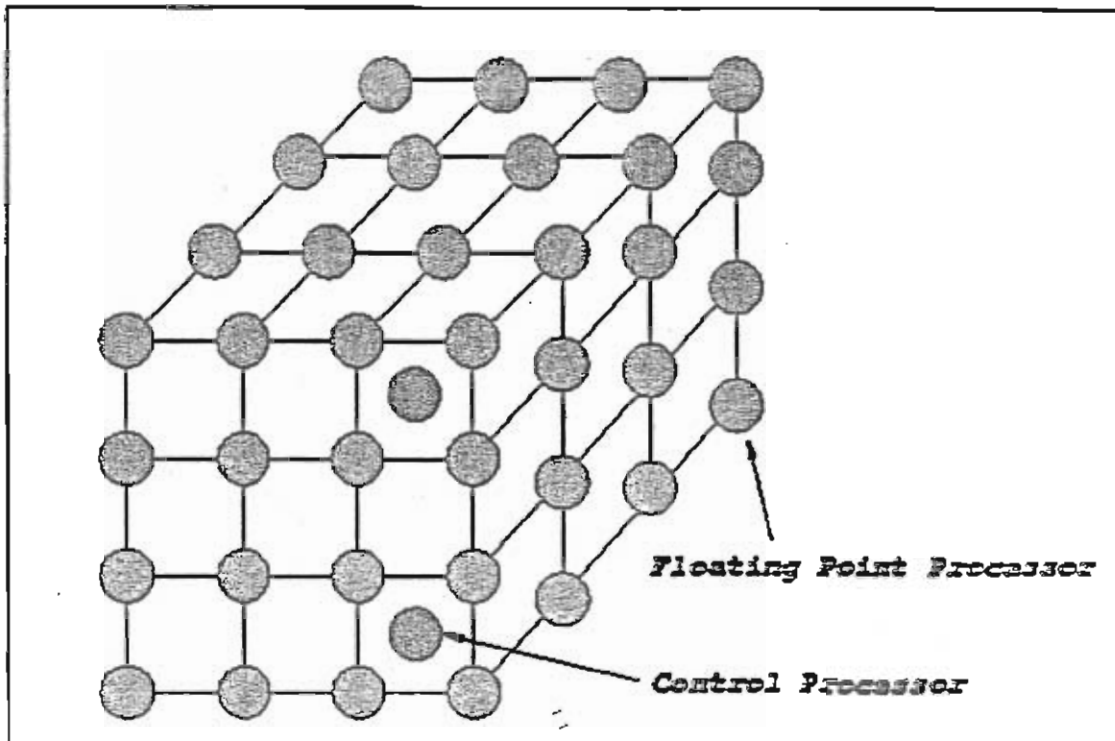
The HW building block of APEmille is a Processing Board that houses 8 nodes (logically arranged at the corners of a cube), a controll processor and a communication controller. In addition, each P.B. has its own Pci-Based host interface. APEmille is hosted by a cluster of networked Linux-PCs . Each PC controls a group of 4 PB (32 nodes). For each PC the APEmille PBs are PCI-peripherals, so every exchange of data is executed throught the C-PCI bus.

## STATUS

Hardware prototypes are under test using preliminary versions of the APEmille operating system and compiler. According to the official planning the first machine, able to perform 60Gigaflops, is scheduled to be ready by the end 1999, several 250 Gigaflops machines are going to be build by year 2000.

# Architecture of APEmille

---

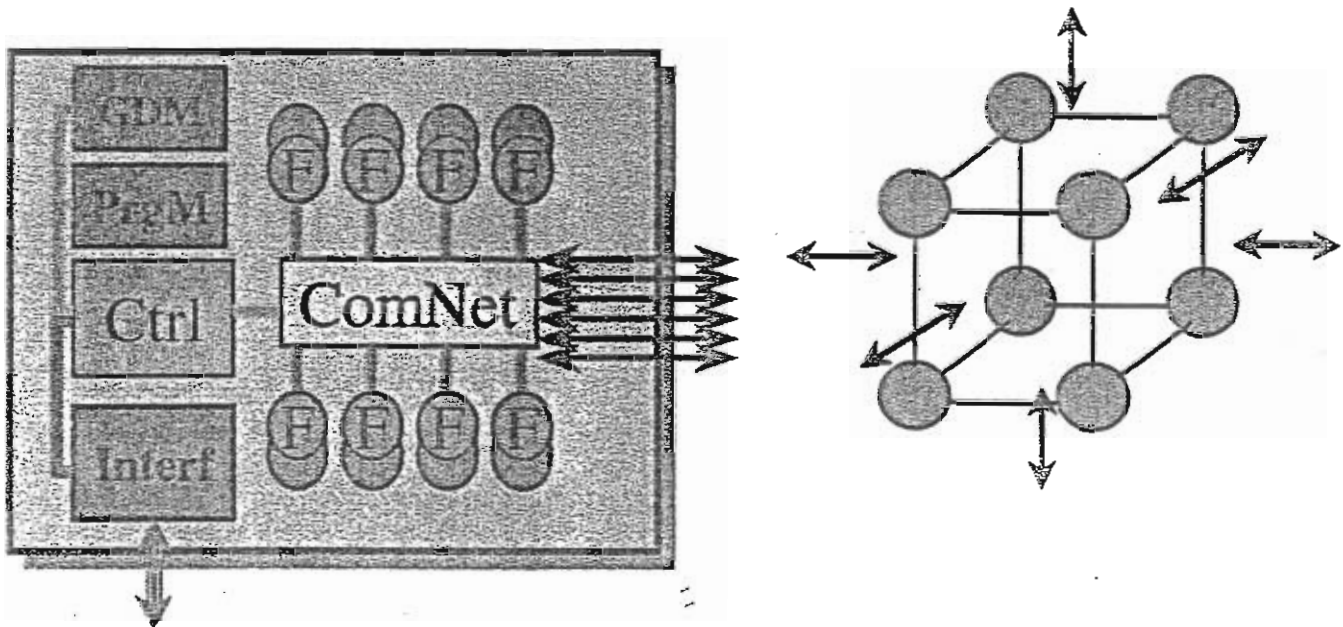


## Characteristics:

- **3D Array of Floating Point Processors:**
  - Large Register File with direct access to local and remote memory
  - Normal Operations  $A*B+C$  for single, double, complex ....
- **Single Instruction Multiple Data (SIMD)**
- **Very Long Instruction Word (VLIW)**

# Processing boards

Board = 8 FP + Controller + Communication Network + Host Interface

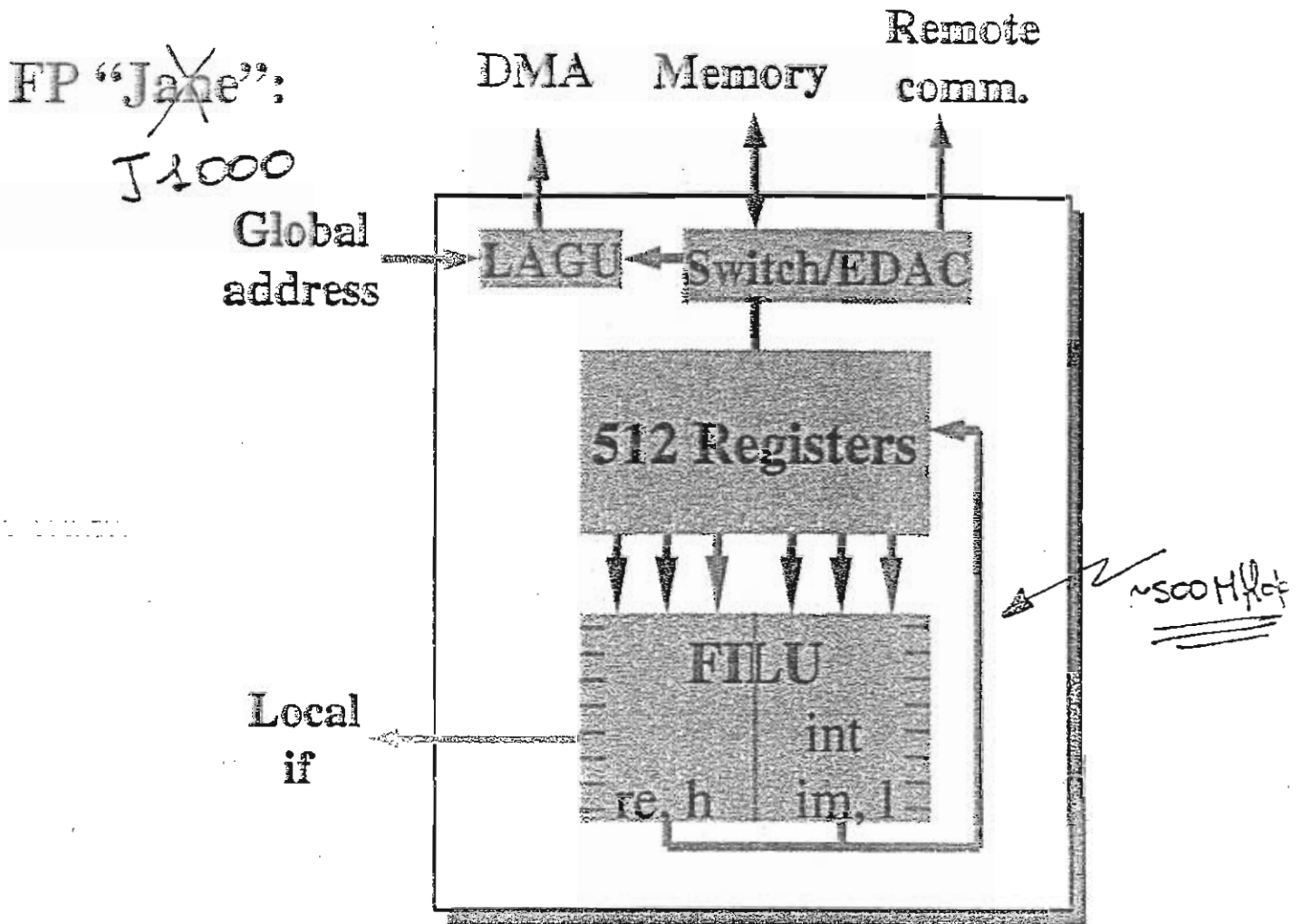


## Processors:

- ASIC design, 0.5  $\mu$  standard cell CMOS
- ca. 400 k gates per FP processor
- ca. 3 W power consumption per FP processor
- Clock: 66 MHz
- Memory:
  - Program: 512 k instructions (96+80 bit) synDRAM
  - Controller data: 128 k words (32 bit) SRAM
  - FP local data: 2-8 M words (32 bit) synDRAM

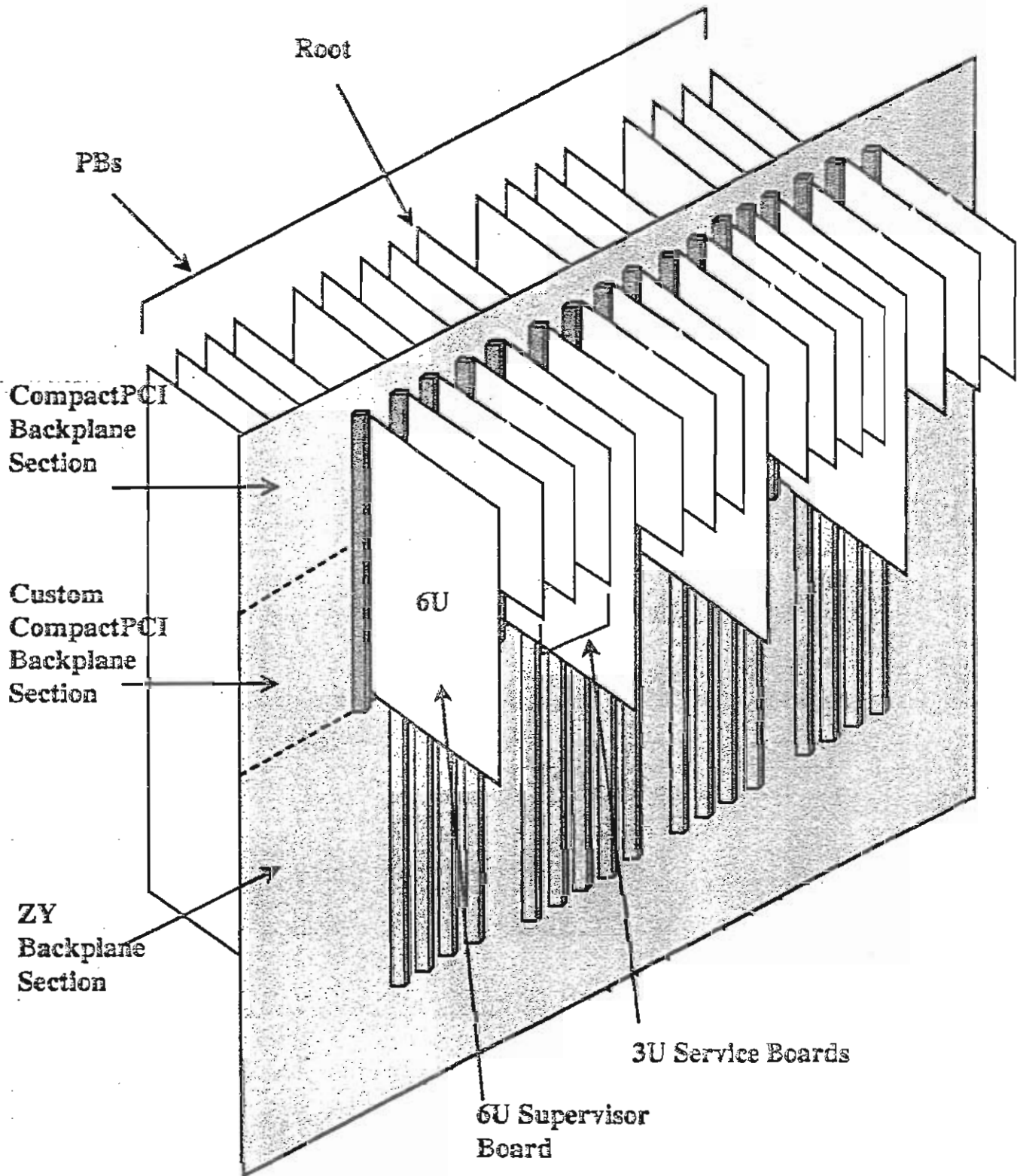


# Processors

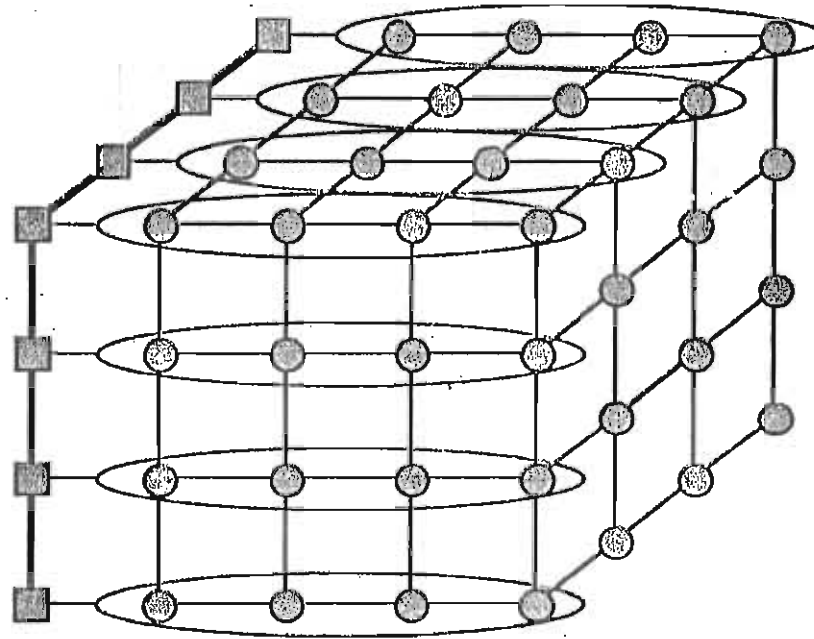


- Normal operations:  $a*b+c$ 
  - single, double, complex, vector
  - IEEE with IBM precision enhancement
  - 1 pipelined normal operation per clock cycle
- Local integers:
  - bit-wise operations
  - independent local AGU
- Large RF:
  - 512 words
  - independent memory access (1 per cycle)

# Crate / Back-plane



# The APEmille host environment:



● = Processing Board

■ = PC

— = Internode link

— = PC link

— = APEchannel

a network of PCs running LINUX





Istituto Nazionale di  
Fisica Nucleare  
*Italian National Institute  
for Nuclear Physics*

---

# APEmille system software

Davide Rossetti  
The APE Collaboration






# APEmille system software - index

---

- APEmille HW:
  - Overview
  - APEmille and clusters: some considerations.
- APEmille System Software:
  - Main features
  - Why Kome ?
  - What is Kome ?
  - Work in progress

## APEmille system software - System SW: main features

### APEmille system software figures:

- A **simple interface**, both tty and graphical, to APEmille distributed HW.
- Full **run-time support** to TAOmille programs.
- **Scalable** from a single PB to the full APEmille.
- **Best I/O performance** from HW:
  - Host  PB (over PCI, ~ 80 MB/s)
  - Host  RAID (over FibreChannel, ~ 50 MB/s)
  - Host  Host (over FLink, ~ 80 MB/s)
- Moderate **partition-ability**.

## APEmille system software - System SW: main features

---

### APEmille System SW simplified:

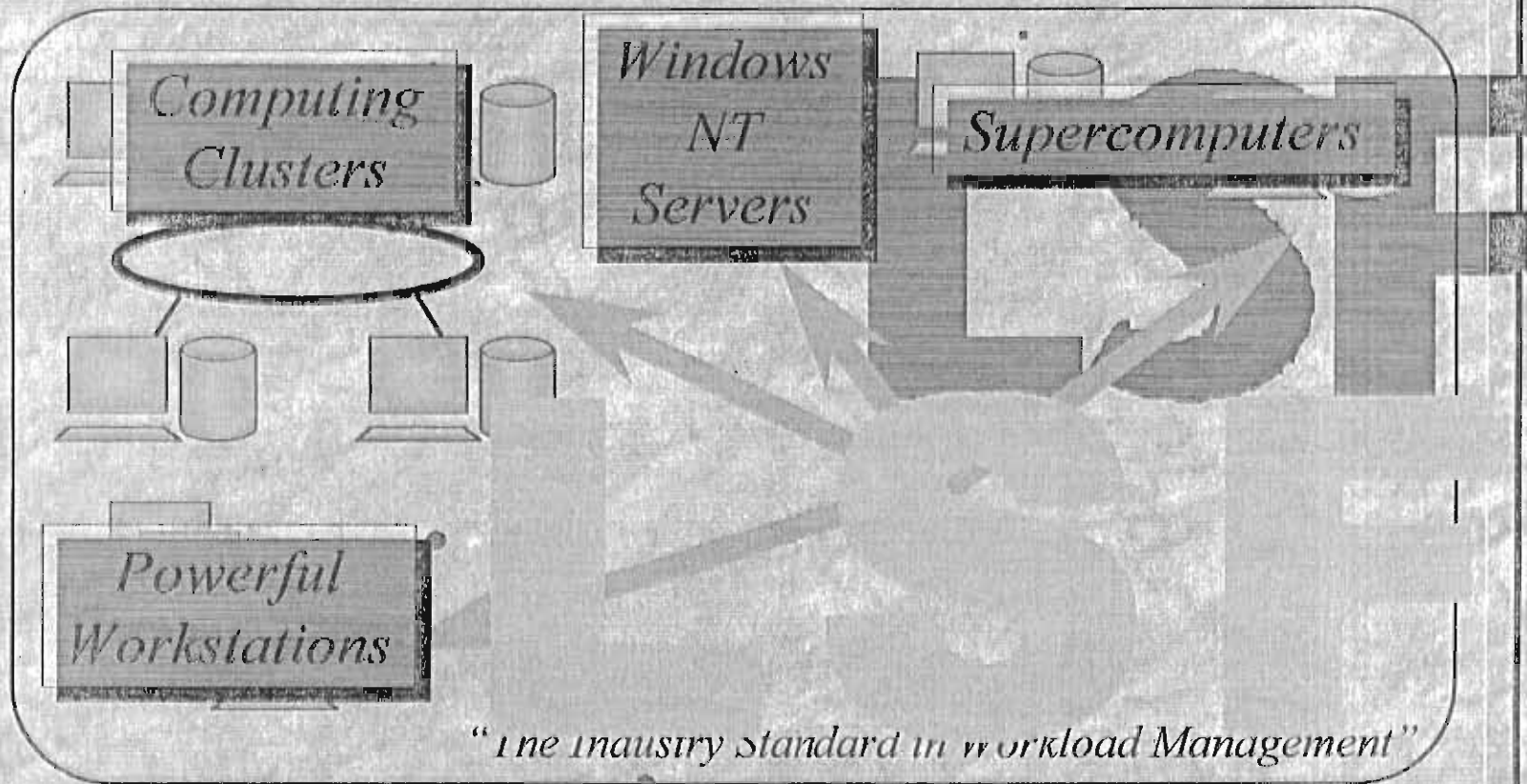
- **Tools:** loader, monitor, debugger.
- **Kome:** distributed objects across Flink or FastEthernet.  
Our enabling technology to distribute the APEmille OS
- **Linux kernel drivers:**
  - APEmille PB's (mixed 90% user / 10% kernel mode).
  - Flink card (both full kernel mode and 90/10% mixed mode).
  - FibreChannel adapter (kernel mode, fully integrated into the Linux SCSI layer).
- **BIOS customization (optional extension):**
  - Console over serial line from the beginning of the boot phase.
  - OS bootstrap from the network.

## APEmille system software - System SW: what is Kome ?

---

- Basic task: *remoting* of C++ objects.
  - Object model: that of C++ language.
  - Calling interface: strictly static.
  - Simple IDL language: supporting both built-in C++ types and aggregated types.
  - Automatic proxy (client stub) / stub (skeleton) generation.
  - Server side execution: multi and single threaded.
  - Transparent support for **APEmille Asynchronous Network** (*Flink* with its efficient and low latency APE protocol or *FastEthernet* using UDP).

# The Virtual Supercomputer

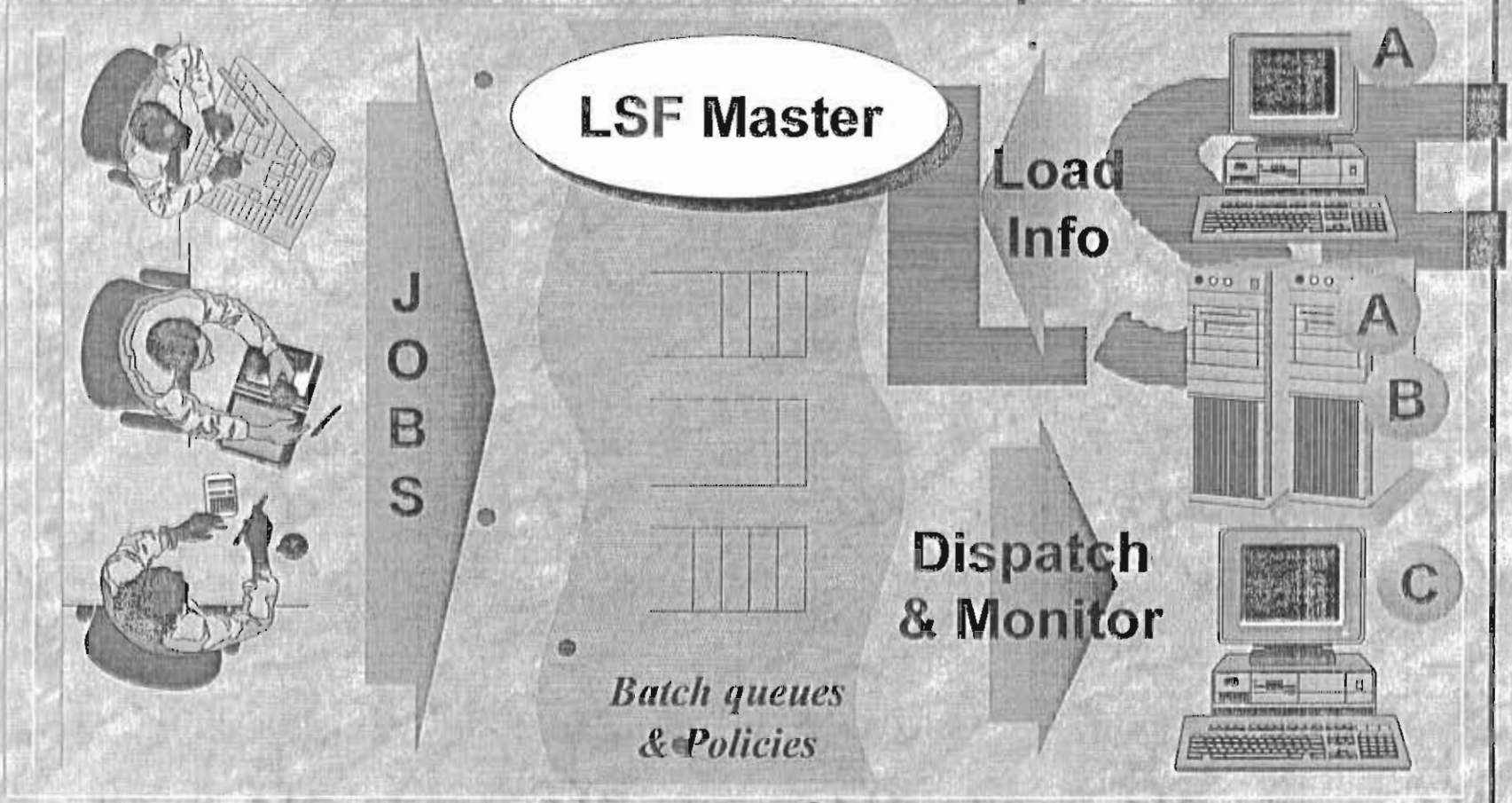


**NICE - Networking Information Communication Engineering**

# Interactive Management



# Batch Management.



# Parallel Management



J  
O  
B

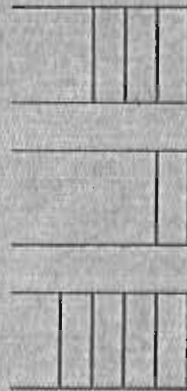
2 Proc. SMP



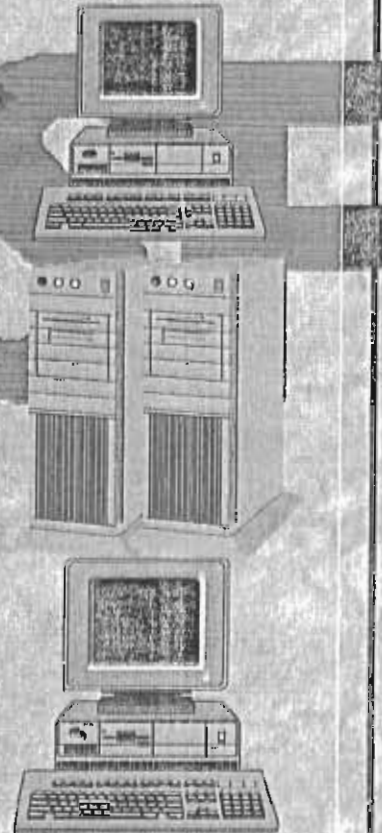
J  
O  
B

4 Proc. MPI

LSF Master

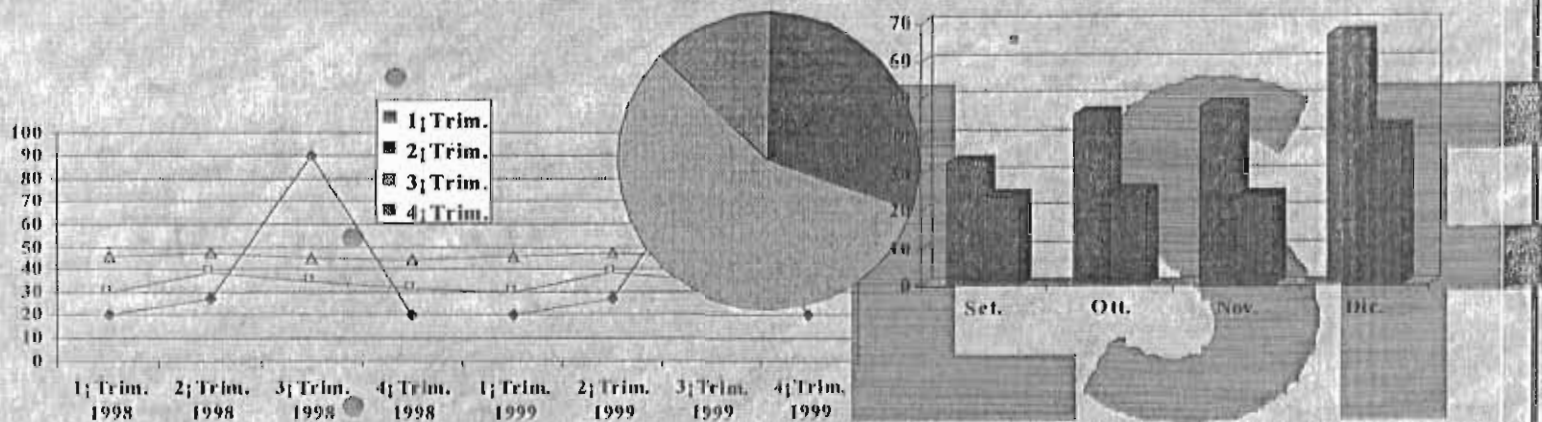


*Batch Queues  
& Policies*





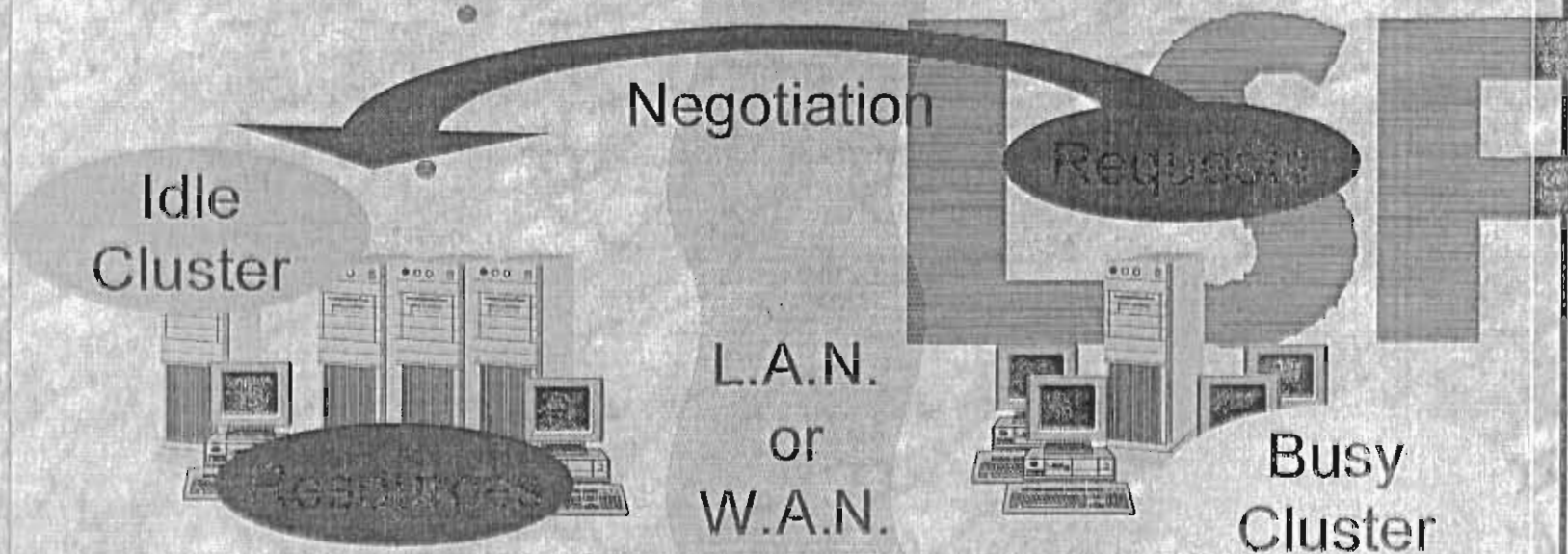
# Contention Management



## ■ Detailed reports through LSF Analyzer

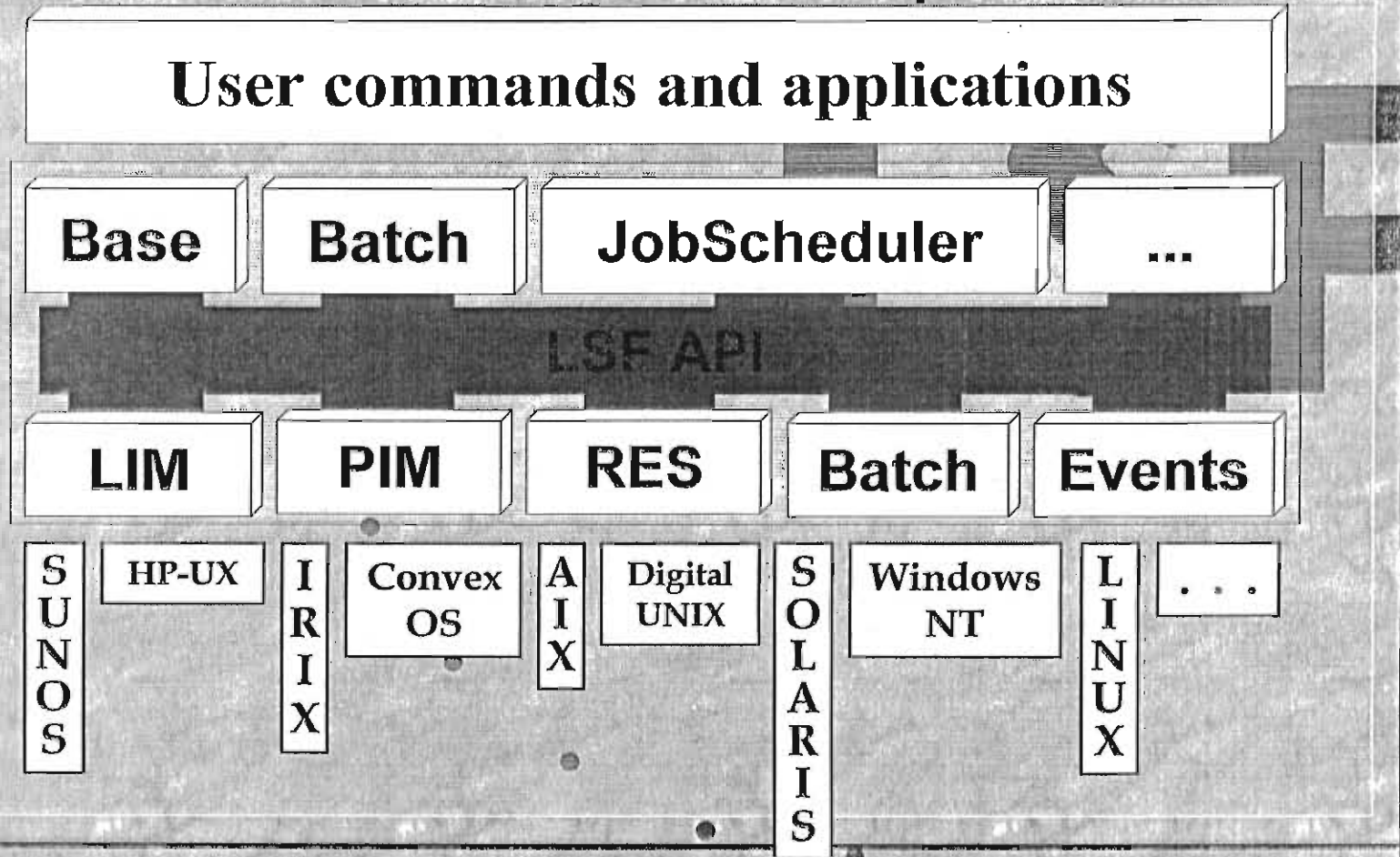
- Decision support for purchase and upgrades
- Chargeback accounting
- Performance analysis

# MultiCluster Management



- Automatic redirect when needed
- Autonomous management

# LSF Architecture



# NICE

Networking  
Information  
Communication  
Engineering

## Company Profile

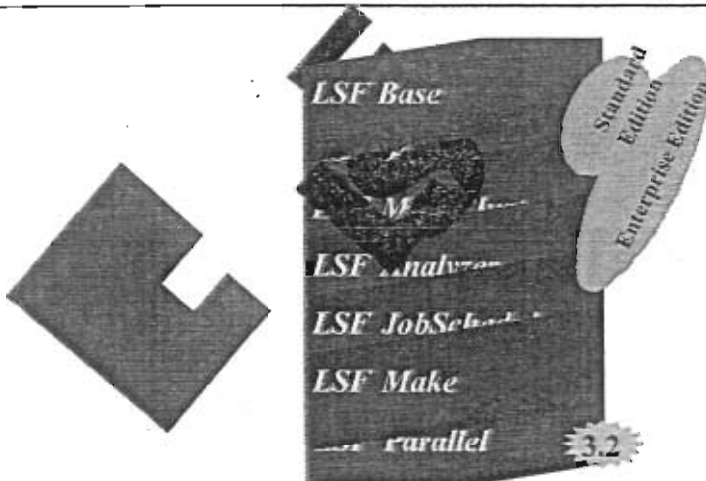
### Mission

NICE understands the demand of information technologies and establishes enduring relationship with customers by jointly solving problems related to **computing resource optimization and rationalization**.

### Competencies

NICE offers industry leader **software products** aimed at company-wide management and optimization of computing resources. NICE **qualified personnel** can provide consulting services related to *Inter-Intranet technologies, heterogeneous systems integration, Java development, technical computing, performance analysis, parallel and distributed computing*.

### Software Products



The LSF Products Suite provides a complete enterprise solution for **Workload Management**, including sophisticated *load sharing, dynamic job scheduling*, and extensive *workload analysis* for distributed heterogeneous Windows NT and UNIX computing environments. On the web: <http://www.platform.com>

NICE is the Italian **Master VAR** for LSF, and provides highly qualified support and services.

### Software Support

According to its role of Italian Master VAR, NICE provides **software support services** to all Italian LSF customers. This contract includes telephone and e-mail first level technical support, problem investigation and transfer for second level issues, and free upgrade for bug fixes, major and minor releases of the products.

## Consulting Products

---

In order to provide best value to its customers, NICE has packaged some of its consulting activities for LSF evaluation and deployment:

<b>NICE/Install</b>	provides a quick startup for the LSF evaluation, including a detailed analysis of previous workload and a preliminary training for the system administrator
<b>NICE/Design</b>	is focused on the design of an optimal LSF implementation based on users' and administrators' feedback, and measurements of the resulting workload
<b>NICE/Tuning</b>	enables production clusters to face new challenges, taking advantage of the latest LSF features and experiences
<b>NICE/Training</b>	provides system administrators the knowledge and hands-on experience needed to support and enhance a fully functional LSF cluster

## Project Management

---

NICE manages specific customers' requirements or targets through projects. By the combination of competencies and cooperative project management, we could deliver state-of-the-art and robust solutions with the following targets:

- LSF functionality extension beyond its limits  
*(Integration with license managers, new software, new hardware, ...)*
- Implementation of High Performance Computing solutions  
*(Distributed parallel computing for CFD, meta-computing solutions on Wide Area Networks, ...)*
- Deployment of a consistent engineering environment  
*(Java/HTML interface to the computing resources, shared file system over WAN, ...)*

## Our Customers

---

<b>Mechanical</b>	Ferrari, FIAT Avio, Centro Ricerche FIAT, Iveco, ELASIS, Pininfarina, UTS
<b>Energy</b>	ABB, Ansaldo, Nuovo Pignone
<b>Pharmaceutical</b>	Pharmacia & Upjohn
<b>Aerospace</b>	Alenia
<b>Electronics</b>	STmicroelectronics
<b>Telecom</b>	Telecom Italia
<b>Consulting</b>	Arthur Andersen, Elsas Bailey
<b>Computers</b>	DEC/Compaq, Hewlett-Packard, Silicon Graphics, Sun
<b>Research</b>	ENEA, ICTP, INFN, CASPUR, CILEA, CNR, Osservatori Astronomici
<b>Education</b>	Scuola Normale Superiore di Pisa, Universities

# The GAMMA Project: Overview and Recent Achievements

Giuseppe Ciaccio  
DISI, Università di Genova  
via Dodecaneso 35, 16146 Genova, Italy

GAMMA (Genoa Active Message MACHine) [1, 3] is a prototype communication system based on the Active Ports paradigm [2], designed for efficient implementation over Fast Ethernet interconnects to be used by Linux clusters of PCs. Thanks to a carefully optimized implementation of the communication protocol and to the adoption of Active Ports (a communication mechanism derived from Active Messages), GAMMA is able to deliver excellent communication performance to message passing parallel applications. The communication latency of GAMMA with DEC 21143-based Network Interface Cards (NICs) and Pentium II 350 MHz PCs is as low as 14  $\mu$ s, whereas the maximum communication throughput is 12 MByte/s.

GAMMA achieves unprecedented performance on Fast Ethernet by using a so-called "lightweight" communication protocol which however has been criticized for not providing reliable message delivery, thus not providing a sufficient quality of service to real-world end users. Moreover, the GAMMA application programming interface (API) is not an industry-standard one, and this implies a substantial rewriting effort to run existing parallel applications atop it.

Both the issues of reliability and presentation of an industry-standard API have recently been addressed. GAMMA communications are now reliable because of a slight enhancement of the underlying communication protocol with a credit-based flow control mechanism. Indeed, in a properly cabled Fast Ethernet LAN, the only source of unreliability is the possible packet overflow at the receiver side of a physical communication (collisions in a shared LAN are

correctly managed and recovered by GAMMA since previous prototypes). The adoption of a credit-based flow control mechanism prevents packet overflow from occurring while requiring only very few additional packet exchanges for the protocol. This results into reliability up to hardware faults without any noticeable performance penalty. Moreover, a porting of the MPICH implementation of MPI, an industry-standard API for message-passing parallel programming, has recently been done, leading to what appears to be the fastest implementation of MPI for Fast Ethernet ever: on a pair of Pentium II 350 MHz equipped with DEC DE500 Fast Ethernet NICs and connected by a Fast Ethernet repeater hub, MPI/GAMMA achieves 17.7  $\mu$ s latency and 11.3 MByte/s maximum communication throughput. This will contribute to substantially widen the range of parallel applications that may effectively run on a low-cost cluster of PCs.

## References

- [1] G. Chiola and G. Ciaccio. GAMMA home page, <http://www.disi.unige.it/project/gamma/>.
- [2] G. Chiola and G. Ciaccio. Active Ports: A Performance-oriented Operating System Support to Fast LAN Communications. In *Proc. Euro-Par'98*, LNCS 1470, 1998.
- [3] G. Ciaccio. Optimal Communication Performance on Fast Ethernet with GAMMA. In *Proc. Workshop PC-NOW, IPPS/SPDP'98*, LNCS 1388, pp.534-548, 1998.

## **The GAMMA project: Genoa Active Message Machine**

### **Main goals**

- \*) Squeeze best performance out of cheapest hardware: PCs, Fast Ethernet.**

**This leads to deep know-how and to innovative, often counter-intuitive solutions.**

- \*) Then, find suitable trade-offs:**

**Performance Vs. Reliability  
Performance Vs. standard APIs.**

- \*) Do all that for point-to-point as well as collective communications.**

- \*) Try all that on next-generation interconnects: Gigabit Ethernet.**

- \*) Byproduct: my PhD thesis.**

## Recent Achievements in the GAMMA project:

### A really fast porting of MPI atop GAMMA

- \*) MPICH, the most used implementation of MPI for clusters, is stacked over (at least) two layers:**

**ADI (implemented atop Channel)  
Channel (implemented atop TCP/IP stack)**

**Performance is poor! Pentium II 350:  
latency 131 usec, max throughput 10 MByte/s**

- \*) It is possible to port MPICH atop GAMMA in two ways:**

**implement ADI layer on GAMMA: better performance  
or  
implement Channel layer on GAMMA: easier to do**

- \*) Second approach is under testing.**

**First approach has led to a working prototype.**



## **Recent Achievements in the GAMMA project:**

### **A really fast porting of MPI atop GAMMA**

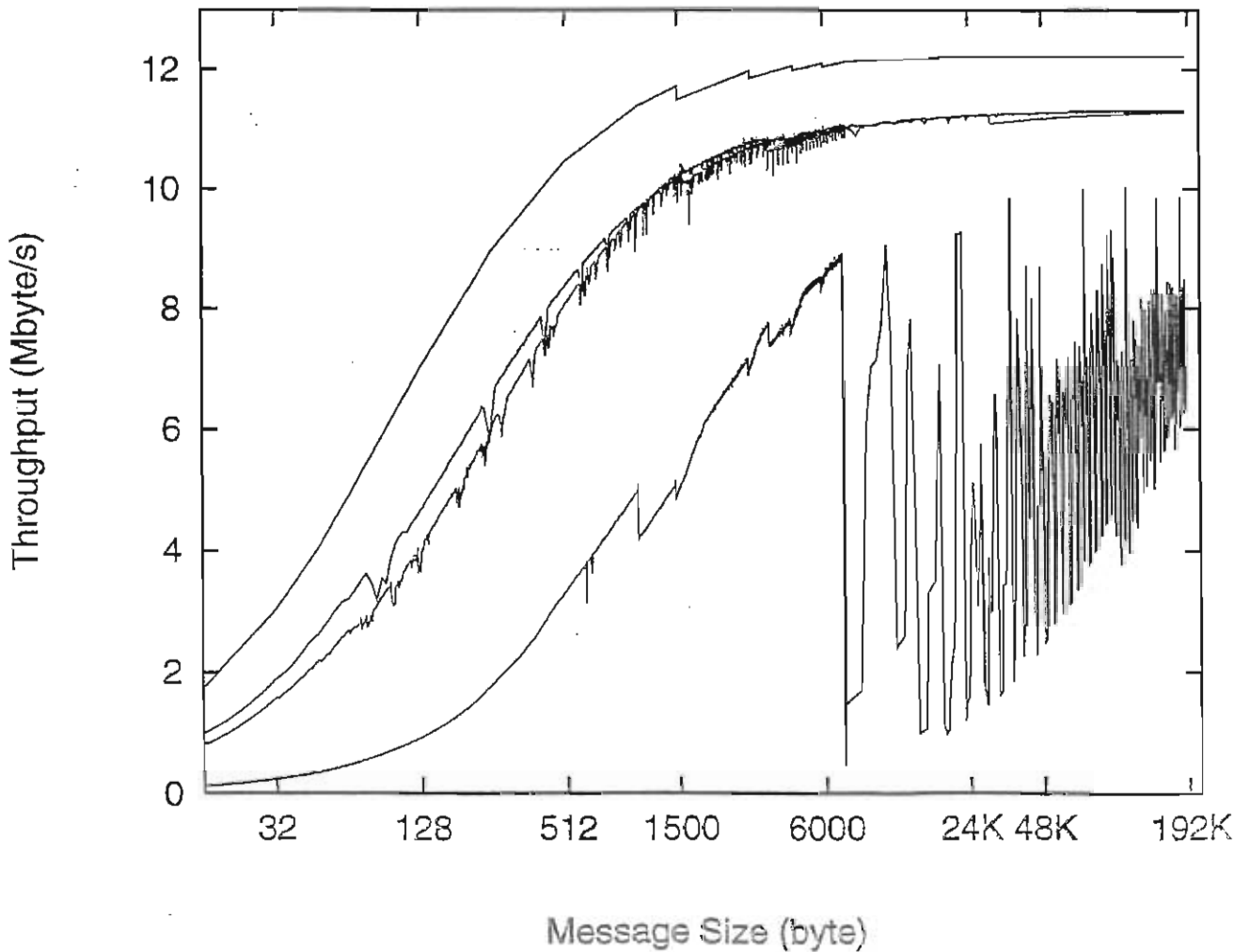
- \*) Substantial rewriting of the ADI layer atop GAMMA.**
- \*) Minimal number of temporary copies of messages:  
no additional copies whenever an incoming message is  
“expected”;  
  
one additional copy on receiver side otherwise.**
- \*) No dynamic allocation of memory for received messages:  
“expected”: stored into final destination directly;  
  
“unexpected”: stored into statically allocated buffers,  
managed by a credit-based policy.**
- \*) Main result:  
The fastest implementation of MPI for Fast Ethernet ever.  
  
MPI “ping-pong”, Pentium II 350, Fast Ethernet hub:  
  
latency: 17.7 usec  
(one order of magnitude improvement)  
  
bandwidth: 11.3 MByte/s (12.0 MByte/s on switch ??)**

## **Recent Achievements in the GAMMA project:**

### **Reliable, flow-controlled communication**

- \*) Added support to credit-based flow control in the GAMMA device driver.**
- \*) Implemented a reliable “send” operation using flow-controlled delivery.**
- \*) Main result:**  
**Adding flow control at driver level has reduced impact on communication performance.**

Optimal Fast Ethernet Throughput: latency 7 usec —  
GAMMA gamma\_send\_flowctl(): latency 14.3 usec —  
MPI/GAMMA: latency 17.7 usec —  
MPICH/P4 on Linux 2.0.29 TCP/IP: latency 131.2 usec —



# Using active messages to port parallel applications on PC clusters.

V.Di Martino<sup>1</sup>,

CASPUR c/o Univ. di Roma "La Sapienza" vincenzo@caspur.it

**Abstract.** The use of cluster of workstation and cluster of PC's is a well established tool in the numerical intensive computer simulation environment. The fast improvement in hardware performances in both processor units and network interconnection it is not followed by an appropriate system software design. Cluster of PC's suffer of the lack of fast communication layers to be competitive with vendor cluster of workstation. Several research group are working to the development of such communication layer, one of them, the GAMMA project obtained the best communication performances on commodity hardware like fast Ethernet board. To test the quality and validity of such approach we ported two parallel code in the Active Messages library developed by the GAMMA project. The porting require two man weeks and for one of the two parallel code we obtained relevant enhancement respect to the PVM parallel version. For the other code, the Flame Front Propagation Problem that solve a Hamilton-Jacobi equation using a domain decomposition approach the communication-computation ratio was low and for this number of node the speedup was almost the ideal already for the PVM implementation. For this code the usage of ports as required by AM reduced the complexity of the communication coding. In the case of the Molecular Dynamics code the long range forces require an high amount of communication all to all. In this code the improvement respect to PVM is large and reflects the low latency of the GAMMA communication protocol. In this presentation we will describe the effort requested to the application programmer to port his application on a AM communication library.

## References

1. V. Di Martino, G. Ruocco, M. Sampoli  
Molecular dynamics of polarizable fluids on parallel systems.  
HPC-ASIA '95 September 18-22, 1995 Taipei, Taiwan.
2. G. Ciaccio, V. Di Martino  
Porting of a Molecular Dynamics application on a Low-cost Cluster of Personal Computers running GAMMA  
Workshop PC-NOW 1998 IPPS/SPDP, March 1998 Orlando, Florida.
3. G. Ciaccio, V. Di Martino, P. Lanucara.  
Porting the Flame Front Propagation Problem on GAMMA,  
HPCN Europe '98, Amsterdam, The Netherlands, April 21-23, 1998.
4. G. Ciaccio, V. Di Martino, Efficient Molecular Dynamics on a Network of Personal Computers,  
VECPAR'98, Porto, Portugal, June 21-23, 1998.

Using Active Messages to port  
parallel application on  
PC clusters

V. Di Martino CASPUR -ROME

G.Ciaccio, P.Lanucara

CASPUR parallel application history

*APE 100*

*Cluster of workstations*

*8 ALPHA + Gigaswitch*

*SP2 8...16...32*

*4 Alpha 2100 + memory channel +2*

*SUN Enterprise 8 + 14*

*PC cluster*

Parallel Languages and libraries

*Tao*

*Linda*

*PVM*

*MPI*

*F90 HPF*

*Open MP , SMP*

Porting on Gamma

*MD , classical applied to Fluids*

*Communication intensive*

*PDE Hamilton Jacobi*

*Domain decomposition*

Porting on Gamma Why

*PC cluster latency too bad*

*Active Message programming "benchmark"*

*Italian group*

GAMMA verification on a real code

*First large application ported on GAMMA*

*Minor problems discovered*

### Short description of the MD-H2O code

*Short distance LJ forces*  
*Long distances Polarization forces*  
*Many communications any to any*  
*Developed using Linda and ported in PVM*  
*It ran on SP2, Convex, T3D, clusters of SMP*

### Porting strategy

*A new PVM collision free SPMD version*  
*Port initialization, cyclic buffer allocation*  
*Extra byte sent to use the broadcast facility*  
*PVM send changed in AM*  
*The receive handler function used to keep the data consistent at each iteration*

### GAMMA HW and SW reliability

*20 hours run with no loss of data*  
*No bug discovered in the system code*  
*A different result respect to vendor PVM*

### Any to Any random communication versus tokens

*Contention on network switching*  
*TCP latency*  
*Hub or switch*  
*60% of the porting effort*

### Port initialization

*Only 255 port available*  
*MD, Iterative procedure*  
*Circular buffer trick to expand the number of reliable communication channels*  
*Port definition easy, more evident in the domain decomposition example*

### PVM message - Active Message

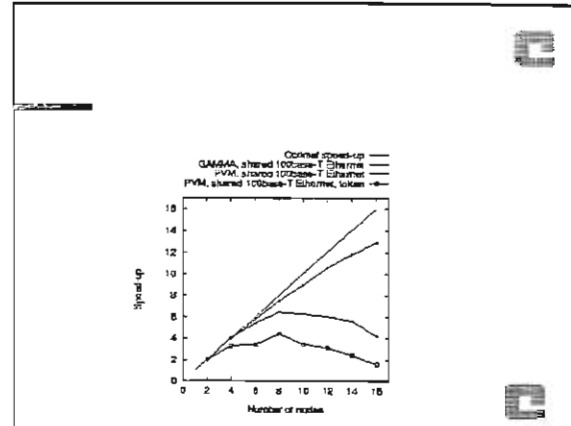
*Send instruction naturally converted*  
*Longer messages to use the broadcast port*  
*Receiver handler coding add complexity on the receiver side, new programming opportunity*

## Data buffers and Receiver side gamma subroutine execut.cost

*Data written on slave process data space*

*Light GAMMA functions to be called in the  
receive handler*

*Caution in data access on receive buffers*



## Conclusion

*Cluster of PC, when*

*First GAMMA release was not "exportable"*

*Very good communication performances*

*MPI extensions are welcome.*

# The INFN Condor experience

S.Parlati

March 24 1999

## Abstract

In this presentation I'll give an overview of the CONDOR batch system and a description of the INFN CONDOR pool and its usage.

While the number of machines and the computational power are rapidly increasing in any institutions or companies, the resources a single user or a group can access are very often limited; often computers are lying on the user's desk, idle most of the time.

The Condor philosophy is to use idle CPU cycles of non-dedicated, preexisting machines in a distributed ownership setting.

Instead of running CPU-intensive jobs in background on their workstations, users submit jobs to Condor which will find an available machine on the network. If that machine becomes no longer available (for instance its owner start working on it), Condor checkpoints the job and migrates it to a different idle machine (figure 2).

Machine's resource offer and job's resource requests can be easy and freely configured through ClassAds expressions: Condor match them to find the best resource available on the net for each particular job.

No changes in the source code are needed to run a program with Condor, just a re-link phase is required.

Condor is developed by the Winsconsin-Madison University and it's available for free for many Unix platforms; port to Windows-NT systems is under way.

The INFN "commissione calcolo" in 1998 approved a project to test the suitability of Condor to satisfy the INFN computing needs. Since the INFN computing resources are geographically distributed it was decided to test the Condor pool efficiency in a WAN environment.

More than 120 workstations and PCs (5 different architectures), located in almost every INFN unit, participate currently in the INFN Condor pool (figure 3).

Figure 4 shows the INFN pool statistics for February 1999: about 40000 hours of CPU have been consumed by Condor jobs.

Many test job have been run in the Condor pool and the first results show that Condor is a robust and reliable computational tool for the INFN needs.

Very intensive I/O jobs have poor performance in a WAN distributed environment, while the impact on the network of large checkpoints is acceptable.

Condor has proved to have very flexible priority mechanism for users or machines: in this way logical "sub-pools" can be easily configured to give, for instance, local machines to local users first.

The collaboration between INFN and Madison University is still continuing in order to adapt Condor to INFN specific needs.



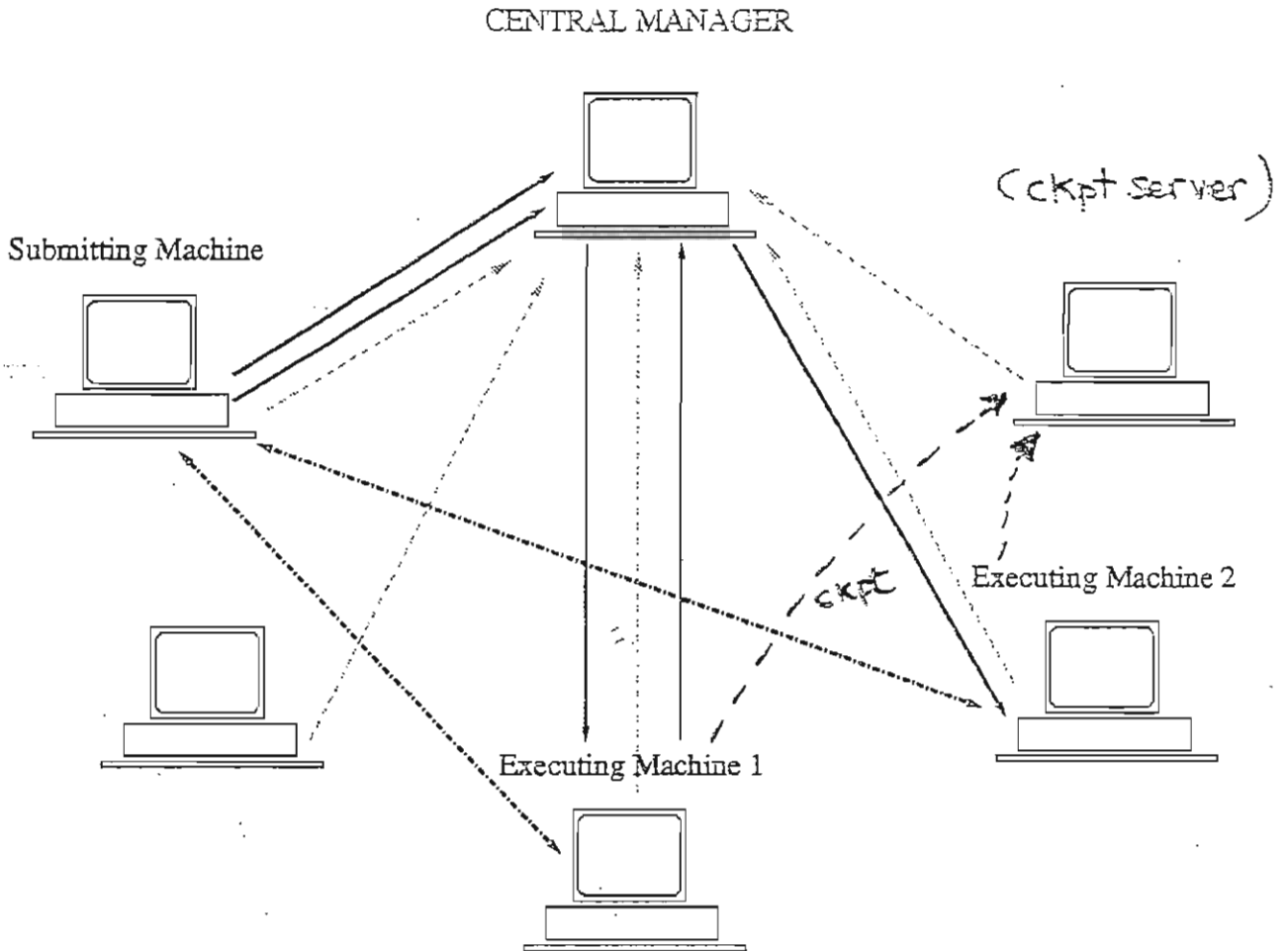
## A hunter of available workstations

Instead of running CPU-intensive jobs in background on their workstations, users submit jobs to Condor.

Condor will find an available machine on the net and begins running the job there.

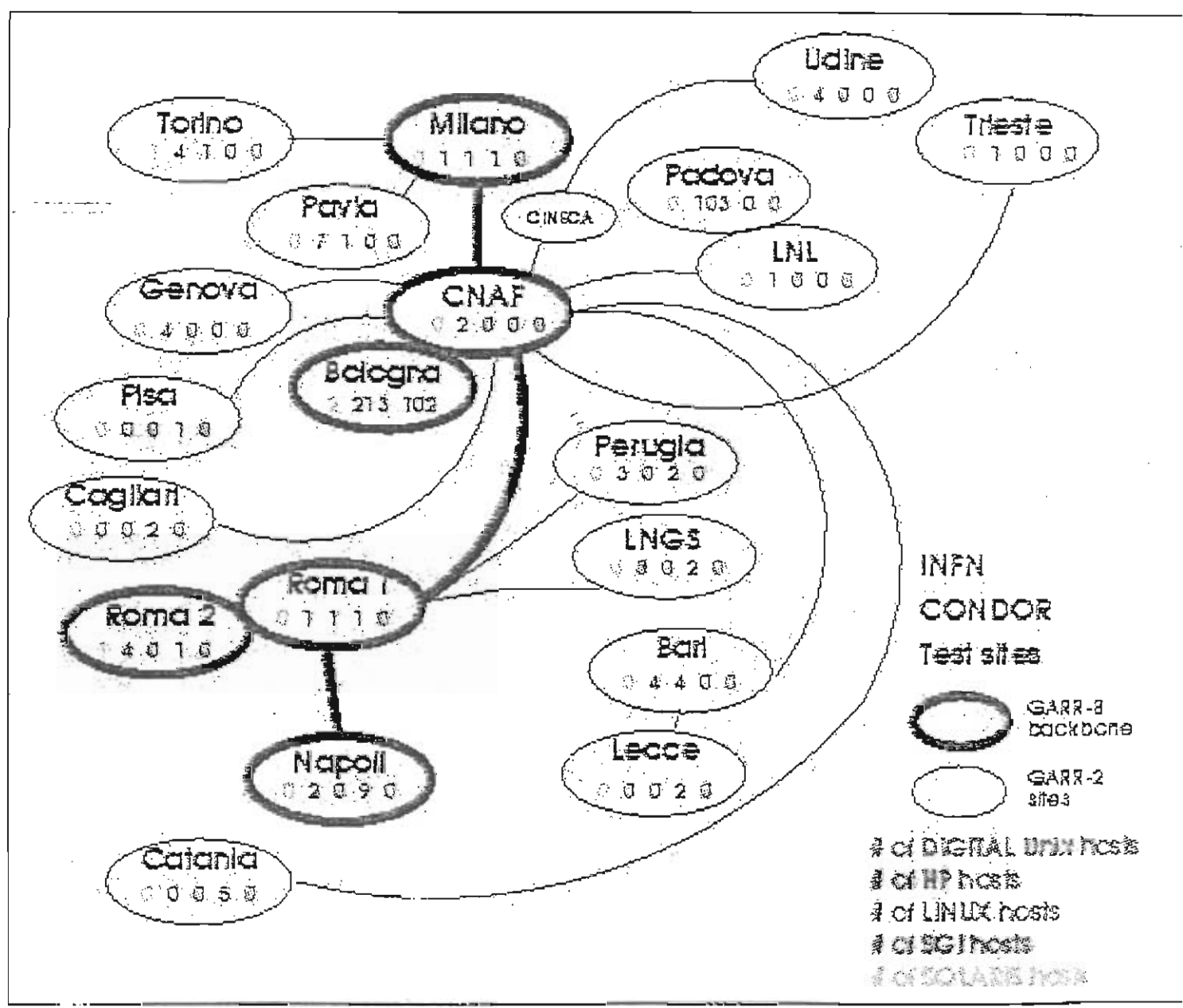
If the machine becomes no longer available, Condor checkpoints the jobs and migrates it to a different machine.

# CONDOR POOL



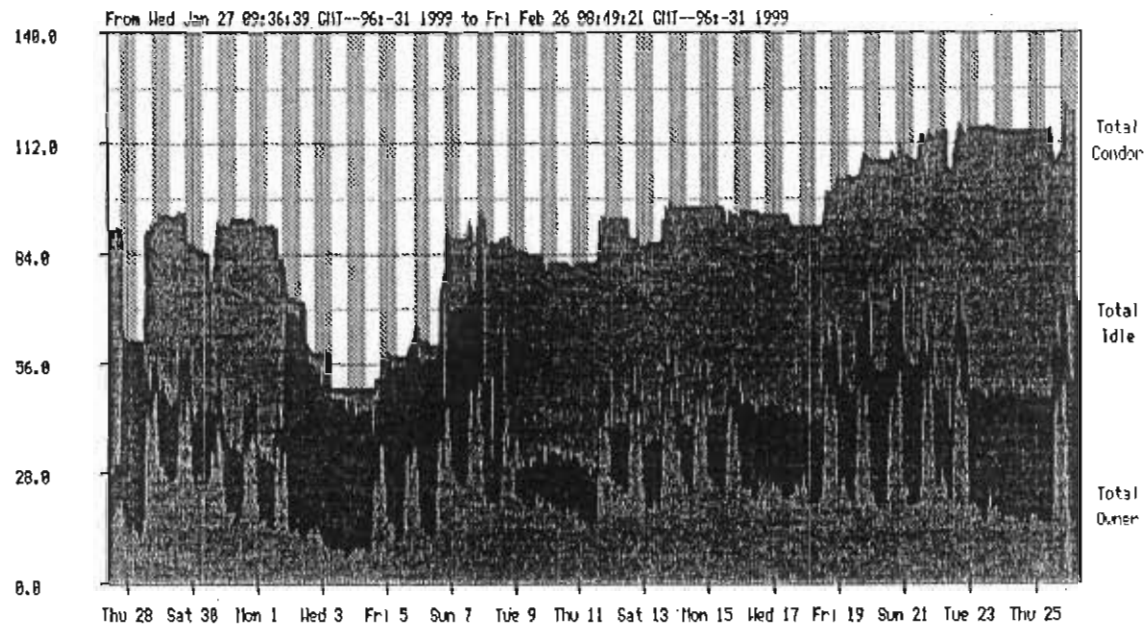
- INFO about CPU load, interactive activity, machine ClassADs
- Job ClassADs
- ..... System calls, checkpoints and data
- > Job execution sequence

CENTRAL MANAGER  
+  
CRPT server



INFN CONDOR POOL

## INFN Condor Pool Machine Statistics for Month



## INFN MAIN pool results

Condor is robust, reliable computational pool for the INFN needs.

Acceptable impact on the network even for large I/O jobs and checkpoints.\*

ClassAds effective to configure sub-pools: local machines to local users first.

\* Very large or frequent I/O gives poor performance!

## A farmer's live or Simulating fluctuating geometries on a PC-Farm

Random geometries play an important role in many fields of theoretical physics. Euclidean quantum gravity, string theory and real-world membranes are prominent examples. Another example are the center vortices in QCD, which are believed to describe the relevant degrees of freedom for the confinement in QCD. These vortices are dual to a hypercubic random surface with, however, a complicated topological structure.

To put these models on a computer it is necessary to switch to a discretized description of the model. A common feature of the class of problems discussed here is that in the discrete description a fluctuation of the geometry is described by a change in the connectivity of the building blocks of the geometry, typically triangles, tetrahedra or square-plaquettes. Therefore the simulation of these models basically requires integer-operations. Floating point calculations are mainly involved in the calculations for possible additional matter fields. As of today to my knowledge no parallel algorithm for this class of problems is known. On the other hand, due to a typically multi-dimensional coupling space one is led to require many independent runs at different couplings. Therefore farming, parallelization by the number of independent systems, is a probate strategy.

Farming does not at all require any sophisticated communications. Many simulation runs last for a week or more and produce only little output. The only requirement is integer performance. To get a cost-effective PC-cluster we reduce the hardware of the compute nodes as far as possible. Except CPU and RAM we only require a network-card. The BIOS forces us to attach a graphics card as well. A hard disk is not required, the booting procedure involves some code put into a Boot-EPROM plugged into the standard socket of the network card.

While the server is being booted from hard-disk, the BIOS on the compute nodes passes control to the code in the EPROM at boot-time. In the first stage the client broadcasts an bootp-request to gather information about its own IP and so on. It then uses the TFTP protocol to down-load and start the client's kernel binary.

The clients root-file-system is shared among all nodes. The configuration files for all nodes are thus identical. The only difference, the host-name, is derived from the hardware Ethernet address, which is unique. In this way the effort to administer the whole cluster does not go beyond the effort for a single Unix-system. The problem of the few files which should be kept independent for the single nodes, like log- and authentication- files, can be addressed using the transfig patch available on the network. Filenames with a hostname attached are accessed from the named node only.

The LAM MPI package provides a convenient working environment for the cluster. It does not only provide a the functionality to control and synchronize processes but allows simple partitioning, debugging and monitoring as well. In the simplest case, the code developed for a work-station can be used with only one modification, initialization of the MPI library in the main program.

However, in all simulations one has the problem, how to distribute efficiently the work-load among nodes and how to deliver run-time parameters to the nodes. For this purpose we developed the FARMLIB. The main ingredient is a dynamic parser tree, which allows user-code to set up a variable name space, which is used to interpret script-files at run-time. The dynamic parser is complemented with an MPI-Object, which manages the exchange of required information among nodes and does some simple load balancing.

We are now using two clusters with 10 compute-nodes each. The one system is operating now for over two years without severe problems. In average three users are working on the systems. The mean time between forced re-boot is more than 100 days. The second cluster was setup four month ago and has never been rebooted since then. The performance of the applications on the PC-nodes is comparable to the performance on average work-stations.

In summary: PC clusters provide an efficient and very cost-effective way to get computational performance for simulations using the farming strategy. The price for a single node is less than 250 Euro. We want to end by stressing that the discussed concept is not only applicable to the mentioned simulations but can, for example, well be used in multimedia environments to compress (independent) audio-streams, render sequences of pictures. Further modifications like local hard-disks allow for simple data-base and data-mining applications.

# Random Geometries

## Thermodynamics of Surfaces

$$Z = \int d[g_{\mu,\nu}] \exp(-S_E) \quad (1)$$

## Examples

- Random Walk  $S = \int dl$
- Nambu Goto String

$$S_{NG}[S(l_i)] = \mu \int_{S(l_i)} dA = \mu \int_{M(l_i)} d^2\xi \sqrt{\left(\frac{\partial x^\mu}{\partial \xi^1}\right)^2 + \left(\frac{\partial x^\mu}{\partial \xi^2}\right)^2 + \left(\frac{\partial x^\mu}{\partial \xi^2} \frac{\partial x^\mu}{\partial \xi^1}\right)^2} \quad (2)$$

- Euclidean Quantum Gravity, Polyakov-Strings

$$S[g, x] = \alpha \int_{M(l_i)} d^D\xi \sqrt{g} \left[ g^{ab} \frac{\partial x^\mu}{\partial \xi^a} \frac{\partial x^\mu}{\partial \xi^b} + \mu \right] + \frac{1}{G_N} R \quad (3)$$

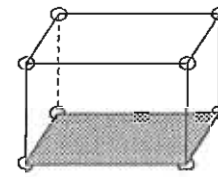
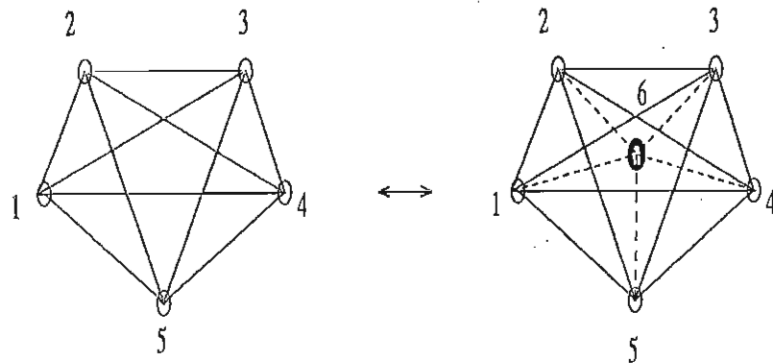
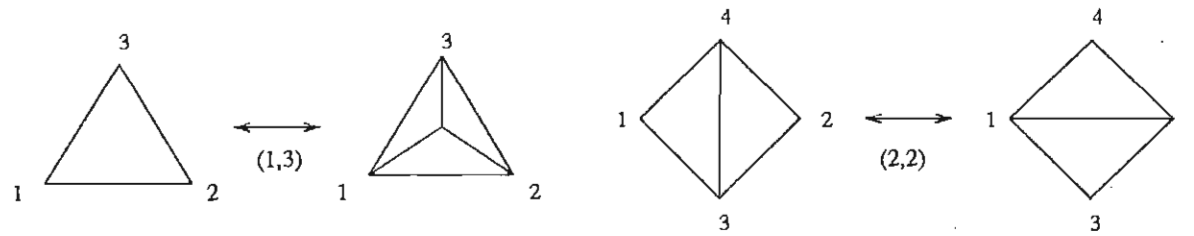
- Fluid Membranes, Oil Films, Polymers
- Confinement in QCD. Center vortices are dual to hypercubic random surfaces



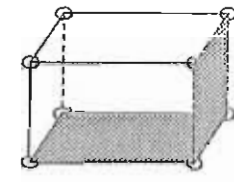
Discretization

$$\int \frac{dg_{\mu,\nu}}{\text{diff}} \rightarrow \sum_{\text{geometries}} \frac{1}{\sigma} \tag{4}$$

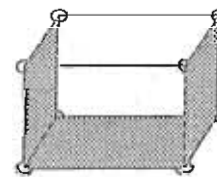
Varying **Connectivity** describes geometry-fluctuations. Algorithms use local updates:



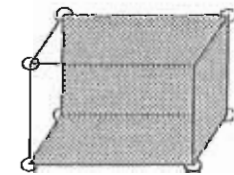
Move 15



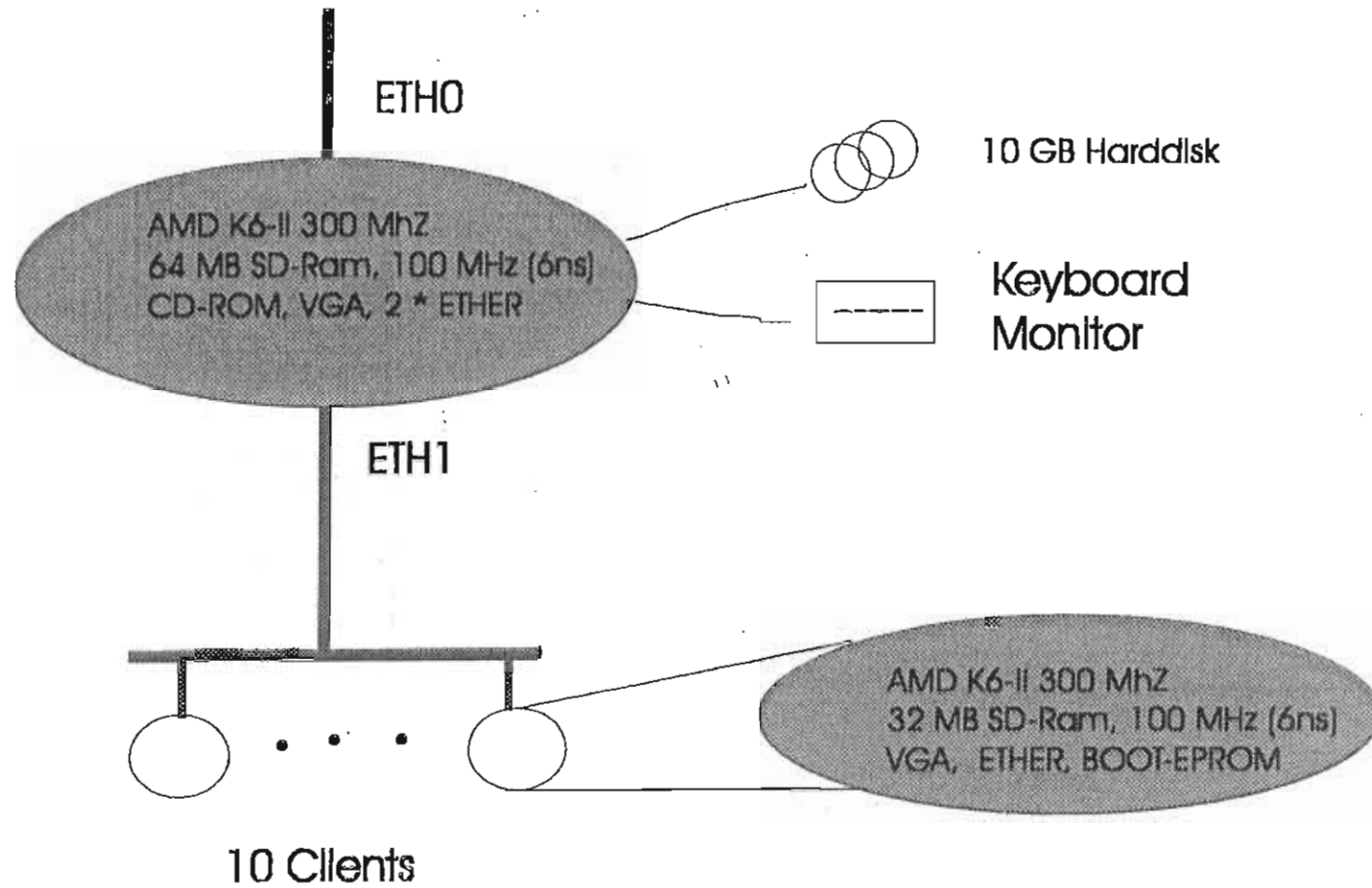
Move 24



Move 33a



Move 33b



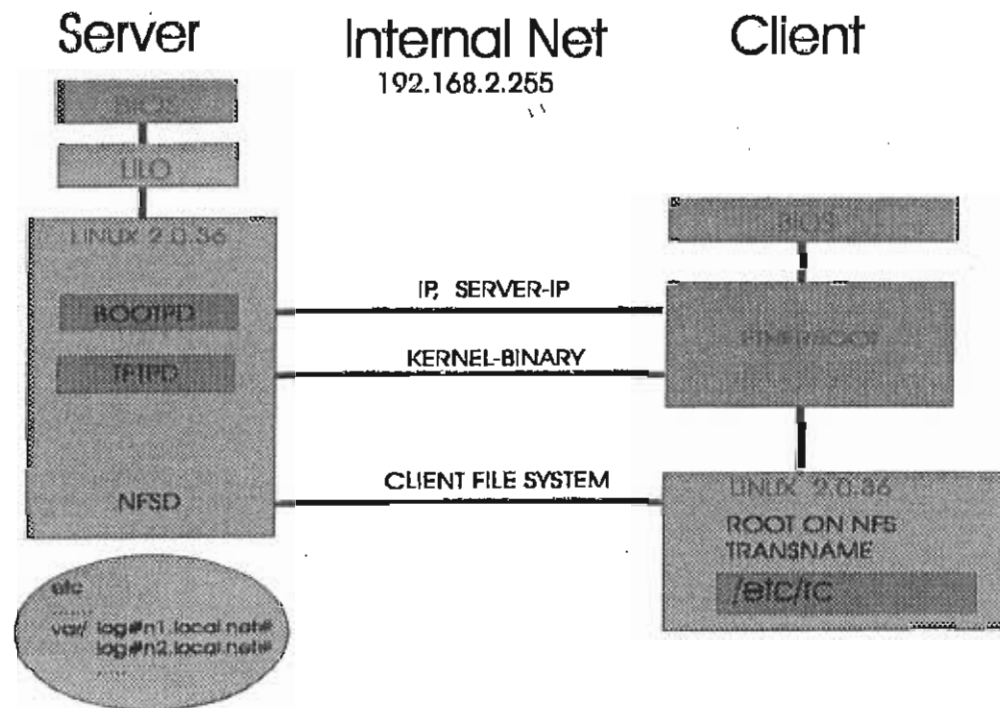
## Server

- RedHat 5.1 typical, 500 MB
- Kernel 2.0.36, Watchdog
- ntimed, nfsd, bootpd, ypserver, tftpd, watchdog
- LAM61, debug

## Client

- RedHat 5.1 minimal 30 MB
- Kernel 2.0.36, Watchdog, transname, root on nfs
- ntimed, ypclient, ntimed, watchdog
- LAM61, debug, libraries

## Cluster Boot Procedure



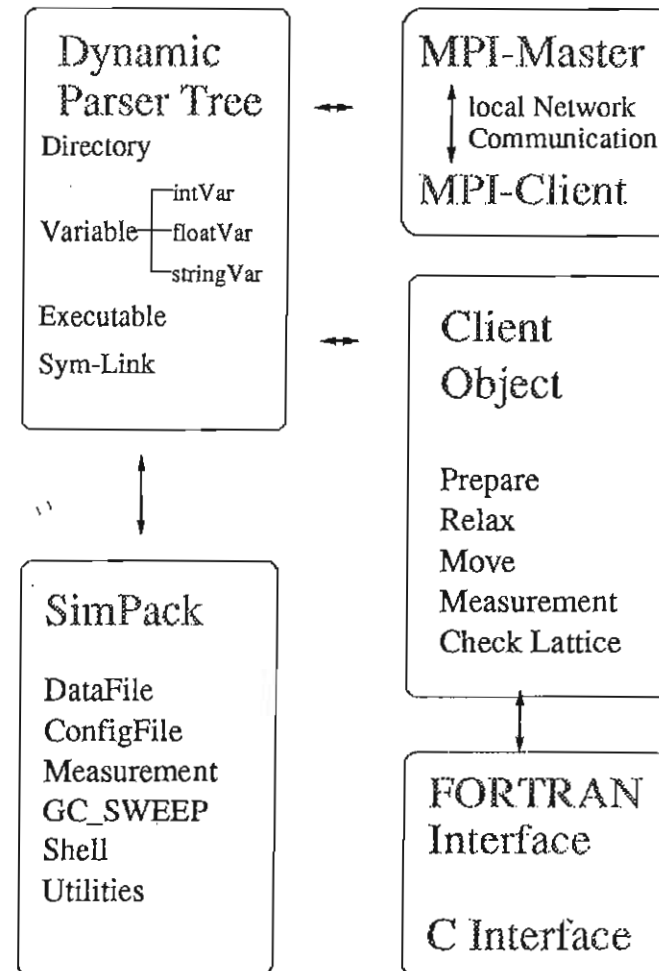
# FARM-LIB

## Sample Script

```

TUNE CYCLE 10000
TUNE PRETUNE 2000
TUNE PREC 0.005
NMEAS 100000
NTERM 5000
TUNE ENABLE TRUE
MEASURE:CANON:DATA:NAME DATA/D020+2700
MEASURE:ENABLE FALSE
CONFIG:NAME AUTO
BETA 4.300
ALPHA 0.0 KAPPA 3.114
ls *
start
ALPHA 0.1 start

```



## Altair Project at CPS-CNR

M. R. Guarracino<sup>1</sup>, G. Laccetti<sup>1,2</sup> and U. Scafuri<sup>1</sup>

<sup>1</sup> Center for Research on Parallel Computing and Supercomputers

<sup>2</sup> University of Naples "Federico II"

**Abstract.** The high computing power of the most recent microprocessors families, together with the 100Mbit/s rate of the fast ethernet network cards, leads to the realization of a new class of distributed parallel systems, the so called Beowulf machines, with performances comparable with the ones of present (and more expensive) supercomputers.

The Beowulf project is a NASA initiative, started in 1994 and sponsored by the HPCC Earth and Space Science Project, to investigate the use of File-of-PCs on computational intensive applications.

These File-of-PCs are effective scalable parallel systems to be used in HPC software development; for their own "nature" they are able to "immediately" use up to date hardware.

In the realization of beowulf systems, only "conventional" hardware and the well known operating system Linux (freely downloadable in source from the Internet) are used. At the moment there are several projects ongoing which aim to the realization of parallel systems based on off the shelf technology. Among others we recall the firsts and best known systems *Neagling* at CalTech, *Loky* at Los Alamos and *Hyglac* at JPL.

Altair is a 16 Pentium Pro system, connected with a fast ethernet switch; operating system is Linux.

First experiences refer to computational applications regarding *Computational Finance* (specifically involving a *collateralized mortgage obligation* problem) and an *Image processing* application, i.e an *image denoising* problem. Results show the system is well suited for such different applications, providing efficient solution for both computational kernels. Moreover some well known benchmarks have been run on the system and compared with the ones obtained on a commercial supercomputer.

## Altair Project @ CPS CNR

*M. R. Guarracino, G. Laccetti, U. Scafuri*  
CPS - CNR, Napoli

**Assergi (Aq) - PC-NETS**  
24-25 Marzo 1999

<http://pixel.dma.unina.it/beowulf.html>

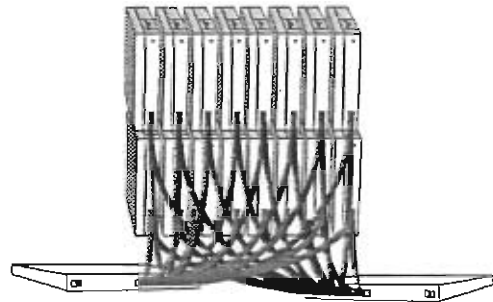
2

## Il sistema Altair

Hardware Sistema Operativo Software

16 nodi con:

- PPro 200 Mhz
- 128 MB EDO RAM
- 256KB cache
- 2.5 GB HD
- 3c905



La connessione avviene tramite uno switch Fast ethernet  
Baynetworks 350T 16 porte autosensing

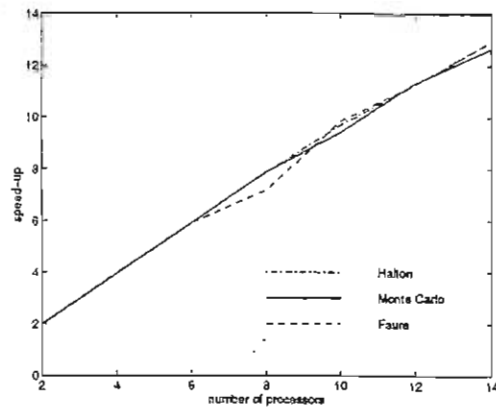
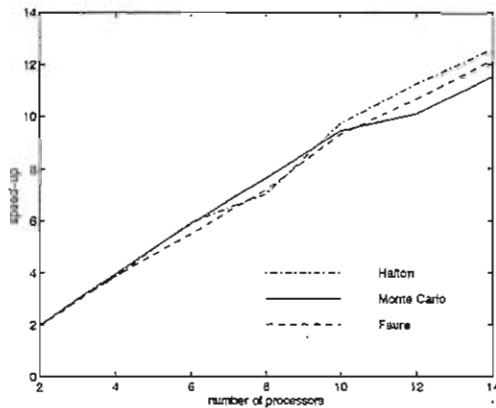
E' presente una rete di servizio

<http://pixel.dma.unina.it/beowulf.html>

# Applicazioni

Computational Finance Image Processing

## Collateralized Mortgage Obligation



speed-up dei metodi di Halton, Faure e Monte Carlo per l'Integrazione in 10 ed 80 dimensioni di

$$f^{(1)}(t) = \exp\left(-\sum_{i=1}^s \alpha_i |t_i - \beta_i|\right)$$

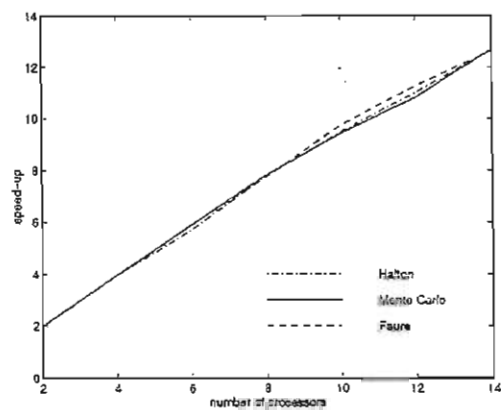
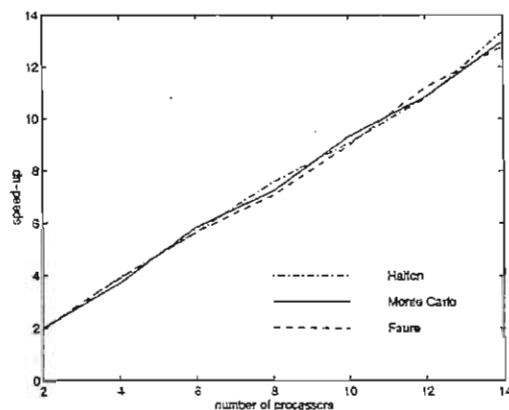
<http://pixel.dma.unina.it/beowulf.html>

4

# Applicazioni

Computational Finance Image Processing

## Collateralized Mortgage Obligation



speed-up dei metodi di Halton, Faure e Monte Carlo per l'integrazione in 10 ed 80 dimensioni di

$$f^{(2)}(t) = \begin{cases} 0 & \text{se } t_1 > \beta_1 \text{ o } t_2 > \beta_2 \\ \exp\left(\sum_{i=1}^s \alpha_i t_i\right) & \text{altrimenti} \end{cases}$$

<http://pixel.dma.unina.it/beowulf.html>

Center for Research on Parallel Computing and Supercomputers

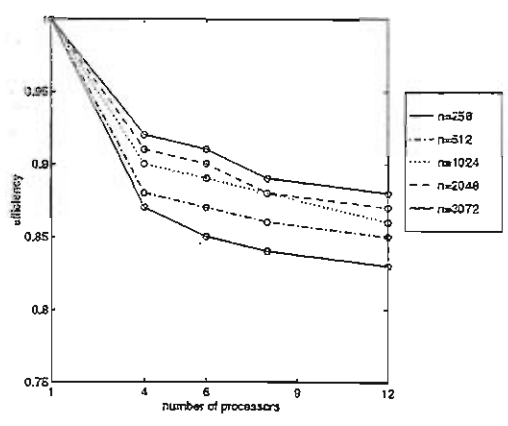
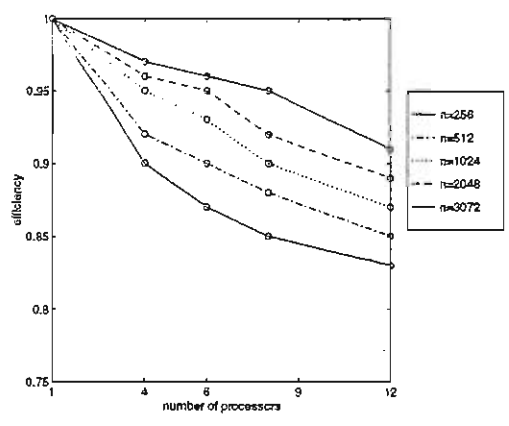
Center for Research on Parallel Computing and Supercomputers



# Applicazioni

Computational Finance Image Processing

## Image Dencising



Filtro locale non lineare

pixel	256	512	1024	2048
Altair	2.10	7.55	28.7	130.20
IBM SP	1.95	6.30	25.01	113.69

<http://pixel.dma.unina.it/beowulf.html>



# Prestazioni

MP Benchmark NAS Parallel Benchmark

	EP	IS	CG	FT	MG	SP	BT	LU
IBM SP	n.a.	13.54	201.78	269.71	635.93	n.a.	n.a.	676.51
Altair	n.a.	2.84	-	-	71.12	n.a.	n.a.	151.10

Classe A - 8 processori

Il rapporto tra i costi delle macchine è 1 a 11.25

Nel benchmark MG l'IBM SP è 9 volte più veloce ma si utilizzano i 2/3 dei nodi dell'IBM SP e 1/2 di Altair; un megaflops sostenuto dell'IBM SP costa il 160% in più

<http://pixel.dma.unina.it/beowulf.html>

Center for Research on Parallel Computing and Supercomputers





## Prestazioni

MP Benchmark NAS Parallel Benchmark

Center for Research on Parallel Computing and Supercomputers

	EP	IS	CG	FT	MG	SP	BT	LU
Altair	39.57	2.62	-	-	126.88	141.67	-	289.88

Classe A - 16 processori

Il costo di ciascun megaflops dell'IBM SP è il 310% di quello di Altair

<http://pixel.dma.unina.it/beowulf.html>



## Conclusioni

Center for Research on Parallel Computing and Supercomputers

L'installazione è semplice e veloce

Tali sistemi sono economici e facili da gestire

Il rapporto costo/prestazioni è "imbattibile"

Tutto è disponibile nel *computer shop* sotto casa

Nei **TOP500** due Beowulf sono nella parte alta della classifica:

- 98° Cplant (SANDIA)
- 114° Avclon (LANL)

- - Introduzione
- - Stato dell'arte
- - Il sistema Altair
- - Applicazioni
- - Prestazioni
- - Sviluppi futuri
- - Conclusioni

<http://www.top500.org/top500.list.html>

<http://pixel.dma.unina.it/beowulf.html>

# Beowulf @ Cagliari: Kalix2 after Kalix1

Alessandro Chessa

UNIVERSITY OF CAGLIARI  
Physics Department



## KALIX

Linux Parallel Cluster Project  
(1996)

---

The Kalix Project is devoted to investigations on the viability of building high-performance computers through the assembling of inexpensive and widely available computing components. At the moment the project is centered around two initiatives, ~~updated~~ Kalix1 and ~~updated~~ Kalix2.

Kalix1 is the name of an already built 8-nodes parallel machine, which works under Linux OS and is used as a testbed both for parallel software and communication hardware. Kalix2 is the name of a new, bigger machine, which builds upon the experience of Kalix1.

### Collaborators:

Prof. Gianni Mula (Physics Department, Cagliari)

Prof. Enzo Marinari (Physics Department and INFN, Cagliari)

Dott. Francesco Zuliani (Physics Department and INFN, Cagliari)

---

E-mail: [alessandro.chessa@dsf.unica.it](mailto:alessandro.chessa@dsf.unica.it)



## KALIX1 MACHINE

---

### Hardware Specifications:

The cluster is made of 8 PCs plus one control machine. Each of them has the following characteristics :

- Triton Motherboard with 100 MHz Pentium
- 96Mb RAM per node
- 1.2 Gb EIDE disks in each node
- 10 Mbit/s ethernet adapter in each node (for NFS and various networking support)
- 100 Mbit/s Fast Ethernet (DEC EtherWORKS 10/100) + 3Com Super Stack II hub 100Mbit

### Software:

- The machines are running Linux as it comes out of Slackware96 distribution.
- PVM and MPI for message passing
- Compilers installed: Gnu C, C++ and FORTRAN (Absoft and g77)
- Libraries: Scalapack, LEDA and Octave libs

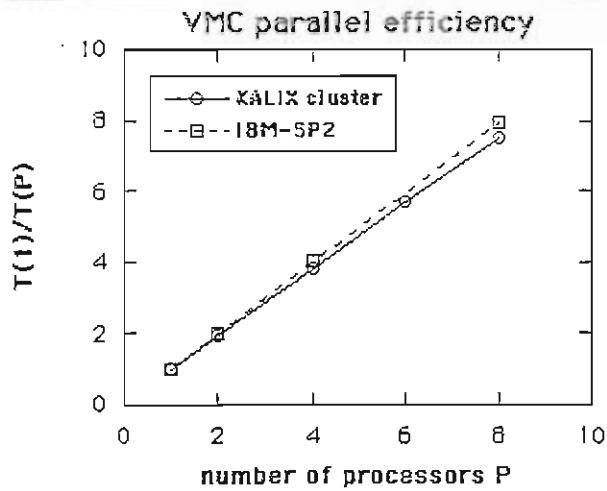


Figure 1: inverse of the execution time on P processors,  $T(P)$ , normalized to the time on one processor,  $T(1)$ , as a function of the number of processors P.

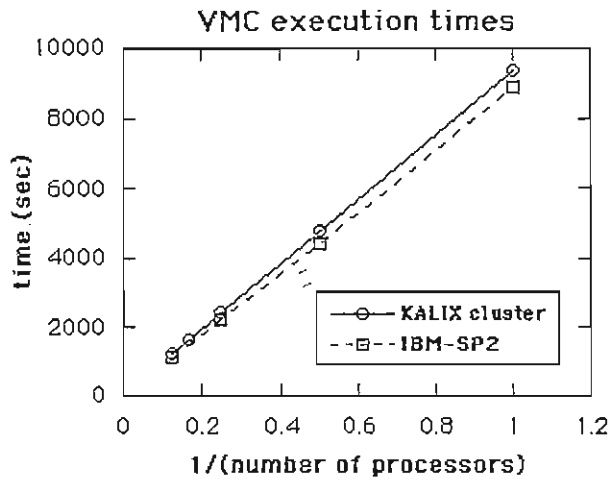


Figure 2: execution times as a function of the inverse number of processors, for our KALIX cluster and for an IBM-SP2.





## KALIX2 MACHINE

---

### Hardware Specifications:

The cluster is made of 16 nodes plus one control machine. Each of them has the following characteristics :

- bi-processor Pentium II 450 Mhz 512 chache
- 256Mb RAM per node
- 10 Gb EIDE disks in each node

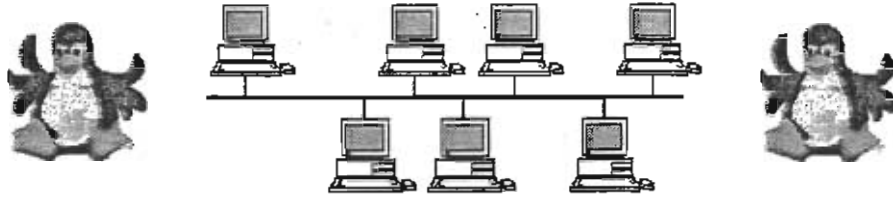
### Networking:

- 3 x DE500 DEC in each node
- switch 3Com 3300 24 port

### Topology:

- One ring with point-to-point connections
- Star connections with the switch

# Parallel Quantum Monte Carlo



on the KALIX cluster

(by **Andrea Bosin** and **Manuela Menchi**)

---

[Up](#) [Index](#)

---

Variational and Diffusion Quantum Monte Carlo algorithms have been implemented on a parallel distributed architecture using the Parallel Virtual Machine package. A system-independent parallel implementation, exploiting the intrinsic parallelism of the two methods, makes Quantum Monte Carlo simulations particularly well suited to distributed computing. This has been first shown for a simple test system; i.e. a valence-only ion with two electrons (see here). The same strategy has then been adopted to perform Variational Monte Carlo simulations on solids (see here).

Here we want to compare execution times and parallel efficiency obtained for a simple Variational Monte Carlo (VMC) valence-only two-electron atom simulation on two different distributed parallel architectures, our KALIX Pentium cluster and an IBM-SP2. On both systems the public domain Parallel Virtual Machine package has been used on a standard 10 Mbit/s Ethernet.

In Fig. 1 we show the inverse of the execution time on  $P$  processors,  $T(P)$ , normalized to the time on one processor,  $T(1)$ , as a function of the number of processors  $P$ , which gives a measure of the parallel efficiency of the VMC program.

In Fig. 2 we compare the execution times on  $P$  processors for the two architectures considered.

We are planning to port the VMC code for solids on the KALIX cluster in the next future to have a more reliable measure of its performance.

# MPI Performance of a PC cluster at the ICTP

Roberto INNOCENTE  
Abdus Salam ICTP/ Sissa  
inno@sissa.it

March 22, 1999

The aim of this presentation is to give some figures about performance of MPI in a real installation and on almost up-to-date hardware. MPI is now an accepted industry standard for message passing. There is a growing number of programs in physics that have been coded with this library on CRAYs T3D/E, IBMs SP1/2, SGI's Origin, etc and that now can be conveniently used on PC clusters (*Refer to Vittoli's presentation*). We will see anyway in the next paragraphs that there are many subtle points in configuring and installing such a system and that many binary distributions available are tuned for very different environments.

## 1 Hardware configuration

We installed a cluster of 20 PCs equipped with a Pentium II at 450 Mhz, 384 MBytes of RAM, a Fast Ethernet 3Com 3c905b card and a 4 GB h/d, all interconnected through a 3Com Super Stack II 3300 Fast Ethernet switch (See *Fig. 1*).

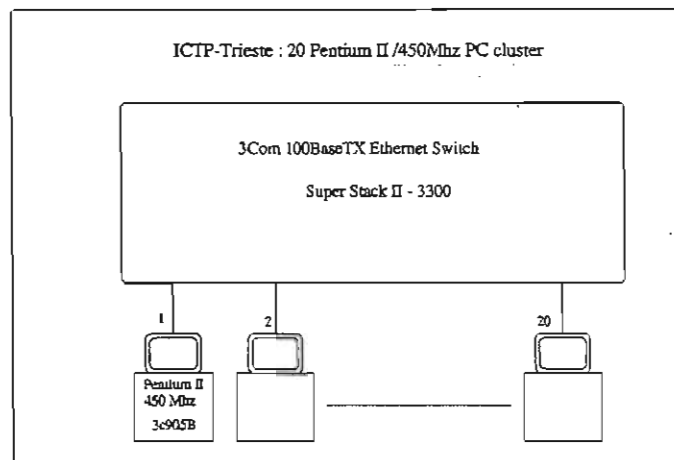


Figure 1: Hardware configuration

## 2 Software configuration

Among the different MPI implementations we chose Argonne's MPICH because it is widely used, it has a clear interface between a device independent layer and a device dependent one (ADI = Abstract Device Interface) and there are good performance reports for it (over Myrinet a bandwidth of over 110 MB/s and a latency of 7 usec are reported).

## 3 Memory Bandwidth

We measured the main memory bandwidth with the *stream* benchmark. As you can see these off-the-shelf PCs have a very respectable memory bandwidth: about 300 MB/s (See Fig. 2). This is sufficient to guarantee that we will not

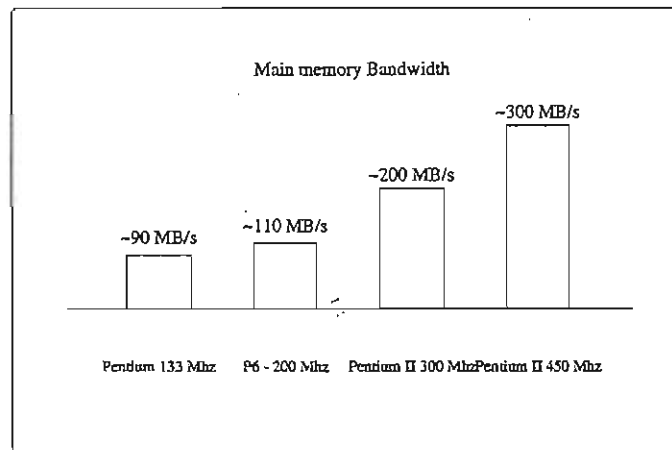


Figure 2: Main memory bandwidth

have problems with the transfer rate involved with one or multiple Fast Ethernet devices.

## 4 Network performance

We measured the UDP/TCP bandwidth between 2 nodes with standard tools like *ttcp* and *netperf*. It comes out that the UDP bandwidth is about 11.7 MBytes/s and the TCP bandwidth is about 10.6 MBytes/s. The UDP bandwidth is about 10 percent higher. This is due to the fact that the TCP header is longer and to the higher processing overhead required by TCP (See Fig. 3).

### 4.1 Using UDP

Using UDP is appealing because of the less overhead and greater efficiency involved. Anyway there are many features of TCP we need to re-implement over UDP if we want to use it as an MPICH abstract device. We need error



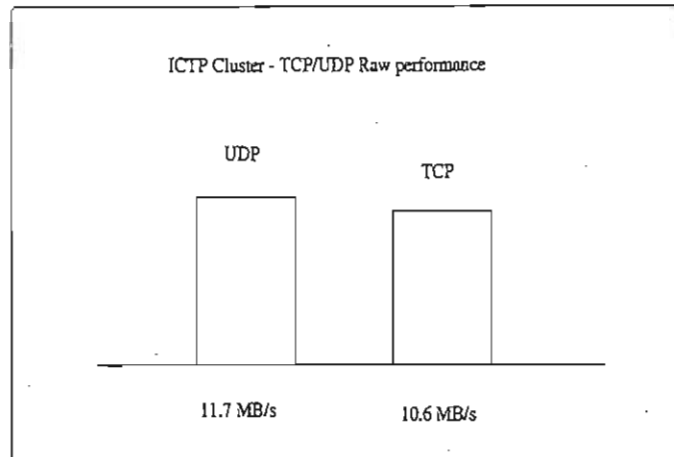


Figure 3: UDP/TCP socket level bandwidth

correction, de-multiplexing and authentication. A preliminary work has been done in a master's thesis by D.Brighwell some years ago, however this has not generated a complete implementation.

## 4.2 Using TCP

An implementation of the MPICH's ADI directly over TCP is planned for the near future. In the meantime the so called *ch\_p4* device is used. This is an implementation of the ADI through Chameleon/P4. What is unfortunate with TCP is that it comes with some congestion control/avoidance mechanism that while essential on overcrowded WANs, are a mess with high speed networking on LANs/SANs (See Fig. 4). We have found that we can easily have a quite stable

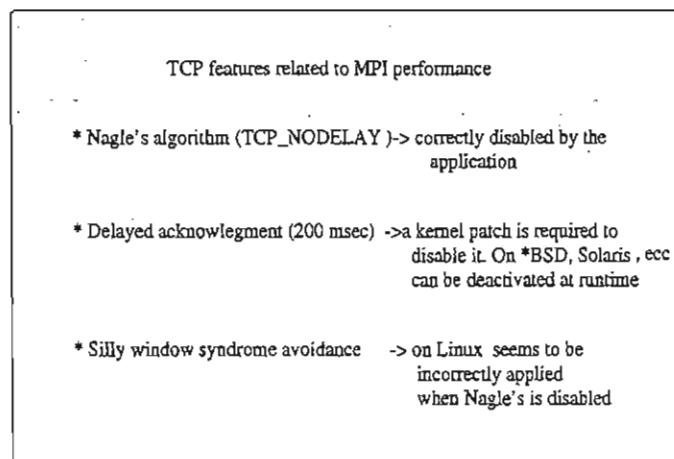


Figure 4: TCP features related to MPI performance

and predictable performance on FreeBSD disabling the delayed ack algorithm (it can be done with *syscontrol*). Linux has a variable delayed ack timeout, this is the reason why it can be difficult to recognize its effects. We can see ordinary 200 msec timeouts as on Berkeley derivatives or *quickacks* with 20 msec timeouts. This 20 msec timeout is for example applied when receiving tiny-grams (segments less in size than half a mss and with the push bit on). The delayed ack algorithm is responsible of 20 msec delays when many small messages are sent only in one direction between 2 nodes. In this case the receiver is delaying his ack in the hope to piggyback the ack to a packet in the other direction (See Fig. 5). On Linux to disable this algorithm it is necessary to

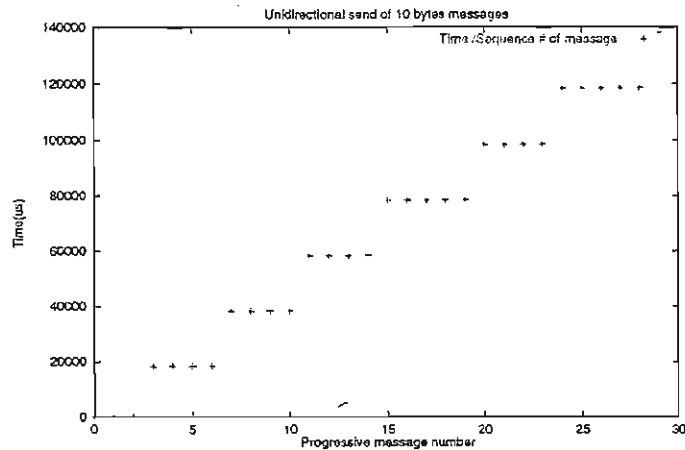


Figure 5: Effect of delayed acks on unidirectional sending

patch and recompile the kernel. We need to recall that the *delayed ack* algorithm is required by the RFCs and so if you disable it your TCP/IP stack shouldn't be used on the global Internet.

## 5 MPICH

MPICH uses different protocols to try to optimize different communication parameters.

### 5.1 MPICH Internal protocols

There is little relation between these internal protocols and the user level blocking/non-blocking MPI taxonomy. These internal protocols try to solve the buffering problem without penalizing too much the latency of small messages. The protocols are called: *eager*, *rendezvous* and *get* (See Fig. 6).

#### 5.1.1 - Eager

In the *eager* protocol as soon as a message is posted, the envelope and the data are sent to the receiver. This requires buffering of the unexpected message

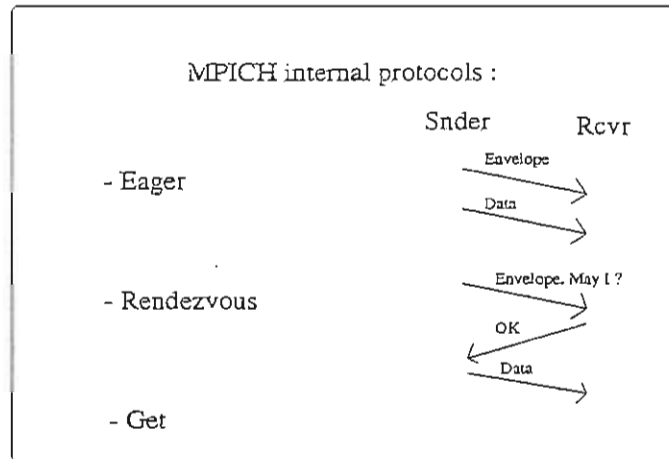


Figure 6: MPICH Internal Protocols

on the receiver side if the receive operation is not yet started and can require an additional copy of the data. This protocol tries to decrease latency and is usually used for small messages (a small variation of *eager* called *short* is used when the envelope and the data all fit in one packet).

### 5.1.2 - Rendezvous

In the *rendezvous* protocol when a message is posted the envelope is sent to the receiver and eventually buffered there. When the receive is posted and the appropriate envelope has already been received, the receiver sends an acknowledge to the sender. Only after having received the acknowledge the sender sends the data. This protocols requires the receiver to eventually buffer only the envelopes. As it synchronize the receiver, it can avoid an additional copy of the data. It is usually used for large messages.

### 5.1.3 - Get

The *get* protocol is used by shared memory implementations or when there is special hardware support for remote memory operations. In this case the receiver gets the message usually via a *memcpy* operation. We will not mention it anymore.

## 5.2 MPICH Performance

Point-to-point bandwidth and bisection bandwidth performance are measured using a *pingpong* test and then dividing by 2 the round trip time obtained.

### 5.2.1 Point-to-point performance

We have found that the MPICH performance for small messages between 2 nodes (using the *eager* protocol) can be approximated with a linear model having a latency of 104 usec and a bandwidth of 5.58 MB/s (See Fig. 7). With large

messages (using the *rendezvous* protocol) the performance can be approximated with a linear model having a latency of 5.23 msec and a transfer rate of 10.6 MB/s (See Fig. 8). The crossover between the 2 linear models is at about

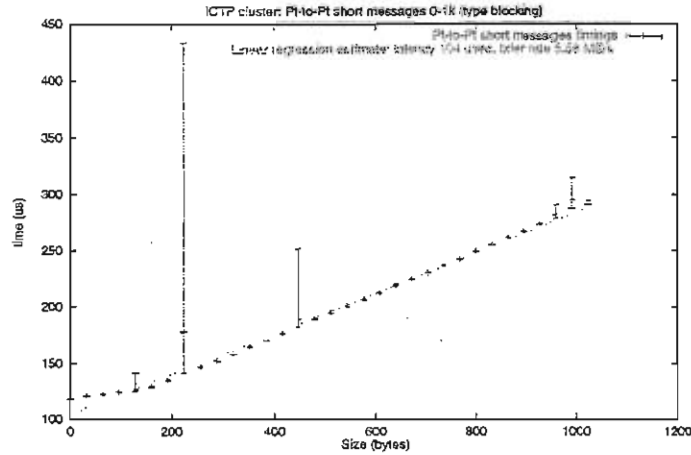


Figure 7: Short messages Pt-to-Pt timings

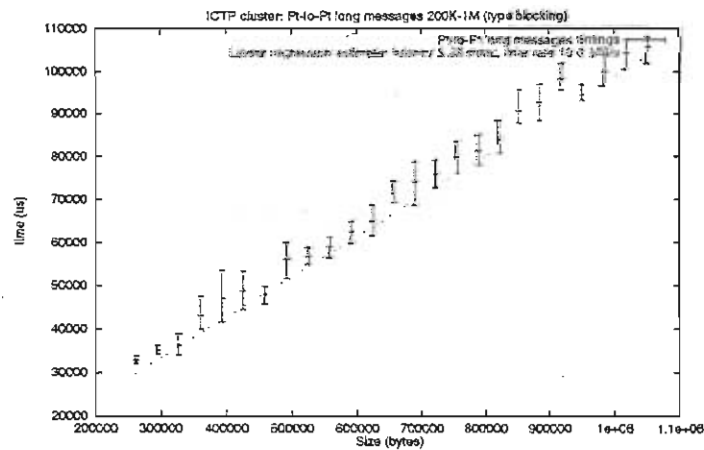


Figure 8: Long messages Pt-to-Pt timings

64 KB where we need to switch between the 2 protocols. This can be specified using an option during the compilation of MPICH.

### 5.2.2 Bisection bandwidth

Bisection bandwidth tests are done by creating a topology of  $N/2$  pairs communicating simultaneously. Then the average of times over the  $N/2$  pairs is taken. These tests stress the communication network. For short messages (using the *eager* protocol) we obtain for 8 nodes :

latency 107 usec, bandwidth 5.64 MB/s  
and for 16 nodes :

latency 108 usec, bandwidth 5.70 MB/s  
that are essentially the figures we obtained for the Pt-to-Pt case (See Fig. 9/ 10).

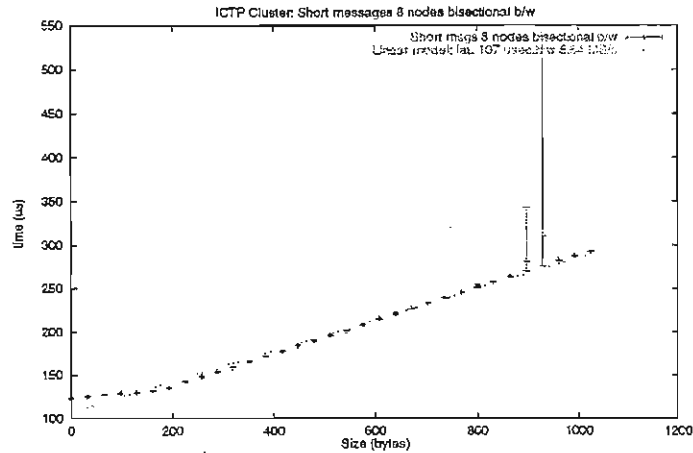


Figure 9: Short messages 8 processors bisection b/w

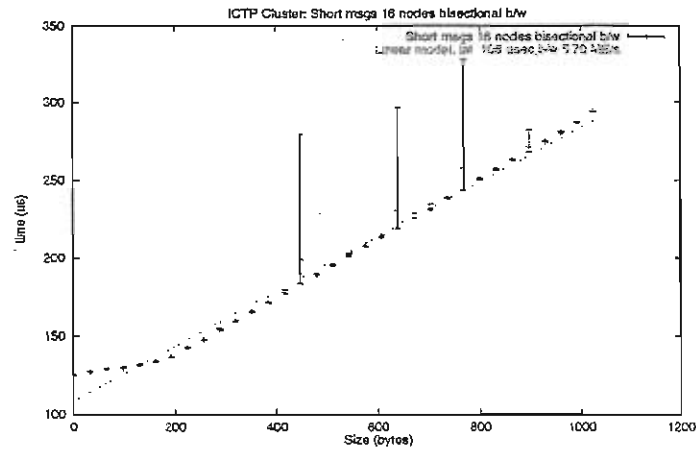


Figure 10: Short messages 16 processors bisection b/w

For long messages (using the rendezvous protocol) we obtain for 8 nodes :

latency 5.6 msec, bandwidth 10.52 MB/s  
and for 16 nodes :

latency 3.84 msec, bandwidth 9.94 MB/s  
again essentially the figures of the Pt-to-Pt case (See Fig. 11/ 12). We can  
conclude that the switch is non-blocking up to at least 16 nodes.

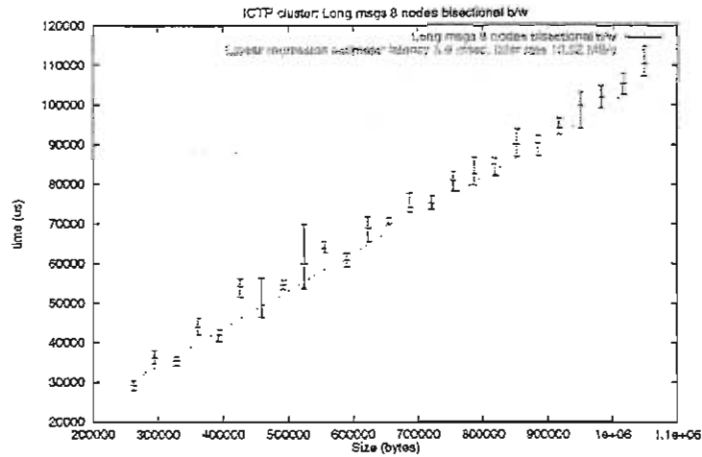


Figure 11: Long messages 8 processors bisection b/w

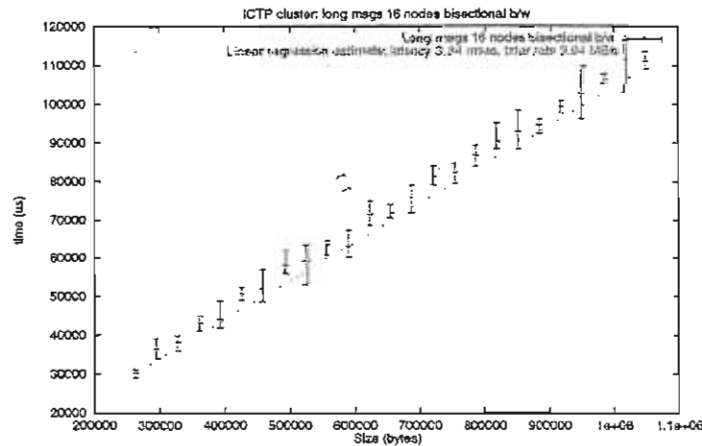


Figure 12: Long messages 16 processors bisection b/w

### 5.3 Broadcast/Reduce algorithms

MPICH can use a broadcast tree algorithm to implement collective communications.

The number of leaves used by each node can be controlled during compilation. So the algorithm can perform like a linear algorithm (one node sends sequentially to all other nodes/one node receives sequentially from all other nodes) if the number of leaves is set to a number greater than the number of processors (this is the right solution if the processors are interconnected through a hub to avoid collisions), or like a tree of height  $\log_2 N$  where the root sends to the process  $N/2$  away, and the root and the receiver become each root of a subtree of size  $N/2$  and send to the processor  $N/4$  away and so on. (See Fig. 13). The latter is the right solution if the nodes are interconnected through a

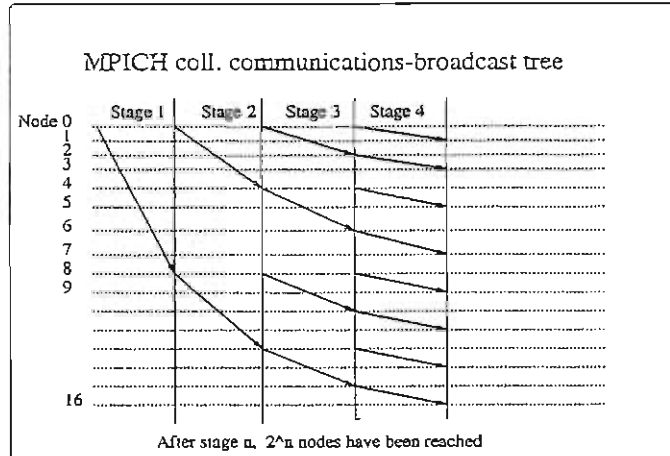


Figure 13: Collective communications broadcast tree

full Fast Ethernet switch like in our case. At stage 4 this algorithm requires an aggregate bandwidth of 8 Pt-to-Pt channels. In this case we can expect times that are  $\log_2 N$  more than Pt-to-Pt communications times. Unfortunately as the communications in this case are essentially unidirectional, if the delayed ack algorithm is not disabled, the performance of repeated broadcasts can be very poor.

## 6 Conclusions

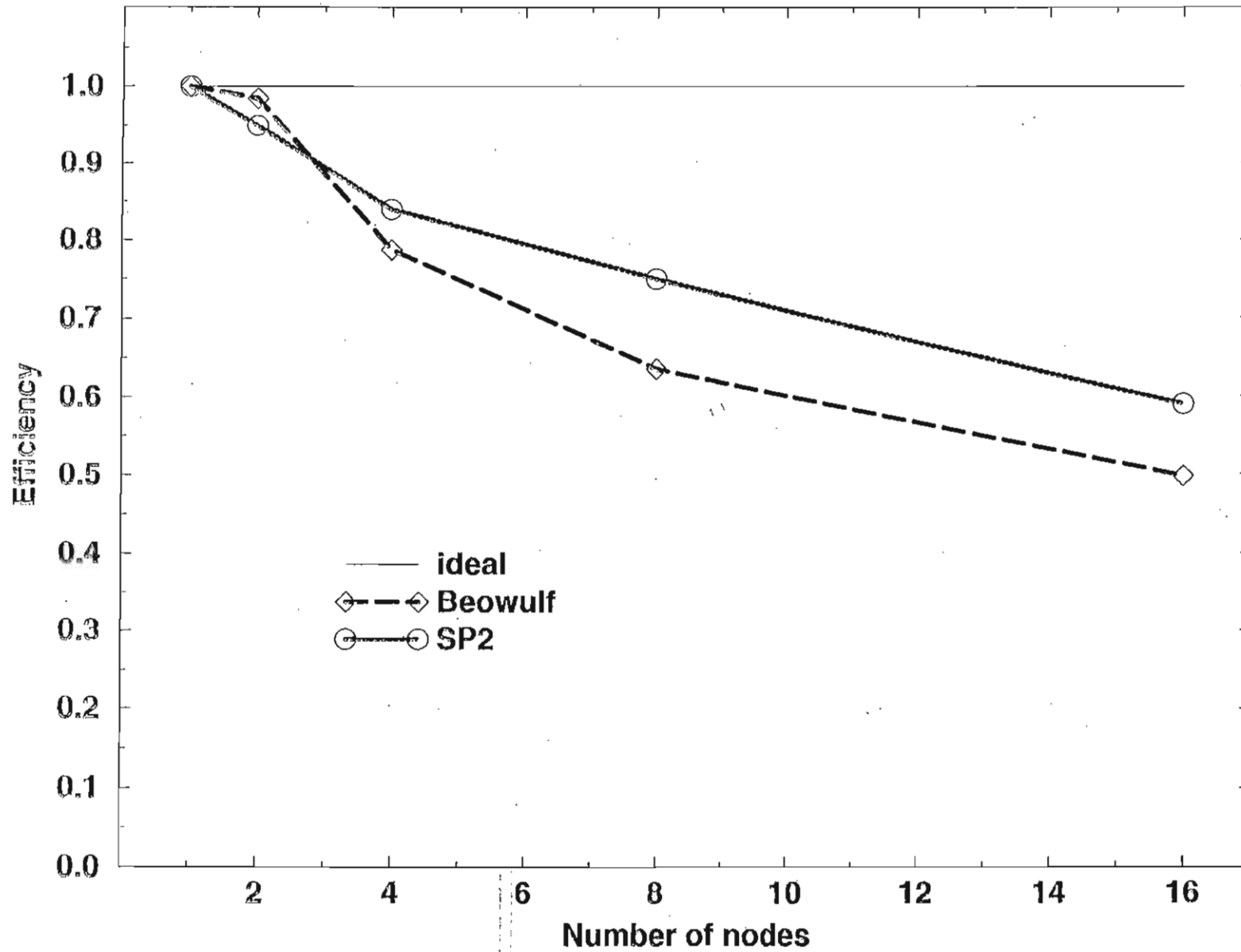
We have seen how the performance of MPICH/ch\_p4 depends on configurable parameters. We have also seen that the performance depends on features of TCP that are required by the RFCs. We will use Linux because of software availability issues. We have not generally disabled delayed acks, but we have prepared a patched version of the kernel with them disabled. We feel that for the time being, delayed acks can be disabled on a computing cluster if access from/to the internet at large is through an RFC compliant application gateway. An implementation of the MPICH's ADI over UDP would solve the problem of the TCP stalls, but would not significantly decrease the latency that is mainly due to the kernel intervention.

## Contents

1	Hardware configuration	1
2	Software configuration	2
3	Memory Bandwidth	2
4	Network performance	2
4.1	Using UDP	2

# Fixed-size efficiency

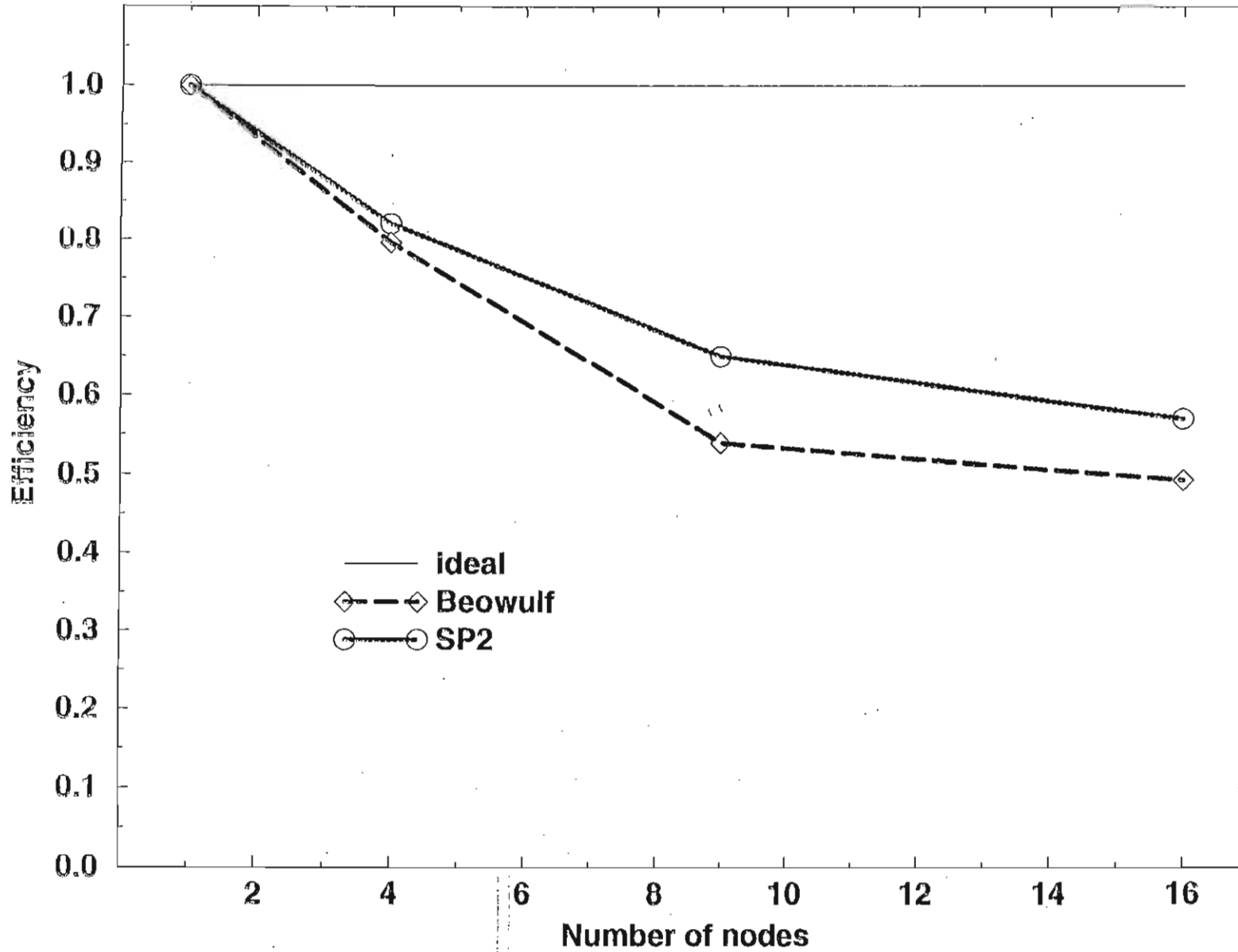
Transport code – Beowulf vs SP2





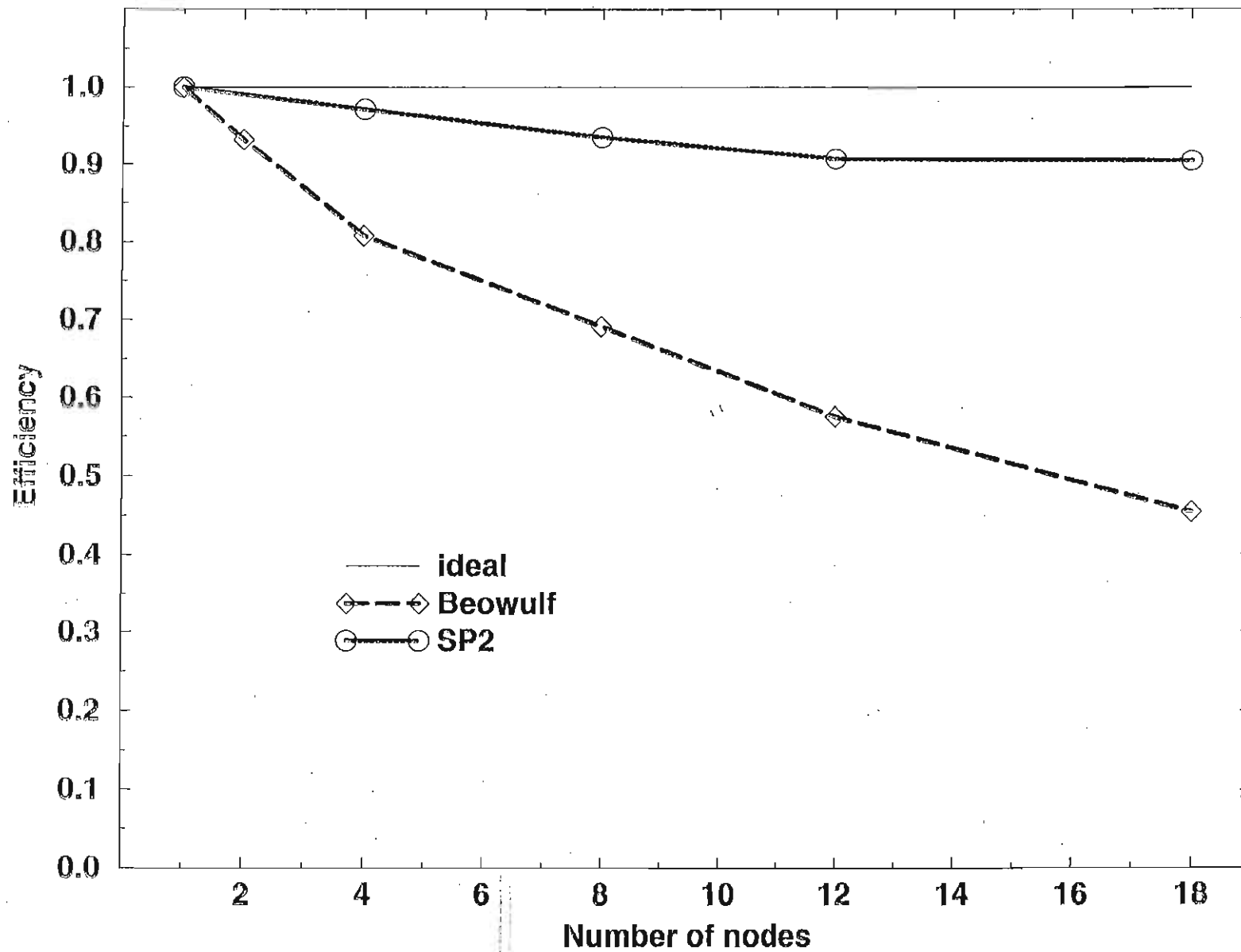
# Growing-size efficiency

Transport code - Beowulf vs SP2



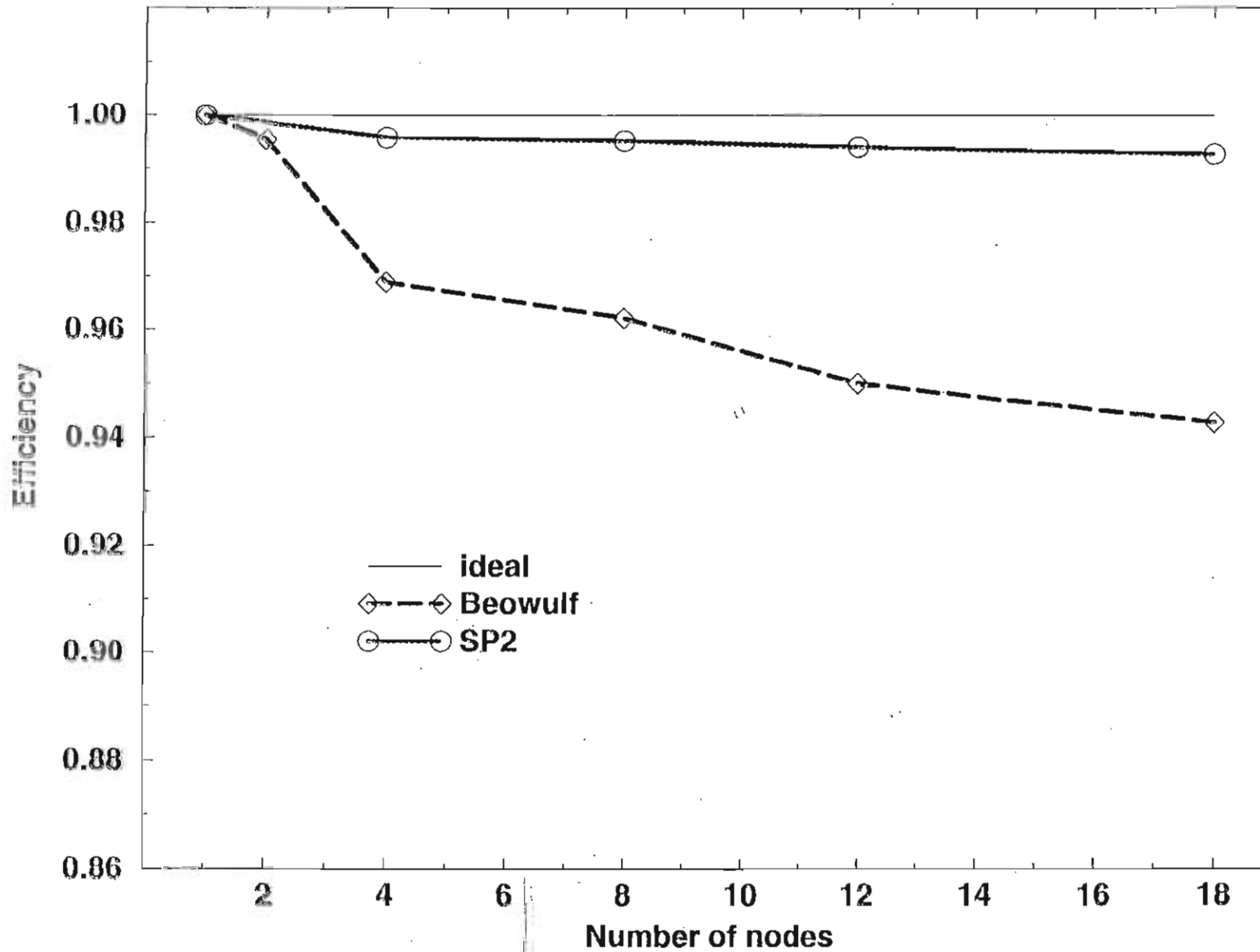
# Fixed-size efficiency

FD-TD code - Beowulf vs SP2



# Scaled-size efficiency

FD-TD code - Beowulf vs SP2



# FD-TD

Finite Difference-Time Domain

PC is 1.7 times faster than the SP2 node

Problem size for 1 node:  $139 \times 139 \times 139$   
2.7 million points — 123 MB for 1 node  
48 million points — 2.2 GB for 18 nodes  
291 Mflops (SP2) vs. 468 Mflops (PC's cluster) on 18 nodes  
4.9 s/time step (SP2) vs. 2.9 s/time (PC's cluster) on 1 node

-----

Problem size:

$201 \times 201 \times 201$  —  $523 \times 523 \times 523$   
8.1 million points — 372 MB for 1 node  
143 million points — 6.5 GB for 18 nodes

Dielectric cube size	1/40	1/3	9/10
efficiency	.95	.57	.71
time/time step (1 node)	8.9 s	9.6 s	26 s

# DISCO — A Status Report

Giovanni Chiola  
DISI, University of Genoa  
via Dodecaneso 35, 16146 Genoa, Italy  
<http://www.disi.unige.it/person/ChiolaG/>  
email: [chiola@disi.unige.it](mailto:chiola@disi.unige.it)

PC-NETS, LNGS, Assergi (AQ), March 25, 1999

This report is published as DISI-TR-98-09, December 1998, and it has been developed in the framework of the European Union by means of the Tender to III/97/31 Lot 5 "Cluster Computing for Data Intensive Application," DISCO project. We survey the state of the art of Cluster Computing based mainly on low-cost PC or workstations technology. Real industrial applications as well as EU funded and international University/Research projects are taken into account in order to provide an overall (although necessarily not exhaustive) view of the current situation and trends.

Cluster computing is mainly obtained by providing a set of PCs or workstations connected by a more or less sophisticated local area network (LAN) with appropriate communications libraries and/or operating system (OS) primitives, and by adopting appropriate application-level languages, libraries, and/or environments to support parallel processing. After a first chapter reviewing already existing industrial applications (that should be regarded as main practical motivations to pursue this approach), we present in subsequent chapters the enabling technologies that have been adopted at the various application, LAN, OS and system library levels. Another chapter reviews past and currently active EU supported R&D efforts in the field, with the intent of reaching a better understanding whether the EU policy in this field is adequate and/or conformant to the one followed by other developed countries. Finally, we conclude with a chapter discussing costs, performance, and trade-offs among the two, in the context of Cluster Computing.

The main motivations for the use of Clusters of PCs both in industrial and research environments can be summarized as follows:

- PC technology is becoming increasingly powerful, less and less expensive;
- Local Area Network technology is becoming faster and faster, less and less expensive;
- Networks of PCs are already found in most environments;
- The same components can be used either as individual, networked workstations, or as cluster components.

High-level programming environments such as standard parallel programming environments and libraries PVM and MPI are supported, and more sophisticated object oriented distributed languages and libraries (e.g., C4, OOMPI, HPC++, High Performance Java, etc.) could also be supported efficiently.

Several NOW/Cluster messaging systems are committed to efficiency and share some performance-oriented features:

- Simplified communication protocols;
- Minimal number of intermediate copies of messages;

- Avoid system call overhead for communication or exploit "light-weight system calls" (which save only a subset of CPU registers and do not invoke the scheduler upon return);
- Poll the network for incoming messages to save the overhead of interrupt launch and service (this is possible in case of synchronous, explicit message receive, and implies an appropriate computational model).

From the software organization point of view, high performance message passing systems running on Clusters can be grouped into three families:

standard interface approach;

user-level approach;

efficient OS support approach.

In particular, the VI (virtual interface) Architecture might become an adequate standard for high performance message passing provided that low-cost NICs conforming to it are introduced for the PC market and that applications exploiting it are developed.

As the basic technology for efficient communication is consolidating, this is probably the right time to start using clusters in real applications. Commercial Cluster applications include computationally intensive image processing, data mining, Web search engines, etc. However normal commercial applications on Clusters are still limited by some "drawbacks":

- Efficient OS support: Linux and Windows NT are the main options for PC technology;
- In scientific environments Linux is a widely assessed substitute for expensive traditional workstations;
- On the contrary, there are yet high barriers for accepting Linux in some corporate worlds:
  - fear of lack of support and of guarantees by a legally established software house;
  - lack of commercial applications;
  - consolidated perception that "Linux is only for hackers"

The situation is however rapidly changing, with support from major computer industries (IBM, Oracle, Intel, etc.) for Linux based applications ...

In conclusions we may say that Clusters of PCs are becoming a viable, low-cost, alternative to traditional high-performance computing platforms as well as to more expensive networks of workstations. The success of clusters is due in part to the better performance/cost ratio that characterizes personal computer hardware and in part to the outcome of various research projects aimed at producing high-speed communication systems out of off-the-shelf local or system area network technology. PC clusters are starting to be used for commercial applications as well as in supported projects in research environments. But there is still a lot to do both in terms of applied research efforts and in terms of development of suitable application environments.

People interested in getting involved in the subject, may refer to the activity of the newly formed IEEE Computer Society Task Force on Cluster Computing, URL:

<http://www.dgs.monash.edu.au/~raj कुमार/tfcc/>

## Quick technology summary

- The basic technology for local computation on nodes is available and well consolidated;
- The basic technology for efficient message passing in Cluster environments is available at research level, and is probably going to consolidate in industrial products in the near future;
- This is probably the right time to start using clusters in real applications.

# Cluster Drawbacks in Commercial Applications

- Efficient OS support: Linux and Windows NT are the main options for PC technology;
- In scientific environments Linux is a widely assessed substitute for expensive traditional workstations;
- On the contrary, there are yet high barriers for accepting Linux in some corporate worlds:
  - fear of lack of support and of guarantees by a legally established software house;
  - lack of commercial applications;
  - consolidated perception that “Linux is only for hackers”



# **Clusters in Commercial Applications: a changing trend**

- The situation is rapidly changing, with support from major computer industries (IBM, Oracle, Intel, etc.) for Linux based applications ...

# EU Policy shortcomings

- In fact the EU's "end-user driven" policy for funding R&D projects does not seem to encourage the development of European products and technologies competitive with the ones developed in the USA
- Real end-user do not want to "take risks" with really innovative technologies: in order to be acceptable to an end-user, a technology should be "mature enough"
- Within a funded project, inevitably developers and researchers are interested in pursuing and/or making profit out of their research and development efforts, rather than on "real applications"

# Perspectives

- From the scientific point of view, one of the main challenges is probably to provide a “single view” for the Cluster from all perspectives:
  - virtual CPU allocation by a generic node that acts as the front-end for that user;
  - coordinated scheduling of processes;
  - virtual RAM for the processes distributed over the cluster;
  - distributed, reliable, redundant File System based on the use of individual nodes' inexpensive disks (RAID-like approach)
  - secure identification of users and cluster resources;
  - privacy, availability and integrity in remote data access

## Perspectives (cont.)

- From the pragmatic user point of view, easy management (installation, maintenance, upgrade, etc.)
- A shift of focus is probably needed from specific computation and communication problems, techniques, and technologies, to a more complete approach in terms of distributed operating systems (there is now a chance to apply research results in this field to real, interesting problems in a production context)
- General interest in the subject is attested by the creation of a new Task Force on Cluster Computing by the IEEE Computer Society:

<http://www.dgs.monash.edu.au/~rajkumar/tfcc/>

LNGS-25th March 1999

## OCTOPUS: the LNGS PC's cluster

G. Di Carlo (LNF) and A.F. Grillo (LNGS)

Our experience in the assembling and use of PC networks, specialised in some sense for computing, started in December 1995. The first machines consisted only in processor boards, CPUs, memory, disk and network interface. We have avoided as much as possible the installation of hardware/software resources not immediately related with calculations, such as keyboard or X-environment. The only access to the computing nodes was through the network (formerly an Ethernet network, now a Fully Switched Fast Ethernet network).

This minimalist aptitude has been maintained during the following years. Up to now we were mostly interested in Farm-like systems; however the fairly good performances now available at low cost for the communication network subsystem, allows us to move the scope towards parallel machines with an increasing level of communications with respect to local computations.

In order to have a totally free software environment we have chosen Linux as operating system from the very beginning. Standard distributions of Linux were used (formerly Slackware, later Red Hat). According to the increase of node communication, communications libraries, other than simple IP, will be needed; MPI and/or Gamma seem to us a reasonable set.

The first array built was a small 5-nodes machine at LNF (PENTIX), based on Pentium-100 CPUs.

Then we moved to a more ambitious project, (OCTOPUS) based at LNGS partly financed by LNGS and by the 'Consorzio di Ricerca del Gran Sasso'; this installation consists at present of 16 computing and 2 front end nodes, connected through two 24-ports Fast Ethernet switches; this dedicated FE architecture allows the future increase of the number of the computing nodes.

These PC nets have been essentially production machines in these years: essentially, no R/D has been performed on them, but physics programs are running on them since the very first days. We enjoyed the full reliability of this hardware in the 4 years of our experience: in particular the PENTIX net is running without any interruptions since 12-1995, apart from general power failures.

The general performances of these non specialised CPUs for physics codes have been quite satisfactory, especially taking in mind the costs (around 60 Mlit for the switched large cluster). On these machines, we run programs of simulations of lattice gauge theories and of evolution of cosmic ray cascades.

25-3-99



OCTOPUS (AND HIS BAND...)  
PIRE  
THE LNGS ~~CLUSTER~~ OF PC

A. F. GRILLO

G. DI CARLO

THANKS

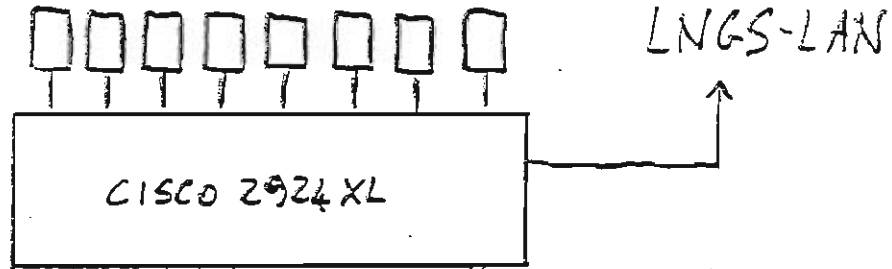
CARLOS E. PIEDRAFITA

AND

MARLO ROSATI (CASPUR)

FOR IDEAS

8 x P6-200



CISCO 2924XL

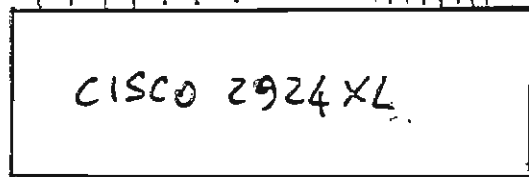
LNGS-LAN

00000000

14 PORTS FOR  
FUTURE EXPANSIONS

8 x FAST ETH. CHANNEL

00000000



CISCO 2924XL

PII-400



8 x PII-266

# THE FUTURE: (TOMORROW MORNING ??)

- FROM A FARM TO A PARALLEL SYSTEM (MPI, GAMMA???)
- MORE USER FRIENDLY ENVIRONMENT
- RE-INSTALLATION (SAME KERNEL ON ALL NODES !!!)

BUT...

THIS IS A "PRODUCTION" MACHINE !!

PHYSICS PROGRAMS RUN ON IT FROM THE FIRST DAY

- MOSTLY LGT
  - SPARSE MATRIX FULL DIAGONALISATION (LANCZOS)
  - FULL MATRIX. " " "

FOR QED WITH DYNAMICAL FERMIONS ( $> 5000 \cdot 14^4$  CONF/YEAR)

AND QCD AT FINITE BARYON DENSITY

- BUT ALSO COSMIC RAYS PHYSICS  
EVOLUTION OF EXTENSIVE AIR SHOWERS (CORSIKA)

AVERAGE LOAD  $\geq 90\%$



## Performance evaluation of the network subsystem of the Caspur Testing/Training Facility (CTF)

Leonardo Valcamonici - CASPUR  
LNGS 25.3.1999



## A bit of history: CASPUR experience in commodity hardware solutions

Our milestones:

- 1993. First PC with UNIX, mine. It ran Linux SLS 1.0 with 0.99pl6 kernel.
- 1994. Our first training room: ten Linux PCs.
- 1995. First Linux PC service machine: a secondary DNS server.

## A bit of history: CASPUR experience in commodity hardware solutions (Cont'd)

Our milestones:

- 1997. First Linux PC cluster: the CTF.
- Today. Almost everyone at CASPUR has a PC with Linux on its desk. Vendor UNIX servers and Workstations are used for intensive computing and heavy graphic applications.
- Tomorrow: More applications on Linux.

## The CASPUR Testing/Training Facility: CTF 1.0 (1997)

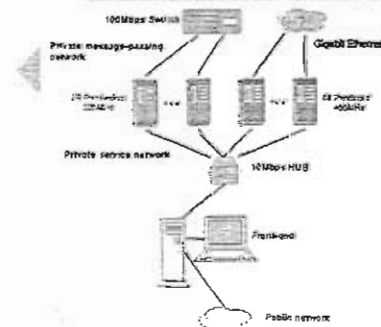
- #1 PentiumPro 200 x 2, 256 MB RAM, 4GB Ultra SCSI system HD, 18GB Ultra2 data HD, two DEC tulip NICs.
- #8 PentiumPro 200, 64MB RAM, 4GB EIDE HD, two DEC tulip NICs.
- #1 DEC Multiswitch 300 8 ports 1.0/100 half/full duplex switch.
- #1 16 ports HP Ethernet hub.

## The CASPUR Testing/Training Facility: CTF 2.0 (1998)

Thanks to the collaboration with the roman INFN group of the ATLAS experiment, we've added:

- #4 Pentium II 450, 256 MB RAM, 6.4GB UltraATA HD, DEC tulip NIC, G-NIC II Packet Engines (Alcatel) Gigabit NIC.

## CTF 2.0 network configuration



### Software on the CTF

- Linux RedHat 5.2, egcs compilers.
- Portland Group (PGI) HPF compiler.
- MPI (MPICH & LAM)
- PVM

### Benchmarking goals

- Evaluating Linux TCP/IP stack implementation.
- Evaluating the performance of the DEC MultiSwitch 300.

### Test bed

- ping - pong between Pentium Pros.
- 100 BaseTX tulip cards involved.
- Crossed cable and switch connection.
- Half and full duplex.
- Linux kernel 2.0.32 and 2.0.36.

### Tools

- ttcp 1.4.
- netperf 2.1.
- netpipe 2.0.

### TCP and UDP performance measures

- Numbers were taken with ttcp, sending 32000 times 16KB long messages.
- Tests were run 30 times and average values are displayed herein.
- Presented results are validated comparing them to the one obtained with netperf.

### What are we expecting ?

The nesting effect of data encapsulation into different protocol layers envelopes and network level frames reduces the network bandwidth an application can exploit. For fast ethernet:

IP layer	TCP layer	UDP layer
16 bytes	40 bytes	40 bytes

### UDP performance

One data stream:

Window size	Throughput	RTT	Loss
1024	1.18 Gbps	1.14 ms	0.0000%
2048	2.36 Gbps	1.14 ms	0.0000%
4096	4.72 Gbps	1.14 ms	0.0000%
8192	9.44 Gbps	1.14 ms	0.0000%
16384	18.88 Gbps	1.14 ms	0.0000%
32768	37.76 Gbps	1.14 ms	0.0000%

Two opposite data streams:

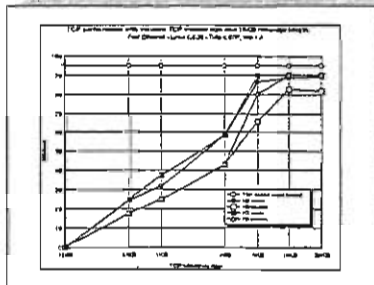
Window size	Throughput	RTT	Loss
1024	1.18 Gbps	1.14 ms	0.0000%
2048	2.36 Gbps	1.14 ms	0.0000%
4096	4.72 Gbps	1.14 ms	0.0000%
8192	9.44 Gbps	1.14 ms	0.0000%
16384	18.88 Gbps	1.14 ms	0.0000%
32768	37.76 Gbps	1.14 ms	0.0000%

### TCP performance with variable window size. One data stream.

TCP parameter	One Stream	Two Streams	One Stream	Two Streams
1024	1.18 Gbps	2.36 Gbps	1.14 ms	1.14 ms
2048	2.36 Gbps	4.72 Gbps	1.14 ms	1.14 ms
4096	4.72 Gbps	9.44 Gbps	1.14 ms	1.14 ms
8192	9.44 Gbps	18.88 Gbps	1.14 ms	1.14 ms
16384	18.88 Gbps	37.76 Gbps	1.14 ms	1.14 ms
32768 (max)	37.76 Gbps	75.52 Gbps	1.14 ms	1.14 ms

Note: due to the usage of integers in TCP buffer management, window size is limited to 32KB in Linux 2.0.32 and 2.0.36.

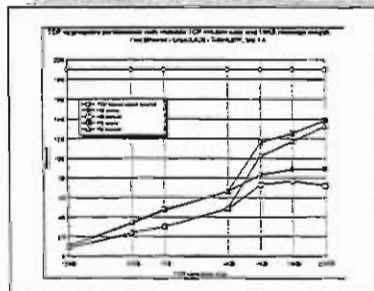
### TCP performance with variable window size. One data stream. (Cont'd)



### TCP performance with variable window size. Two opposite data streams.

TCP parameter	One Stream	Two Streams	One Stream	Two Streams
1024	1.18 Gbps	2.36 Gbps	1.14 ms	1.14 ms
2048	2.36 Gbps	4.72 Gbps	1.14 ms	1.14 ms
4096	4.72 Gbps	9.44 Gbps	1.14 ms	1.14 ms
8192	9.44 Gbps	18.88 Gbps	1.14 ms	1.14 ms
16384	18.88 Gbps	37.76 Gbps	1.14 ms	1.14 ms
32768 (max)	37.76 Gbps	75.52 Gbps	1.14 ms	1.14 ms

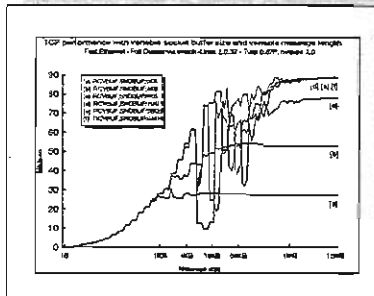
### TCP performance with variable window size. Two opposite data streams. (Cont'd)



### Testing with netpipe ...

- It's a ping-pong class test.
- It sends n times a data block c, from size i to size j with step k.
- For any given c, measures are taken for c-p, c and c+p, where p is a perturbation factor (usually 3 bytes).

... reveals a Linux TCP/IP bug.



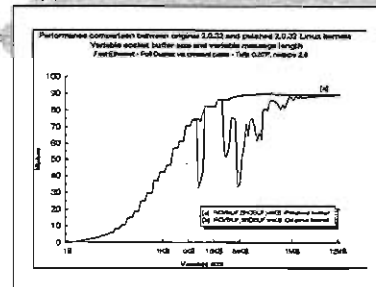
Where is the bug ?

- Kernel 2.0.32 has a flaw implementation of the Nagle's algorithm: partial TCP packets can only be sent when all previous segments of the message are acknowledged.
- It's particularly nasty in combination with the TCP window slow start algorithm.

Fixing ...

- An appropriate kernel patch fixing the problem is now available on Usenet.
- Kernel 2.0.36 is free of this bug.

Fixed !



Latency

Latency has been measured with the TCP\_RR netperf test:

Kernel	Latency (ms)
2.0.32	~75
2.0.36	~15

Note: the switch is "store and forward".

Conclusions

- Extensive benchmarks have shown that both the NIC driver and the TCP stack perform very well under Linux 2.0.32 (patched) and 2.0.36.
- TCP/IP in terms of performance and stability seems to be a modern and high grade implementation of the standards.

### Conclusions (Cont'd)

- In a one way test, Linux 2.0.32/2.0.36 can achieve the 96.2% of the theoretical UDP over Fast Ethernet bandwidth and the 94.9% of the TCP over Fast Ethernet one.
- Latency tests aswell report very high performance values.

### Conclusions (Cont'd)

- As to the Digital Equipment Corp. Multiswitch 300, tests have shown its performance are nowadays not comparable to ones declared for modern Fast Ethernet or Gigabit Ethernet switching devices. Its limitations especially show up in terms of latency, aggregated bandwidth and supported configuration options.

### Therefore we can say:

- Linux is a valuable alternative for building a general purpose *network of workstations* for distributed and parallel computing.
- Application level performance numbers show up such a configuration could work effectively with distributed applications not requiring intensive intercommunication.

# The DISCO project

P. Rossi

ENEA HPCN Project

Via Martiri di Monte Sole, 4, Bologna, Italy

April 9, 1999

## 1 The problem

The idea is to develop a very low cost archival and retrieval system based on a cluster of PC equipped with Linux operating system and public domain data base software. The technology adopted is an evolution of two scientific projects ( PARMA2 and GAMMA<sup>2</sup> ) carried out in the last couple of years by the University of Parma and the University of Genova. The specific application shows the general validity of the technology consisting of off the shelf, low cost commodities and freeware software.

For the problem at hand the issue is to set up a multimedia data base to handle technical drawings coming from the industrial activities of a mechanical industry involved in producing packaging machines for pharmaceutical products. The technical solution consists in a cluster of PC connected via a fast ethernet network. Each node runs Linux as the operating system, suitably endowed with a light weight protocol developed at the University of Genova to support low latency and high transfer rate. Upon this basic software there is superimposed a multi media data base, developed at the University of Parma. The demonstrator has been assembled by ENEA in Bologna and is currently visible via a connection to the Internet. The user interface for the data base queries has been developed in standard HTML and Java, so to allow any remote computers, equipped with a browser, to access the information stored.

## 2 The HW platform

The hardware architecture of DISCO currently comprises four nodes and a front-end; in Figure 1 there is a schematic representation of it. The front end, a pentium 200MMX decouples the nodes from internet, acting as a fire-wall, while the nodes are accessible directly from the LAN. Each node is an Intel Redwood motherboard with dual PentiumII 300Mhz, 128MByte of RAM a Adaptec AIC-7880 Ultra SCSI host adapter and two network interface cards. The SCSI adapter serves a 4.5GByte hard disk from Quantum. Of the two interface cards, one is an Intel EEpro 100 integrated on board, the other is a 3Com905. The EEpro 100 is connected to a 10Base ethernet hub and through that to the LAN while the 3C905 connects the nodes through a 100BaseTX switch. The ethernet network is primarily used for system administrative tasks and NFS mounting of user directories, while the fast ethernet network, accessible only within the nodes, is used by the high performance applications.

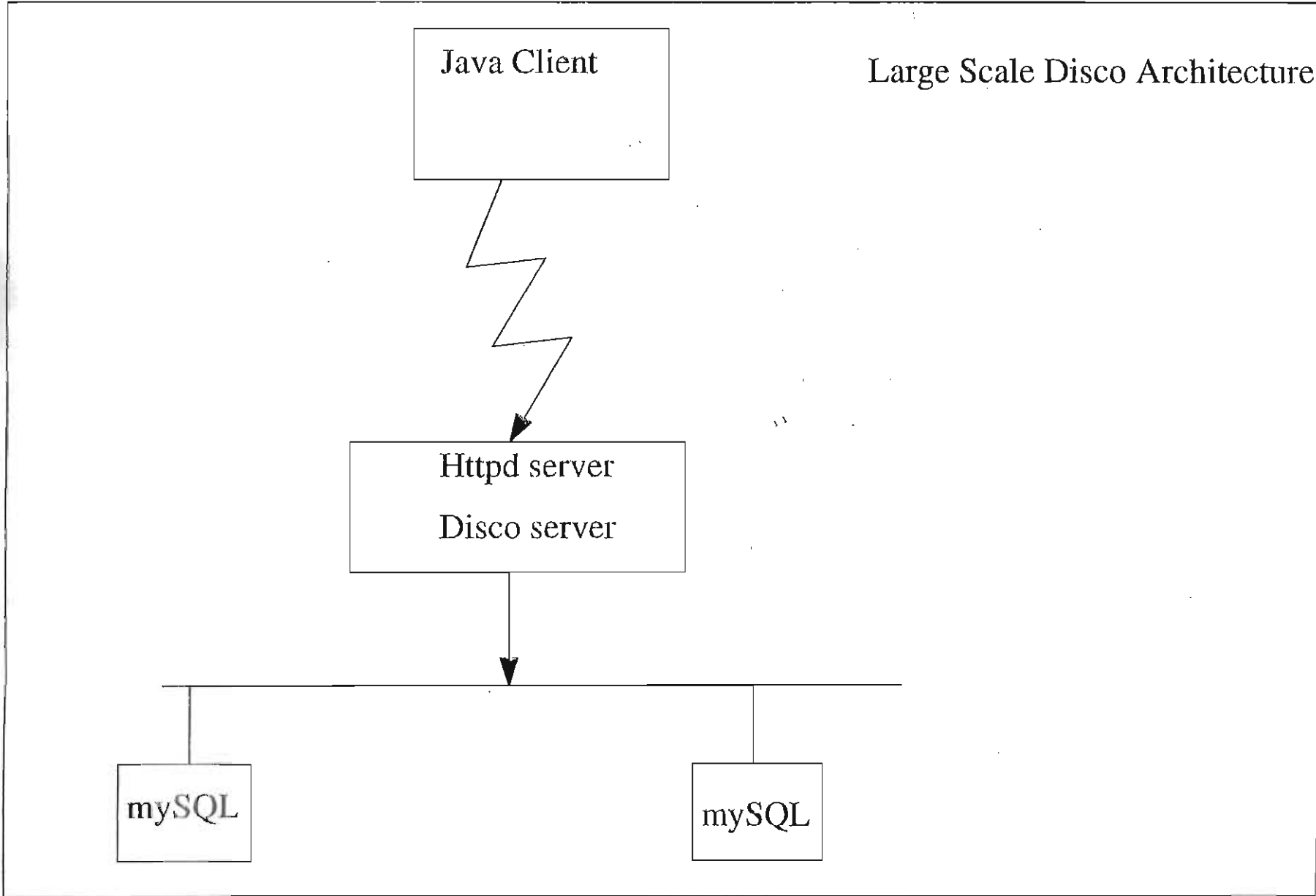
Large Scale Disco Architecture

Java Client

Httpd server  
Disco server

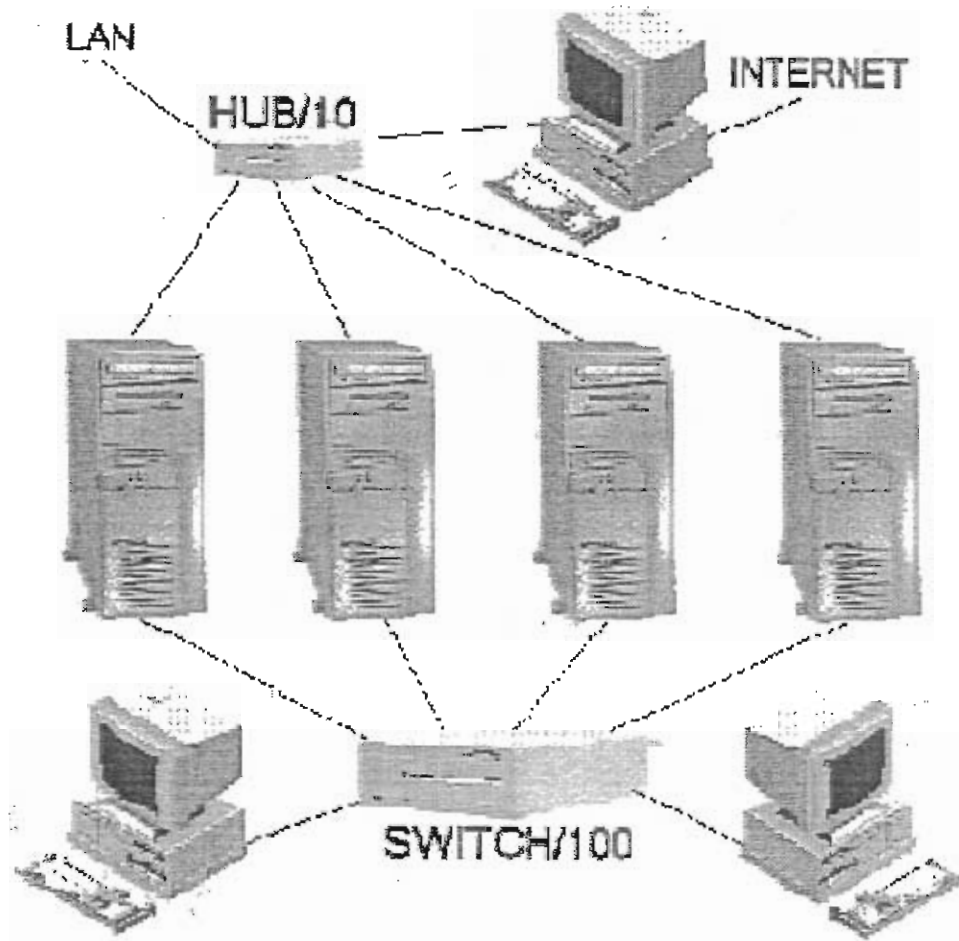
mySQL

mySQL





## Disco Hardware Layout



Hubert Simma  
APE Group INFN/DESY-Zeuthen

### Physics Requirements

During the next years Lattice Gauge Theory applications (QCD and other non-perturbative problems) are expected to require heavily CPU limited simulations with dynamical fermions on up to  $50^3 \times 100$  lattices and also quenched calculations on very fine grained lattices (up to  $100^3 \times 200$ ) for heavy quark physics which is memory limited. A future QCD compute engine could consist of several independent machines, each with a few TFlops sustained, 1-10 TBytes of memory and adequate mass storage access.

### Benchmarks

The computing performance for these applications depends critically on the speed of floating point arithmetics, of memory access, and of remote communications. Preliminary benchmarks on a Dual Pentium II system indicate about 30 % efficiency for LGT kernels. On a dual processor system we observed almost twice the performance of a single processor. The L2 cache gives only moderate gain for realistic lattice sizes. However, the results strongly depend on the compilation (e.g. gcc vs egcs and options) and the execution environment. The effect of remote communications requires further studies, but network interfaces in the order of Gbit/sec and 3-d interconnect topologies seem to be mandatory.

### Commercial Solutions

Interesting examples of commercial PC-cluster solutions, scalable up to hundreds of processors, are the *AltaCluster* (with Myrinet) from Alta Technologies [[www.altatech.com](http://www.altatech.com)] or the *hpcLine* (with SCI) from Siemens [[www.siemens.de/computer/hpc](http://www.siemens.de/computer/hpc)]. The estimated costs of such systems today are about 7-8 M Euro per TFlops peak performance where roughly half of the cost is for the required high-bandwidth interconnect (including optimised message passing libraries).

### Custom Solutions

The pure hardware costs for such a system might be strongly reduced by putting together commodity PC's or boards (e.g. Dual Pentium III for about 1800 Euro) and a custom network performing close to the maximum PCI bandwidth (like the Flink card as used in the APEmille host network with components that cost about 200 Euro). However, considerable efforts for integration, SW support and maintenance of the system will be required.

### Perspectives

One may hope that in 2 years from now (multi-processor) commodity boards with around 1 GFlops sustained performance for LQCD codes could be available. The integration of a 2 TFlops machine with  $O(2000)$  boards raises mainly problems of reliability, power consumption ( $\approx 200$  kW), and space ( $\approx 32$  racks).

We conclude that PC Clusters have become an interesting and promising platform for medium-scale LGT computations. Deploying LGT optimised (i.e. tightly interconnected) machines with a few hundred GFlops should not be a problem on principle. Such an effort would be a necessary prerequisite in order to assess the feasibility of PC-based Multi-TFlops machines.

# PC's for Multi-TFlops LGT Compute Engines?

Hubert Simma

APE Group INFN/DESY-Zeuthen

## 1 Physics Requirements

- Lattice QCD and other non-perturbative problems
- Lattice sizes for “light”-physics (dynamical fermions) and B-Physics (quenched)
- Main computational steps and costs: Dirac operator
- Profile of a QCD Compute Engine with multiple TFlops in 2003
- Advantages of PC-based solutions

## 2 PC Benchmarking

- Basic questions (memory+cache system, FP performance, remote comm.)
- Naive performance factors
- Results from a Dual Pentium II system
- Missing work

## 3 Possible PC Solutions

- Basic architecture with commodity boards
- Examples of available commercial solutions (AltaCluster, hpcLine)
- Example of a custom network (FLINK)
- Non-commercial solutions?
- Extrapolation

## 4 Summary

- Open problems
- Perspectives

# Physics Requirements

## LQCD Problems:

- Phenomenology of light and heavy Hadrons
- Electroweak Matrix Elements
- Running Coupling Constant and Quark Masses

## Other non-perturbative Problems:

- Supersymmetric Yang-Mills Theories
- Phase Transitions (electroweak, quark-gluon)

## Problem Sizes:

- "Light"-Physics: dynamical fermions  
 $L = 2 \dots 3 \text{ fm}, a = 0.1 \dots 0.05 \text{ fm}$   
 $\Rightarrow$  Lattice Size  $50^3 \times 100$   
Performance limited: 5–10 TFlops sustained
- B-Physics: quenched approximation  
 $L = 1.5 \dots 2 \text{ fm}, a = 0.1 \dots 0.02 \text{ fm}$   
 $\Rightarrow$  Lattice Size  $100^3 \times 200$   
Memory limited: 1 fermion = 38 GByte

## Profile of a QCD Compute Engine:

Several machines by the year 2003, each with:

- 2–3 TFlops sustained
- Double Precision FP
- Memory: 1–10 TByte
- Disk I/O: few GByte/sec

## Advantages of PC-based Solutions:

- Profit from impressive PC evolution (incl. FP performance)
- Price/Performance
- Availability of HW components
- Reliable and free Linux SW environment (incl. compilers)
- Educational value

# PC Benchmarking

Naive Performance Factors:

Floating-point Calculus

$$t_{FP} = \left( \frac{400MHz}{F_{peak}} \right) \left( \frac{V_{loc}}{4^3 \cdot 100} \right) \cdot 24 msec$$

Memory Access

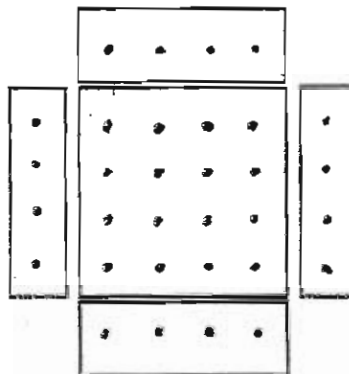
$$t_{MA} = \left( \frac{100MHz \cdot 8B}{B_{loc}} \right) \left( \frac{V_{loc}}{4^3 \cdot 100} \right) \cdot 25 msec$$

Remote Communications

$$t_{RC} = \left( \frac{S_{rem}}{0.25} \right) \left( \frac{100MB/sec}{B_{rem}} \right) \left( \frac{V_{loc}}{4^3 \cdot 100} \right) \cdot 25 msec$$

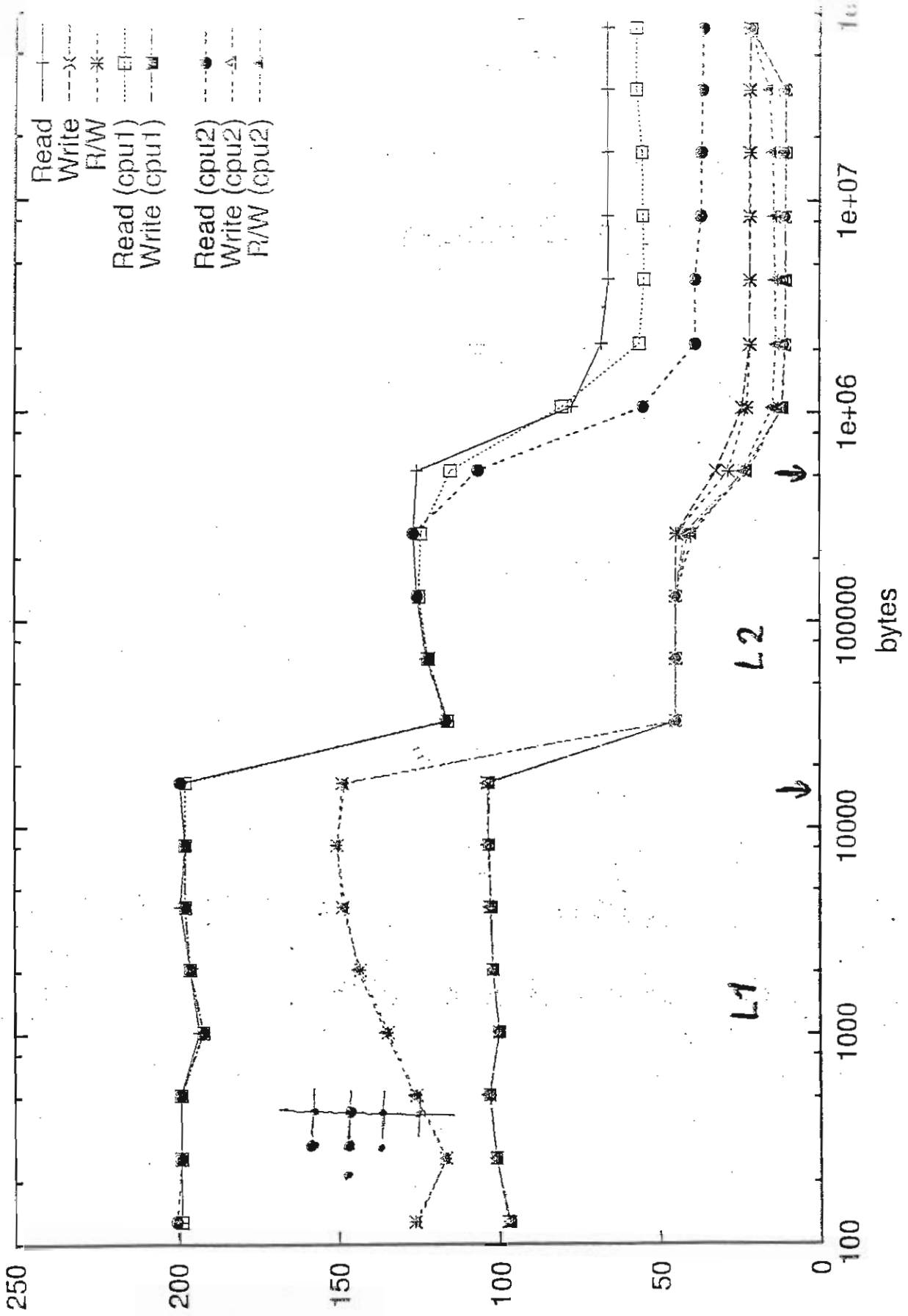
where  $S_{rem}$  = fraction of remote spatial neighbours

and frame buffers are assumed for  $\Phi$  (and  $U$ )



# Effect of caches and data cont. (Pentium II routine)

Memory <-> Register I/O (16 doubles blocked)

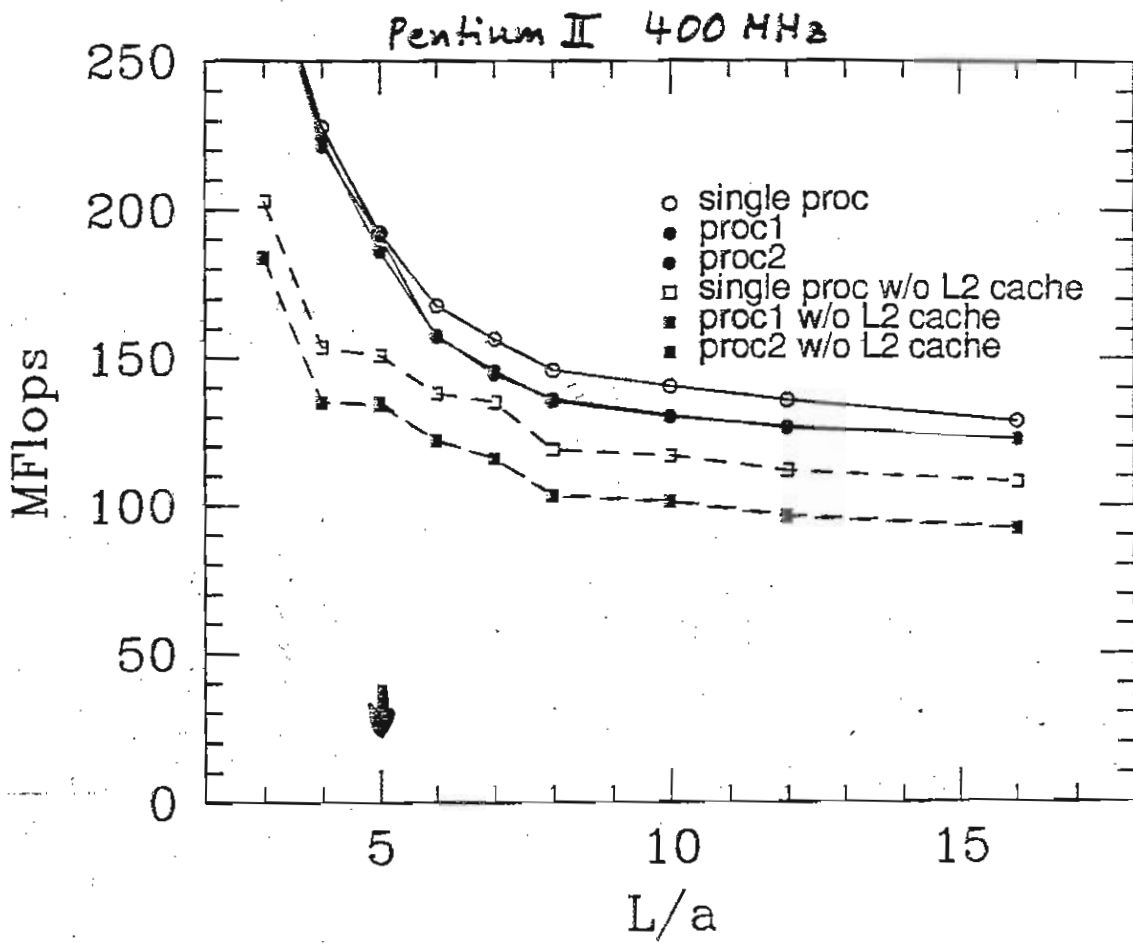


Fstp %st(0)  
 Fldl %eax,%eax

top

# SU(3) Dirac Operator

(1754 Flop /side)





## Non-Commercial Solutions?

E.g. Dual PentiumII (1.8 k Euro) + Flink (200 Euro)

Pay half the price but work on:

- Mechanical integration (fans, power supply,...)
- Optimized message passing library  
(or buy for 50–100 k Euro)
- OS (exception handling, parallel I/O, ...)
- System administration tools (booting,...)

## Extrapolation:

- New Processor Architectures (Merced, ...) ???
- Next Generation IO ???
- Assume commodity boards in 2001 with:  
2–4 Processors at 1 GHz  
⇒ about 1 GFlops/board sustained  
⇒ need about 2000 boards!

# Summary

## Open Problems:

- Integration of  $O(2000)$  multi-processor boards:
  - Reliability
  - Power ( $>200$  kW)
  - Space ( $>32$  racks)
- Cost of Communications (% and \$)
- Operating System (IO hosts, Bios, ...)
- Administration

## Perspectives:

- PC Networks are interesting and promising for LGT computations
- Several-hundred-GFlops machine should not be a principal problem
- Would have to build it (now!) in order to assess feasibility of Multi-TFlops machine