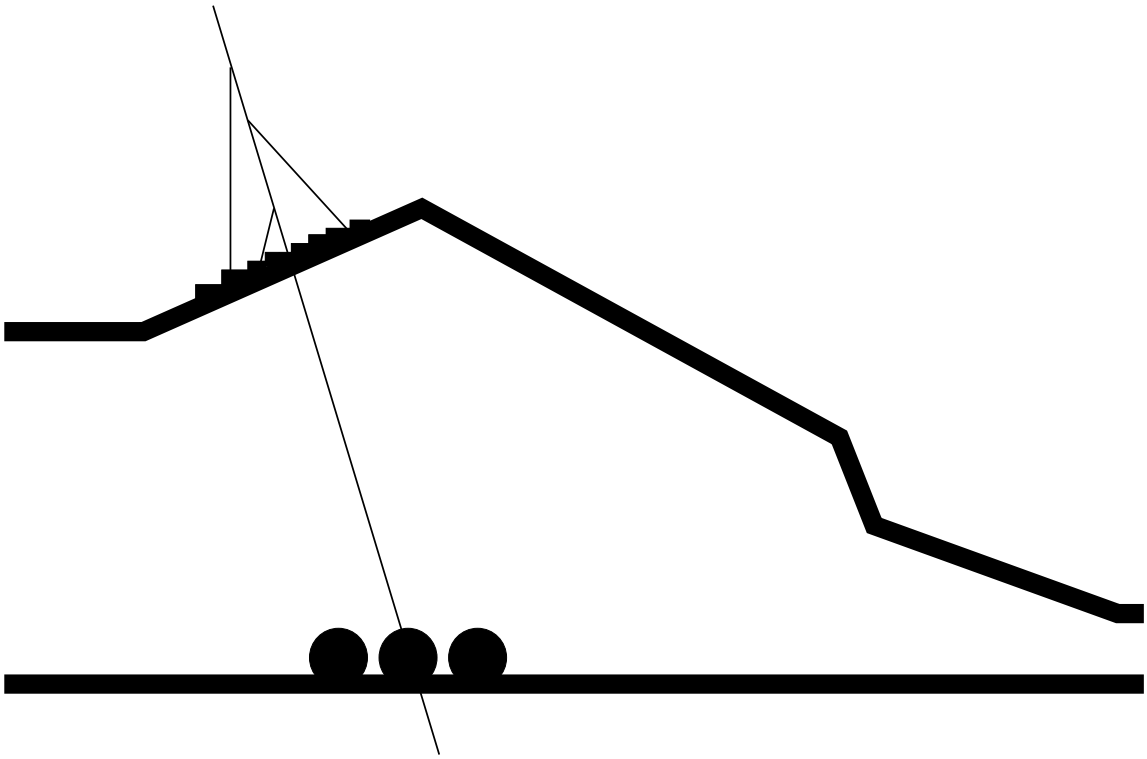
INFN/TC-00/21
6 Dicembre 2000



PcNets2000

A workshop aimed at discussing
applicative, software, hardware aspects of cluster computing.

April 26–28 2000

Giuseppe di Carlo, Andrea Donati, Maria-Paola Lombardo, Pietro Rossi
Organizers

INFN - Laboratori Nazionali del Gran Sasso

INFN - Istituto Nazionale di Fisica Nucleare
Laboratori Nazionali del Gran Sasso

INFN/TC-00/21
6 Dicembre 2000

PcNets2000

A workshop aimed at discussing
applicative, software, hardware aspects of cluster computing.

April 26–28 2000

Giuseppe di Carlo, Andrea Donati, Maria-Paola Lombardo, Pietro Rossi
Organizers

Abstract

This booklet includes notes and material from the talks presented at the meeting. Most of the contributions have been prepared by the speakers, some consist of an 'organizers' choice' of a few significant transparencies. In either case we refer to the complete set of trasparencies, available on the conference WEB site:

`http://www.lngs.infn.it/site/meesem/pcnets2000.htm`

Foreword

Pc-Nets2000, the second workshop on applicative, software, hardware aspects of cluster computing, took place at the Gran Sasso National Laboratory, in Italy, from April 26 to April 28, 2000.

Our invitation to participate in this informal gathering was accepted by many colleagues working in many different areas, from computer sciences, to fundamental physics, either in academic and commercial environments. We are grateful to all of the participants for their contribution to the success of this meeting.

It is a great pleasure to thank those who agreed to advise us in setting up the program: Giovanni Chiola (DISI, University of Genova), Thomas Lippert (University of Wuppertal), Agostino Mathis (HPCN-ENEA, Roma), Mirco Mazzucato (University and INFN, Padova), Roberto Petronzio (University and INFN, Roma II), Agostino Poggi (University of Parma), Vittorio Rosato (HPCN-ENEA, Roma), Klaus Schilling (University of Wuppertal and NIC, Juelich), Hubert Simma (DESY) and Raffaele Tripiccione (INFN, Pisa).

We also gratefully acknowledge the support of the Istituto Nazionale di Fisica Nucleare and the Laboratori Nazionali del Gran Sasso. In particular we wish to thank the LNGS Scientific Secretariat, Vincenzo Fantozzi and Ersilia Giusti, as well as Barbara Sartini and Ilaria Spagnoli of the Consorzio Ricerca del Gran Sasso.

Some of the contributions included in this booklet have been prepared by the speakers, some consist of an 'organizers' choice' of a few significant transparencies. In either case we refer to the complete set of trasparencies, available on the conference WEB site:

http://www.lngs.infn.it/site/meesem/pcnets2000.htm

for full information.

We do hope that these proceedings as well as the conference itself fulfil their main purpose: fostering communication among people who share an interest in computers and computing.

Giuseppe di Carlo, Andrea Donati, Maria-Paola Lombardo, Pietro Rossi

October 2000

Laboratori Nazionali del Gran Sasso
lstituto Nazionale di Fisica Nucleare

Pc-Nets2000

April 26 to April 28, 2000

A workshop aimed at discussing applicative, software, hardware aspects of cluster computing.

Local Organizing Committee :

Giuseppe di Carlo, Andrea Donati, Maria-Paola Lombardo, Pietro Rossi

Program Committee :

Giovanni Chiola (DISI, University of Genova), Thomas Lippert (University of Wuppertal), Agostino Mathis (HPCN-ENEA, Roma), Mirco Mazzucato (University and INFN, Padova), Roberto Petronzio (University and INFN, Roma II), Agostino Poggi (University of Parma), Vittorio Rosato (HPCN-ENEA, Roma), Klaus Schilling (University of Wuppertal and NIC, Juelich), Hubert Simma (DESY), Raffaele Tripiccione (INFN, Pisa)

Coordinator: Enzo Fantozzi (LNGS)
Secretariat: Barbara Sartini (CRGS), Ilaria Spagnoli (CRGS)

Program

26 April

Chair : Stefano Fantoni

11.15 - 12.00	G. Bilardi	Universality and portability in high performance computing
12.00 - 12.30	R. Petronzio	QuantumChromoDynamics
12.30 - 13.10	T. Lippert	The ALICE project at Wuppertal
13.10 - 13.30	N. Eicker	Operating a cluster system in an University environment

13.30 - 14.30 Lunch

Chair : GianCarlo Rossi

14.30 - 15.10	G. Visconti	Atmospheric Physics
15.10 - 15.30	M. Verdecchia	Real time weather forecast
15.30 - 16.20	V. Rosato, F. Valentinotti	Heterogeneous Computing as a key feature for high performance computing. The GENESI-2 project and applications on environmental modeling

16.10 - 16.30 coffee break

16.30 - 18.30	APEmille Tutorial - 1st part

18.30 - 19.30 Buffet

19.30 - 21.00	APEmille Tutorial - 2nd part

---

27 April

Chair : Aurelio Grillo

9.30 - 10.10	S. Fantoni	Quantum Monte Carlo
10.10 - 10.30	F. Pederiva	New perspectives in Quantum Monte Carlo methods for many fermion systems
10.30 - 10.50	E. Panizzi	The C++ language for APEmille
10.50 - 11.10	G. Ciaccio	GAMMA on Gigabit ethernet
11.10 - 11.30	V. Di Martino	GROMACS, a testbed code for the MPI-GAMMA library

11.30 - 12.00 coffee break

Chair : Edwin Laermann

12.00 - 12.30	K. Jansen	Fermion algorithms
12.30 - 12.50	P. Rossi	Data mining
12.50 - 13.10	C. Best	Improving cache performance on cluster computers using multigrid algorithms

13.30 - 15.00 Lunch

Chair : Steven Gottlieb

15.00 - 15.40	B. Monien	Graph partitioning
15.40 - 16.00	R. Lueling	Hpc-line, a new cluster system
16.00 - 16.20	W. Schroers	What do we gain from systolic algorithms for long-range interactions on cluster computers
16.40 - 17.00	N. Paschedag	SW/HW recent developments of FLINK

17.00 - 17.30 coffee break

Chair : Yutaka Ishikawa

17.30 - 18.00	S. Kim	Kool MPI
18.00 - 18.30	S. Katz	The PMS project: Poor Man's Supercomputer
18.30 - 19.00	T. Warschko	High performance cluster computing
19.00 - 19.20	B. Ugolotti, L. Genoni	SecureNAT Linux cluster solution

Dinner

---

28 April

Chair : Burkhard Monien

9.30 - 10.10	S. Gottlieb	Benchmarking MILC QCD code on clusters and supercomputers
10.10 - 10.40	A. Vicere'	Computing for VIRGO
10.40 - 11.10	C. Palomba	VIRGO data analysis for periodic sources of gravitational waves

11.10 - 11.40 coffee break

Chair : Enrico Bellotti

11.40 - 12.20	Y. Ishikawa	An overview of SCore 3.0 cluster system software
12.20 - 13.00	S. Morante & G.C. Rossi	Modeling biological systems by computer simulations: the case of phospholipid bilayers

13.00 - 14.00 Lunch

Final Session

Chair: Klaus Schilling

Chairman's introduction:	Computational science along with computer scientists
P. Capiluppi	The Grid project and the LHC computing challenge
A. Ghiselli	The Grid as advanced solution for WAN distributed computing
R. Tripiccione	APEmille & APEnext

Discussion

16:00 Conference ends

ALiCE
The Alpha-Linux-Cluster-Engine at Wuppertal University

Thomas Lippert
Department of Physics
and
Institute for Applied Computer Science
University of Wuppertal

ALiCE

The Alpha-Linux-Cluster-Engine at Wuppertal University

Thomas Lippert

Department of Physics
and
Institute for Applied Computer Science
University of Wuppertal

Pc-Nets2000
April 26 to April 28, 2000
Laboratori Nationali del Gran Sasso

Outline

1. Why Cluster-Computing ?

2. Wuppertal Parallel Computing History

3. Wuppertal Profile and Benchmarks

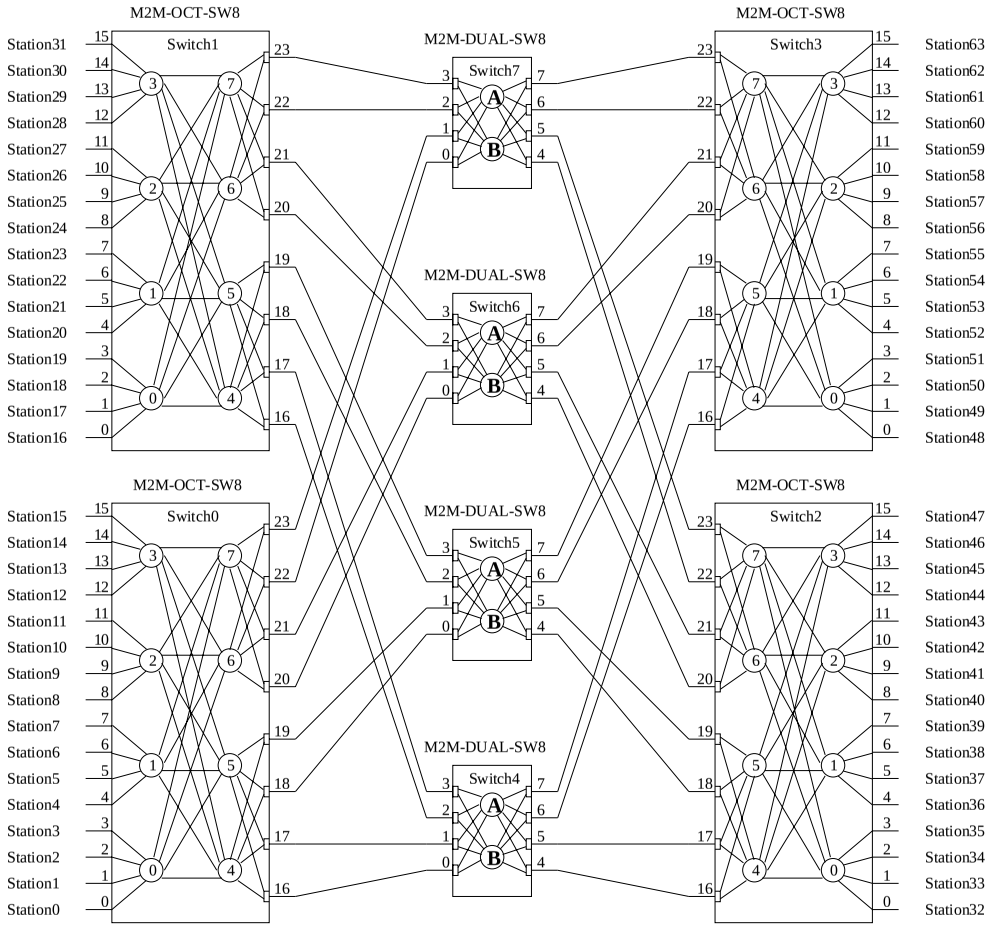
4. Selected Applications

   (a) CQED Simulations: Alpha-Linux-Cluster-Engine vs. Cray T3E
   (b) Matrix Multiplication
   (c) QCD

5. Still to do. . .

6. Summary

The Myrinet Topology



Myrinet achieves fully scalable bi-sectional bandwidth. Under ParaStation, one can reach about 150 Mbytes/s using the 21264 CPU.

4 Selected Applications

Compact QED Simulations: Alpha-Linux-Cluster-Engine vs. Cray T3E

- U(1) Phase trans: Standard simulation algorithms get stuck due to metastabilities
  We need to enhance the number of flips between the two phases to decide which order!

- Our idea: We merge the multicanonical algorithm with the hybrid Monte Carlo

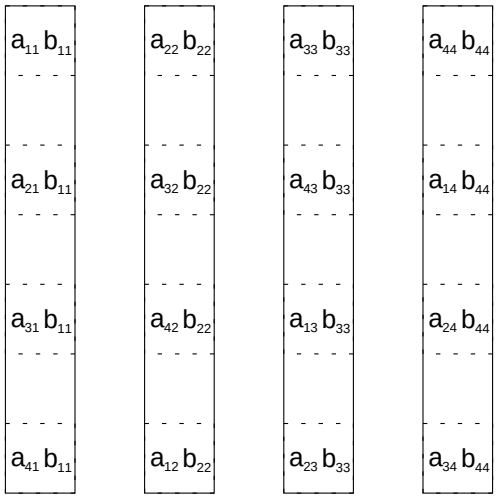
- Flip rate: Follows power law instead SCSD exponential
  Improvement by factor of order 100-1000 on 24^4 lattice

Hyper-Systolic Matrix Multiplication

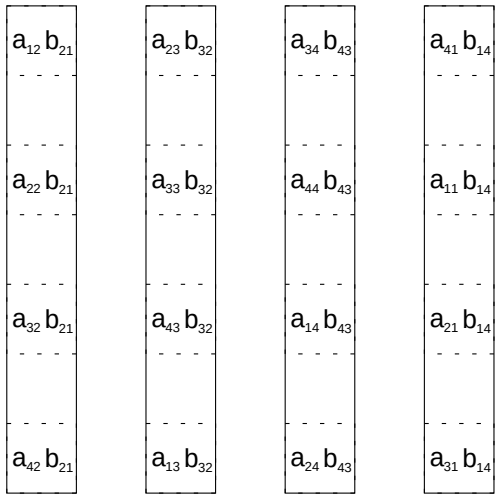
- H-S Algorithms: for details ⇒ see talk of W. Schroers

Simple systolic example

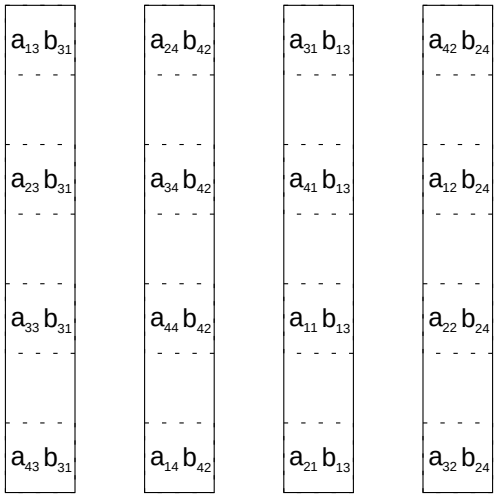
$$c_{i,j} = \sum_{k=1}^{m} a_{i,k}\, b_{k,j}, \quad i,j = 1,\ldots,n.$$



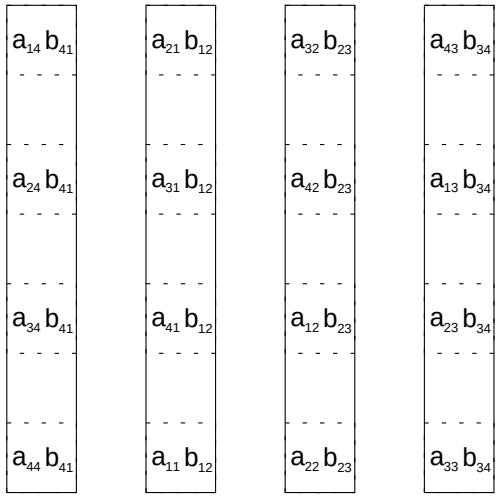
a) t=1



b) t=2



c) t=3



d) t=4

Operating a Cluster System in a University Environment

Norbert Eicker
Department of Physics
University of Wuppertal

Outline

I. Necessities in a University Environment

- Standard Programming Languages
- Commodity Operating System
- Standard Communication Libraries

II. Batch system

III. Visualization

IV. Parallel filesystem

V. Summary

I. Necessities

- Standard Programming Languages

  Students know (and need) standard languages like C, Fortran and C++.

  - egcs 1.1.2 (C, C++, F77)
  - gcc/egcs's optimization is know to be poor on Alpha.
  - Compaq ports its True64 compilers to Linux.
  - Fortran (f77 & f95) and C compilers already available.
  - C++ is available as Beta.
  - $\sim$ 20% performance improvement.
  - Compaq's `cpml`– & `cxml`–library also available (BLAS etc.).

- Commodity Operating System

  Students know Linux from their PC at home.

  - SuSE 6.3 / AXP (glibc 2.1 based)
  - Linux 2.2.14

  ALiCE behaves like a desktop PC.

- Standard Communication Libraries

  We must teach the students a standard and not proprietary solutions.

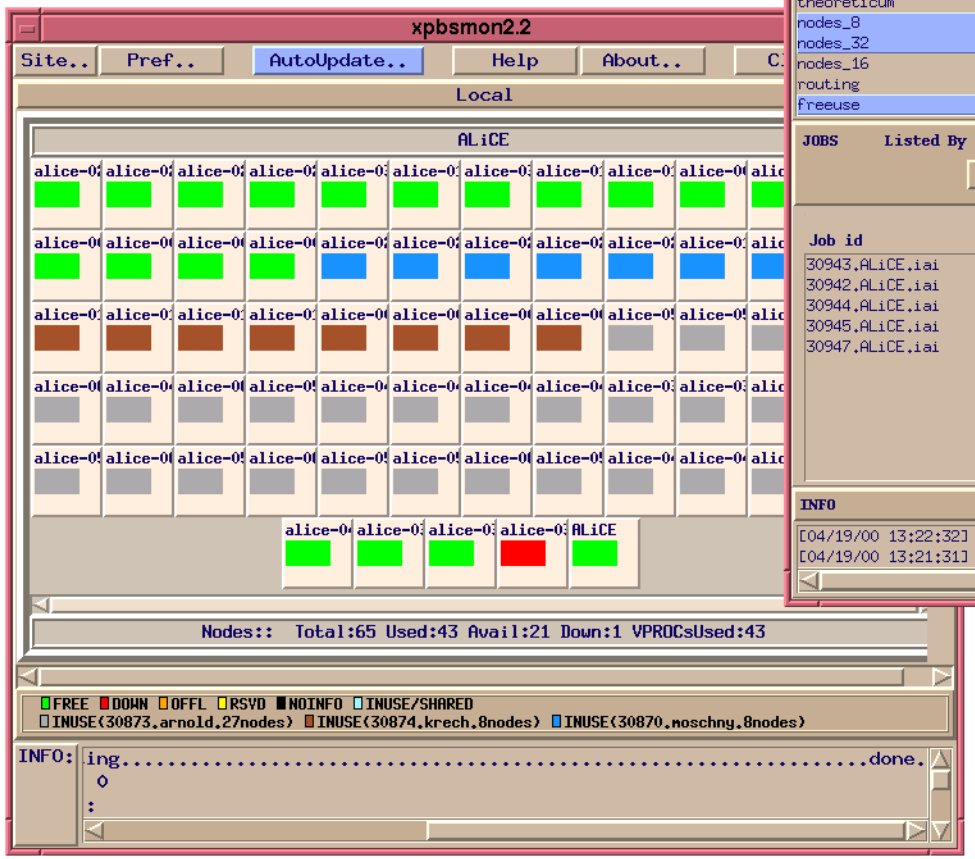
  - Use MPI instead of less portable communication libraries.
  - ParaStation is fast, secure, stable and supports MPI.

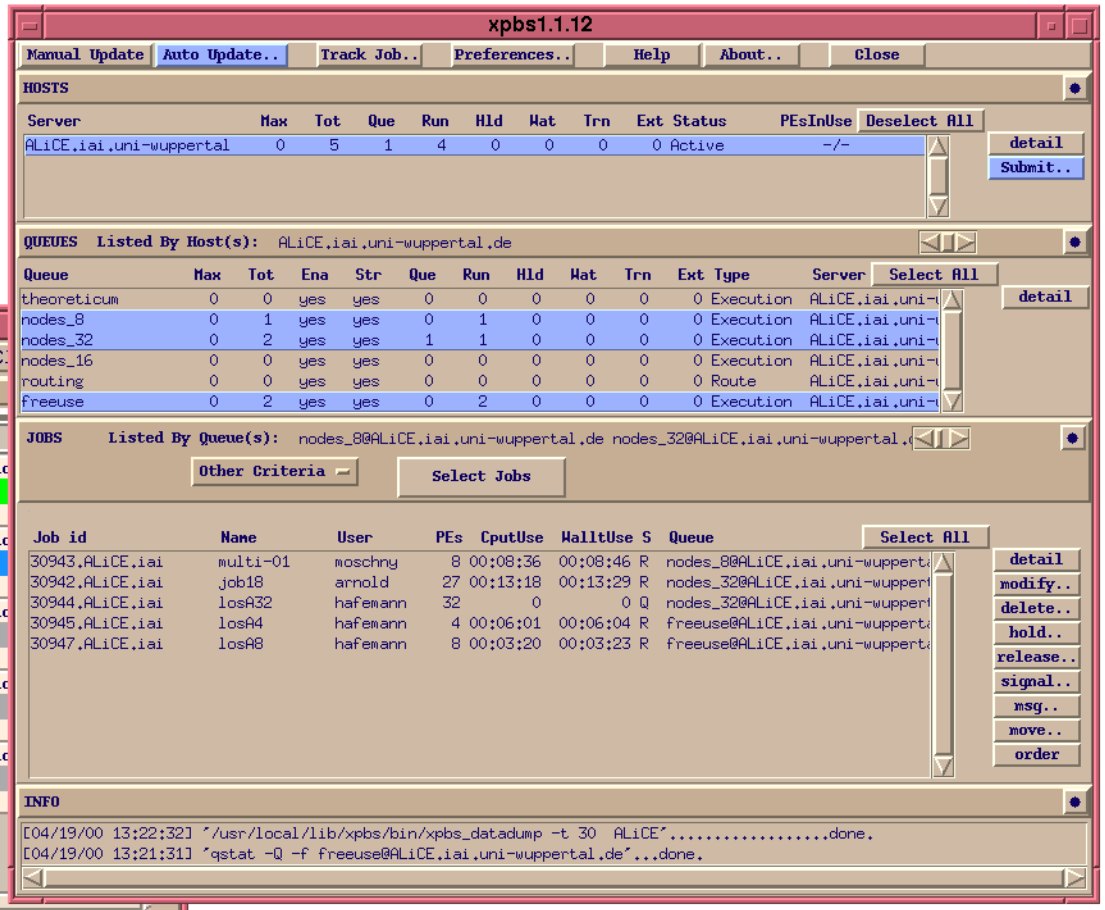
  You can program ALiCE like a Cray T3E.

```
ALiCE:~>cat script

#!/bin/bash
#PBS -N LookingGlass
#PBS -l nodes=4
#PBS -j oe
#PBS -o output
cd /home/eicker/Programs/c/Wilson
./linsolv_SSOR -np 4

ALiCE:~>qsub script
```





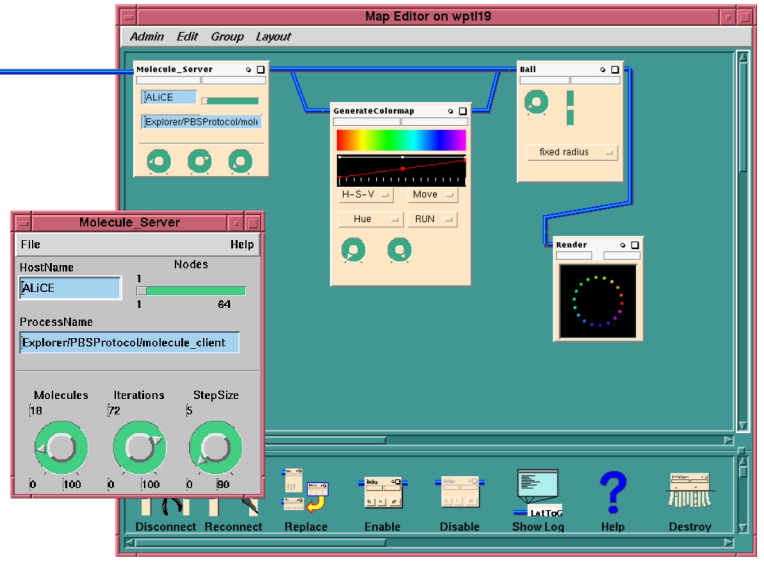

```
ALiCE:~>qstat

Job id          Name             User            Time Use S Queue
--------------- ---------------- --------------- -------- - -----
30943.ALiCE     multi-01         moschny         01:04:20 R nodes_8
30961.ALiCE     losA16           hafemann               0 Q nodes_16
30963.ALiCE     job18            arnold          00:02:43 R nodes_32
30964.ALiCE     losA4            hafemann        00:10:41 R nodes_8
30965.ALiCE     IRIS_Explorer    eicker          00:00:07 R theoreticum
```

```
#include <cx/PortAccess.h>
#include <cx/DataAccess.h>
#include <cx/UI.h>
#include <LookingGlass.h>
static int sock_fd=-1;
void *io_handle=NULL;
void molecule_server(char *hostName, char *appName, int noNodes,
                     int noMolecules, int noIncrements, int stepSize)
{
    if (sock_fd == -1){
        /* No valid fd yet, try to open client */
        sock_fd=open_client(hostName, appName, noNodes);
    }
    if (! io_handle)
        io_handle = cxInputAdd(sock_fd, cx_InputReadMask, io_handler, NULL);

    write_client(sock_fd, &noMolecules, sizeof(noMolecules));
    write_client(sock_fd, &noIncrements, sizeof(noIncrements));
    write_client(sock_fd, &stepSize, sizeof(stepSize));
}
```

LookingGlass

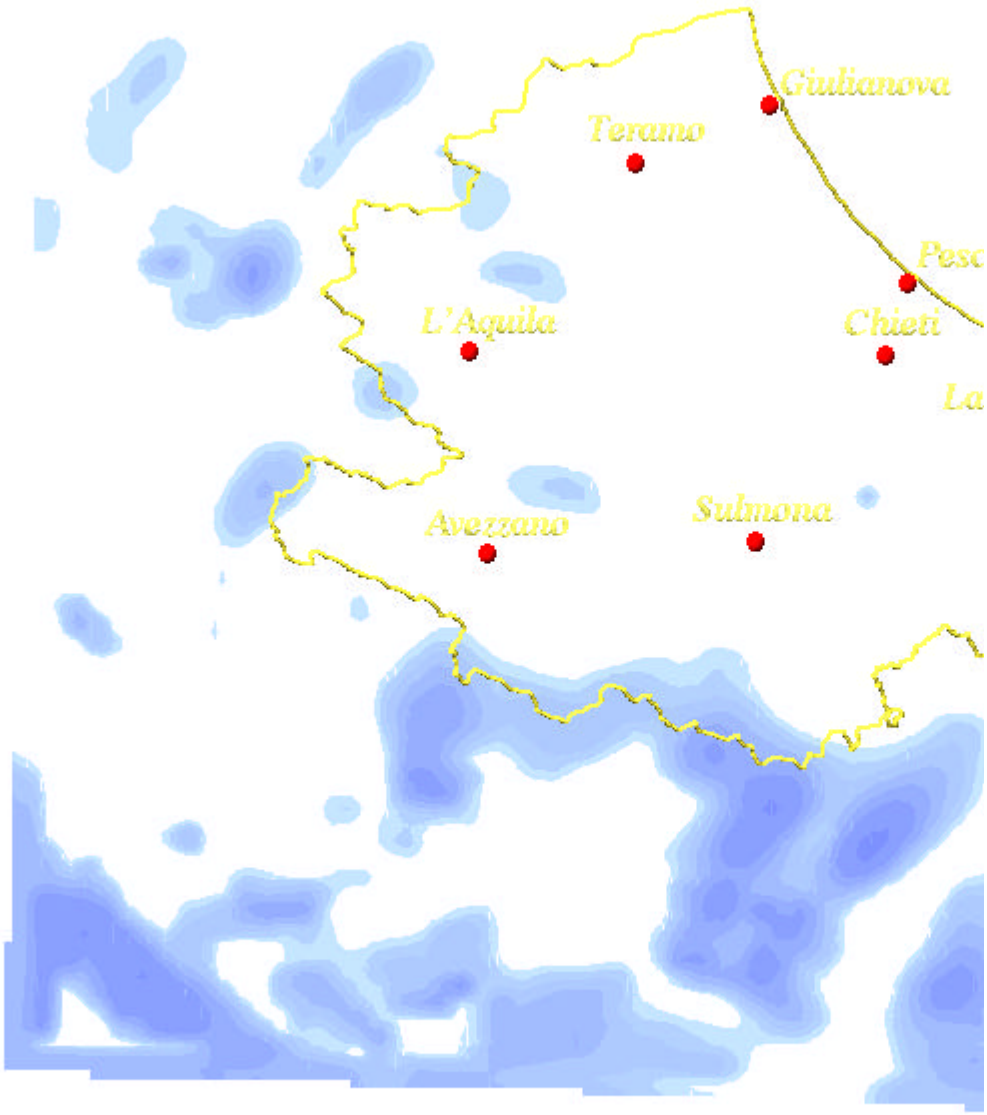



```
#include <stdio.h>
#include <LookingGlass.h>
int main (void)
{
    int sock_fd;
 sock_fd=open_server();
    if (sock_fd == -1){
        printf("Something went wrong within 'open_server'\n");
        exit(1);
    }
    while(1){
        read_server(sock_fd, &noMolecules, sizeof(noMolecules));
        if(noMolecules == -1)
            break;
        read_server(sock_fd, &noIterations, sizeof(noIterations));
        .
        .
        .
        write_server(sock_fd, &noMolecules, sizeof(noMolecules));
        write_server(sock_fd, coord, sizeof(coord));
    }
    close_server(&sock_fd, -1);
    return 0;
}
```

Real Time Weather Forecast

M. Verdecchia
University of L' Aquila
and
Parco Scientifico e Tecnologico d' Abruzzo

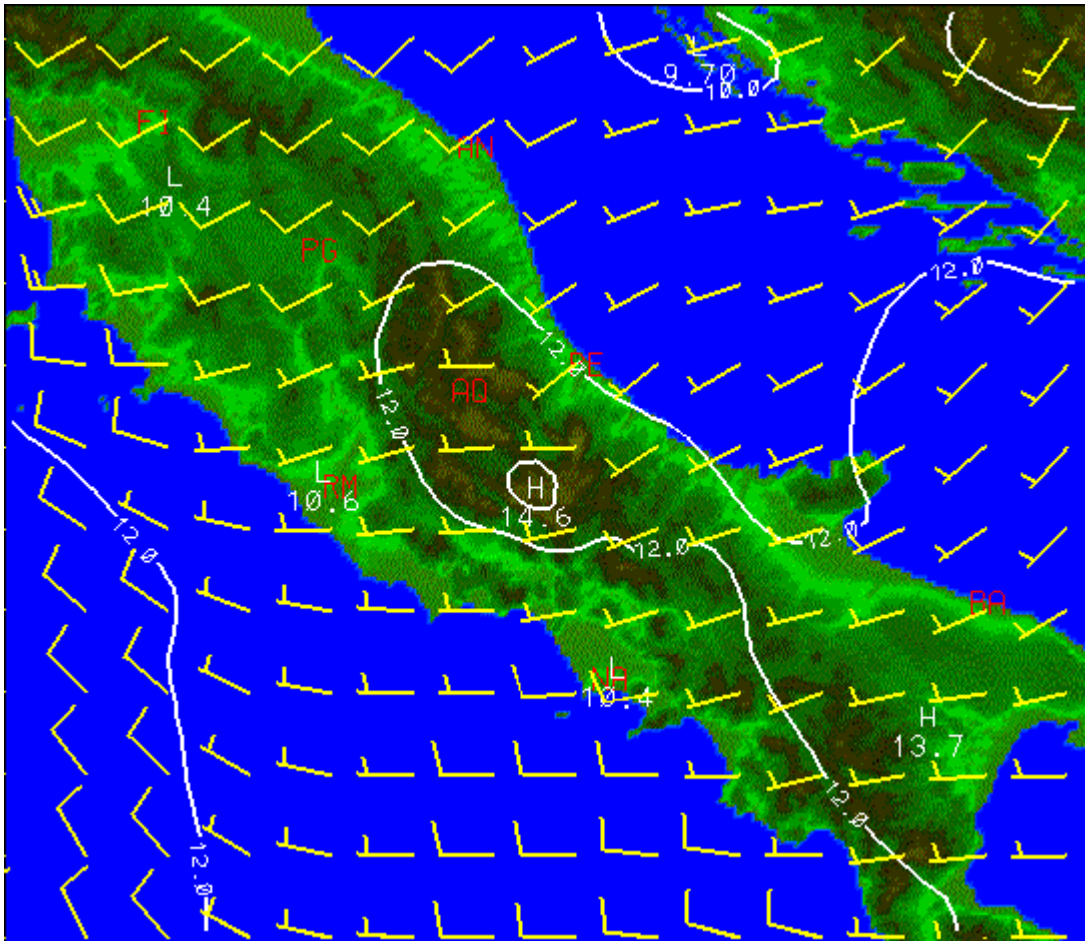
ore 20 di Martedi 11 APR



DEPARTMENT OF PHYSICS - UNIVERSITY OF L'AQUILA
in cooperation with
SCIENTIFIC AND TECHNOLOGY PARK OF ABRUZZO

Barbs of Horizontal Wind [m/s] - smooth=0
Pressure level = 850 mbar
(First nested domain: 9.0 Km resolution)

12 UTC of Monday 29 May 2000 - MM5 Real-Time Weather Forecast



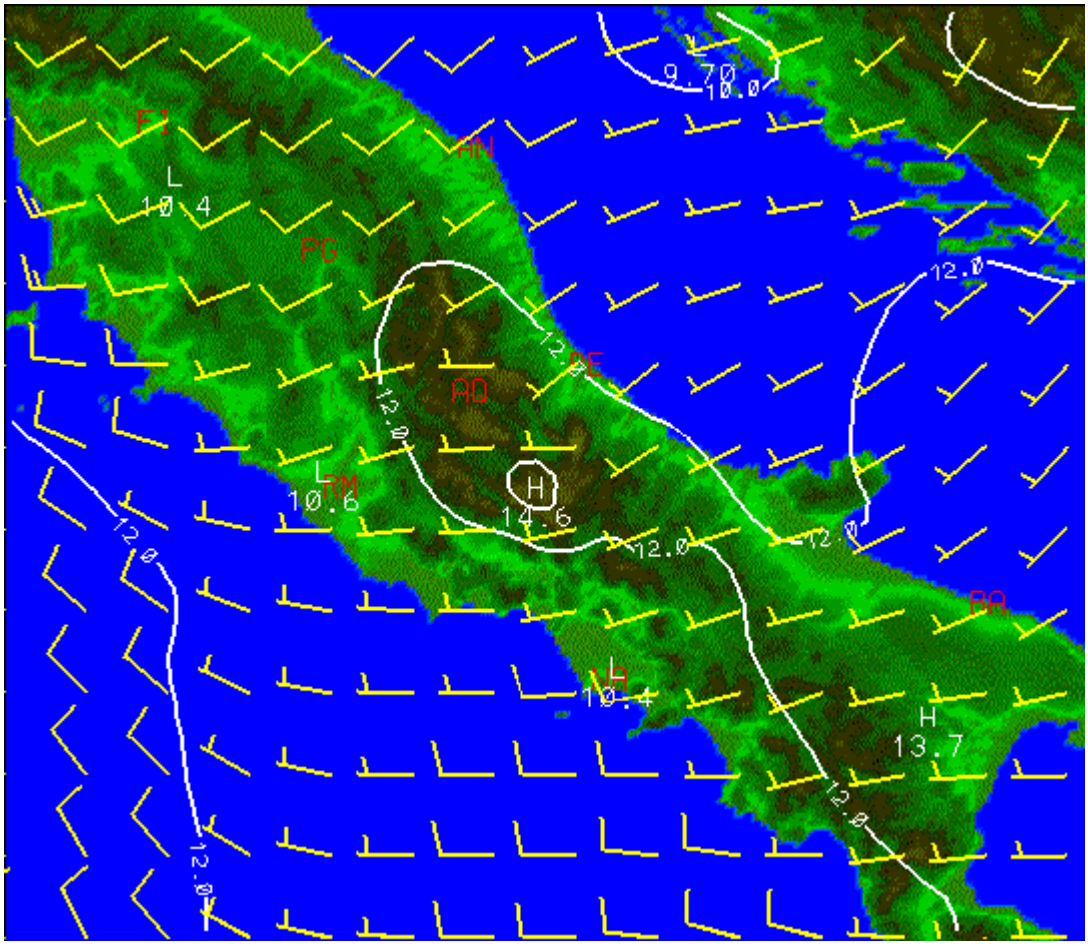
Click on the arrows to see previous or next images
Check this one for simpler but faster gif images
or click here for an animated view of the field

Automatically generated HTML code
Comments and feedback: Livio Bernardini

DEPARTMENT OF PHYSICS - UNIVERSITY OF L'AQUILA
in cooperation with
SCIENTIFIC AND TECHNOLOGY PARK OF ABRUZZO

Barbs of Horizontal Wind [m/s] - smooth=0
Pressure level = 850 mbar
(First nested domain: 9.0 Km resolution)

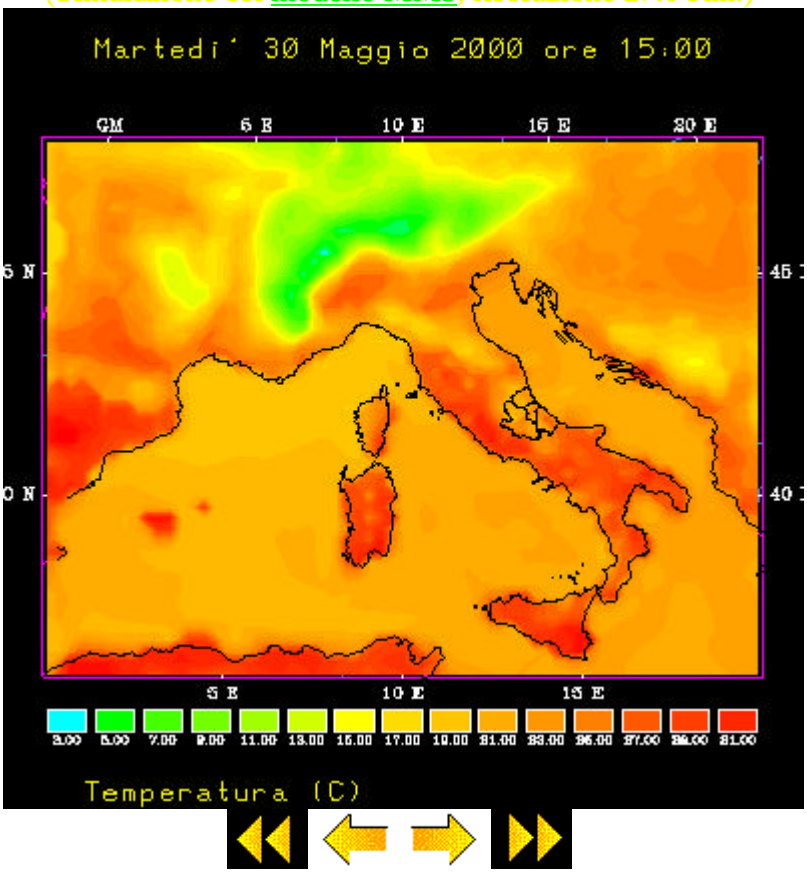
12 UTC of Monday 29 May 2000 - MM5 Real-Time Weather Forecast



Click on the arrows to see previous or next images
Check this one for simpler but faster gif images
or click here for an animated view of the field

Automatically generated HTML code
Comments and feedback: Livio Bernardini

(Simulazione col modello MM5, risoluzione 27.0 Km.)



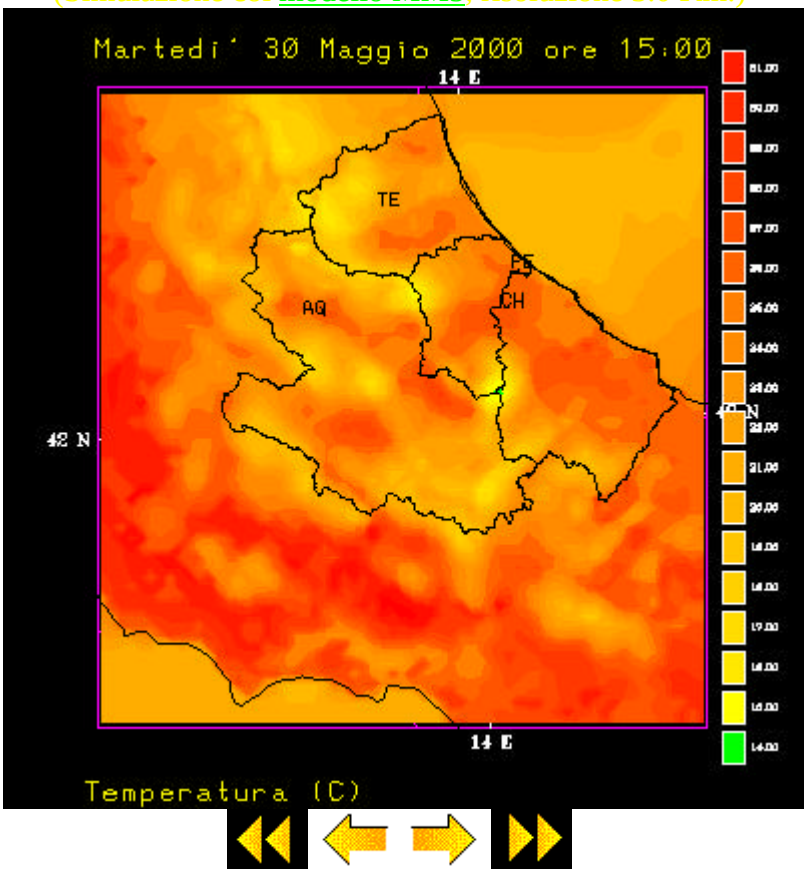
Clicca sulle frecce per avere le immagini precedenti/successive
Clicca sull'immagine per avere il dominio successivo
Clicca qui per le mappe animate
Clicca qui per la spiegazione del grafico

Automatically generated HTML code.
MM52web software, version 2.01 by: Marco Verdecchia, Tiziana Paolucci and Barbara Tomassetti

---

Pagina Principale

(Simulazione col modello MM5, risoluzione 3.0 Km.)



Clicca sulle frecce per avere le immagini precedenti/successive
Clicca sull'immagine per avere il dominio successivo
Clicca qui per le mappe animate
Clicca qui per la spiegazione del grafico

Automatically generated HTML code.
MM52web software, version 2.01 by: Marco Verdecchia, Tiziana Paolucci and Barbara Tomassetti

---

Pagina Principale

Heterogeneous Computing as a key feature for high performance computing. The GENESI-2 project and applications on environmental modeling

V. Rosato, F. Valentinotti
HPCN - ENEA

Heterogeneous Platform

We define heterogeneous a platform composed by the assembly of component platforms: :

- characterized by different architectures
- connected by a large bandwidth-low latency interconnect
- displaying a broad spectrum of machine granularities g_m

Recipe to implement a task on a heterogeneous system

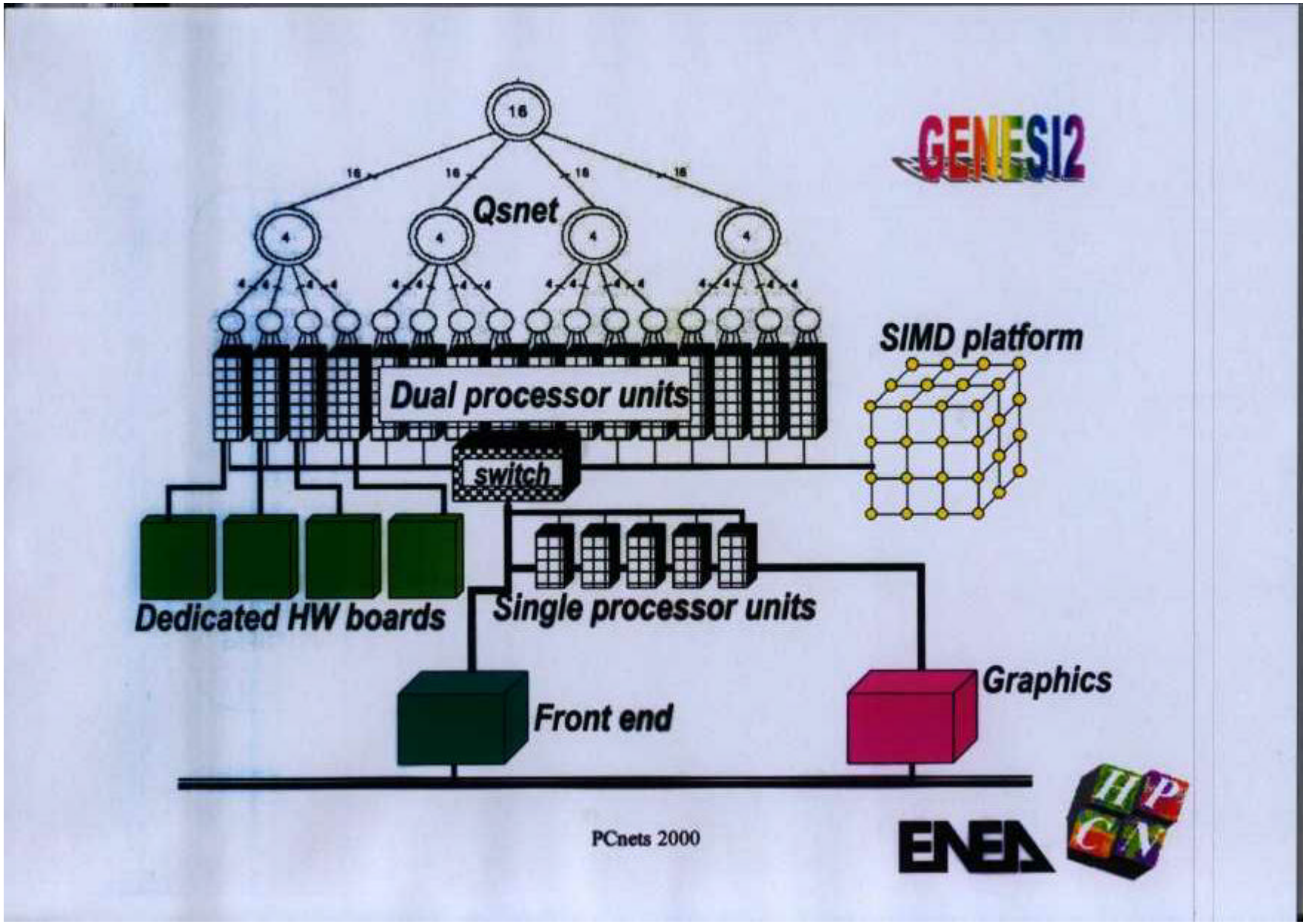
(1) consider the highest level of detail for the system and the task graphs

(2) ordinate the node tasks in descending order of computational complexity

(3) for each node of the task graph, select the system node whose architecture matches the task computational paradigm

(4) choose the node with the highest computational power which satisfies the condition

(5) assign the selected task to the selected node


16
16
16
16
16
Qsnet
4
4
4
4
GENESI2
SIMD platform
Dual processor units
switch
Dedicated HW boards
Single processor units
Graphics
Front end
PCnets 2000
ENEA
H P
C N

Technical details GENESI-2 docking unit

- 16 dual Alpha EV6.7 667 MHz, 1 GB RAM, 4MB L2 cache, 10 GB HD, Tsunami chipset (2.6 GB/sec internal bandwidth)
- 6 Compaq X1000 workstations (Alpha EV6.7 667 MHz, 256 MB RAM, 4MB L2 cache, 10 GB HD)
- QsNet 16 ways (200 MB/sec, 5μsec latency)
- 24-ways Fast Ethernet Switch with Gigabit up-link (service network)
- Linux RedHat xxx OS
- Compaq-Linux F90 and C compilers
- MPI-CH message passing library
- SCALAPACK mathematical library
- AFS (file system)
- LSF (load sharing)

PCnets 2000

Scientific Applications

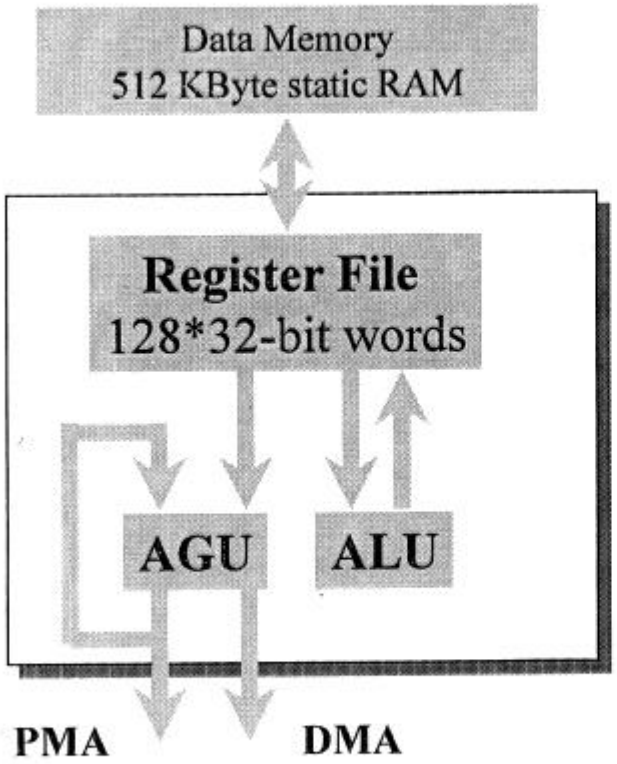
(1) Fluido-dynamics (oceanography, weather forecast, climate modeling)
(2) Combustion studies (Fluent)
(3) Electromagnetics (mobile telephone technology, em dosimetry)
(4) Computational Materials Science (CP, TBMD)
(5) Biotechnology (Gromacs, InsightII)
(6) Genomics (use of dedicated devices)
(7) Computational Astrophysics (n-body, tree-codes)
(8) Structural analysis (FEM)
(9) HW co-design (design of the dedicated devices)
(10) City traffic modeling (link to virtual reality engines)

APEmille architecture

A. Bartoloni

Bartolini

HW: processing board – T1000



for both T1000 and J1000

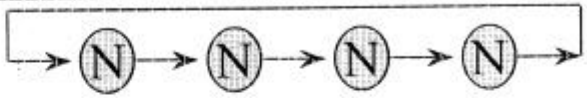
Highlights:

- An ASIC component used as:
  - Control Unit
  - Integer ALU
  - Address Generation Unit
- Pipelined Architecture
- **VLIW** Very Long Instruction Word
- Synchronous Operations at 66 MHz

26/4/2000

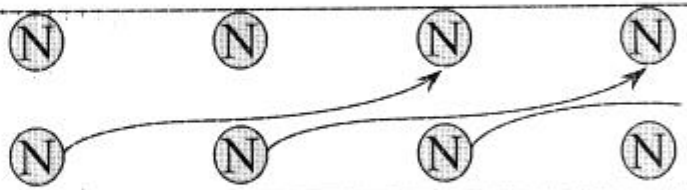
HW: synchronous network – C1000

next-neighbor communication



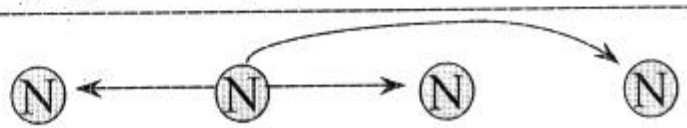
In order to improve the communications features of APEmille the basic 6-nearest neighbours approach has been generalized to a rigid addressing scheme of arbitrary distance.

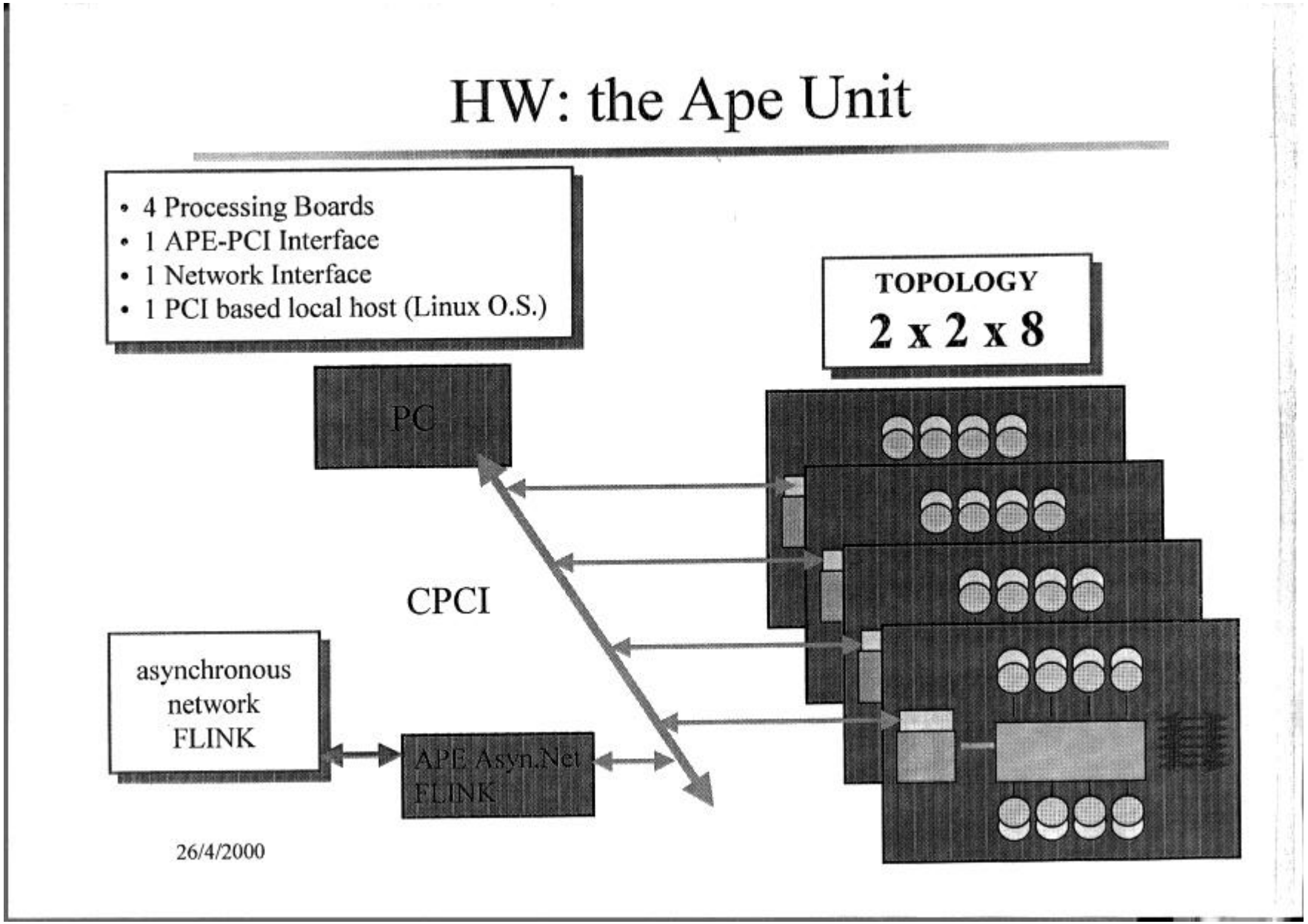
automatic routing for homogeneous transfers



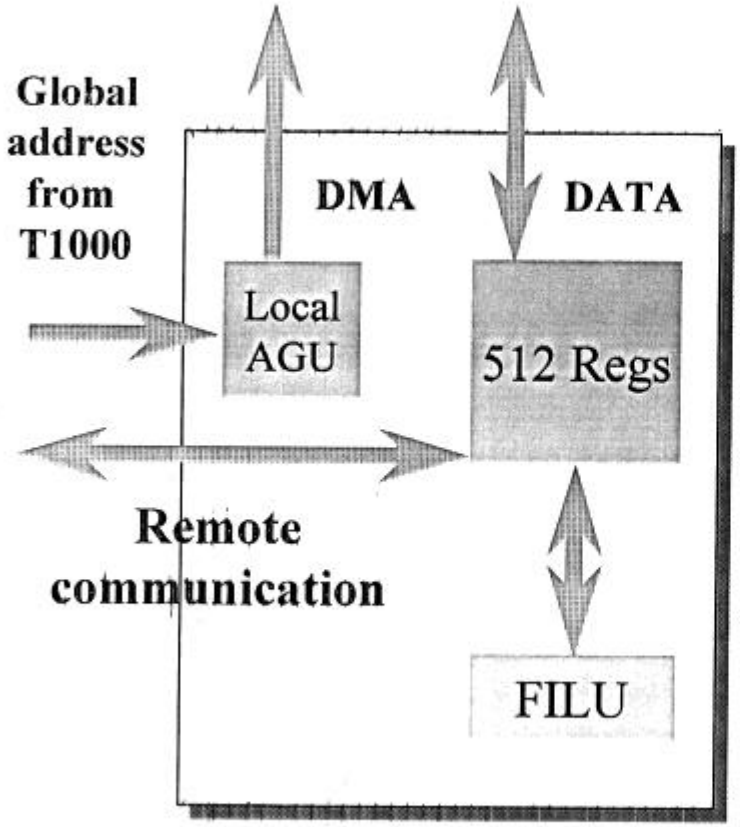
The possibility is given to broadcast a data word from one node to the rest of the machine, or to broadcast the content of a plane of processors along the direction ortogonal to it.

broadcast




HW: the Ape Unit
• 4 Processing Boards
• 1 APE-PCI Interface
• 1 Network Interface
• 1 PCI based local host (Linux O.S.)
TOPOLOGY
2 x 2 x 8
PC
CPCI
asynchronous network FLINK
APE Asyn.Net FLINK
26/4/2000

HW: processing board – J1000



Data types:

- **integer**, 32 bits (2)
- **real**, single precision IEEE (133 / 8)
- **double**, double precision IEEE (133 / 10)
- **complex**, in single precision (528 / 10)
- **vector$_2$**, two-elements real (266 / 10)

Highlights:

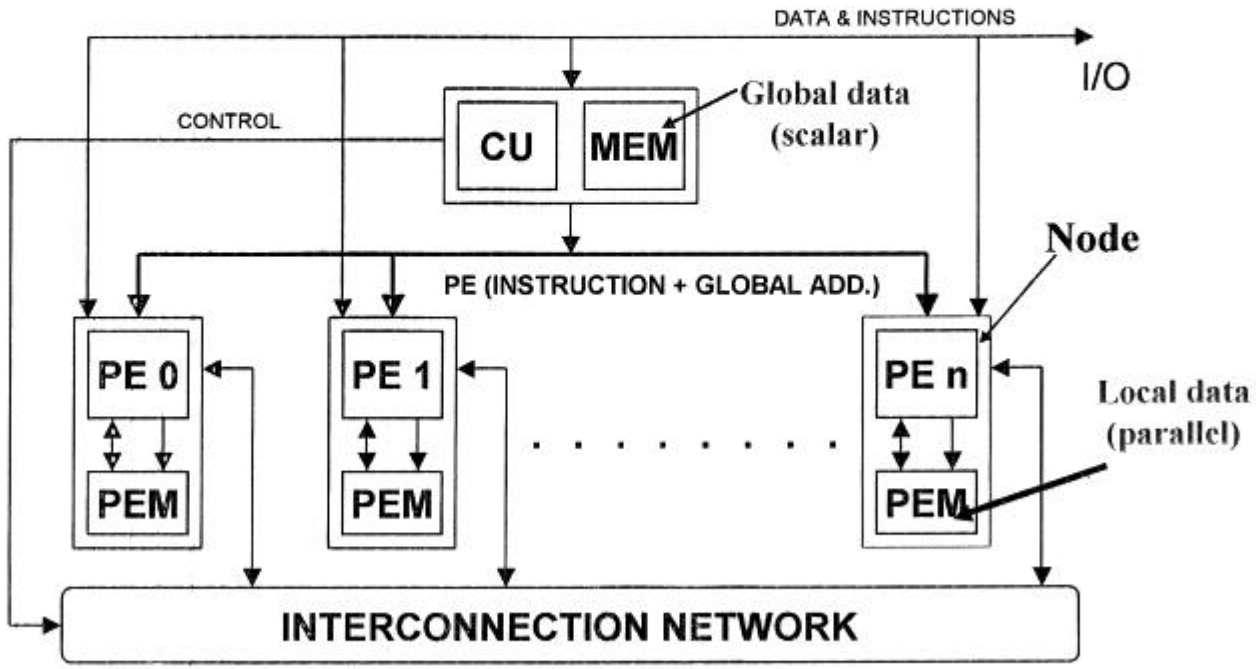
- one cycle **A*B+C** for all types
- pipelined Architecture
- **VLIW** Very Long Instruction Word
- synchronous Operations at 66MHz
- 528 Mflops peak performance
- 8-32MB SDRAM local data memory

26/4/2000

Multiple SIMAMD architecture

- Multiple users can partition the machine or
- one program can activate Multiple SIMD threads

Multiple Addressing thanks to the local integer capability

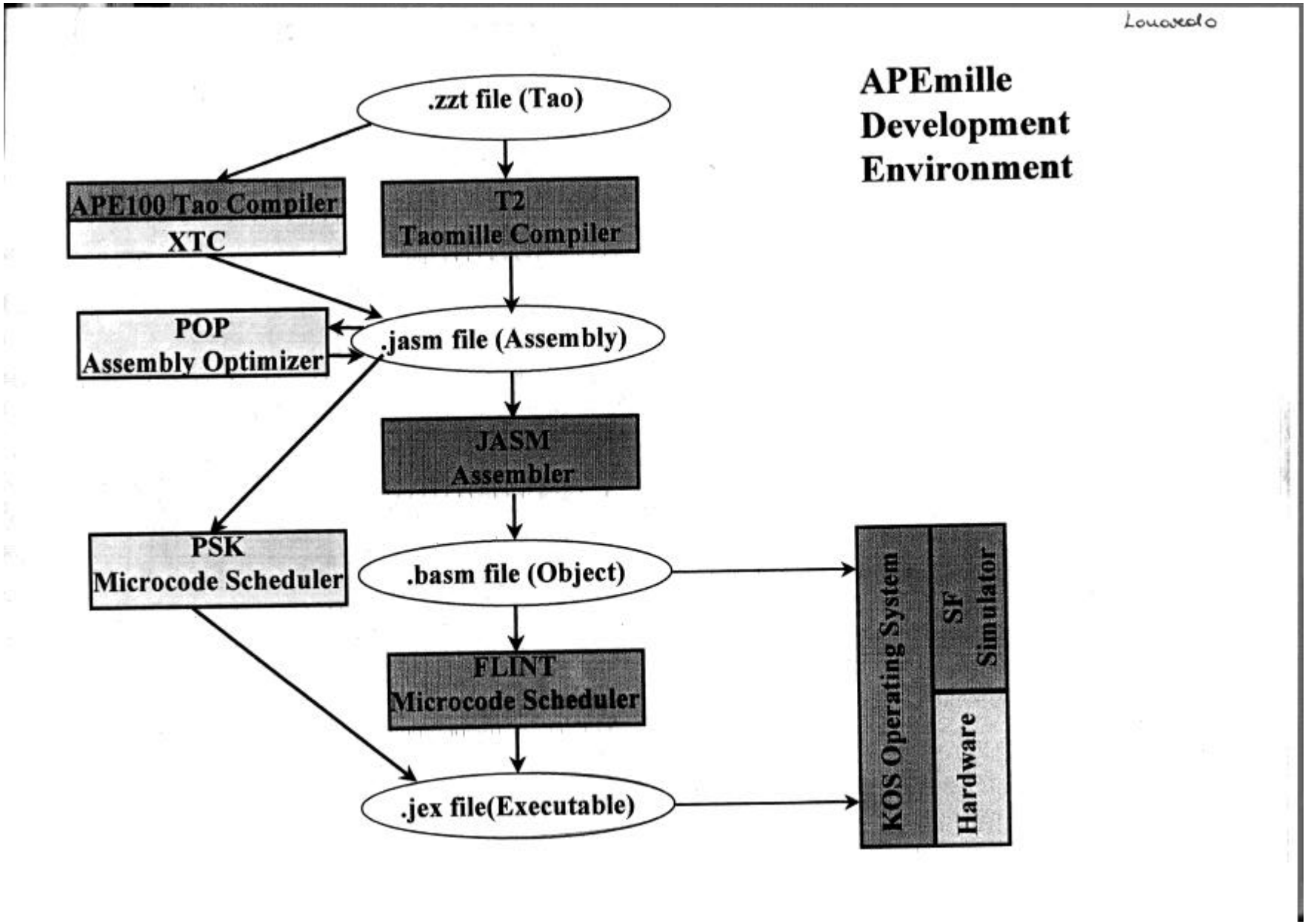


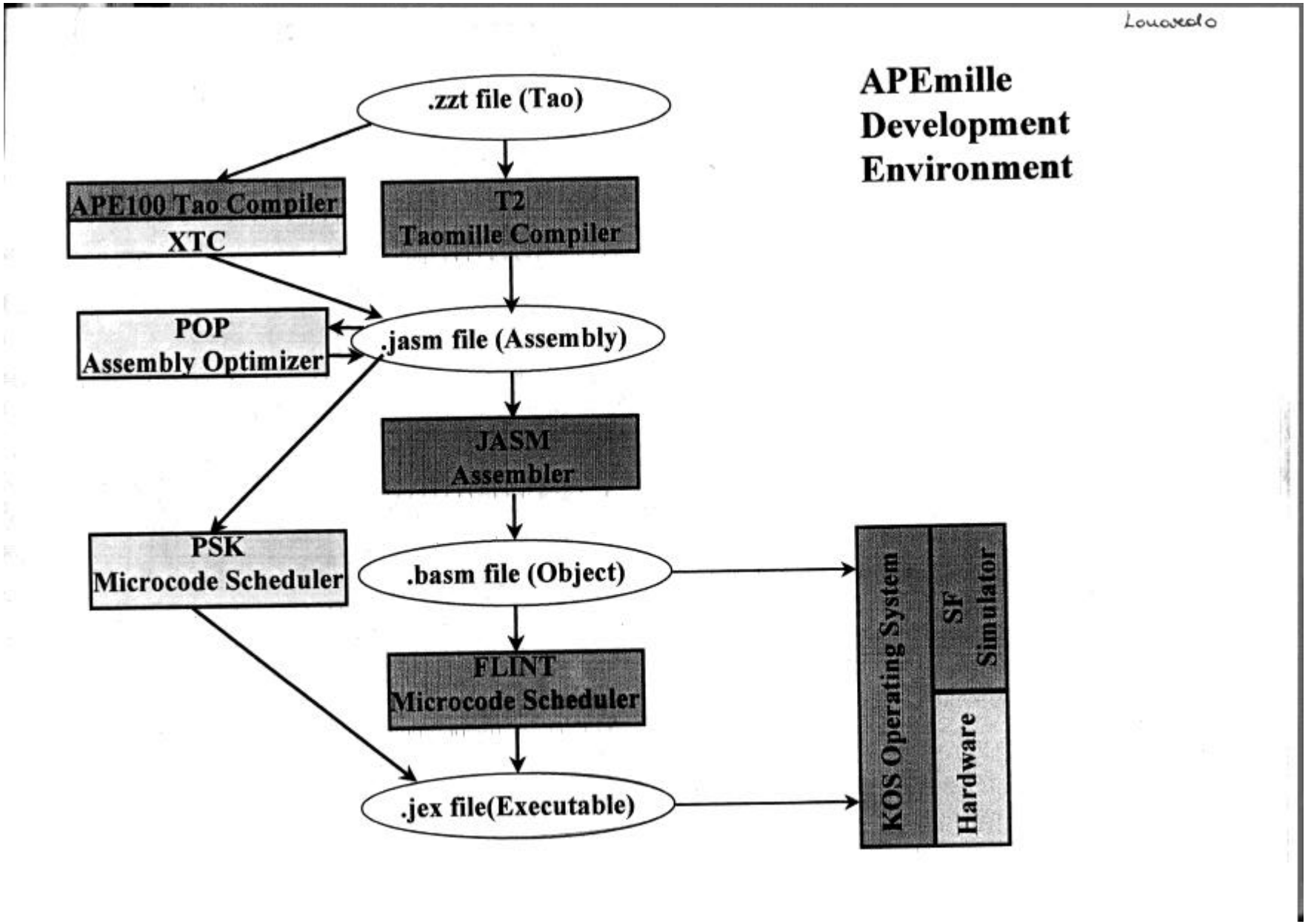
26/4/2000

support to put to "sleep" some nodes while the others remain active

TAO compiler chain and functional simulator

A. Lonardo


Lonardo
APEmille
Development
Environment
.zzt file (Tao)
APE100 Tao Compiler
XTC
T2
Taomille Compiler
POP
Assembly Optimizer
.jasm file (Assembly)
JASM
Assembler
PSK
Microcode Scheduler
.basm file (Object)
FLINT
Microcode Scheduler
.jex file(Executable)
KOS Operating System
SF
Simulator
Hardware


Lonardo
APEmille
Development
Environment
.zzt file (Tao)
APE100 Tao Compiler
XTC
T2
Taomille Compiler
POP
Assembly Optimizer
.jasm file (Assembly)
JASM
Assembler
PSK
Microcode Scheduler
.basm file (Object)
FLINT
Microcode Scheduler
.jex file(Executable)
KOS Operating System
SF
Simulator
Hardware

T2 TAOmille compiler

Based on Zz dynamic parser, supports new TAOmille data types:
- **localint** *(integers allocated on arithmetic processors)*
- **double** *(double precision floating point)*
- **vector** *(couple of single precision floating point)*

and the local addressing (i.e. the use of localints in the addressing of arrays).

Performs a series of optimizations on the generated assembly (CSE, "normalization", copy propagation, costant folding, dead code removal).

Useful compilation flags: `t2 -fcdm filename`

-f, --constant-fold optimize constants
-c, --cc-load optimize conjugate loads
-d, --dead-code-remove removal of dead branches
-m, --ct-matrix-index only compile-time indexing for matrix

Documentation:
http://www.ifh.de/computing/nic/Docu_e.html (for the TAO language)
http://apemaia.roma1.infn.it/intranet/taomille.html (for the APEmille extensions to the TAO)

T2 TAOmille compiler

Based on Zz dynamic parser, supports new TAOmille data types:
- **localint** *(integers allocated on arithmetic processors)*
- **double** *(double precision floating point)*
- **vector** *(couple of single precision floating point)*

and the local addressing (i.e. the use of localints in the addressing of arrays).

Performs a series of optimizations on the generated assembly (CSE, "normalization", copy propagation, costant folding, dead code removal).

Useful compilation flags: `t2 -fcdm filename`

-f, --constant-fold optimize constants
-c, --cc-load optimize conjugate loads
-d, --dead-code-remove removal of dead branches
-m, --ct-matrix-index only compile-time indexing for matrix

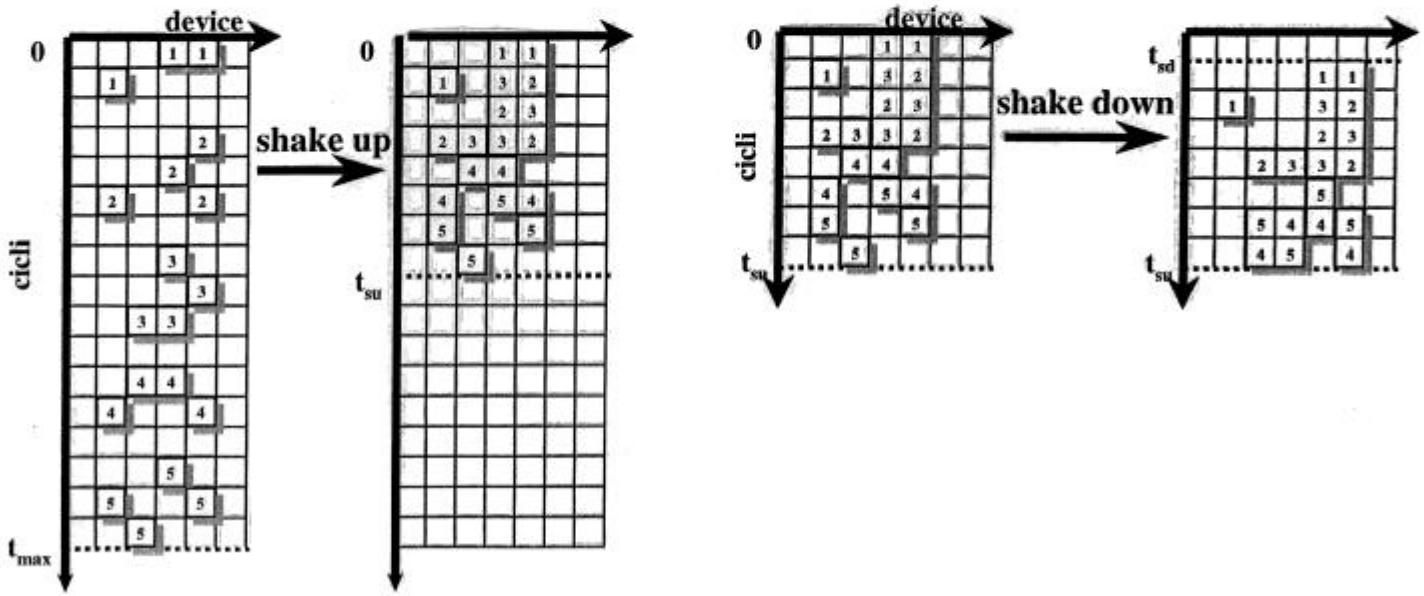
Documentation:
http://www.ifh.de/computing/nic/Docu_e.html (for the TAO language)
http://apemaia.roma1.infn.it/intranet/taomille.html (for the APEmille extensions to the TAO)

FLINT

Reads a .basm file and produces an APEmille microcode program according to an optimizing scheduling algorythm (SHAKER) which tries to minimize the code length (and thus the program execution time). It performs register allocation.

Useful options:

-s : instruction are temporized without shaking phase (for debugging)



POP

parses an APEmille assembly file and performs two optimizations: "normalization" and unnecessary move instructions removal. This module is now included in PSK.

PSK

Reads an APEmille assembly file and generates an executable file (VLIW microcode). Uses an optimizing scheduling algorythm (SHAKER) to shorten program length (and to speed-up program execution), register allocation is also performed.

Useful options: `psk -U n filename`

-U n Set Shaker Upper Bound to n (Default n=32).

XTC

Uses APE100 Tao compiler with modified libraries to generate an assembly file that is coverted into APEmille assembly.

Supports APE100 Tao syntax plus the double data type.

Does not perform optimizations on generated assembly code (relies on POP) but compiles correctly very large programs.

Useful options: `xtc -t -j filename`

-t execute taocomp before translation
-j execute jasm upon translation

Profiler and Performances

A. Petricola

Petricola

KPROF

Kprof is a static profiler for the APE micro-code.

To use Kprof, first we have to invoke FLINT with the '-l' option to produce the '.lld' (labels linkig dump) file.

Usage: kprof [-t] [-s threshold] [-p threshold] [-v] filename

-t: Print ZZT code corresponding to each label if the jasm file exists.
-s: Print statistic only for block with weighted efficiency less than threshold.
-p: Print statistic only for block with weighted efficiency more than threshold.
-v: Verbose mode: print out various info.

The '.jex' and '.lld' files must be present in the same directory.

Output format ('.pfr' file)

LABEL = Label name of the block
BEGINCYCLE = Block Starting cycle
LENGTH = Block Lenght
wE = Weighted Eficiency
absE = Absolute Efficiency
DFLOP = Total FLOP of Double operation
SFLOP = Total FLOP of Single operation
MIP = Total MIP of Integer operation
JIO = Total Jmille IO operation
Rs = Ratio (SFLOP+MIP) / N_MEMOP_SINGLE
Rd = Ratio DFLOP / N_MEMOP_DOUBLE
Dummy_MIN = Dummy Memory IN operation
Dummy_MOUT = Dummy Memory OUT operation

LABEL	BEGINCYCLE	LENGTH	wE	absE	DFLOP	SFLOP	MIP	CONV	JIO	Rs	Rd	MDummy_IN	MDummy_OUT
*4133	0x00001066	162	0.93%	0.23%	0	3	0	0	3	Na	Na		
*4134	0x00001108	60	0.00%	0.00%	0	0	0	0	0	Na	Na		
*4136	0x00001144	1322	9.08%	1.81%	0	142	49	0	18	19.10	Na	3	1
*4138	0x0000166e	176	9.38%	3.91%	0	44	11	0	10	Na	Na		
*4139	0x0000171e	26	0.00%	0.00%	0	0	0	0	0	Na	Na		
*4140	0x00001738	178	9.27%	3.86%	0	44	11	0	10	Na	Na		
*4141	0x000017ea	28	0.00%	0.00%	0	0	0	0	0	Na	Na		
*4142	0x00001806	176	9.38%	3.91%	0	44	11	0	10	Na	Na		
*4143	0x000018b6	18	0.00%	0.00%	0	0	0	0	0	Na	Na		
*4144	0x000018c8	176	9.38%	3.91%	0	44	11	0	10	Na	Na		
*4145	0x00001978	18	0.00%	0.00%	0	0	0	0	0	Na	Na		
*4137	0x0000198a	18	0.00%	0.00%	0	0	0	0	0	Na	Na		
*4135	0x0000199c	24	0.00%	0.00%	0	0	0	0	1	Na	Na		
*4146	0x000019b4	78	0.00%	0.00%	0	0	0	0	0	Na	Na		
*4148	0x00001a02	142	5.63%	0.79%	0	2	7	0	5	Na	Na		
*4149	0x00001a90	18	0.00%	0.00%	0	0	0	0	0	Na	Na		

Speedup of APE programs

H. Simma
Desy

Simma

Speeding up APEmille Programs

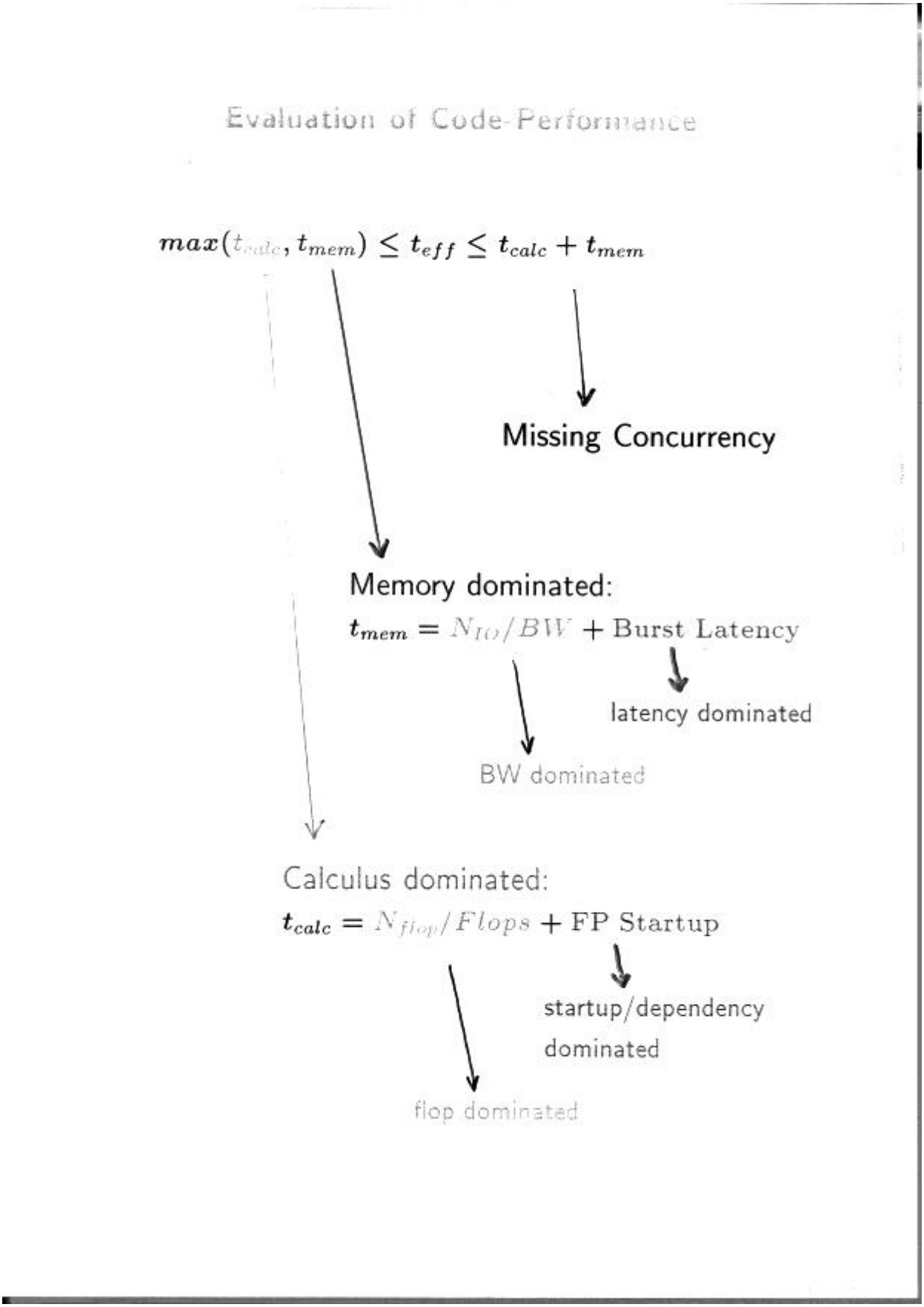
APEmille ...

- is a Micro-Programmed Architecture:
  assembler instruction = micro-program
- has many concurrent functional units, e.g.
  - flow-control
  - Address Generation Unit (addresses, constants)
  - Arithmetic Logical Unit (integer)
  - FP-Unit
  - Memory-Access
- is heavily pipelined

t_{eff} =	t_{exec} +	t_{start}	
3	1	2	integer
4	1	3	localint add
10	1	9	localint mul
12	1	11	complex norm
12+2+N	N	12+2	complex read access
8+N	N	8	complex write access

Main optimization tasks:

- understand characteristics of code
  - calculus vs. memory access
  - data dependencies
- maximise concurrency + hide t_{start}
  - unrolling
  - extract/replace
  - re-order memory access
  - pre-load physical registers
  - pre-calculate addresses
- minimise dependencies
  - re-order arithmetics (e.g. tree summation)
  - vectorisation


Evaluation of Code-Performance
$max(t_{calc}, t_{mem}) \leq t_{eff} \leq t_{calc} + t_{mem}$
Missing Concurrency
Memory dominated:
$t_{mem} = N_{IO}/BW$ + Burst Latency
latency dominated
BW dominated
Calculus dominated:
$t_{calc} = N_{flop}/Flops$ + FP Startup
startup/dependency dominated
flop dominated

TAO Tricks

- `/for i = 0 to N { ... }`
- `extract` *matrix* `from` *array*
- Declaration of physical registers:
  `physreg` *FP-type varname*
- Assignment complex-from-real:
  `c = ( r1, r2 )` (NOT `c.re = ...`)
- Complex Conjugation "on the fly":
  r = c[i]~
- Latency Optimization for local memory access:
  `r = local(a[i] )`
- Remote Communications with arbitrary distance:
  `set remote` *myneighbour* = [2 , 0 , 27]
  r = a[i + myneighbour]

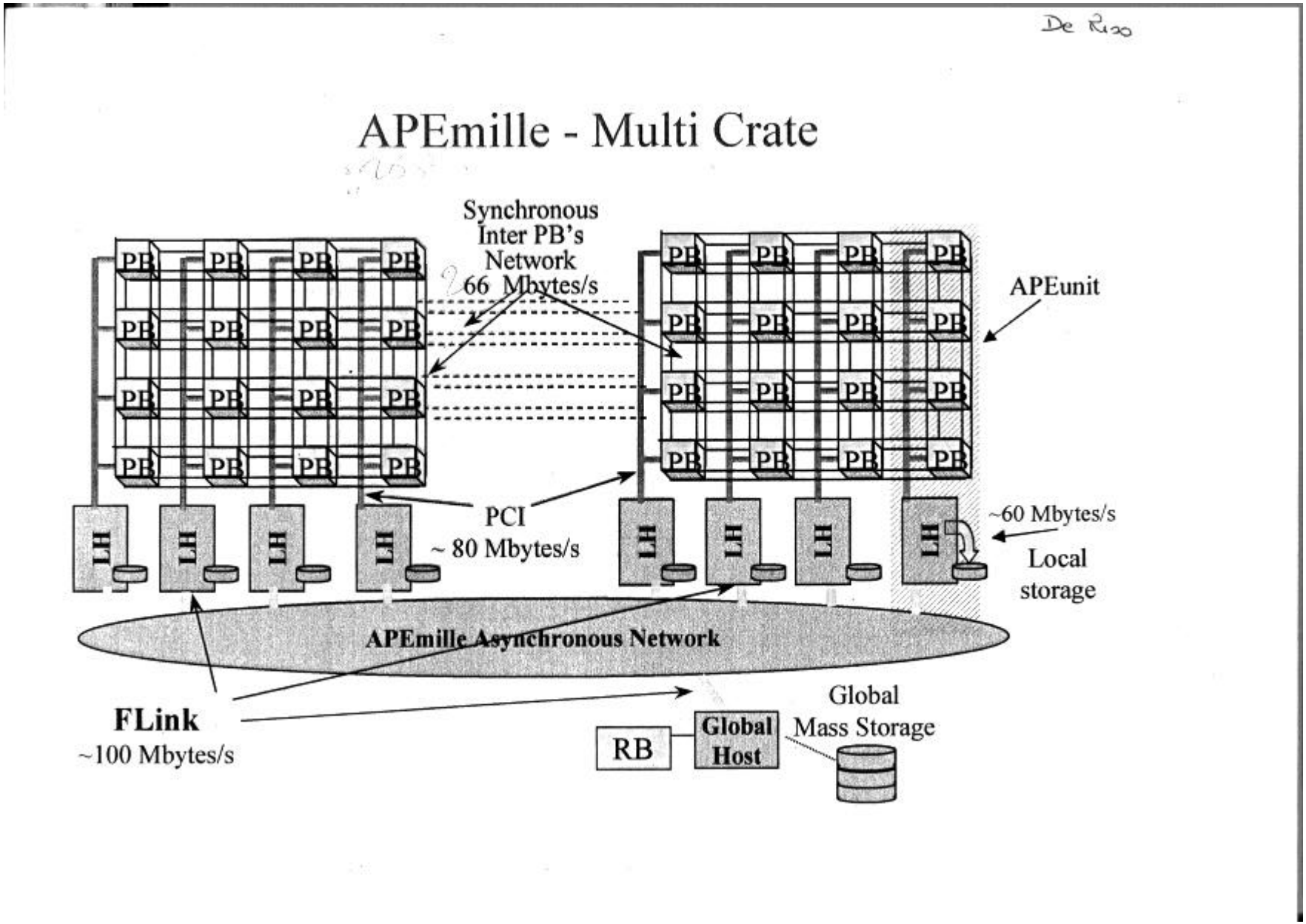
Summary

APEmille codes can be tuned to similar efficiency as on APE100, but ...

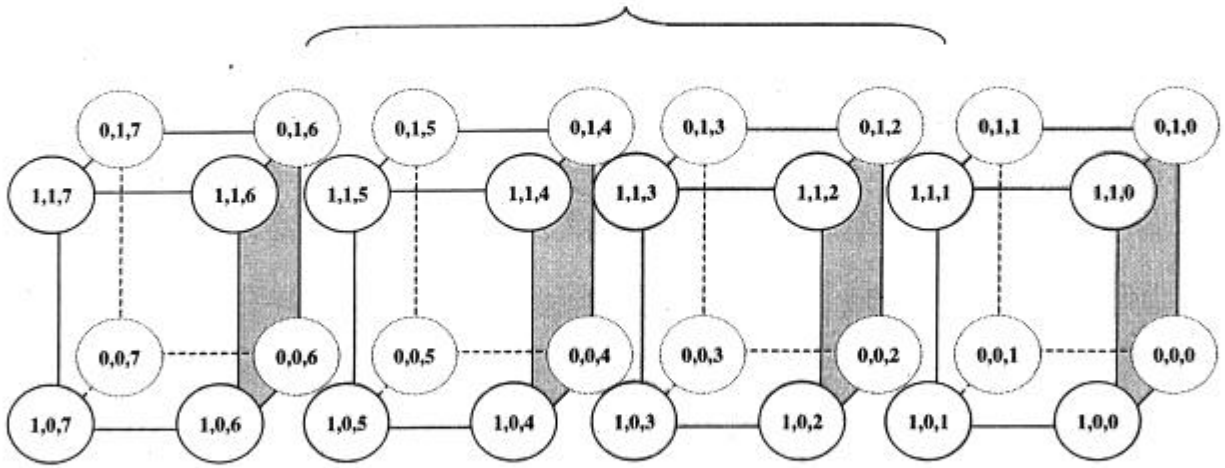
- might be slightly harder
- needs learning somewhat different tricks
  - − memory latencies larger
  - \+ more registers available
  - \+ custom communications
- compiler chain needs further fine tuning

APEmille topology

P. de Riso


De Riso
APEmille - Multi Crate
Synchronous
Inter PB's
Network
66 Mbytes/s
APEunit
PB
LH
PCI
~ 80 Mbytes/s
~60 Mbytes/s
Local
storage
APEmille Asynchronous Network
FLink
~100 Mbytes/s
RB
Global
Host
Global
Mass Storage

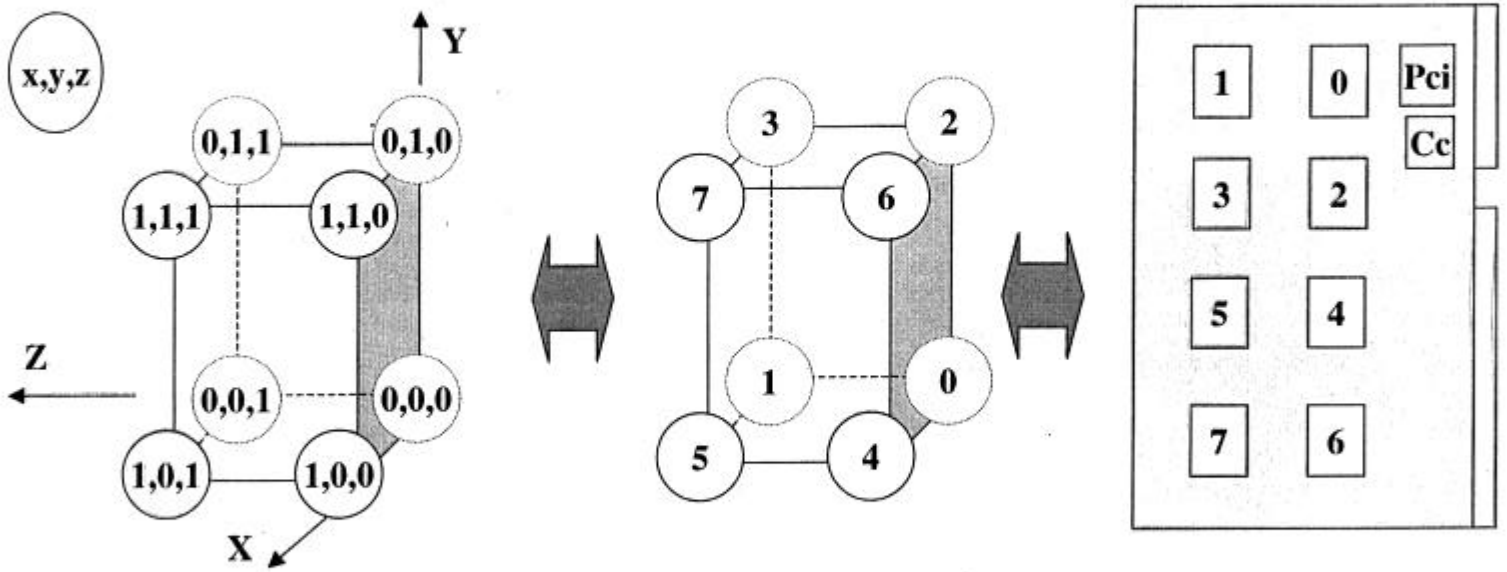
APEUnit 0



PB 3 **PB 2** **PB 1** **PB 0**

Single Unit: 2x2x8

Processing Board: 8 nodes - 2x2x2



Example of Machine topologies

1 PB .... 8 nodes 2x2x2

2 PBs .... 16 nodes 4x2x2

4 PBs (ApeUnit) 32 nodes 2x2x8
(8x2x2)

8 PBs (2 ApeUnit) 64 nodes 4x2x8

16 PBs (4 Ape Unit = 1 Crate) 128 nodes 2x8x8
(8x2x8
8x8x2)

APEmille operating system and run

D. Rossetti

Rossetti

APEmille system SW

Available items

- Loader/Server Daemon (krun,kunitd)
- SF simulator (libSf)
- Tools (kdeb,monitor,testmem)

People involved

Davide Rossetti Andrea Michelotti
Norbert Pashedag Giuseppe Sacco
Sandro Tassa

APEmille system SW - loader

KRUN is the Linux executable which acts as the APEmille loader and run-time Operating System.

It is able:

- To load **.JEX** programs onto the APEmille machine, then executing it.
- To run **.BASM** files using the SF simulator.
- To serve all the APE-style system services.
- To catch, decode and clear all the APE run-time exceptions.

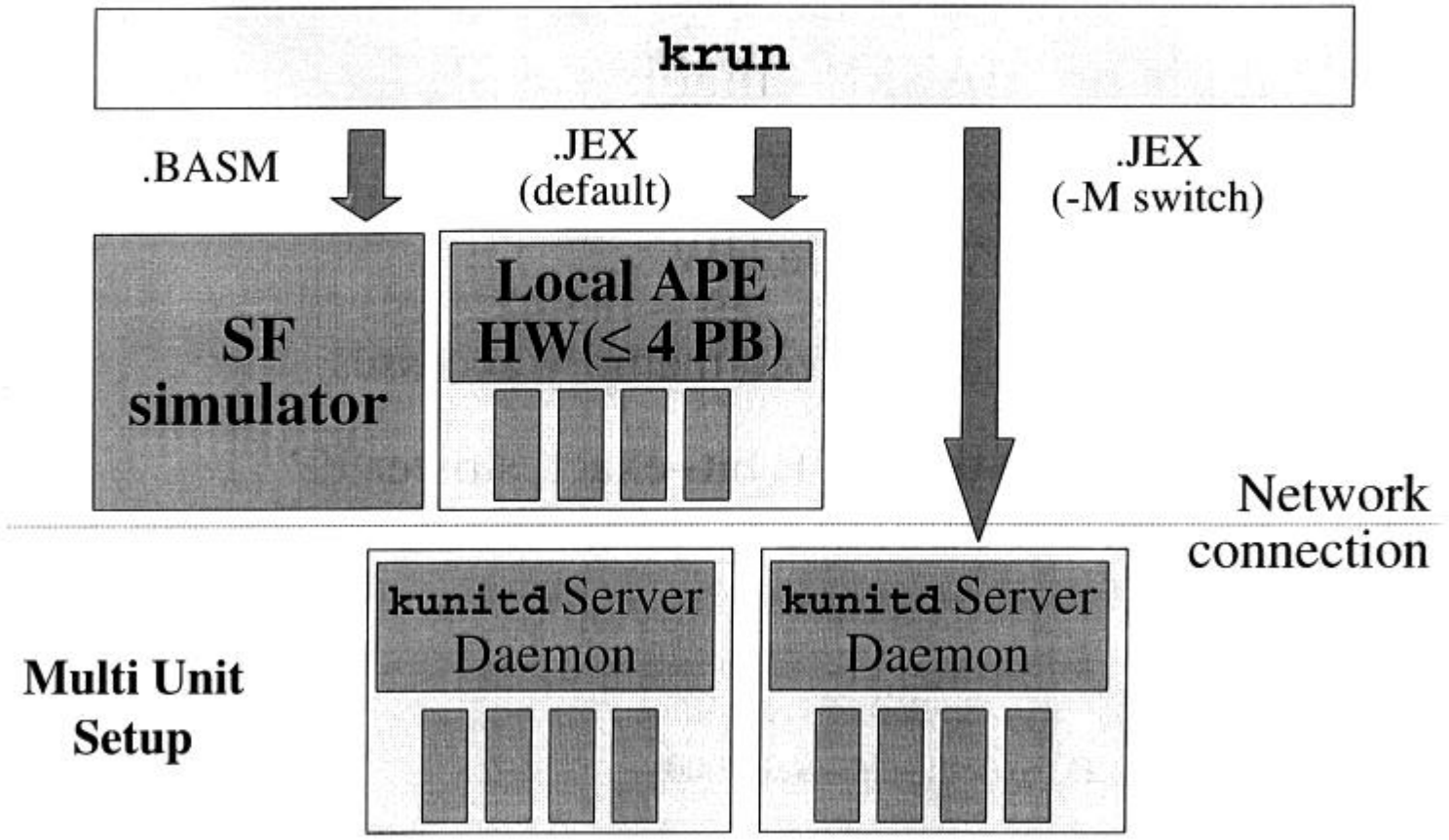
APEmille system SW - loader

It is usually wrapped inside a shell script.

KRUN supports a huge number of switches and environment settings. `KRUN -h` lists all of them.

- Most of them are really **low level**, like those to access HW registers (`--set-tz-mask`).
- Some are **mid level**, like choosing machine geometry (`-G` switch) or controlling debug tracing.
- Others are of **common use** (`-H` to issue a reset before running or `-O` to choose topology).

APEmille system SW - loader



APEmille system SW - SF

Using KRUN on .BASM - that is via SF simulator - can be tremendously useful:

- No need of the real APEmille HW.
- Easy catching of un-initialized memory accesses.
- Choice between a APE bit-exact, slower FP engine (`SF_USE_FILU`) and the built-in, faster math ops.

APEmille system SW - SF

SF is dynamically configurable as of:

- Number of simulated nodes (`-G` switch). From a single `node`, a `board` (2x2x2 nodes), a `crate` (2x8x8) up to a `rack` (4x8x8). **CPU hogs!!**
- Memory sizes (`SF_JANE_DM_SIZE`, `SF_TZ_DM_SIZE`). **Memory hogs are here!!**
- Exception masking (`SF_MASK_EXC`).

Single node is useful: it is faster and ligther for the Linux CPU so it is suitable for *memory intensive* or *production level* runs. More nodes may be necessary to test *communication logic*.

Quantum Montecarlo

S. Fantoni
SISSA, Trieste

PC-Nets 2000
LNGS - INFN

Quantum Monte Carlo

S. Fantoni (SISSA-Trieste)

- Brief review of QMC (in condensed matter)
- Motivations of QMC for Nuclear Astrophysics
- Auxiliary field QMC
- Application to neutron matter & neutron droplets.

DMC is one way of simulating the diffusion equation on a computer.

- ψ and not ψ^2 corresponds to the density of diffusing objects
- these objects are "walkers" that move in a $3N$-dim. space. The "walkers" do not interact
- $V(R)$ is an external influence $\Rightarrow$ creation or absorption probabilities per unit time.
- Model a continuous wf as the average density of an ensemble of "walkers"

Mixed Energy $E(\tau) = \dfrac{\int dR\, \Psi^*(R,0)\, \hat{H}\, e^{-\hat{H}\tau}\, \Psi(R,0)}{\int dR\, \Psi^*(R,0)\, e^{-\hat{H}\tau}\, \Psi(R,0)}$ $\tau \to \infty$

$\Big\|$

Growth energy $E_g(\tau) \equiv \dfrac{e^{-E_0\tau} \int dR\, \Psi^*(R,0)\, \Psi(R,\tau)}{\int dR\, \Psi^*(R,0)\, \Psi(R,\tau)}$

The normalization $a_0 e^{-E_0\tau}$ is proportional to the number of "walkers" present.

QMC for nuclear systems

Motivations:

- Important processes occurring in extreme conditions in stellar and primordial environments (not exp. reproducible)
  - weak proton capture reactions (sun)
  - radiative neutron capture by neutron rich nuclei (synthesis of heavy elements → r-process)
  - three-body reactions ($3\,^4He \rightarrow {}^{12}C$)
  - Matter with very large densities of up 10^{15} g/cc and beyond (compact white dwarfs, neutron stars)
  - EOS and neutrino and photon opacities (evolution of supernovae & neutron stars)
  - Pion condensation in neutron matter
  - superfluidity of neutron matter

The Algorithm

1. Sample $|R,S\rangle$ initial walkers from $|\langle\Psi_T|R,S\rangle|^2$ using Metropolis Monte Carlo.

2. We propagate in the usual diffusion Monte Carlo way with a drifted gaussian for half a time step.

3. For each walker, we diagonalize the potential matrix.

4. Loop over the eigenvectors, sampling the corresponding Hubbard-Stratonovich variable and update the spinors for half a time step. We use the expectation value of $\langle\Psi_T|\vec{\sigma}_i|R,S\rangle$ to introduce approximate importance sampling of the Hubbard-Stratonovich variables.

5. Repeat 2, 3, and 4 in the opposite order to produce a reversible propagator to lower the time step error.

6. Combine all weight factors and evaluate new value of $\langle\Psi_T|R,S\rangle$. If the real part is less than 0 enforce constrained path by dropping the walker.

7. Evaluate the averages of $\langle \Psi_T | R, S \rangle$, and $\langle \Psi_T | H | R, S \rangle$ to calculate the energy.

8. Repeat as necessary.

New perspectives in Quantum Monte Carlo methods for many fermion systems

F. Pederiva, M. H. Kalos
Department of Physics and INFN
University of Trento

THE SIGN PROBLEM

Diffusion Monte Carlo (DMC) calculations for fermionic systems suffer of the so-called sign problem .

Diffusion Monte Carlo projects the state with lowest eigenvalue out of a trial function which in general is a superposition of eigenstates of a given many body problem.

$$\Psi_T = \sum_i c_n \phi_n$$

The projection is obtained propagating the trial function in imaginary time

$$e^{-(\tau \hat{H} - E_0)} \Psi_T = \sum_n c_n e^{-\tau (E_n - E_0)} \phi_n \underset{\tau \to \infty}{\sim} c_0 \phi_0$$

When we deal with fermions, we seek the antisymmetric solution with lowest eigenvalue. Usually in DMC one computes the energy as

$$E_0^A = \frac{< \Psi_T^A | \hat{H} \phi_0 >}{< \Psi_T^A | \phi_0 >} = \frac{\int dR \phi_0 \frac{\hat{H}\Psi_T^A}{\Psi_T^A} \Psi_T^A}{\int dR \phi_0 \Psi_T^A}$$

Because Ψ_T^A is antisymmetric, the components along symmetric ϕ_n disappear, and we are left with the correct antisymmetric ground state, provided we choose a proper E_0^A.

What happens to the variance of E_0^A ?

$$\sigma^2(E_0^A) = | < \hat{H}^2 > - < \hat{H} >^2 |$$

This expression contains terms like

$$\frac{\int dR \phi_0 \frac{\left(\hat{H}\Psi_T^A\right)^2}{\Psi_T^A} \Psi_T^A}{\int dR \phi_0 \Psi_T^A}$$

which diverge exponentially with τ , because they contain components along the symmetric states

TEST CASES

- He atom triplet state

  wave functions with the correct nodes, but modified.
  Eigenvalue: -2.175227(58) a.u. (exact: -2.175229 a.u.)

- Be atom

  wave function from optimized VMC calculations both for the bosonic and fermionic ground state.
  Eigenvalue: -14.78(30) a.u. (exact: -14.667 a.u.)

- 7 Free Fermions

  trial functions: $\phi_s = 1$, ϕ_A = Slater determinant of modified plane waves (in order to start with the wrong nodal structure).
  Eigenvalue: 2.91285 ± 0.00049 (analytic result: 2.912712)

- 14 ^{3}He atoms

  Trial functions:

  $$\phi_S = \exp\left[\prod_{i<j}\left(\frac{b_j}{r_ij}\right)^5\right] \quad \phi_A = \phi_S||e^{-i\mathbf{k}\cdot\mathbf{r_i}}||$$

  Eigenvalues:

b_j	E_0
1.15	-2.256±0.001
1.145	-2.2585±0.017
1.135	-2.2581 ±0.020

  - Runs are found to be stable for a total imaginary time >> 1000 decay times
  - Long runs may show a sudden qualitative change in stability, with a strong reduction of the fluctutaion in energy and improved linearity in the growth of the denominator ("superstability"). We are attempting to obtain superstability in a systematic way using a second stage importance sampling.

CONCLUSIONS

- Correlated dynamics, cancellation and different importance sampling for plus–minus walkers give a walkers distribution with a stable overlap with an antisymmetric solution of the many body Schrödinger equation.

- It works for physically meaningful systems.

- We need more efficient sampling (understanding the origin of superstability is one possible way).

The C++ language for APEmille

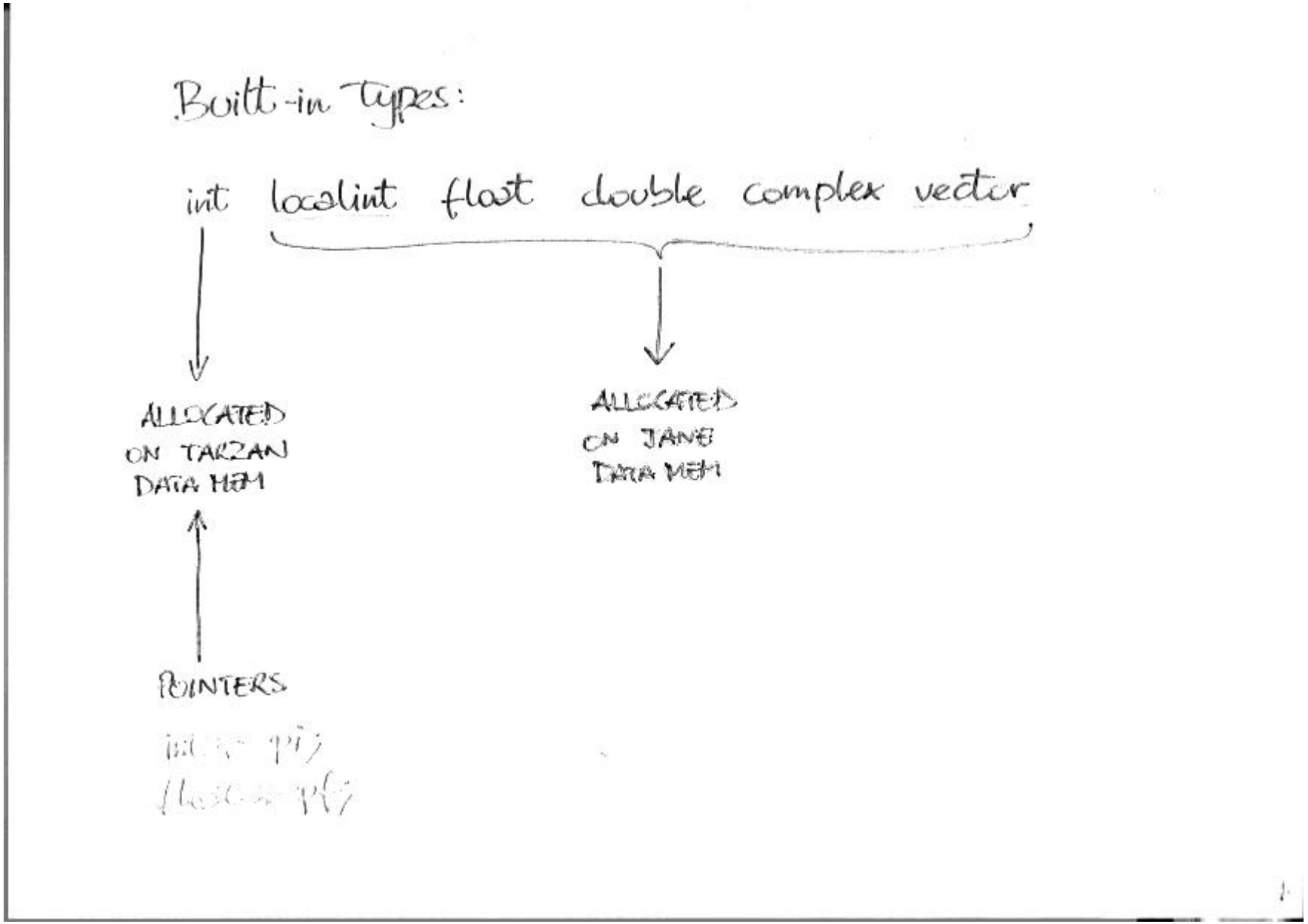
A. Lonardo, E. Panizzi, B. Proietti
Department of Electric Engineering
University of L' Aquila
and
INFN-Roma I

C / C++

- standard language
- object oriented
- emerging language for future physics codes

- no use of classes to implement parallelism
- GOAL: as few new instructions as possible.

INTENDED FOR: G4 PROGRAMMERS
TAO PROGRAMMERS


Built-in Types:
int
localint float double complex vector
ALLOCATED ON TARZAN DATA MEM
ALLOCATED ON JANE DATA MEM
POINTERS

Conditions:

- global: · based on comparison of Taxon values

```
int i,j;
if (i<j)
{//...
}
```

· aggregate of comparisons of Jane values: { all, any, none

```
float x,y;
if (all (x<y))
...
```

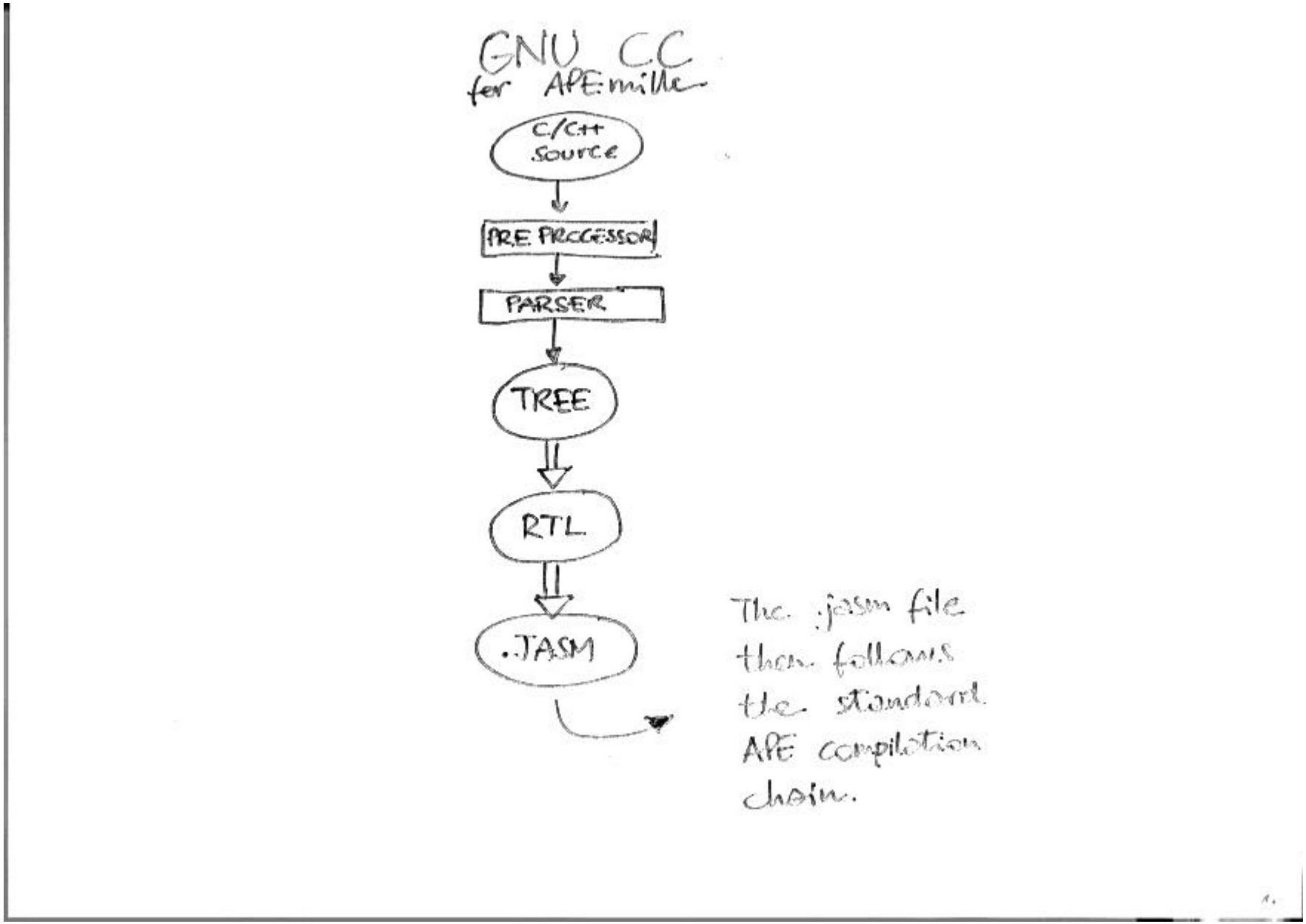
- local: · based on local (JANE) condition: where.

```
float x,y;
where (x<y)
{ //...
}
elsewhere
{ //...
}
```

{no jump}

APE like remote communications:

```
#define NODE_UP 0x03000000
complex v[10], c;
c = v[5+node_up];
...
```


GNU CC
for APE:mille
C/C++ source
PRE PROCESSOR
PARSER
TREE
RTL
.JASM
The .jasm file then follows the standard APE compilation chain.

GAMMA on Gigabit Ethernet

G. Ciaccio
CASPUR - University of' Roma "La Sapienza"
and
DISI - University of Genova

GAMMA:
Genoa Active Message MAchine

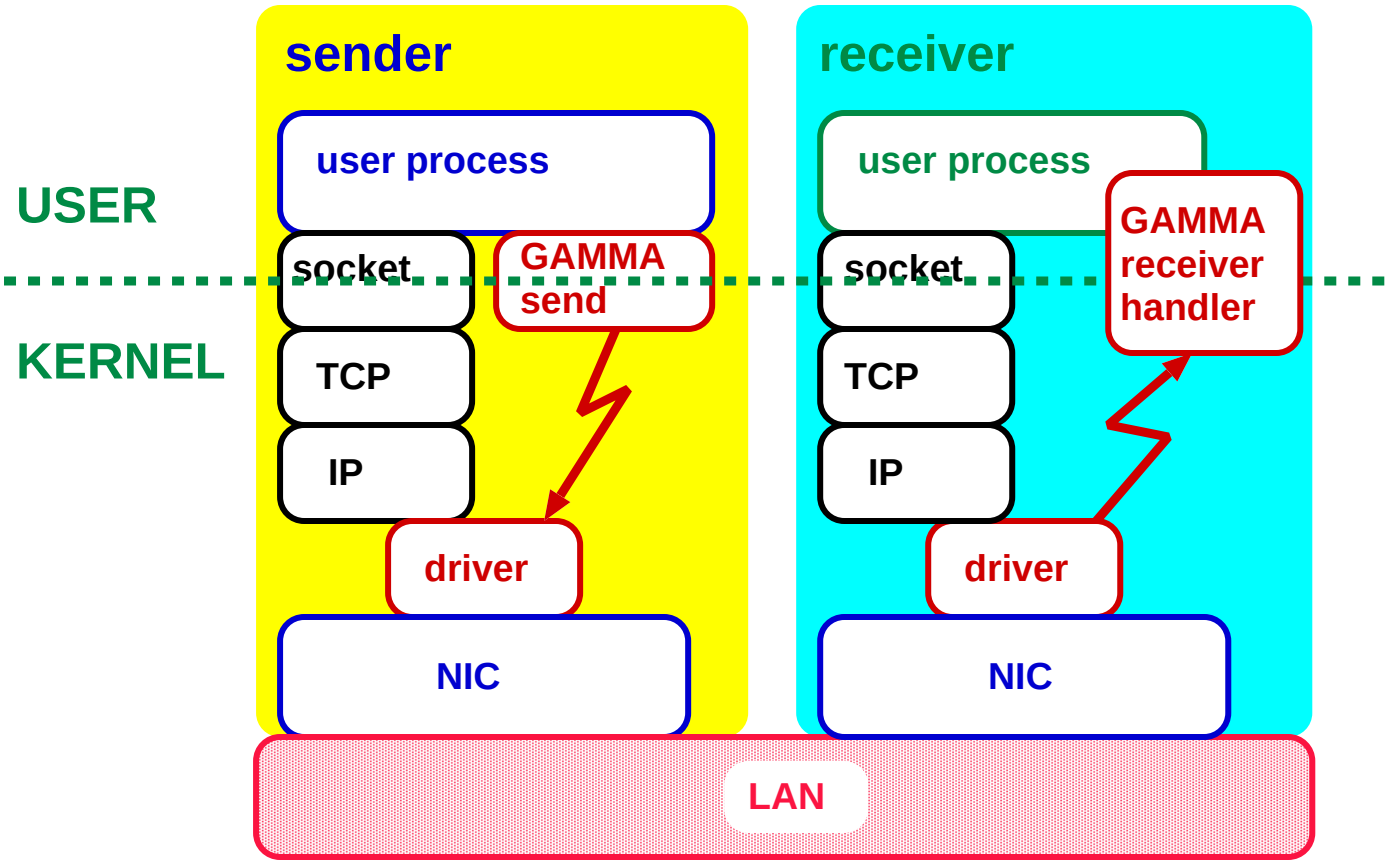
A very efficient messaging system for Linux clusters

Goals: Low latency, high throughput. Multi-task environment.

Strategy: Reduced protocol complexity, monolithic structure, reduced data copies, credit-based flow control.
Reliable up to LAN faults.

Kernel-level implementation:
enhanced Linux device driver + small programming library.
Support for IP is preserved.

MPI/GAMMA: an industry-standard API for message passing.
A low-overhead MPI for GAMMA is now available



GAMMA Driver + "lightweight" system calls:
Point-To-Point and Broadcast communications

GAMMA user-level library:
Collective Communications (SPMD parallel model)

Preserve standard environment (multi-task, TCP/IP), simply add a new kind of network abstraction

Supported Fast Ethernet NICs:
DEC DE500, 3COM 3c905.
Experimental: Intel EtherExpress Pro/100

Supported Gigabit Ethernet NICs:
Packet Engines GNIC-II (no longer in production)
Alteon AceNIC (Netgear GA620, 3COM 3c985)

DISCUSSION

Gigabit Ethernet does not improve over latency, compared to Fast Ethernet: a substantial fraction of latency still depends on the specific NIC

TCP/IP: a severe throughput bottleneck on Gigabit Ethernet, regardless of the NIC and PC

GAMMA + old-fashioned PCs: throughput bottleneck is the CPU-memory bus, due to memory-to-memory copies

GAMMA + modern PCs: throughput bottleneck is the NIC itself, due to message fragmentation (NIC overhead)

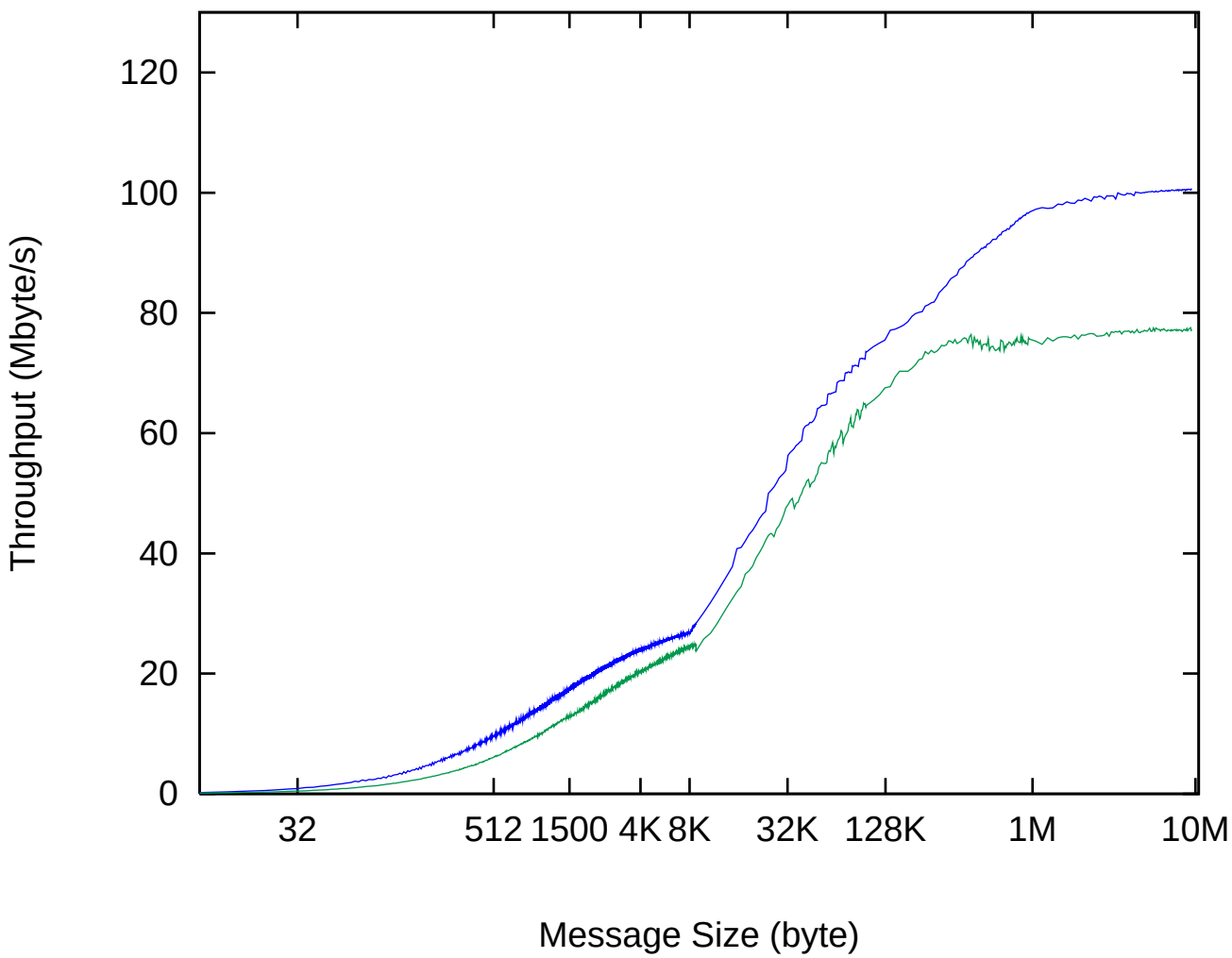
‘‘Jumbo frames’’ might help. Good NICs could help even more!

Ping-Pong (end-to-end throughput)

AMD Athlon 500 MHz, Alteon AceNIC, back-to-back

gamma_send_fast_flowctl(), Jumbo frames: latency 31.4 usec

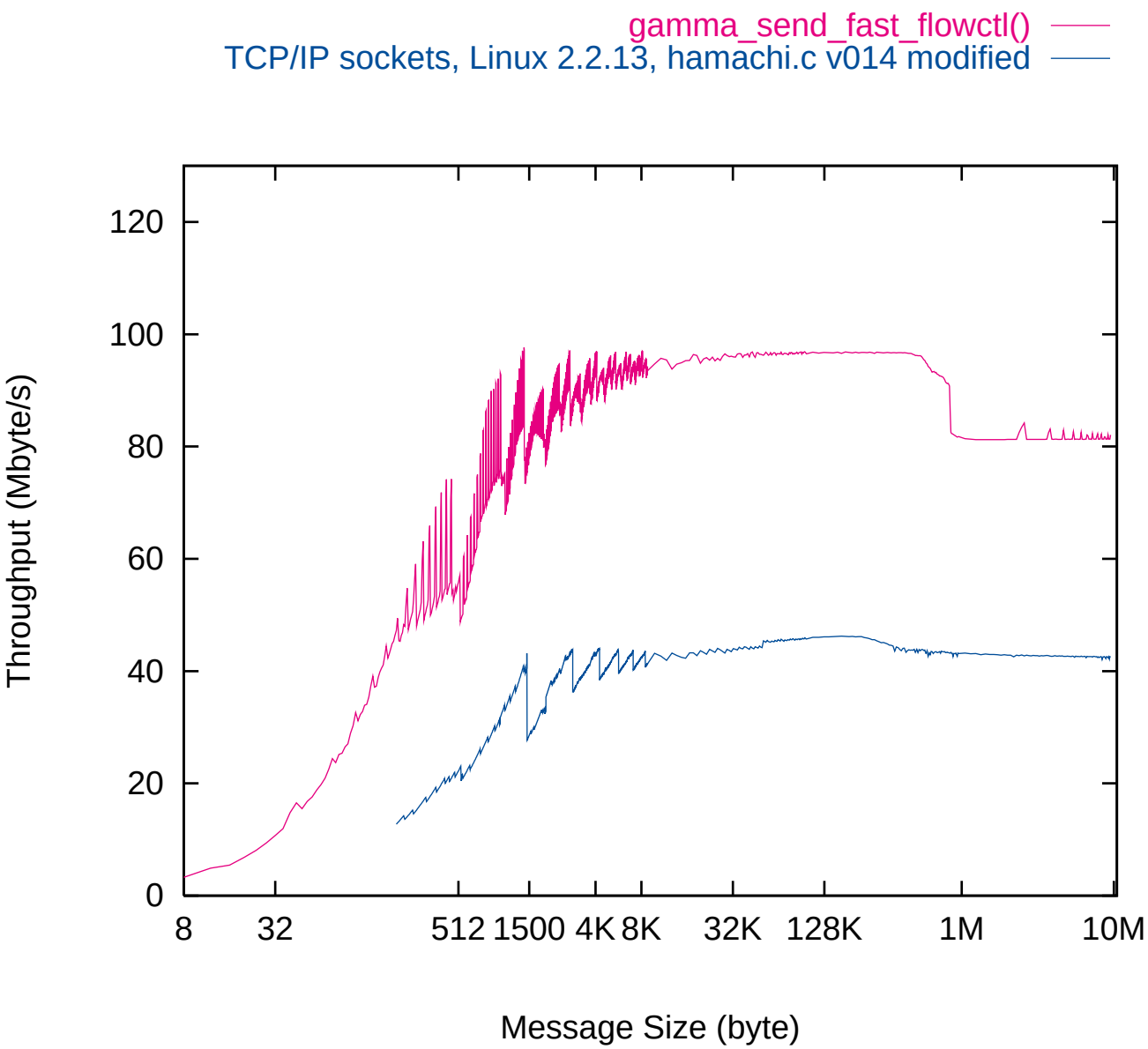
TCP/IP, Linux 2.2.12, acenic.c v038, Jumbo frames: latency 68.2 usec



Unidirectional stream (transmission throughput)

Pentium II 450 MHz, Packet Engines GNIC-II NIC

Intel Express Gigabit switch



Fermion Algorithms

K. Jansen
CERN
TH Division

I give an overview on the two most challenging numerical problems we are facing presently in studyingquantum chromodynamics on the lattice.The first difficulty concerns dynamical fermions while the second is connected with simulations of actions having an exact lattice chiral symmetry.
Dynamical fermions are already a long standing problem.However,it was only relatively recently pointed out that besides their very large computational cost, they also have a conceptual problem:it is well known that accidental (almost)zero modes of the lattice Dirac operator can appear in practical simulations. In these situations, standard algorithms like hybrid Monte Carlo face a dilemma: either these algorithms never accept such configurations in which case one may question the ergodicity of these algorithms, or, if they accept such configurations, physical fermionic observables that are proportional to the inverse of the (almost) zero modes receive exceptionally large values, spoiling the statistical sample. In the talk, the method of the PHMC algorithm as proposed by Frezzotti and Jansen is explained, which cures this conceptual problem by treating the (almost) zero modes exactly through a correction factor.
Exact chiral symmetry is very important in QCD. As became evident only recently, a lattice chiral symmetry can be realized by lattice Dirac operators that obey the so-called Ginsparg-Wilson (GW) relation. In the talk, the numerical treatment of one solution to the GW relation, Neuberger's operator, is discussed and the practical example of the study of spontaneous chiral symmetry breaking in quenched lattice QCD is given. It is demonstrated that, because of the complicated form of Neuberger's operator, its numerical evaluation becomes very computer time demanding with a cost that is comparable to dynamical fermion simulations themselves.Therefore the theoretical and conceptual advantage of such operators have to be weighed against the numerical effort before physical problems are to be addressed.

Difficulty # 3

nature/physicists like symmetries

$\Leftarrow$ we can break them

$\Rightarrow$ many properties of models in physics follow from symmetries (or from breaking them)

also QCD has a symmetry: chiral symmetry

however, naive lattice QCD *does not have chiral symmetry*

it took us 20 years to find out how chiral symmetry can be implemented on the lattice

$$M \to D = 1 - \frac{M}{\sqrt{M^\dagger M}}$$

complicated new matrix through the $\sqrt{}$

matrix D is evaluated by

- computing lowest and largest eigenvalues of $M^\dagger M$ (by using a CG method, Simma's favourite toy to test the APE's)

- construct a polynomial $P_n(M^\dagger M)$ (take a Chebyshev polynomial tailored in the range $[\lambda_{\min}, \lambda_{\max}]$)

the new matrix D is complicated:

to compute physical observables like

$$\langle \bar{\Psi}_x \Psi_x \rangle = \mathrm{Tr} D^{-1}$$

linear solver: in *each* iteration we have to evaluate the Chebyshev polynomial $P_n(M^\dagger M)$

$\Rightarrow$ cost=$n_{\mathrm{conf}} \times n_{\mathrm{iter}} \times n$

with the degree n of the polynomial $\mathrm{O}(100)$
$\Rightarrow$ cost of a quenched simulation is similar to
a full QCD simulation

collision of

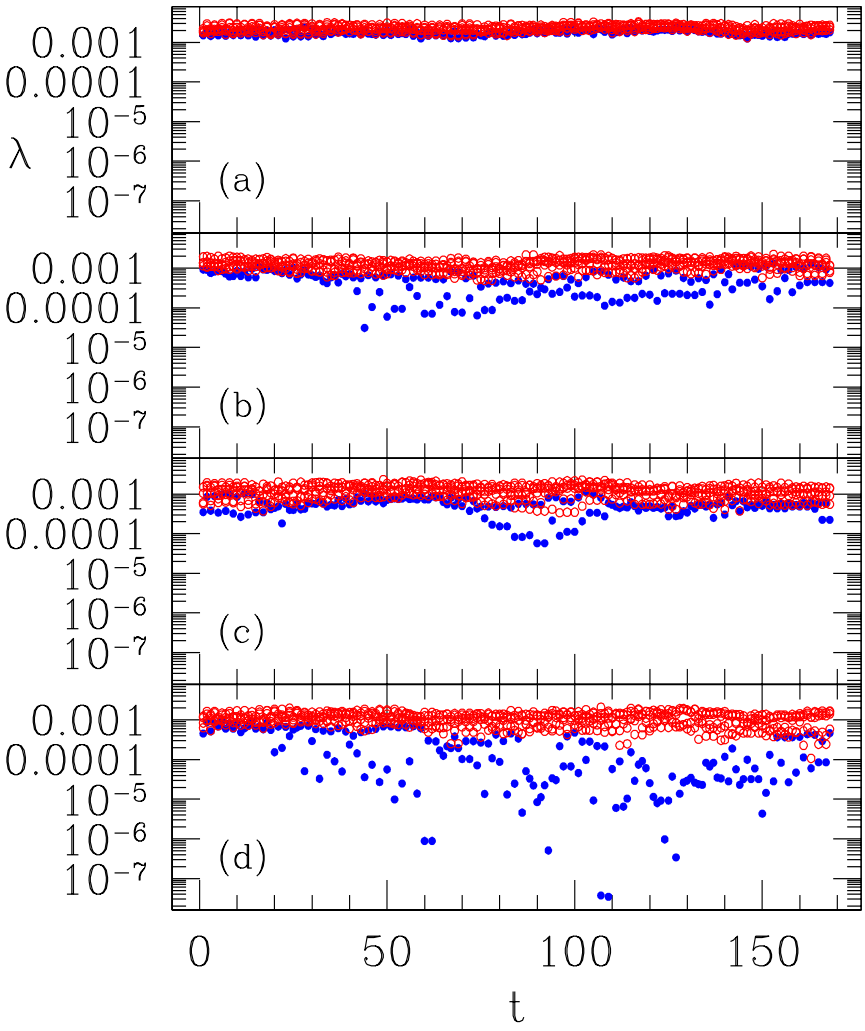
- theoretical advances of having an exact chiral symmetry
- dramatic increase of computational cost

$\cdots$ just one example of the *no free lunch theorem*

still, there are problems in physics where chiral symmetry is so important that we want to use D instead of M

Difficulty # 4

there are dangerous animals on the lattice



- isolated small modes of $M^\dagger M$ (← related to topology)
- accept/reject step: $\Delta H \propto \left[M^\dagger M\right]^{-1} \propto \lambda^{-1}$
  $\Rightarrow \Delta H$ becomes very large and the new configuration is not accepted
- observables: proportional to $\left[M^\dagger M\right]^{-1}$ as well
  $\Rightarrow$ achieve very large value and spoil statistical sample

A Polynomial Hybrid Monte Carlo Algorithm

let $Q^2 = M^\dagger M$: aim is to compute

$$\prod_x \int d\Phi_x e^{-\Phi Q^{-2} \Phi} = \det(Q^2)$$

Rewrite

$$\det(Q^2) = \frac{\det\left[Q^2 P_n(Q^2)\right]}{\det\left[P_n(Q^2)\right]} \ .$$

where $P_n(Q^2)$ is a polynomial such that

$$\left[\det Q^2\right]^{-1} \approx \det P_n(Q^2)$$

If $P_n(Q^2)$ is a Chebyshev polynomial one can find an exponential convergence rate for all eigenvalues in the interval

$$\epsilon \leq \lambda < 1$$

Bound on the accuracy

$$\delta = 2\left(\frac{1-\sqrt{\epsilon}}{1+\sqrt{\epsilon}}\right)^{n+1}$$

integrate each det separately

$$\mathcal{Z} = \int \mathcal{D}U\mathcal{D}\phi^\dagger\mathcal{D}\phi\mathcal{D}\eta^\dagger\mathcal{D}\eta \exp\left\{-S_g - \phi^\dagger P_n(Q^2)\phi\right.$$

$$\left. -\eta^\dagger[\ \underbrace{Q^2 P_n(Q^2)}_{+1-1+Q^2P_n(Q^2)}\ ]^{-1}\right\}\eta$$

$$\mathcal{Z} = \int \mathcal{D}U\mathcal{D}\phi^\dagger\mathcal{D}\phi\mathcal{D}\eta^\dagger\mathcal{D}\eta W \exp\left\{-S_g - \phi^\dagger P_n(Q^2)\phi - \eta^\dagger\eta\right\}$$

with the correction factor

$$W = \exp\left\{\eta^\dagger\left(1 - \left[Q^2 P_n(Q^2)\right]^{-1}\right)\eta\right\} \quad .$$

Data mining

P.Rossi
HPCN - ENEA

The Problem at Hand

We have a very large data base.

We look for a reasonable way to group entries in Classes that have something in common. (But we don't know what it is)

The Goal

We need a probability density function:

$$P(J, T, V|X)$$

That describes the probability to have:

- A number of classes J
- The classes J are of type $T_1, \ldots, T_J$
- Class j is described by the set of parameters V_j
- Under the condition that we have "OBSERVED" X.

The Goal

If we had such a function we "could":

- Choose a set of models so to maximize $P(J, T, V|X)$
- Each class model is described by a p.d.f $L(x_i|T_j, V_j)$ describing how likely it is for the i^{th} element to take the "observed" value x_i, given that belong to class T_j with parameters V_j
- Assign class membership by looking, for each element, to the most probable match.

ALG-EF

- Step 0] Select a random startin point μ_j, σ_j

- Step 1] For each x_i compute

$$w_{ij} = \frac{\pi_j P(x_i | x_i \subset C_j)}{\sum_{j=1}^{J} \pi_j P(x_i | x_i \subset C_j)}$$

- Step 2] compute

$$\begin{aligned} \mu_j &= \frac{1}{N_j} \sum_{i=1}^{N} w_{ij} x_i, \\ \sigma_j^2 &= \frac{1}{N_j} \sum_{i=1}^{N} w_{ij} \left(x_i - \mu_j \right)^2 \end{aligned}$$

- Step 3] Go to Step 1.

The Parallelisation

Computation are needed in Step 1 and Step 2 og ALG-EF. It is obviuos that the simplest strategy one could think of, that is distributing the database across node, will work just fine. In Step 1, the computation is completely local, provided we have had the common sense of replicating class parameters, while Step 2, requires some global communication due to the global sums. By inspection we realize that the number of communication is independent from the size of the data base, (number of rows) but depends solely on the number of columns (attributes), thus in presence of a given shape for a data base, that is a fixed number of attributes, when the number of rows grows larger and larger the algorithm becomes embarassingly parallel and we achieve linear speed up. Results show that with 20 columns we reach linear scaling with as little as 1000 lines, while a more challenging data base that we have just analyzed, with 219 columns, reached asymptotic behaviour around 10000 rows.

MULTIGRID METHODS FOR DISORDERED SYSTEMS
A Way to Improve Cache Performance on Cluster Computers?

C. Best
John von Neumann Institute for Computing/DESY

The biggest performance problem in solving QCD problems on cluster computers comes from the limited memory-to-cache bandwidth available on off-the-shelf workstations. Even on an Alpha 21264 system, performance for large problems is about 1/2 to 2/3 of the performance for problems that fit into cache. Higher CPU frequencies and multi-processor boards will aggravate this problem in the future.
Multigrid approaches can be applied to reduce the memory footprint of an algorithm in favor of the number of arithmetic operations. They are also expected to reduce the problem of critical slowing down, e.g. close to the chiral limit. Earlier multigrid attempts were based on blocked basis functions which were difficult to obtain for disordered problems. We propose to use algebraic multigrid methods and demonstrate how a coarse-grid operator that emulates a given fine-grid operator can be determined numerically. Such operators can not only be used in classical V/W-cycle operation, but can also serve in renormalization-group analysis and in the construction of chirally improved fermions.

Benchmark: Fermion matrix inversion

- BiCGStab method with / without SSOR preconditioning (Norbert Eicker's code):
  - vector operations
  - strongly memory-bandwidth / latency limited
  - very limited cache use
- Characteristics:
  - Memory-to-computation

$$\frac{N_{\text{flops}}}{N_{\text{words}}} \approx 2.5$$

  - Communication-to-computation:

$$\frac{N_{\text{flops}}}{N_{\text{words}}} \approx 70 \frac{n_z n_t}{n_z + n_t}$$

  - Memory-to-communication:

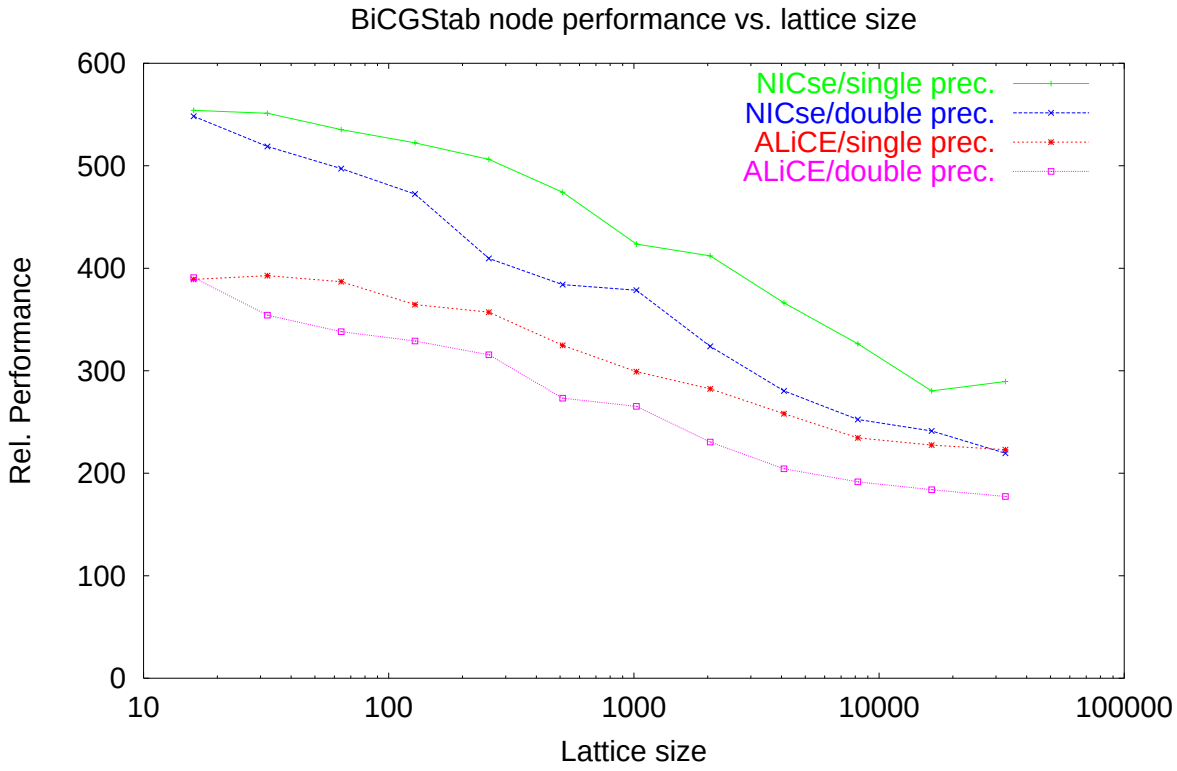
$$\frac{N_{\text{mem.words}}}{N_{\text{comm.words}}} \approx 30 \frac{n_z n_t}{n_z + n_t}$$

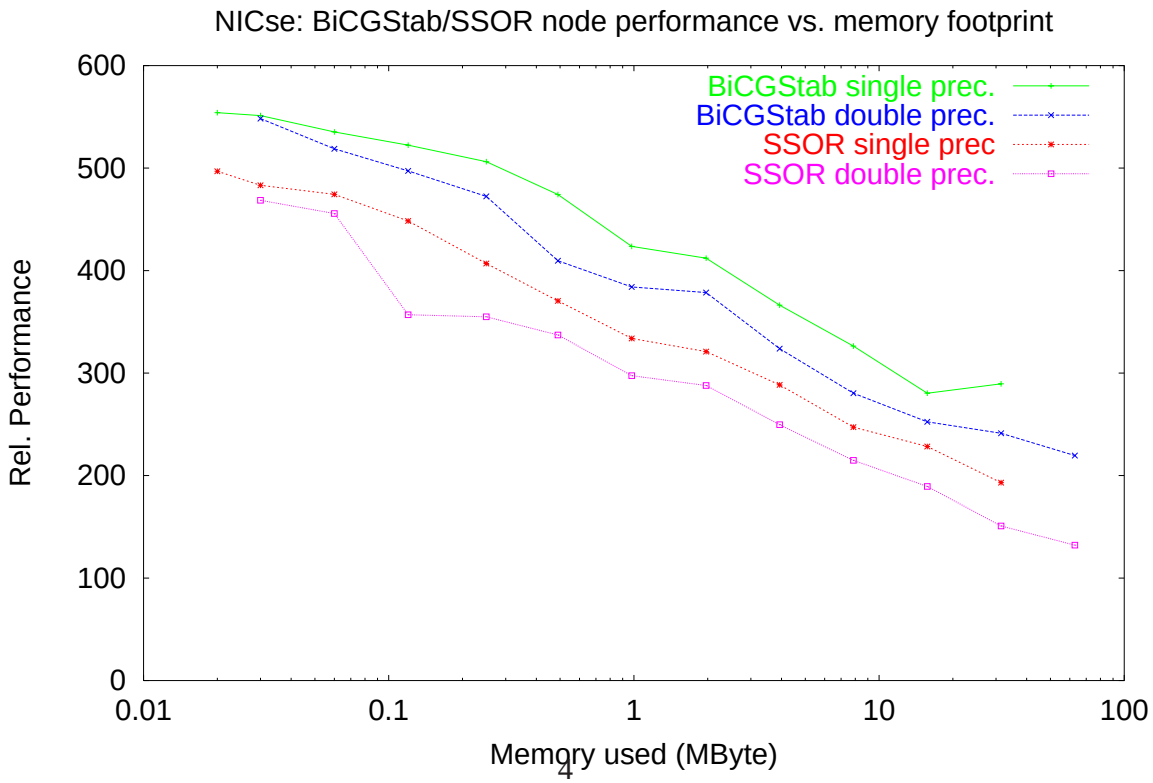
$\Longrightarrow$ O(100) diff. between memory and interconnect performance well satistfied in Myrinet

$\Longrightarrow$ performance limited by mem. bandwidth:

$$P_{\text{flops}} \approx 0.3 \, \frac{\text{MFlops/s}}{\text{MByte/s}} \, P_{\text{mem.bandwidth}}$$

On-node performance





Numerical procedure

Numerical methods to find a good approximation to

$$M_{\text{eff}} = M_{11} - M_{12} M_{22}^{-1} M_{21}$$

IDEA:

- Generate lots of pairs u, a such that $u = M^{-1} a$
- Coarsen them: $\bar{u} = Au$, $\bar{a} = Aa$
- Look for an operator $\bar{M}$ such that

  $$\left| \bar{M} \bar{u} - \bar{a} \right|^2 \qquad (*)$$

  is minimized

Expand a general operator into a basis of path operators:

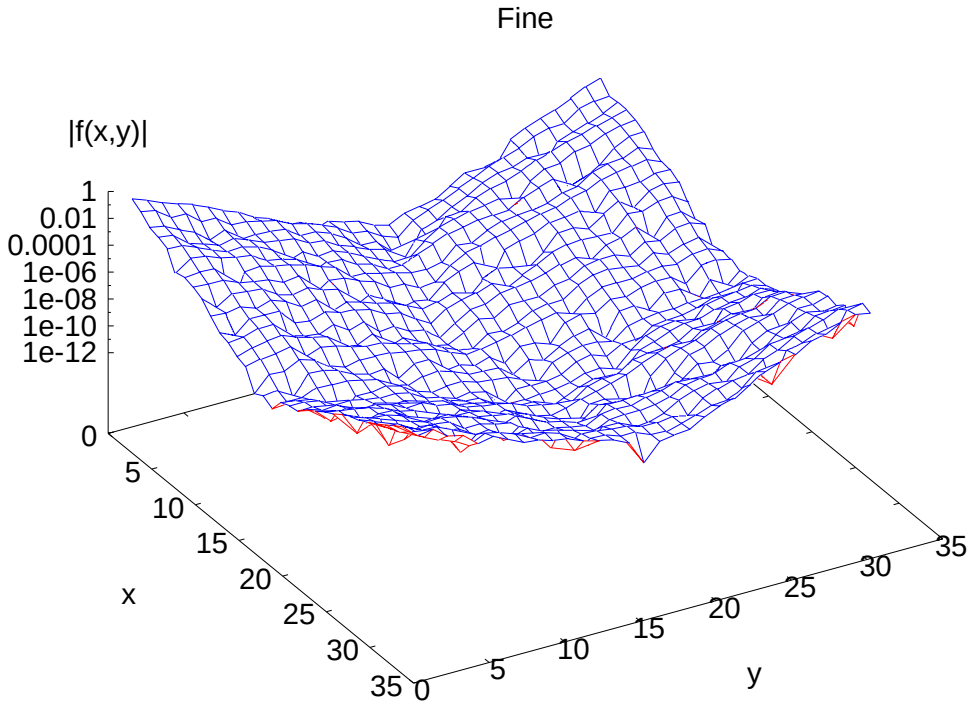
$$\bar{M}(U) = \sum_k \alpha_k \underbrace{M^{(k)}(U)}_{\text{single path}}$$

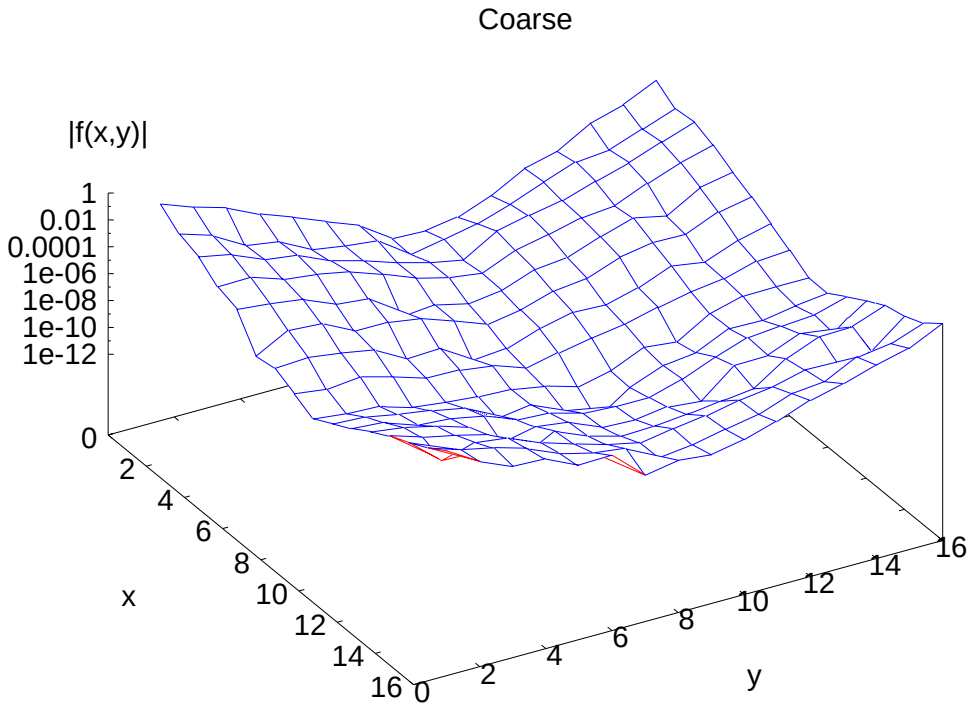
Optimize α_k numerically such that $(*)$ is minimal

Note: Similarity to perfect actions and to chirally-improved operators (e.g. Gattringer hep-lat/0003005, Bietenholz).

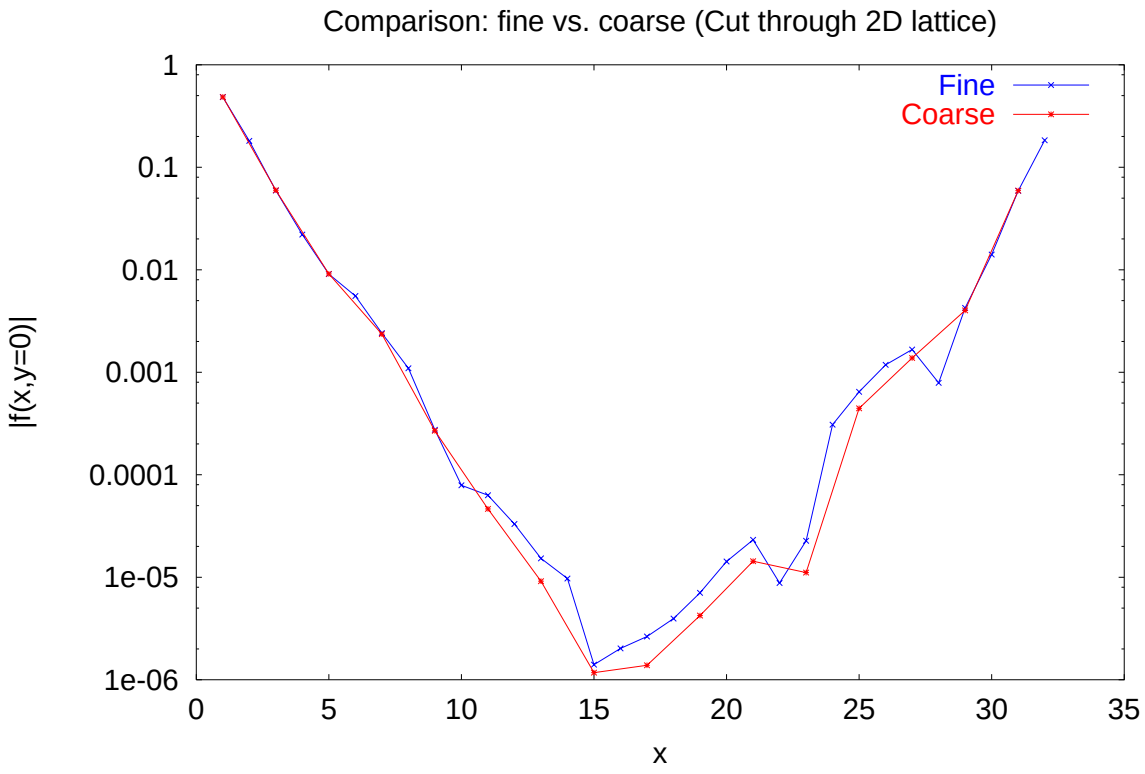
Results

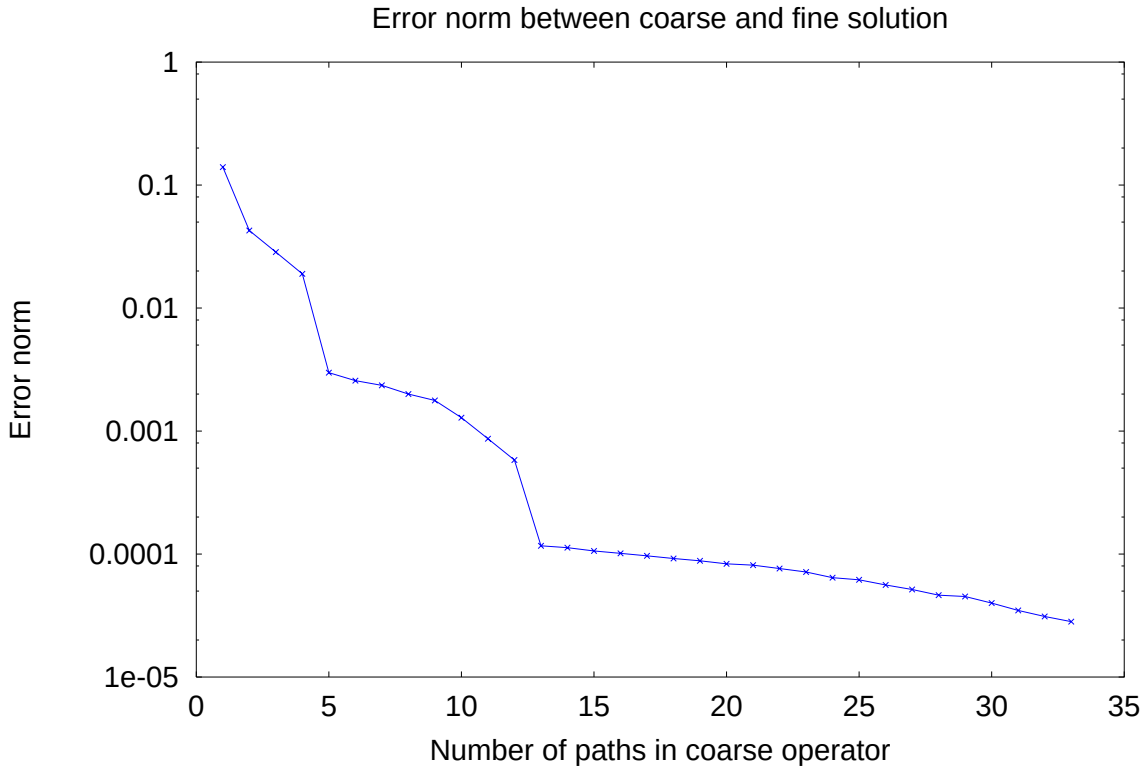
Model problem: 2-D bosonic problem with $\mathbb{Z}_2$ disorder





Results II


Comparison: fine vs. coarse (Cut through 2D lattice)
Fine
Coarse
|f(x,y=0)|
1
0.1
0.01
0.001
0.0001
1e-05
1e-06
0
5
10
15
20
25
30
35
x


Error norm between coarse and fine solution
Error norm
1
0.1
0.01
0.001
0.0001
1e-05
0
5
10
15
20
25
30
35
Number of paths in coarse operator

Graph partitioning

B. Monien
Department of Mathematics and Computer Science
Paderborn Center for Parallel Computing
University of Paderborn, Germany

Efficient use of Parallel Systems

static

dynamic

Mapping

load characteristic known a priori

Distributed Algorithm = Network of communicating processes

Parallel System = Network of communicating processors

==> Efficient mapping of process graph onto processor graph

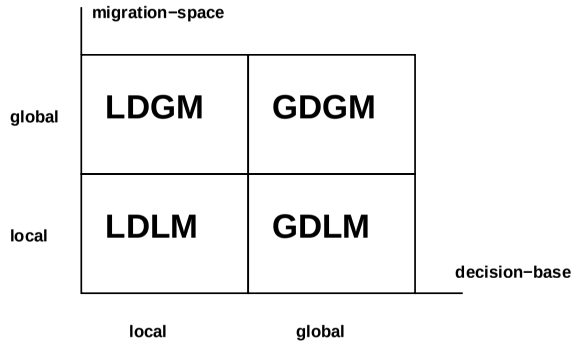
methods : graph partitioning + graph embedding

Load Balancing

unpredictable load behaviour

Dynamic Load Balancing =

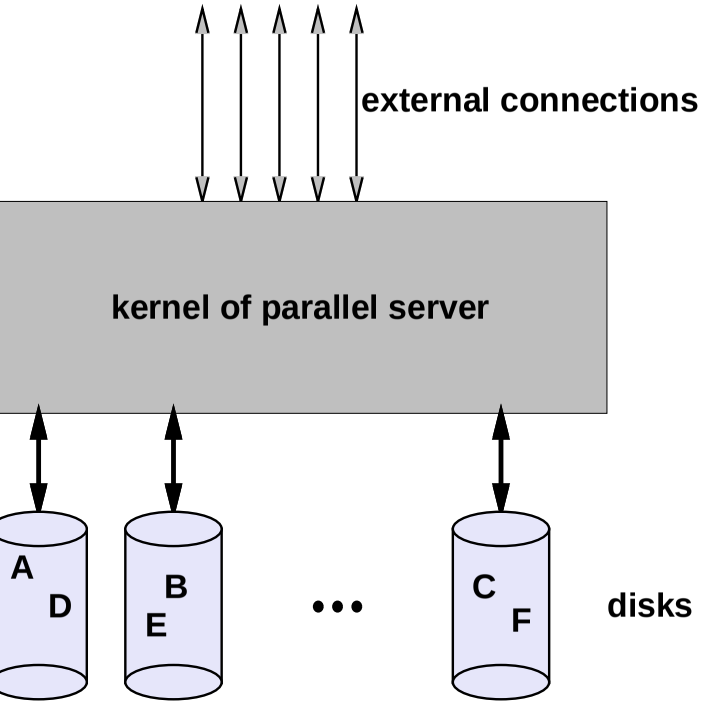
Decision−Base + Migration−Space



pplication: File Placement

oblem:

- parallel data server
- parallel request of two files on same disk $\rightarrow$ *collision*
- place files, goal: minimization of collisions



$\leftrightarrow$

Max-Cut:

- node $u \leftrightarrow$ file u
- edge-weight $w(u,v) \leftrightarrow$ number of collisions of files u, v

$\leftrightarrow$

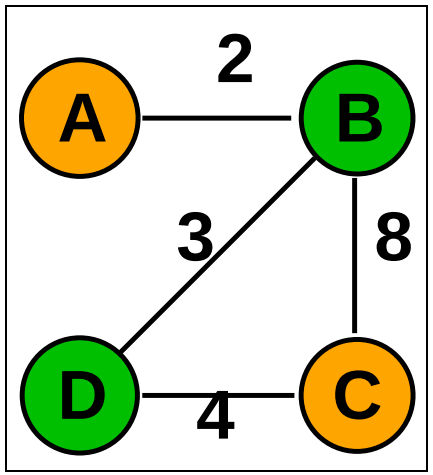
collision matrix:

files				
B	2			
C	0	8		
D	0	3	4	
	A	B	C	

files

$\leftrightarrow$

weighted graph:



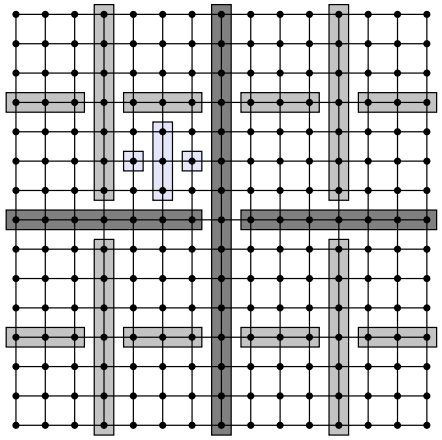
Ordering Methods – Nested Dissection

Remarks:

1) The problem of determining an optimal ordering is NP-complete (Yannakakis, 1981).
2) For a positive definite matrix the ordering does not influence the numerical stability of the elimination process.

Nested Dissection: (George and Liu, 1978)

- recursive algorithm
- relies on vertex separators
- top-down approach



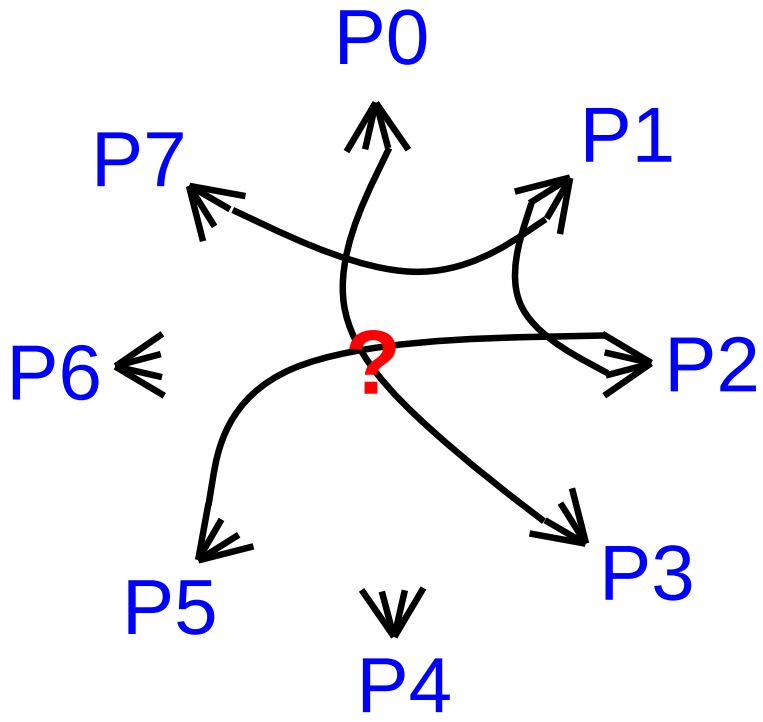
Motivation:

No fill edges between nodes belonging to different components.

Drawback:

In general, S forms a clique in the filled graph $\Rightarrow$ S should be small.

onstruction of Efficient Processor-Networks



Efficiency depends on

- Diameter
- Expansion
- Bisection Width

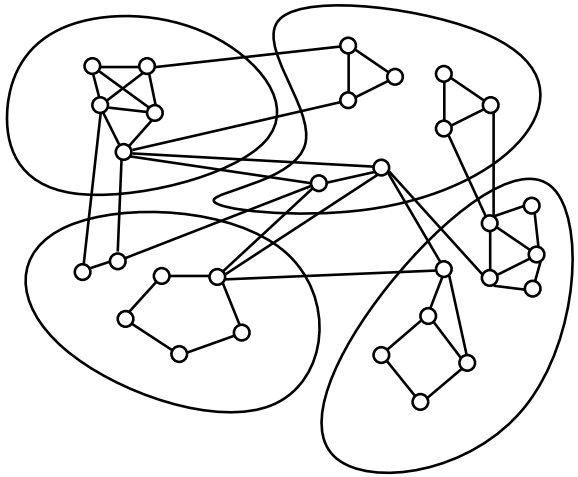
Question: How do I connect the processors?

2. Static Partitioning

- Application Graph $G = (V, E)$, P Processors
- Find partition $\pi : V \to \{0, \ldots, P-1\}$ with

  i) $|V_i| - |V_j| \leq c \quad \forall i, j \in \{0, \ldots, P-1\}$ *(load)*

  ii) $|\{e = \{u, v\} \in E \mid \pi(u) \neq \pi(v)\}| \to \min$ *(cut size)*



ften solved by recursive bisection)

Hpc-line, a new cluster system

R. Lueling
Department of Computer Science
University of Paderborn

The ”Paderborn Center for Parallel Computing, PC^2” at the University of Paderborn is an interdisciplinary research and service center focussing on parallel and distributed computing. One of the main objectives of the PC^2 is the development, integration and provision of high performance computing systems. These systems are provided to the users of the University as well as to external users.

In this effort, the PC^2 has developed together with industrial partners a number of closely connected parallel computing systems. Since 1996, in-depth investigations of cluster architectures are being done at the PC^2. At that time the first cluster system consisting of 22 PC processors has been connected as a SCI cluster, using the Dolphin SCI routers and PCI interface boards. Using the results of this study, in 1997 a first larger system was set up that contains 64 Pentium II, 300 MHz processors. This system was build up by Siemens Nixdorf and Scali using the results of the studies of the former SCI systems.

Today, the PC2 operates a 96 node SCI cluster system, named hpcLine and manufactured by Fujitsu Siemens Computers. Each node contains two Intel Pentium II processors of 450 Mhz and 512 Mbyte of main memory for a node. The nodes are connected by a 8 x 12 two dimensional torus network build up by an SCI network. The system achieves a peak performance of about 45 Gflop (Linpack performance benchmark) and is currently the only purely European manufactured system in the Top-500 list of the most powerful computers world-wide.

The main important feature of the SCI network (SCI = Scalable coherent interface) is the integration of a shared memory between all connected nodes. This allows to support the program development in an easy way. Although the definition of the SCI protocol foresees a cache coherent protocol, the current implementation using PCI cards for each node does not support the coherency between nodes as the PCI protocol destroys this.
The SCI clusters at the PC2 are all managed using the CCS system developed by the PC2. CCS is a full management system for shared memory parallel computing systems that allows to support the users as well as the administrator of a HPC system. It is build up as a distributed system that not only allows to manage one machine, but provides transparent access to a number of machines connected by a LAN. It has open interfaces for the integration of scheduling and other maintenance systems and will be provided as an Open Source version in the future.

High Performance Cluster computing - The SCI cluster hpc*Line*

Reinhard Lüling

rl@upb.de

Paderborn Center for Parallel Computing, PC²

University of Paderborn

PADERBORN CENTER FOR PARALLEL COMPUTING

PC² - Paderborn Center for Parallel Computing

PC2:

Research and service center at the University of Paderborn

Activity:

Competence center for parallel and distributed computing in Northrine Westfalia

Working areas:

- **HPC Systems**
- **HPC Applications**
- **Distributed HPC / Grid Computing**
- **Distributed Media Systems**
- **HPC Services**

User :

~ 250 users in 120 projects, about 40% external users

Parallel Computing Systems :

- **Massively parallel systems with up to 1024 processors (Transputer processors)**
- **SCI Cluster Systems with up to 200 processors (Intel processors)**

Objectives of PC²:

Development of scalable cluster systems, integration into open networks and service provision

PC² PADERBORN CENTER FOR PARALLEL COMPUTING

Primergy High Scalable Compute Server (PSC-192)

1999

Fujitsu Siemens hpc*L*ine:

Distributed-memory machine with
192 Intel processors

Prototype of hpcLine product

Installation base today : > 20 systems

System:	86,4 GFlop/s peak,
	48 Gbyte RAM,
Compute nodes:	96 Primergy Server with 2 x Intel Pentium II 450 MHz
	and 512 Mbyte DRAM
Interconnect:	SCI (Scalable Coherent Interface): 500 MByte/s, 3µs
Topology:	8 x 12 torus with distributed switches,
OS:	Solaris, Linux

The next phase of HPC

- Cluster systems:
  - *Standard Compute Nodes: Intel, Alpha, ...*
  - *Standard OS: Linux*
  - *Standard Communication: MPI*
  - *Communication hardware: SCI, Myrinet, Gigabit Ethernet, ...*
- Next step:
  - *From isolated and “personal” HPC systems to computing power “on-demand”*
  - *Grid Computing*
- Idea of Grid Computing:
  - *Use the Internet to deliver Compute Resources to the user independently from time and place*
  - *First prototype results are available for various applications*

Grid Computing

- Grid Computing - Research:
  - *Scheduling of resources and jobs on networks of HPC systems*
  - *Communication issues: QoS*
  - *Data Storage, Security, ...*
- Close relationship to research areas in Internet computing !!
- Discussion forum in US: Grid Forum www.grid.org
- Discussion forum in Europe: E-Grid
  - *Close link to US activities*
  - *Definition of working groups, co-operation according to IETF model*
  - *First workshop in Poznan on beginning of April, next workshop in June*
- ToDo:
  - *go to www.egrid.org*
  - *subscribe to mailing list and join the workshops and discussions*

What do we gain from hyper-systolic algorithms on cluster computers?

W. Schroers, Th. Lippert, and K. Schilling
Fachbereich Physik, Bergische Universität Wuppertal
D-42097 Wuppertal, Germany

What do we gain from hyper-systolic algorithms on cluster computers?

W. Schroers,[†] Th. Lippert, and K. Schilling[‡]

Fachbereich Physik, Bergische Universität Wuppertal, D-42097 Wuppertal, Germany

The principle problem of parallel computing, the degradation of performance by internode-communication, is aggravated on modular cluster-architectures with off-board communication cards. In this contribution we consider so-called N-squared problems, the prototype being the N-body computation with long range interaction. We demonstrate that hyper-systolic algorithms are able to enhance the performance of such computations on parallel machines by alleviating the communication bottleneck, at the cost of moderately increased memory. We determine the efficiency of implementations on the Alpha-Linux-Cluster-Engine ALiCE at Wuppertal university compared to the SIMD system APE100.

1. INTRODUCTION

The exact computation of mutual interactions between n elements leads to n^2 interaction terms. The computational effort growing with the power of two, such "n^2 problems" truly are a high performance computing task *sui generis*. With the current trend in HPC going from costly proprietary devices towards cost-efficient clusters built from work-stations or PC's, it appears very interesting to study such type of computations on these devices.
To be more specific, let us consider the computation of an array $\{y\} = y_1, \ldots, y_n$ from an array $\{x\} = x_1, \ldots, x_n$ by means of a function $f(\cdot, \cdot)$:

$$y_i = \sum_j f(x_i, x_j), \qquad (1 \le i, j \le n)\,. \tag{1}$$

The computation of the function $f(\cdot, \cdot)$ is required for all pairs x_i, x_j. Therefore, the computational complexity is of order $O\left(n^2\right)$.
The class of n^2-problems considered in (1) occurs in a wide variety of fields, e.g. molecular dynamics, where $f(\cdot, \cdot)$ is the (long range) force between the bodies. Molecular dynamics is of importance in the areas of astrophysics, thermodynamics and plasma physics; for general overviews see the e.g. [1, 2]. Other important fields of application are polymer chains with long-range interactions [3], or genome analysis. Also signal processing [4] and statistical analysis of time series [5] falls into this class. Furthermore, one can generalize the form of (1) by choosing two different arrays $\{x\}$ and $\{z\}$ as the arguments of $f(\cdot, \cdot)$ [6].
In order to implement the exact computation of (1) on a parallel machine with p processing elements (PEs), basically three parallelization strategies are known:

Replicated data method. The algorithm requires the complete array $\{x\}$ to be broadcast at the begin of the calculation, such that all the PEs contain the complete arrays $\{x\}$. PE $k = 1, \ldots, p$ computes $f(x_i, x_j)$ for $j = 1, \ldots, n$ and $i = (k-1)\frac{n}{p} + 1, \ldots, k\frac{n}{p}$. Thus, the communication complexity is of order $O(np)$.

Systolic array computation. Instead of a broadcast, only nearest-neighbour communication is required, while the process is split into n time steps. To each PE $k = 1, \ldots, p$ a sub-array $\{\tilde{x}_k\}$ with n/p elements is assigned. The sub-array is shifted time-step by time-step to all other PEs where the functions $f(\cdot, \cdot)$ are evaluated after each time-step. The communication complexity again is of order $O(np)$.

Hyper-systolic array computation. As an extension of the systolic array computation (see chapter 3), this method allows for the reduction of the communication complexity to order $O\left(n\sqrt{p}\right)$. The price to pay is an increased memory consumption of order $O\left(\sqrt{p}\right)$.

In general, n^2 problems are dominated by computation for large n (when n/p is still large) and by communication for large p (when n/p becomes small). The meaning of "large" and "small" is of course machine and implementation dependent. In chapter 4 we will present some general guidelines to decide which problem sizes n/p are the apropriate choices for a given machine.

[†]Electronic address: wolfram@theorie.physik.uni-wuppertal.de; URL: http://www.theorie.physik.uni-wuppertal.de/~wolfram/
[‡]Also at: John von Neumann-Institut für Computing, Forschungszentrum Jülich, D-52425 Jülich, Germany

2. SYSTOLIC ARRAY COMPUTATION

We map the system of n elements onto a logical 1-dimensional ring of p PEs, each of which can carry out a local computation and a shift to its neighbor PE on the ring, see Ref. [7, 8] for a detailed description. The arrays $\{x\}$ and $\{y\}$ are partitioned into p sub-arrays $\{\tilde{x}_\alpha\}$ and $\{\tilde{y}_\beta\}$ with $\alpha = 1, \ldots, p$ of n/p elements and are homogeneously distributed to the PEs. Note that by virtue of (1) a systolic system of n processors can be mapped onto a system of p processors. This procedure is called hierarchy mapping.
The algorithm consists of the following steps:

1. Compute $\{\tilde{y}_\alpha\} = \sum_{k\in\{\alpha\}} f\left(\{\tilde{x}_\alpha\}, \{\tilde{x}_{\alpha,(k)}\}\right)$ locally on each node.
2. Shift the sub-arrays along the 1-dimensional ring topology with cyclic boundary conditions $\{\tilde{x}_\beta\} \leftarrow \texttt{cshift}\left(\{\tilde{x}_\alpha\}\right)$.
3. Compute the function $\sum_{\{\beta\}} f\left(\{\tilde{x}_\alpha\}, \{\tilde{x}_\beta\}\right)$ and sum up the result to $\{\tilde{y}_\alpha\}$ on each node.
4. Repeat steps 2–3 $(p-1)$ times.

The result is $\{\tilde{y}_\alpha\} = f\left(\{\tilde{x}_\alpha\}, \{x\}\right)$ on each node.

Step	$\{\tilde{x}_\gamma\}$ on PEs							
0	1	2	3	4	5	6	7	8
1	8	1	2	3	4	5	6	7
2	7	8	1	2	3	4	5	6
3	6	7	8	1	2	3	4	5
4	5	6	7	8	1	2	3	4
5	4	5	6	7	8	1	2	3
6	3	4	5	6	7	8	1	2
7	2	3	4	5	6	7	8	1

TABLE I: The distribution of the data packets $\{\tilde{x}_\alpha\}$ on the PEs during the systolic computation.

The communication complexity of this algorithm is of order $O(np)$. Symmetries or anti-symmetries of $f(\cdot,\cdot)$ are not yet exploited. If the evaluation of $f(\cdot,\cdot)$ is very costly, one may thus loose up to a factor of two in computation time. Furthermore, we note that during the systolic process the combination of elements (i,k) could be found n times, if one had stored intermediate arrays. However, this specific combination is only required for the PEs i and k. The load is equally distributed to all PEs and no idle cycles occur. This fact renders the algorithm suitable on both SIMD and MIMD architectures.
The method is also known as "Orrery-algorithm". Its extension, the Half-Orrery-algorithm, allows the exploitation of symmetries of $f(\cdot,\cdot)$. However, the Half-Orrery-algorithm is equivalent to a specific case of the hyper-systolic algorithm to be discussed in the next section.

3. HYPER-SYSTOLIC ARRAY COMPUTATION

3.1 General concept

Next, let us try to remove the "redundancy" of combinations which occurs in the previous case. We still want to keep the advantages of the algorithm, namely the symmetric implementation on all PEs. We now keep the intermediate arrays in memory, and during each step we consider *all* combinations of pairings between the data packets on the current PE. This strategy will of course not reduce the number of operations required, but we can expect that the number of rows required (and thus the number of communication operations) becomes smaller.

THEOREM 1 *The minimum number of rows required to compute (1) is given by*

$$h \geq \sqrt{p - \frac{3}{4}} - \frac{1}{2} \tag{2}$$

if the function $f(\cdot,\cdot)$ is symmetric or antisymmetric in its arguments and by

$$h \geq \sqrt{2p - \frac{7}{4}} - \frac{1}{2} \tag{3}$$

if no symmetries are exploited.

Proof: The minimum number of pairings required is given by $p(p-1)/2$ (or $p(p-1)$ if there are no symmetries), the number of possible combinations between the elements of a given column is $\binom{h+1}{2}$. Since there are p such columns, the following inequality holds:

$$\frac{p(p-1)}{2} \leq \binom{h+1}{2} p,$$

or, respectively

$$p(p-1) \leq \binom{h+1}{2} p.$$

□

Since the problem is homogeneous for all PEs (and thus also for the data packets $\{\tilde{x}_\gamma\}$ initially distributed on the PEs), it is sufficient to consider the treatment of data packet $\{\tilde{x}_{\alpha=1}\}$. The problem is solved if we generate the combinations with all other data packets $\{\tilde{x}_{\beta\neq 1}\}$. A possible solution for this problem is shown in table II. The configuration can be reached if the sub-array $\{\tilde{x}_{\beta=1}\}$ is shifted by strides of $(1,1,2)$ (instead of only strides of 1 as in the previous chapter); thus the array is *not* distributed to all PEs (only to 1, 2, 3 and 5).

Line	$\{\tilde{x}_\gamma\}$ on PEs							
0	1	2	3	4	5	6	7	8
1	8	1	2	3	4	5	6	7
2	7	8	1	2	3	4	5	6
3	6	7	8	1	2	3	4	5
4	5	6	7	8	1	2	3	4
5	4	5	6	7	8	1	2	3
6	3	4	5	6	7	8	1	2
7	2	3	4	5	6	7	8	1

TABLE II: The minimal number of rows required to have all pairings of data packets $\{\tilde{x}_{\beta\neq 1}\}$ with $\{\tilde{x}_{\alpha=1}\}$ occur at least once. Some combinations with $\{\tilde{x}_{\beta\neq 1}\}$ may occur more than once. This table has been constructed under the assumption that the function $f(\cdot,\cdot)$ is symmetric.

However, so far all components of $\{\tilde{y}_\gamma\}$ have been computed, but some have been accumulated on the wrong PEs (e.g. the pairing $(1,4)$ on PE 5 must be shifted to the PEs 1 and 4 and added to $\{\tilde{y}_{\gamma=1}\}$ and $\{\tilde{y}_{\gamma=4}\}$ to complete the calculation). Therefore, we have to store and finally shift the resulting partial arrays $\{\tilde{y}_{\gamma,a=1,\ldots,h}\}$ in the same way backwards as we previously shifted the partial arrays $\{\tilde{x}_\gamma\}$ forward and sum them up at the correct place. Thus, the total amount of communications required is 6 for this algorithm (in contrast to 7 for the previous case). In the general case for p processing elements we need h shifts of $\{\tilde{x}_\alpha\}$ and h shifts of $\{\tilde{y}_\alpha\}$. On the other hand the total memory requirement has increased to $h-1$ arrays of $\{\tilde{y}_\alpha\}$! Below we are going to show that always a solution for h of order $\sqrt{p}$ exists, thus the communication cost for the new algorithm is of order $O\left(n\sqrt{p}\right)$.

3.2 Hyper-systolic bases

The recipe to find the minimum number of rows required to solve the complete problem can be cast in terms of Additive Number Theory. We define the hyper-systolic base $A_h = (a_0 = 0, a_1, \ldots, a_h)$ to be an ordered set of numbers where the $a_{l=0,\ldots,h}$ are the strides of the shifts introduced in section 3 3.1. Then we can formulate the problem in the following way:

THEOREM 2 *Let I be a set of integers $I = \{0, 1, \ldots, p-1\}$. Find the ordered set A_h of $h+1$ integers with h minimal, such that each $m \in I$ can be represented at least once as the ordered partial sum*

$$\begin{aligned} m &= a_i + a_{i+1} + \cdots + a_{i+j}, \\ m &= p - (a_i + a_{i+1} + \cdots + a_{i+j}), \qquad 0 \le i + j \le h, \qquad i, j \in \mathbb{N}. \end{aligned}$$

The second equation has to be fulfilled to exploit symmetries of $f(\cdot,\cdot)$.

This optimization problem is equivalent to the h-range $p(h, A_h)$ of the extremal basis A_h with only ordered partial sums allowed as formulated in theorem 2. It reminds one of the postage stamp problem where no ordering was required [9, 10]. This class of problems is NP-complete, however, so finding the exact solutions become increasingly costly for large p. However, we may find a base which is valid for arbitrary p and whose length still only grows as $O\left(n\sqrt{p}\right)$.

THEOREM 3 *The regular base*

$$A_{2K_{\mathrm{hys}}-1} = \left(0, \underbrace{1, \ldots, 1}_{K_{\mathrm{hys}}-1}, \underbrace{K_{\mathrm{hys}}, \ldots, K_{\mathrm{hys}}}_{K_{\mathrm{hys}}} \right)$$

with $K_{\mathrm{hys}} \ge \sqrt{p}$ ($K_{\mathrm{hys}} \ge \sqrt{p/2}$) solves the (symmetric) problem. The communication complexity still is $O\left(n\sqrt{p}\right)$.

Proof: The base $A_{2K_{\mathrm{hys}}-1}$ contains $K_{\mathrm{hys}} - 1$ elements $a_i = 1$. Thus, any number $0 \le r \le K_{\mathrm{hys}} - 1$ can be represented as an ordered partial sum. Furthermore the partial sums

$$\sum_{l=i}^{j} K_{\mathrm{hys}} = (j - i + 1) K_{\mathrm{hys}}$$

are integer multiples of K_{hys} with $0 \le j - i + 1 \le K_{\mathrm{hys}}$. Therefore any $m < p$, $m \in \mathbb{N}$ can be represented by $A_{2K_{\mathrm{hys}}-1}$. In the symmetric case, we can use the second condition in theorem 2 to show that summing up only until $p/2$ is sufficient for the base to be complete. □
Starting from this base one can compute better solutions using *simulated annealing* methods [11].

3.3 Cost functions

In the previous section the bases found are most effective if all communications have equal costs, regardless of the stride. In the general case, we have to minimize the cost function:

$$C_\Sigma(A_h) = \sum_i \left(C^{\{\bar{x}_\alpha\}}(a_i) + C^{\{\bar{y}_\alpha\}}(a_i) \right),$$

where $C^{\{\bar{x}_\alpha\}}(l)$ and $C^{\{\bar{y}_\alpha\}}(l)$ are the costs to communicate $\{\tilde{x}_\alpha\}$ and $\{\tilde{y}_\alpha\}$ by a stride of l.
Again simulated annealing methods may help to find good solutions.

4. IMPLEMENTATIONS AND RESULTS

Table III gives a list of the strides for the shortest bases (under the assumption that $f(\cdot,\cdot)$ is symmetric). We see that for some values of p the shortest base length obtained from eq. (2) cannot be realized. Furthermore we note that a solution for a given value of p is not necessarily valid for $p - i$, $1 \le i < p$ [12].
As a test-case for the efficiency of hyper-systolic routing on parallel computers we give results for a Kepler-force computation, implemented on two different machines.
Figures 1 and 2 show results for two APE-100 parallel systems. These machines are SIMD machines with a static networking topology, designed as a 3-dimensional torus. They deliver a performance of 50 MFlops/node in single precision. The code has been optimized, and the force computation has been carried out in single precision. The summation, however, is in double precision. The results have been taken from [13]. The APE architecture offers a very low latency. Therefore, small particle numbers per node (less than 100) can be efficiently simulated on APE.

p	$\mathbf{A}_h$	p	$\mathbf{A}_h$	p	$\mathbf{A}_h$	p	$\mathbf{A}_h$
1	(1)	17	(1, 1, 2, 8)	33	(1, 1, 1, 3, 10, 11)	49	(1, 1, 3, 19, 9, 3, 8)
2	(1)	18	(1, 1, 3, 6)	34	(1, 1, 1, 4, 5, 8)	50	(1, 2, 5, 9, 11, 4, 6)
3	(1)	19	(1, 1, 4, 3)	35	(1, 1, 1, 5, 4, 9)	51	(1, 1, 3, 6, 7, 12, 8)
4	(1, 1)	20	(1, 1, 1, 3, 4)	36	(1, 1, 3, 7, 2, 6)	52	(1, 1, 1, 1, 2, 8, 7, 9)
5	(1, 1)	21	(1, 3, 10, 2)	37	(1, 1, 2, 6, 5, 7)	53	(1, 1, 1, 1, 3, 14, 8, 15)
6	(1, 2)	22	(1, 1, 1, 4, 4)	38	(1, 1, 1, 1, 4, 6, 9)	54	(1, 1, 1, 1, 5, 6, 6, 10)
7	(1, 2)	23	(1, 1, 1, 4, 4)	39	(1, 1, 2, 9, 5, 15)	55	(1, 1, 1, 1, 2, 13, 7, 21)
8	(1, 1, 2)	24	(1, 1, 1, 4, 8)	40	(1, 1, 1, 1, 5, 5, 10)	56	(1, 1, 1, 1, 7, 5, 17, 6)
9	(1, 1, 2)	25	(1, 1, 1, 5, 4)	41	(1, 1, 1, 1, 5, 6, 10)	57	(1, 2, 10, 19, 4, 7, 9)
10	(1, 1, 3)	26	(1, 1, 3, 4, 6)	42	(1, 1, 1, 1, 5, 6, 10)	58	(1, 1, 1, 4, 14, 12, 4, 13)
11	(1, 1, 6)	27	(1, 1, 3, 8, 9)	43	(1, 1, 1, 1, 6, 5, 11)	59	(1, 1, 1, 3, 7, 8, 14, 9)
12	(1, 2, 4)	28	(1, 3, 11, 5, 2)	44	(1, 1, 1, 3, 10, 11, 11)	60	(1, 1, 2, 5, 6, 10, 5, 12)
13	(1, 2, 6)	29	(1, 1, 1, 1, 5, 5)	45	(1, 1, 1, 2, 7, 6, 8)	61	(1, 1, 1, 4, 8, 10, 11, 9)
14	(1, 1, 1, 4)	30	(1, 1, 1, 1, 5, 10)	46	(1, 1, 1, 3, 12, 7, 13)	62	(1, 1, 2, 6, 22, 7, 7, 5)
15	(1, 1, 1, 4)	31	(1, 2, 5, 4, 6)	47	(1, 1, 1, 2, 11, 6, 18)	63	(1, 1, 4, 2, 12, 18, 3, 13)
16	(1, 1, 3, 3)	32	(1, 1, 1, 4, 4, 8)	48	(1, 1, 3, 4, 11, 6, 10)	64	(1, 1, 3, 9, 2, 18, 8, 17)

TABLE III: Hyper-systolic bases for $p \leq 64$ for a function $f(\cdot, \cdot)$ symmetric in its arguments.

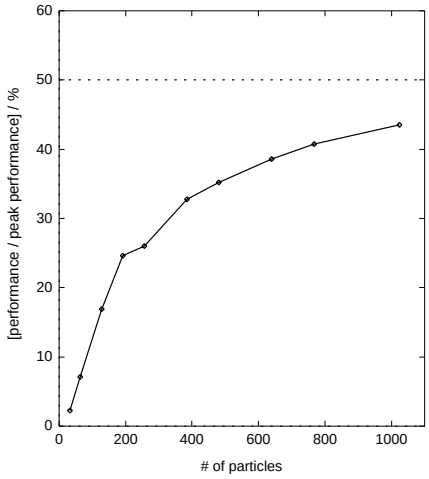


FIG. 1: Relative performance on the 32-node APE-100 Q4 (peak: 1.6 GFlops/single).

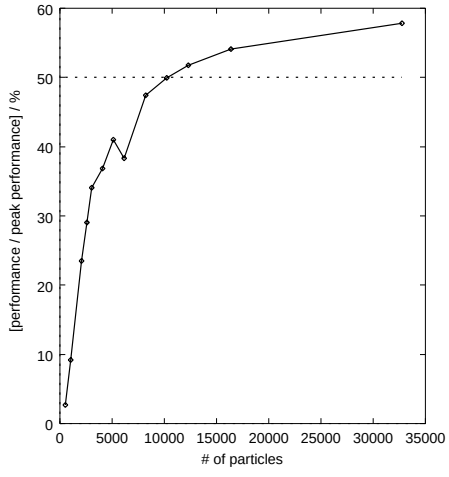


FIG. 2: Relative performance on the 512-node APE-100 QH4 (peak: 25.6 GFlops/single).

Next let's consider results from the Wuppertal Alpha-Linux-Cluster-Engine (ALiCE). This system consists of a cluster of Compaq DS10 workstations, equipped with Alpha 21264 EV6 466 MHz processors with a 2nd level cache of 2 MBytes. Each node delivers a peak performance of 932 Mflops and they are connected by a fat tree Myrinet network. The implementation of the Kepler force is in standard Fortran 90 using MPI without any futher optimizations beyond those the compiler offers. The entire calculation is performed in double precision.
Figure 3 shows the total times of a single evaluation of the forces for different numbers of particles per node. Figure 4 gives the relative performances achieved for these cases. For small particle numbers the results are contaminated by latency effects (the Parastation software used at this stage of the project showed a latency around 55μs).
On ALiCE, particle numbers of less than hundred particles per node do not make efficient use of the system. Local particle numbers of the order of $O\left(10^3\right)$ particles per node are required for effective pipelining of communication and memory-to-cache-data-transfer.
We observe a decrease of the hyper-systolic curve at about 8200 particles/node. At this point the total size of data involved is about 1.2 MBytes which is 60% of the 2nd level cache of the machine. As a consequence we observe a performance degradation due to cache effects once the data size reaches about $50-60\%$ of the total cache size.
However, since we could exploit symmetries, the total number of floating point operations required to compute the resulting force is still smaller than for the systolic algorithm and thus the total timing is still advantageous in figure 3.

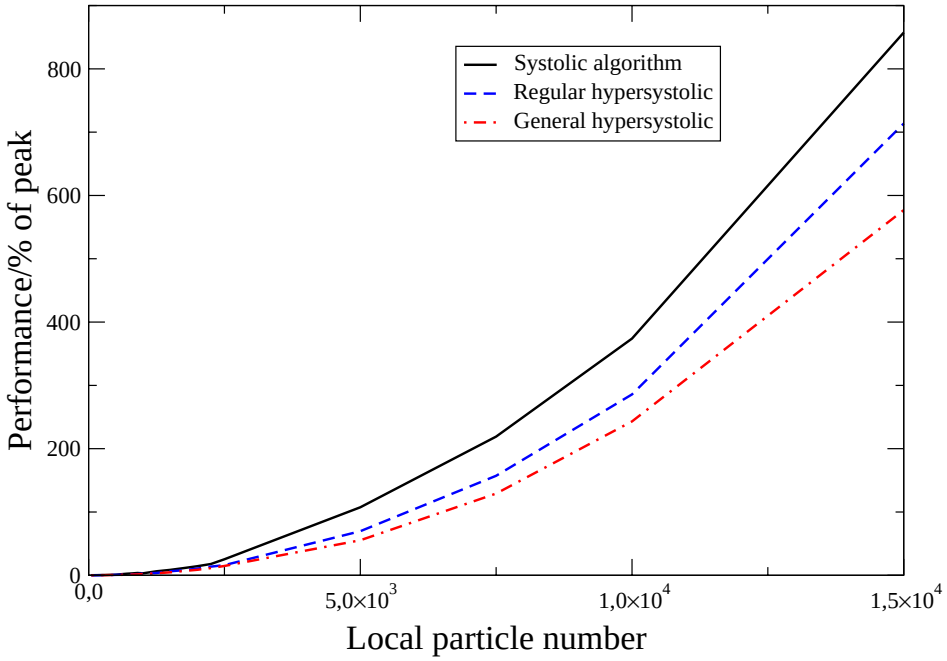


FIG. 3: Timing on a 32-node partition of ALiCE.

5. CONCLUSIONS

In this talk we covered different routing strategies for different parallel computing architectures. We have motivated the use of hyper-systolic algorithms and discussed their implementation on two different parallel architectures. They allow parallization of n^2 problems at minimal communicational cost, but have increased memory consumption. The low latency on APE machines allows small data sizes per node, while on computer clusters larger numbers of particles per node are more advantageous.

ACKNOWLEDGMENTS

The authors wish to thank N. Eicker for his support on ALiCE and his valuable comments on compiler usage and software optimization. The authors furthermore wish to thank H. Simma for his important remarks on compiler

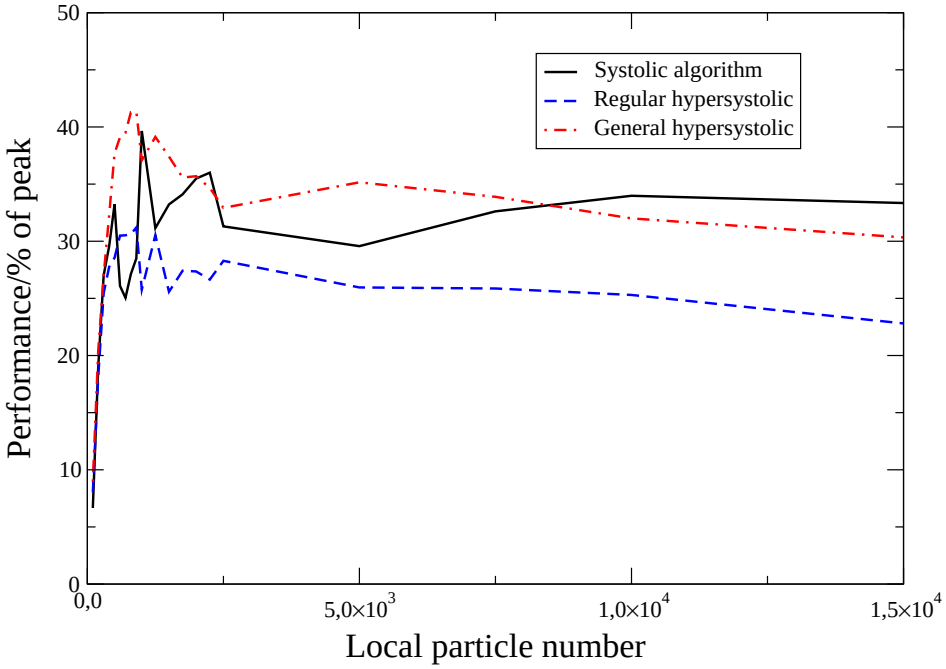


FIG. 4: Relative performance on the 32-node partition of the ALiCE.

optimizations and P. Ueberholz for sharing his experience in performance analysis with us.

REFERENCES

[1] W. Smith, Computer Physics Comm. **62**, 229 (1991).
[2] P. Hut, astro-ph/9704286 (1997).
[3] M. Levitt, Current Opinion in Structural Biology **1**, 224 (1991).
[4] L.R. Rabiner and B. Gold, *Theory and application of Digital Signal Processing* (Englewood Cliffs, N.J.: Prentice Hall, 1975).
[5] T.D. Dontje, Th. Lippert, N. Petkov and K. Schilling, Parallel Computing **18**, 575 (1992).
[6] Th. Lippert, N. Petkov, P. Palazzari and K. Schilling, *Hyper-Systolic Matrix Multiplication* (2000), In preparation.
[7] N. Petkov, *Systolische Algorithmen und Arrays* (Akademie-Verlag, Berlin, 1993).
[8] Th. Lippert, *Hyper-Systolic Parallel Computing*, Ph.D. thesis, Rijksuniversiteit Groningen (1998).
[9] H. Scheid, *Zahlentheorie* (BI-Wissenschaftsverlag, Mannheim, Germany, 1991).
[10] M. Djawadi and G. Hofmeister, Mainzer Seminarberichte, Additive Zahlentheorie **3**, 187 (1993).
[11] P. Palazzari, Th. Lippert and K. Schilling, in *Proc. NATO Advanced Research Workshop High Performance Computing, Technology and Applications*, edited by L. Grandinetti et.al. (Cetraro, Italy, 1996).
[12] W. Schroers, Th. Lippert and K. Schilling, *Hyper-systolic bases for different network topologies* (2000), In preparation.
[13] Th. Lippert, U. Glaessner, H. Hoeber, G. Ritzenhöfer, K. Schilling and A. Seyfried, Int. Journal Mod. Phys. **C**(7), 485 (1996).

FLINK - A high speed PC network interface
Recent HW/SW developments

K.H. Sulanke, G. Sacco, S. Hummel, N. Paschedag
DESY

FLink – a high speed PC network interface
Recent HW/SW developments

Hardware Design
Karl–Heinz Sulanke
DESY Zeuthen, Platanenallee 6, 15738 Zeuthen, Germany

Linux Character Device Driver
Giuseppe Sacco
INFN, Sezione di Roma, P.le A. Moro 2, I–00185 Roma, Italy

Linux Point–to–Point Network Driver
Sven Hummel
FH Brandenburg, Magdeburger Str. 50, 14470 Brandenburg, Germany

Linux C/C++/perl API
Norbert Paschedag
DESY Zeuthen, Platanenallee 6, 15738 Zeuthen, Germany

In order to provide an alternative to expensive commercial high–speed network adapters, the FLink project was initiated with the intention to exploit modern channel–link technology to bring the inter–node link speed into the same performance range that is seen for the internal PC busses.
We present an introduction to the design of the FLink PCI network adapters, the unusual FLink network topology, and show performance results obtained with the current prototypes.
Although only basic TCP/IP support is provided yet, custom API's are available and allow to exploit transfer rates above 60 megabytes per second.

Concepts

A switchless LAN network interface for data transfers at speeds comparable to internal bus speeds

Ring network topology

Allow parallel access by many nodes (No Tokens)

FLink I (prototypes)

reliable for long duration tests

low-level latency of ~300ns

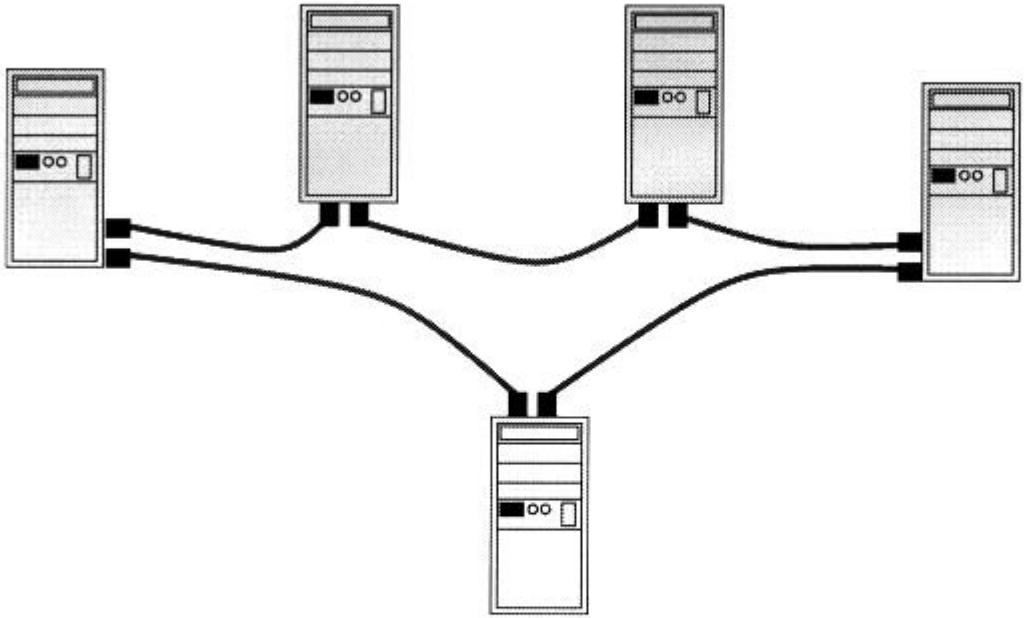
link speed 66 MB/s

DMA speed 120 MB/s

Uploadable firmware

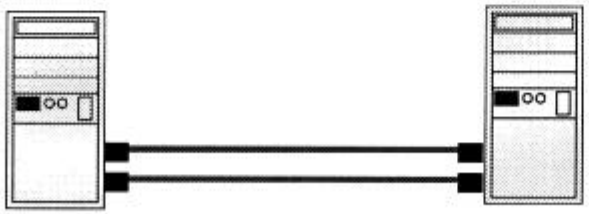
Hardware costs / node : ~ 200 €

FLink Network



FLink Ring

(up to 255 nodes)



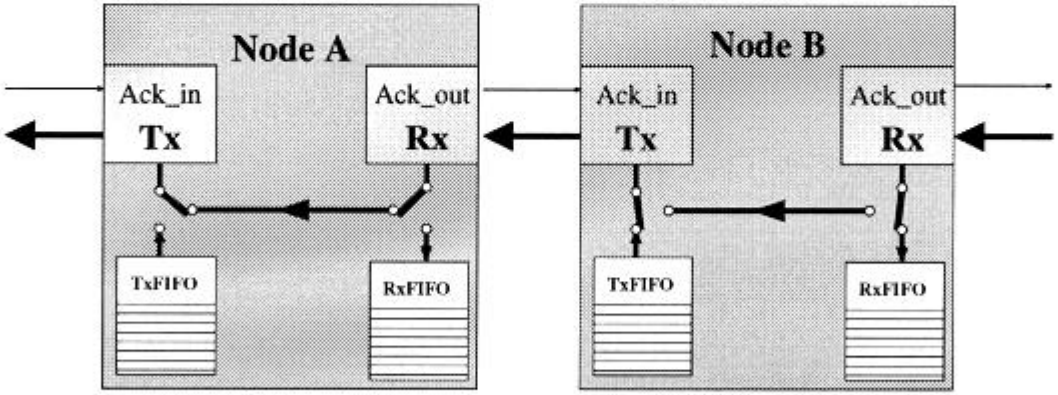
Point to Point

Principle of ring connection

Point-to-point link , max. distance 10 m

link acknowledge signal for synchronization

"zipper principle" for send / pass-through arbitration



FLink II outlook

Link speed increase to ~ 240 MB/s

FIFO size of at least 64 kB

PC memory to link latency ~ 250ns

DMA speed max 132 MB/s

Hardware costs / node : ~ 250 - 300 €

Availability (samples) : ~ June 2000

KMPI (Kool MPI): Toward an optimized MPI implementation for the Linux clusters

M. Kim, S. Kim
Department of Physics
Sejong University
R.of Korea

KMPI (Kool MPI): Toward an optimized MPI implementation for the Linux clusters

Myunggyu Kim *and Seyong Kim †

February 15, 2000

http://hana.etri.re.kr/~mgkim/kmpi

This document was generated from the file *kmpi.tex*. If you visit KMPI homepage, you can look at this document in a hierarchical form using hyperlinks.

Abstract

KMPI (Kool MPI) is an optimized implementation of MPI for the Linux clusters. KMPI is the set of a MPI library and a monitoring and management tool for MPI processes.

KMPI is simple and fast.

LAM -6.1 is faster than MPICH-1.1.2 for small messages and MPICH -1.1.2 is faster than LAM-6.1 for large messages.

KMPI version 0.2 is up to four times faster than MPICH-1.1.2 and LAM-6.1 for small messages but has a reliability problem.

KMPI whose version is greater than 0.5 is more than 20 % faster6 than LAM-6.1 and MPICH-1.1.2 for small messages and 10 % faster than MPICH-1.1.2 and LAM-6.1 for large messages. KMPI is reliable and stable.

The tests were carried out on a switched Fast Ethernet network cluster of 8 Alpha machines running Linux.

The KMPI project is in progress. Myunggyu and Seyong are looking for few volunteers to help the development of the KMPI. If you are willing to join the KMPI project, please contact Myunggyu Kim or Seyong Kim .

*Electronics and Telecommunications Research Institute (ETRI)

†Sejong University

3 Overview of KMPI

KMPI whose version is greater than 0.5 is implemented using TCP/IP. The TCP guarantees reliability. The TCP sockets used in KMPI are optimized for LAN Linux clusters by turning the Nagle algorithm off and by modifying the send/receive buffer size adaptively. In this version, C and Fortran bindings are implemented.

KMPI has a built-in short/large protocol. The built-in short protocol makes KMPI faster for short messages. The built-in large protocol makes KMPI faster for large messages than using the short protocol for large messages.

KMPI is simple. Reading the KMPI source files will make you surprised by their simplicity and clearness. This feature benefits easy debugging and improvement.

The TCP dependent part in KMPI is well separated from others. It is our next goal to improve the MPI performance by modifying this TCP dependent module into an optimized protocol for LAN Linux clusters.

3.1 features

- KMPI is faster6 than MPICH and LAM for the Linux clusters.
- KMPI is reliable and stable.
- KMPI is simple and thus clear.
- KMPI is scalable. We have successfully run more than 128 processes on a 8 node cluster stably.
- KMPI shows the line-buffered stdout messages and unbuffered stderr messages of the processes.
- KMPI accepts the user input from the process of rank 0.
- KMPI provides a monitoring and management tool for MPI processes.
- KMPI ($\geq$ 0.6) processes works in the directory where *mrsh* is started after **MPI_Init** .
- KMPI ($\geq$ 0.6) accepts the user-defined command line arguments.

3.2 limits

- If an error occured during a call, the current version does not return an error code but aborts the MPI session.

4 Performance of KMPI

The performance tests were done on a switched Fast Ethernet network cluster of eight Alpha 21164 600 MHz with 128 - 256 M SDRAM running Linux whose kernel version is 2.2.1. More detail specification for the cluster is described in the subsection 4.1.

All the MPI libraries and test programs were compiled using the compiler egcs version 1.0.3a with -O optimization. All the test programs are C programs.

MPICH was configures with the ch_p4 device.

All LAM tests used the -c2c, -nger, -O and -w switches to mpirun. The -c2c selects client-to-client mode in which the LAM library bypasses daemon and establishes direct connections for MPI communication. The -nger disables GER (Guaranteed Envelope Resources) communication protocol and error reporting. The -O informs the LAM/MPI library that the cluster is homogeneous and hence turns off data conversion. The -w waits for all processes to complete before exiting mpirun and reports any abnormal exit codes.

Figure 4: Picture of the cluster

Visit the site http://hana.etri.re.kr/~mgkim/kmpi.

Table 2: Hardware configuration of the cluster

CPU:	Alpha 21164 (EV5.6) 600 MHz	(Samsung)
Main Board:	164 LX with 2MBytes cache	
Memory:	from 128 to 256 MBytes SDRAM	(Samsung)
SCSI Controler:	AHA2940 Ultra wide	(Adaptec)
SCSI HDD:	from 2 to 4 GBytes	(Seagate)
CD-ROM Drive:		(Samsung)
Network adapter:	3c905 Boomerang 100baseTx (node 0 – node 7)	(3Com)
	EtherExpress Pro 10/100 (node 0)	(Intel)

4.1 Specification of the cluster

The figure 4 shows the picture of the cluster that the proformance test of KMPI has been done on. The cluster is located at the Department of Physics, Sejong University . Samsung Electronics Co. donated the cluster to Seyong Kim and he upgraded some parts of the cluster. The cluster consists of 8 Alpha PCs. The hardware configuration of the cluster is given in the table 2. The nodes are networked using the Intel Express 510 T (24port) 10/100 Mbps switch.

The nodes are running the Alzza Linux 5.2a with the kernel 2.2.1.

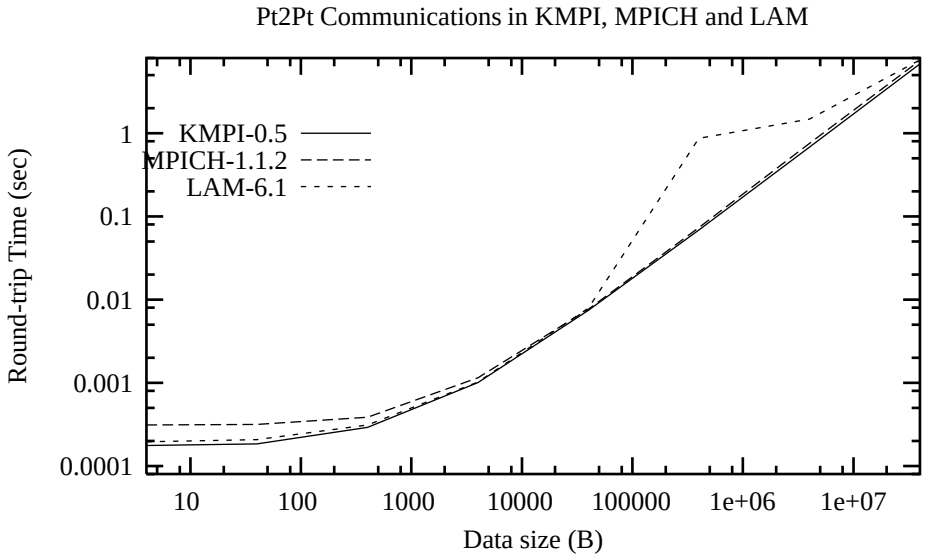
The cluster has the following parallel comutation environments:

- KMPI (version 0.5)
- MPICH (version 1.1.2)
- LAM (version 6.1)
- PVM (version 3.4)

Table 3: Average round-trip time of message in seconds

size (B)	KMPI-0.5	MPICH-1.1.2	LAM-6.1
4	0.000176	0.000311	0.000196
40	0.000184	0.000316	0.000207
400	0.000291	0.000384	0.000311
4000	0.001005	0.001146	0.001022
40000	0.007450	0.007627	0.007872
400000	0.068741	0.073035	0.865750
4000000	0.680646	0.753016	1.477553
40000000	6.798747	7.500379	7.595774

Figure 5: Average round-trip time of message in seconds



4.2 Pt2Pt communications

The performance of the point-to-point communications is tested using a ping-pong program.

In this program, one process is run on one node of the cluster and the other on another node. Process rank 0 loops calling **MPI_Send** with destination rank 1 followed my **MPI_Recv** from source node 1. Process rank 1 loops calling **MPI_Recv** with source rank 0 followed by **MPI_Send** with destination rank 0.

The send and receive is done 100000 times for 4,40,400,4000 and 40000 byte data and 1000 times for 400000, 4000000 and 40000000 byte data.

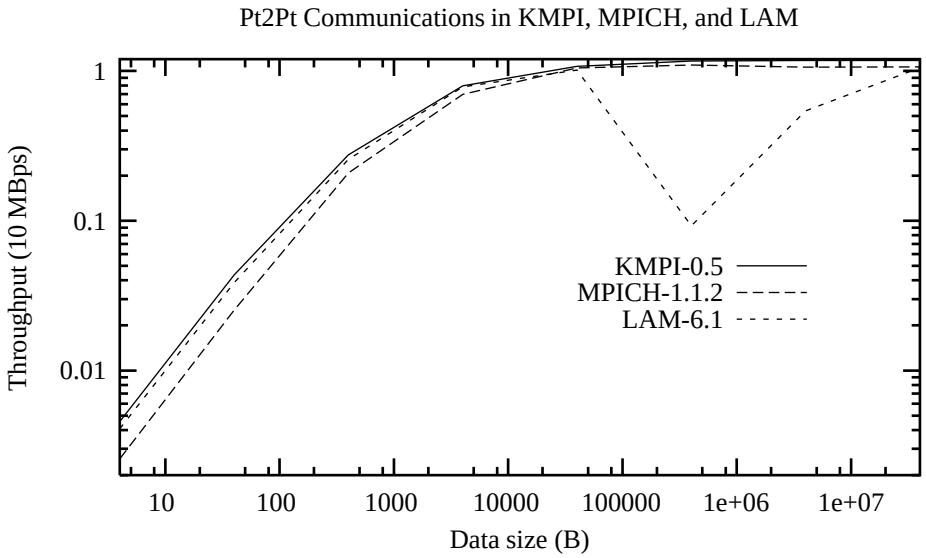
This program has be run more than 10 times for KMPI, MPICH and LAM. In the following tables, the numbers are those of a typical run. The average round-trip time is shown in the table 3. This table shows that KMPI-0.5 is up to almost two times faster than MPICH-1.1.2 for small messages and approximately 10 % faster for large messages. KMPI-0.5 is more than 20 % faster than LAM-6.1 for small messages and much faster for large messages.

The numbers are plotted in the figure 5.

Table 4: Average throughput of message in 10 M Bps. Here M does not mean 2^{20} but 10^6.

size (B)	KMPI-0.5	MPICH-1.1.2	LAM-6.1
4	0.004557	0.002573	0.004085
40	0.043464	0.025356	0.038637
400	0.275026	0.208130	0.257219
4000	0.796073	0.698237	0.782763
40000	1.073852	1.048885	1.016239
400000	1.163781	1.095363	0.092405
4000000	1.175355	1.062395	0.541436
40000000	1.176687	1.066613	1.053217

Figure 6: Average throughput of message in 10 M Bps



The average throughput for a send or a receive obtained from these numbers is shown in the table 4. Here Mega does not mean 2^{20} but 10^6. The number 1.177 for 40000000 in KMPI-0.5 is 10 % improvement over 1.067 in MPICH and more close to the Fast Ethernet hardware limitation 1.31. LAM is worst for large messages.

The numbers are plotted in the figure 6. The figure 6 shows that the throughput is saturated beyond 100 k. This implies that beyond this point the bottleneck is not the software but the Fast Ethernet hardware.

The PMS project: Poor Man's Supercomputer

F.Csikor, Z.Fodor, P.Hegedus, V.K.Horvath, S.D.Katz, A.Piroth
Eotvos University
Budapest

Overview

Main features of PMS

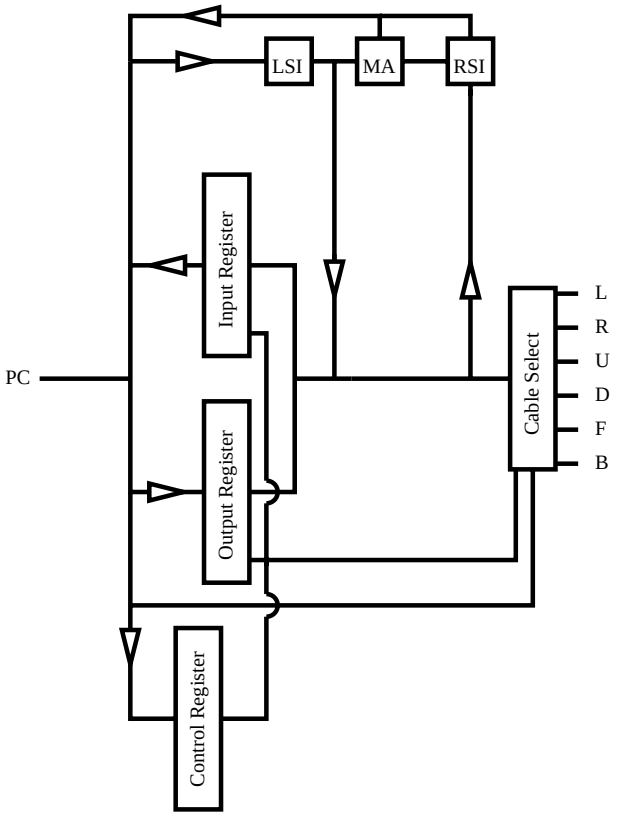
- parallel supercomputer using PC's as nodes because
  - PC's have excellent cost/performance ratio
  - PC elements can easily be upgraded
- nodes are arranged in a 3D matrix
  - convenient for physical applications
- Fast communication between nearest neighbors (similar to APE)

  Comput. Phys. Commun. 57 (1989) 285;

  A. Bartonoli et al., Nucl. Phys. B (Proc. Supl.) 60A (1998) 237;

  http://chimera.roma1.infn.it/ape.html; F. Aglietti et al., Nucl. Instrum. Meth. A389 (1997) 56.
  - Special communication cards have been developed
  - good for local problems with nearest neighbor interactions (Lattice Gauge Theories)
- Scalable

Hardware Architecture of PMS1



Block diagram of the communication card

Two 16-bit ISA cards → max. speed is 2MBytes/sec.

1. PMS CPU card (main circuits to transfer data)
2. PMS RELAY card (connectors for flat cables)

Main circuits

- Input/Output registers
- Cable select circuit
- LSI, RSI registers, MA bit
- Control register

Performance

Peak performances

Computation:

Double precision (DP):
450 MHz CPU's, 1 FP operation / 2 cycles →
225 MFlops peak performance /node
Total:
32 nodes → 7.2 GFlops peak performance

Single precision (SP):
3DNow MMX instruction set: 4 FP ops. / cycle →
1.8 GFlops peak perf. /node
Total:
32 nodes → 57.6 GFlops peak performance!

Communication:

2 MBytes/s for each channel
protocol overhead negligible → 2 MB/s real speed

Full bandwidth:
16 pairs → 32 MB/s
32 times faster than simple Ethernet

Price/Performance Ratio of PMS1

Price:

One node:
$350 for PC elements
+ $40 for communication cards

Total:

$12500

Excellent price/performance ratio:

Double Precision:

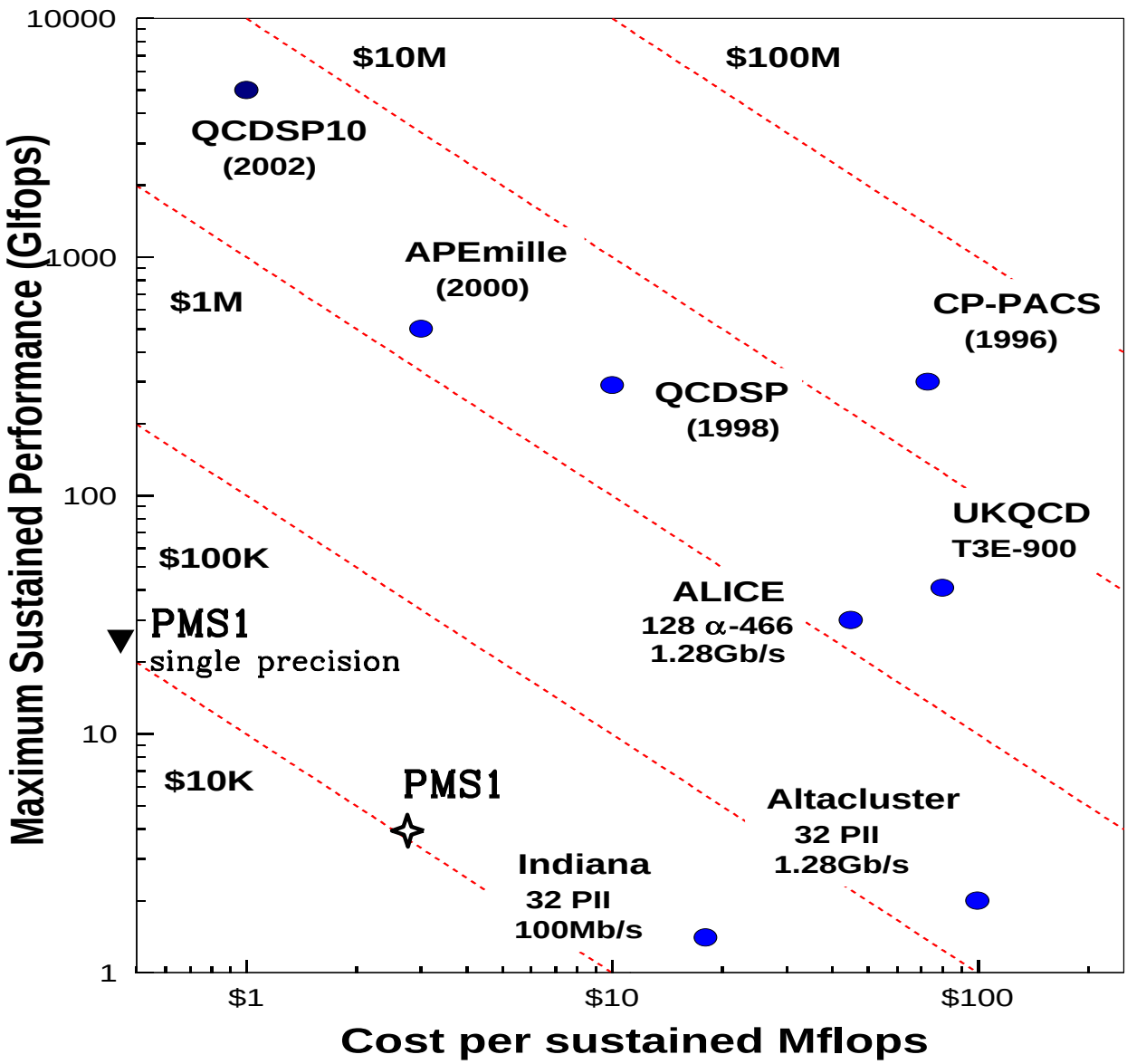
≈ $3 / MFlops

Single Precision:

≈ $0.45 / MFlops !

Comparison of different Supercomputers

N. Christ, hep-lat/9912009.



High performance cluster computing

T. Warschko
Department of Informatics
University of Karlsruhe

ParaStation2 Overview

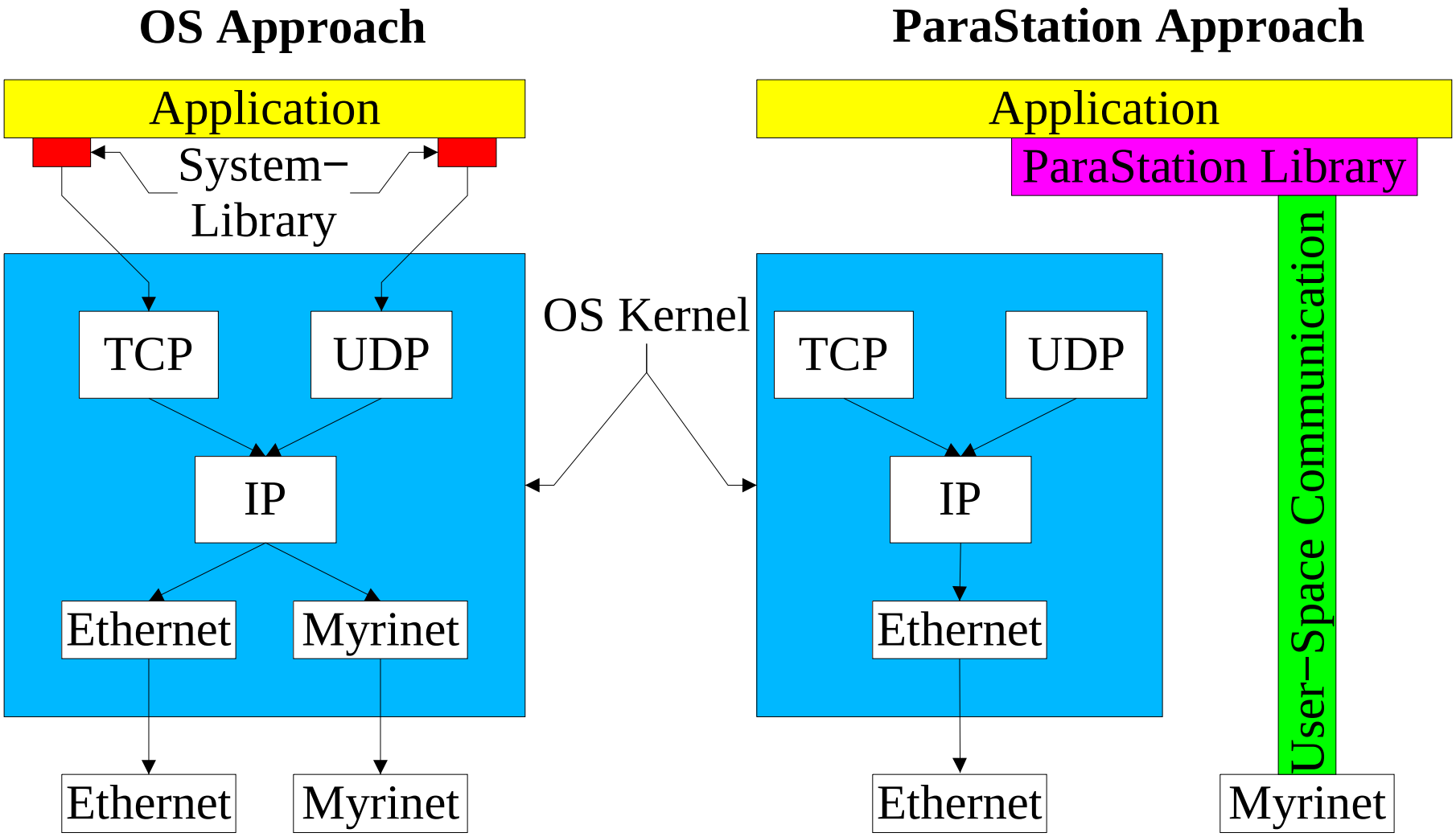
- **Hardware:**
  - Off the shelf Alpha−Stations or Intel− & AMD−PCs
  - Myrinet Interconnect
- **Operating Systems:**
  - Compaq Tru64 Unix (V. 4.0D, 4.0E, 4.0F, 5.0)
  - SuSE & RedHat − Linux (Kernel 2.0.x and 2.2.x)
- **Software:**
  - High Performance Communication System
  - ParaStation Cluster Environment

ParaStation2 Goal

Efficient Parallel Computing in Workstation Clusters

- using a second network (Myrinet) for parallel computing
- novel communication system based on the user−space communication principle (low latency, high throughput)
- mutiprocess, multiuser and multiprotocol environment
- portability through well known and standardized interfaces (Unix sockets, PVM, MPI, Java−RMI)
- integrated management and administration
- ParaStation2 one system image

User−Space Communication



ParaStation2 vs. Beowulf (1)

Both build Clusters using commodity hardware

BUT:

- Beowulf stresses the »cost argument«
  - Fast Ethernet (switched) as network (100 Mbit/s)
  - cheap to medium priced nodes
- ParaStation emphasises high−performance, scalable networks as well as administration and management of clusters.
  - Myrinet as network (1280 Mbit/s)
  - high order topology with high bisection bandwidth
  - low latency and high throughput
  - fastest nodes available (Alpha 21264, Pentium−III, Athlon)
  - integrated management and administration

Future

- Hardware:
  - new and faster processors (>700Mhz)
  - new and faster networks (> 2Gbit/s)
- Operating System:
  - Linux 2.4
  - SMP support for Alpha−Linux
- Software:
  - **ParaStation3:** Kernel based high performance communication system
  - Improvements targeting administration
  - Improvements targeting management
  - Redundancy using spare nodes
  - Fault tolerance through reconfiguration
  - Accounting

SecureNAT Linux cluster solution

A. Ugolotti
Nice
L. Genoni
Secure NAT inc.

SecureNAT developed a parallel cluster solution that allows the management of one single virtual machine, composed by the cluster nodes. This way the system administration is fully centralizzed, and the cluster has for every single node a fully coherent image (software upgrades and system configurations are as simple as if it was a single computer). Users have access to a multinodes virtual machine, like it was a multiprocessor supercomputer, with all his capabilities, but at lower costs. Kernel NFS vers.3 and the new netfilter technology included in Linux kernel 2.4.x are the key components of the cluster architecture. MPI and PVM are fully integrated as well as Load Sharing Facility (Batch base from Platform Co.) as distributed batch queueing subsystem. This cluster solution scales up to 8 master nodes in high availability and 64 computational nodes for every single master.

Linux Clustering

- System Management *desiderata*
  - *Centralized System Maps*
  - *Centralized File Systems Management*
  - *Centralized System and User Level Applications Updating*
  - *Better Security Control*
  - *Better-in-time installation, backup, recovery procedure*
  - Use *only* Standard Models (protocols, system image, administrative tools).

SecureNAT Solutions

- **SecureNAT** offers:
- Highly scalable clustering solution based on
  - Industry standard components
  - Best state of art on kernel development
  - Architectural frameworks to allow your network to grow, to meet your performance and reliability demands
  - No *black box* solutions

SecureNAT Solutions

- **SecureNAT cluster technology is based on**:
  - loose node coupled cluster model
  - each node runs his own copy of kernel
  - single system and user level application images
  - strongly centralized sub-system configuration
  - kernel NFS vers.3
  - strong security control on services

SecureNAT Solutions

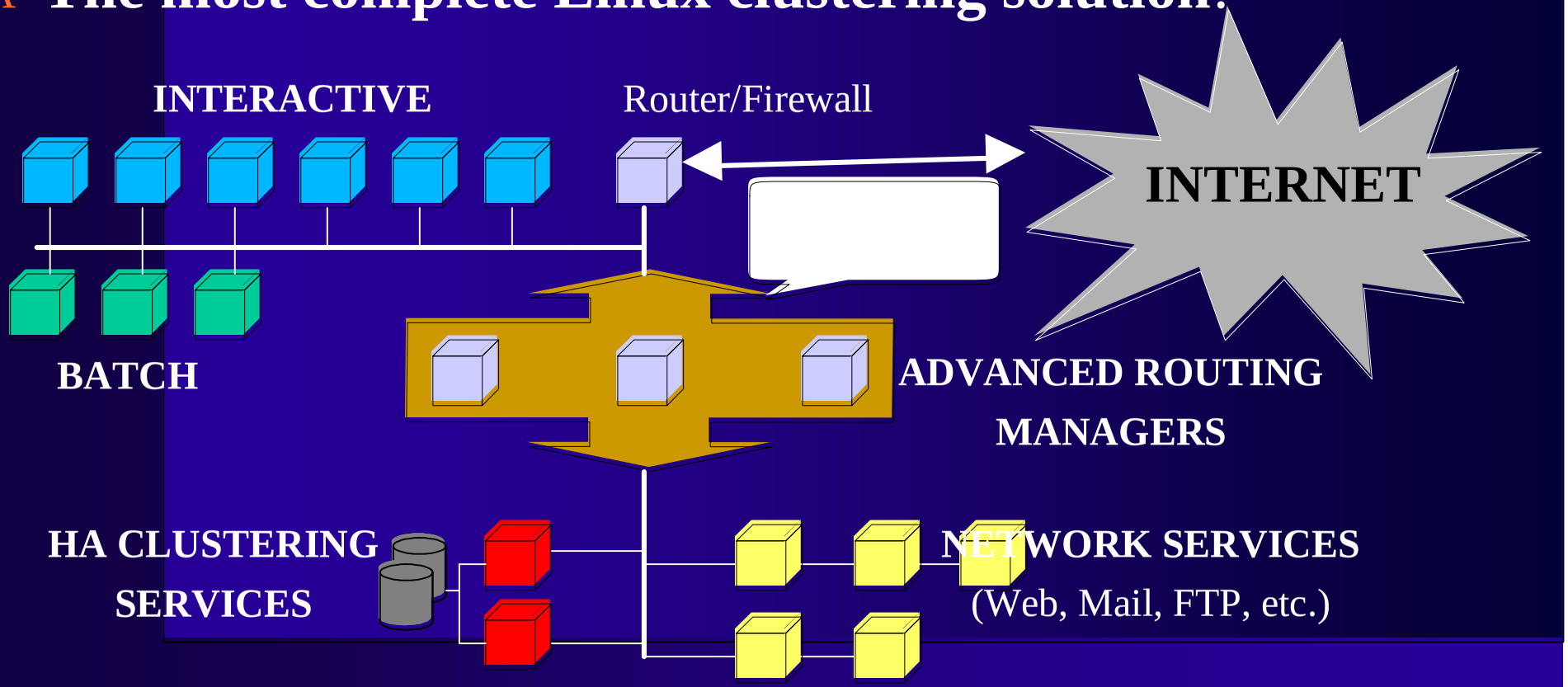
- **SecureNAT cluster solution offers**: (1)
  - an innovative *Cluster Topology Block*, that consist in:
    - 1 *Master (or more with High Availability solutions)*
    - up to 63 *nodes* per Master divided in:
      - Interactive nodes
      - Batch nodes
      - Service nodes

SecureNAT Solutions

- **SecureNAT cluster solution offers**: (3)
  - *High Availability Clustering*, based on innovative issues in the Linux kernel vers. 2.4 (to be released).
  - Platform Co. Load Sharing Facility (Batch base) as distributed batch queueing subsystem.
  - Assistance on startup.
  - Training for system administrator.
  - Advanced expertize on clustering projects.

SecureNAT Solutions

- **SecureNAT cluster solution offers:** (6)
- **The most complete Linux clustering solution!**



Benchmarking MILC QCD code on clusters and supercomputers

S. Gottlieb
Department of Physics
Indiana University

CANDYCANE

CPUS
AND
NETWORK
DO
YOUR
CALCULATION
AND
NOTHING
ELSE

$50K funding to physics dept. to build 32-node cluster

Sept. '98 build and test 4 node cluster

Oct. '98 put out bids for components

Nov. '98 last component arrives
Wednesday and Friday of Thanksgiving Break
Build 34 nodes and start cluster running.

Actual cost $693/node PII350, 4.3GB, 64MB Ram
Fast ethernet
HP Procurve switch $2,000
$25,000

Today <$400/node ~$15K
1.2 - 1.5 GF 10-12.5 $/MF

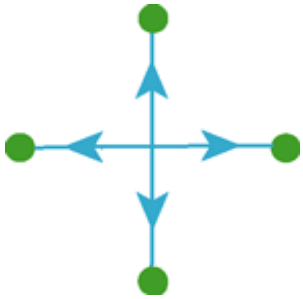
Physics Computing 2000 (next slide)

Back to Outline

Know Your Problem

Is it amenable to a parallel solution?
Where is most of the time spent?

Example: Lattice QCD
Conjugate Gradient-Sparse Matrix inversion



Blue lines with arrows represent 3 x 3 complex matrices, i.e., "links"

Green dots are 3-component complex vectors, i.e., "matter fields"

M × V

M = 18 reals = 72 bytes
V = 6 reals = 24 bytes
Total input = 96 bytes
Total output = 24 bytes

36 real multiplies
30 real adds
66 flops

1.45 bytes / flop input

0.36 bytes / flop output

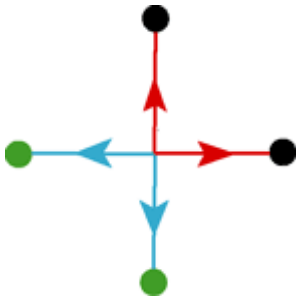
The heart of conjugate gradient is (sparse matrix) x vector. To parallelize, use domain decomposition. The system has four dimensions: three space and time. However, for simplicity, the diagrams will only show neighbors in two dimensions. Assume there are L^4 sites/node. The data is stored so that at each site there are link

matrices to connect the current site and neighboring sites in positive directions, and there is the vector matter field. In other words, if a link is pointing in a positive direction, it is stored at its tail end site. If a link is drawn with its arrow in a negative direction, it is the adjoint of the matrix with the link pointing the opposite way. That means it is actually stored at its head end.

Thus, in the diagram below, the red links and the black vectors are stored at different sites and the vectors must be gathered to the central site. On the other hand, each blue link and the corresponding green vector are stored together, but the temporary result vector stored at that site will have to be moved to the central site to accumulate all the contributions to the final result at the central site.

Because of the domain decomposition, whenever a result has to be gathered or scattered the values for sites at the edges of the domain that must be moved to another domain will put into a message and sent to the node that owns the other domain.

There is an even-odd, or red-black decomposition of the problem, so that if the central site is even, then the neighboring sites at the ends of the arrows are odd.



Let's assume that the central site is even. The strategy to overlap communication and computation is to start to gather the vectors from the odd sites, i.e, the black dots at the end of the red arrows. While those messages are passed, we start the local computation of multiplying the blue links by the green vectors for all odd sites. (This computation must be done for all four directions of links.) Once these local computations are done, we start sending the results which are at odd sites to the even sites at which they are needed. Before we can start the second stage of the computation, we must wait until the first set of messages have arrived (the black dots). We then start the second stage of the computation multiplying the red links at even sites by the vectors that have just arrived. Then, we wait until the matrix-vector products from the first stage of the computations have arrived and accumulate all the results.

We summarize the tasks during the first stage in the three steps below,

and calculate the time for each step assuming the bandwidth for passing messages is MB and the rate for matrix times vector is MF. Note that MB and MF are achieved rates, not maximum rates.

a) For every odd site, M × V in each negative direction
b) For every even site on (+) boundary, receive a vector
c) For every odd site on (-) boundary, send a vector

t_a = L^4 / 2 × 66 Flops × 4 / MF = 132 L^4 Flop / MF (microsec)

t_b = L^3 / 2 × 24 bytes × 4 / MB = 48 L^3 bytes / MB (microsec)

t_c = t_b

To completely overlap computation and communication, require that $t_a = t_b$

$$132\ L^4 / MF = 48 L^3 / MB \quad \text{or}$$

$$MB = 48\ MF / (132\ L) = 0.364\ MF / L$$

If the communications is not fast enough you must wait for message to arrive. We are assuming full duplex communication, so that steps b and c can proceed at the same time.

This is a simplification!

- **MF and MB depend on L**
- **On fewer nodes don't have neighbors in all directions.**
- **No time alloted to processor overhead required to pass message.**

Keys to Performance (next slide)

Back to Outline

Keys to Performance

- **Single node floating point speed.**
  - **quality of CPU**
  - **cache performance, size**
  - **compiler**
  - **memory bandwidth**
- **Message passing performance**
  - **latency**
  - **peak bandwidth**
  - **processor overhead**
  - **messaging software**

Single Node Performance (next slide)

Back to Outline

Benchmarks

A web site for MILC benchmarks may be found at http://physics.indiana.edu/~sg/milc/benchmark.html. Additional graphs and explanations may be found there.

The simple performance model presented above can help us guess when the communication and floating point are in reasonable balance, but it is no substitute for real benchmarks.

Key variables

- **problem size L^4/node**
- **# of CPU's or nodes**

All benchmarks are for Single precision Kogut-Susskind Conjugate Gradient. Use the back button on your browser to return to this page. These graphs are in PostScript. Other tables and axis are available from the MILC benchmark site.

- **CANDYCANE**
- **Comparison of Fast Ethernet and Myrinet on Roadrunner with one CPU per node**
- **Comparison of Fast Ethernet and Myrinet on Roadrunner with two CPUs per node**
- **Comparison of MPICH and MVIA under Fast Ethernet on CANDYCANE**
- **Cray T3E 900**
- **IU IBM SP Winterhawk II (4 way SMP**
- **Comparison of different IBM SP models**
- **SGI Origin (195 MHz), note that 250 MHz is also available**
- **Sun E10000**
- **LLNL Teracluster, with some assembly code and 2 EV67 667 MHz processors per box**
- **Comparison of several commercial machines for L=8**

Price/Performance (next slide)

Back to Outline

Price-Performance Ratios

Caveats

- **Not so easy to get prices for supercomputers. Some quotes are old.**
- **Myrinet estimates were done some time ago.**

		$/MF
Intel Fast Ethernet	(single processor)	10-13
	(dual processor)	worse
Myrinet	(single processor)	~27
	(dual processor)	~22

AMD may be better than Intel, but dual processing not yet available.

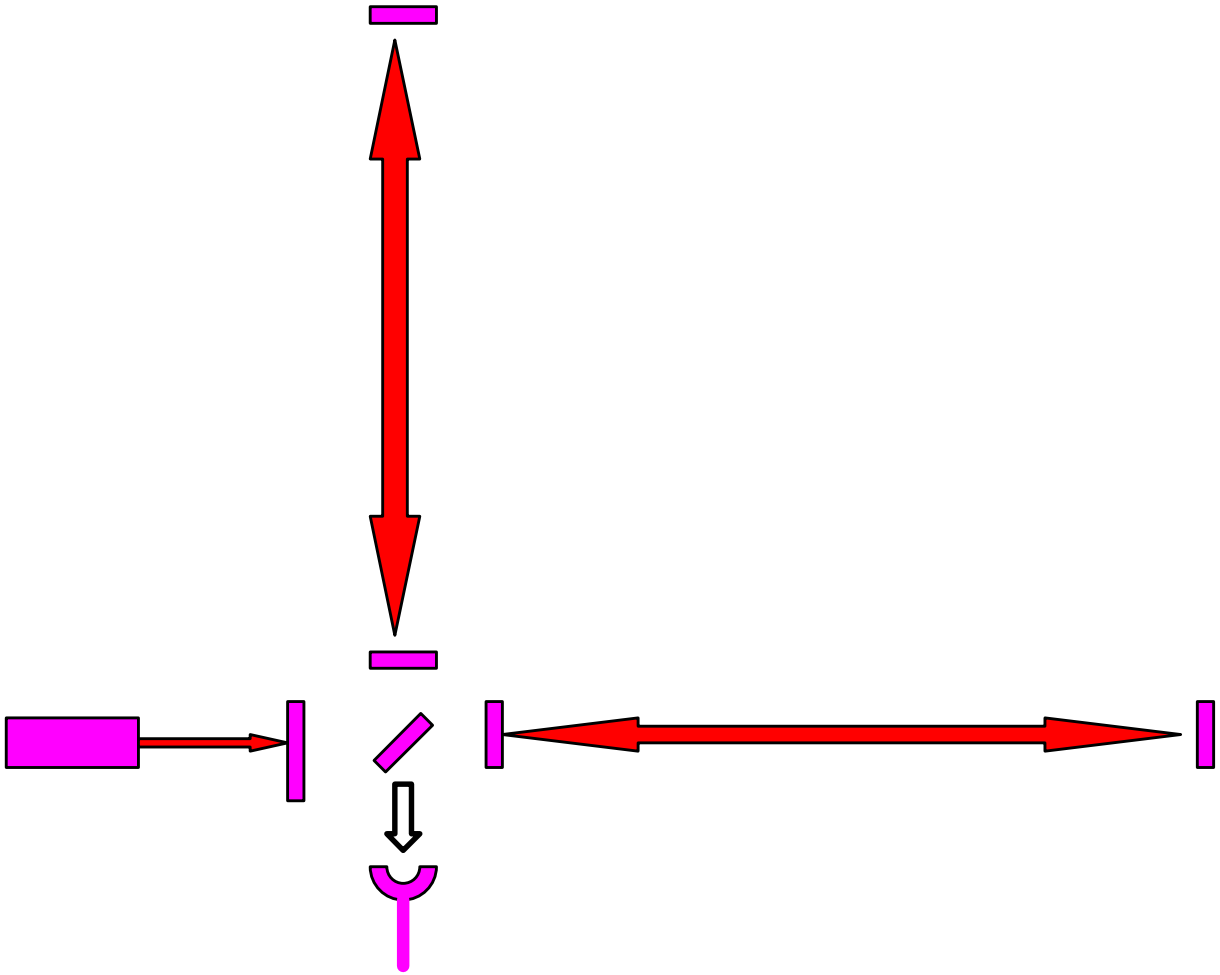
2/99	64 node SGI origin (250 MHz) million @ 113 MF/node	193
2/99	44 node Cray T3E $1.9 million @ 90 MF/node	480
	256 node IBM SP Power 3 @ 173 MF/node	166 (91 discount)
	Compaq Alpha Server SC 64CPU $2.7 million @ 280 MF/node	150

Back to Outline

Computing for VIRGO

A. Vicere'
INFN - Pisa

What is VIRGO in one slide

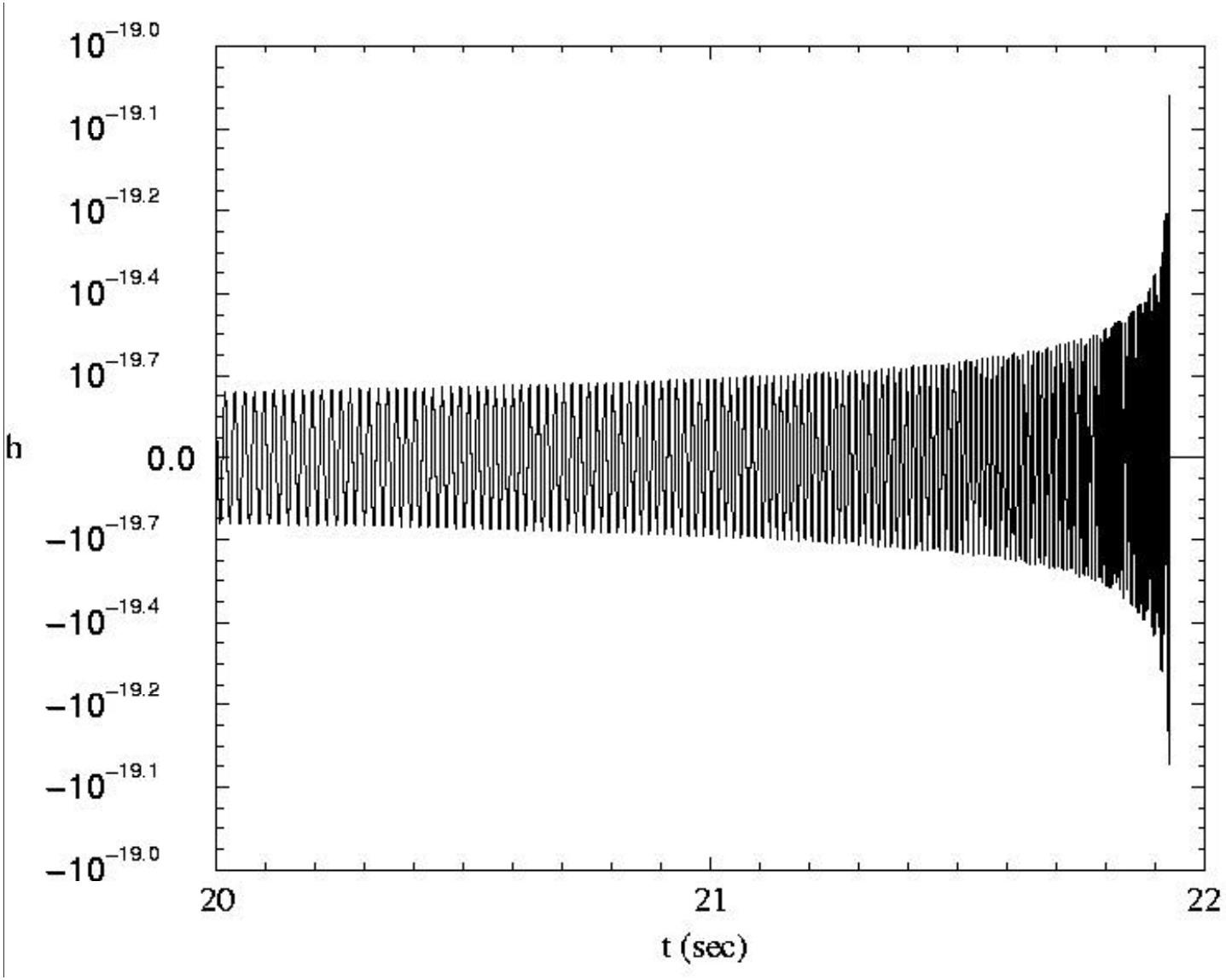


- Michelson ITF
- Fabry-Perot cavities in the arms
- Translates metric distortion *h* into modulations of the signal at the ITF out
- Sensitivity: 10^{-23} at 100 Hz
- Bandwidth: from 4 to 10000 Hz

Data analysis tasks

- Reconstruct h signal from the diode output and control signals
- Subtract the effect of known noise sources when possible
- Characterize and monitor the noise signal
- Detect burst events
  - **Of short duration (SuperNova explosions)**
  - **Of long duration (Coalescences of binary stars)**
- Analyze looking for periodic signals from spinning neutron star
  - **Talk by Cristiano Palomba**

Coalescing binaries: expected signals



- Known wave forms, depending on several parameters
- Good theoretical model -> matched filtering
- Longer duration, heavier computation cost

MF computational issues

- Embarassingly parallel algorithms
  - **Each CPU can deal with a different portion of the parameter space**
  - **Result fusion needed to look for the most probable candidate**
- Low single node efficiency
  - **FFTs and scalar products -> I/O dominates over CPU**
  - **Possible to improve, for instance matched filters can be generated on the fly**
- Good scaling
  - **To enlarge the parameter space, it is sufficient to add more nodes**
  - **Current proposed architecture: master-slave**
- Typical cost scales
  - **CPU power: ~50 Gflops sustained**
  - **Memory: ~30 Gwords for templates**
  - **I/O bandwidth ~2 Gbytes/sec**

Complications

- The analysis may require more complicated templates
  - **To take into account other parameters**
  - **To coherently analyze data also from other interferometers**
- Possible solution: a hierarchical search
  - **First layer, simplified templates. Low efficiency, low thresholds to avoid losing too many events**
  - **Second layer, more refined templates.**
- Computational issues very different
  - **Need of dynamically allocate processes in the second layer, as dictated by the first layer candidates**
  - **Need of communication between processes, to perform the likelyhood maximization exploiting the correlation between templates**
  - **Still a cluster can be a right solution, with PVM or MPI for task instantiation and synchronization.**

VIRGO data analysis for periodic sources of gravitational waves

C. Palomba, S. Frasca, E. Majorana, C. Palomba, F. Ricci,
G. Amati, F. Massaioli
Department of Physics
University of Roma I "La Sapienza"

The detection of periodic gravitational waves, like those emitted by spinning symmetric neutron stars, is one of the main purpose of the VIRGO interferometer. However, due to the very low amplitude of the expected signals, which, tipically, will be buried in the detector noise, this comes out to be a computationally heavy problem. Moreover, the received signals are not monochromatic, due to the Earth motion (Doppler effect) and to the intrinsic variations of the source rotation frequency (spin-down).
As a consequence of the huge resulting parameter space the optimal filtering technique, based on the matched filter, is not feasible.
The proposed strategy consists of a hierarchical method, realized alternating coherent (based on FFT) and incoherent (based on the Hough Transform) steps. The basic idea of the procedure is that of performing an optimal analysis only on a limited portion of the parameter space. In the coherent stage a short FFT data base (SFDB) is built and from this a frequency-time peak map is obtained. Then, the Hough Transform (HT) is applied, resulting in a set of candidate sources. For the narrower parameter space corresponding to these candidates, a more refined FFT data base is produced (i.e longer FFTs), on which the HT is applied again. These two steps are repeated until the total observation time, and then the maximum sensitivity, is reached. This procedure is embarassingly parallel (we can operate independently on different frequency bands and portions of the sky) and our present goal is to build a computer farm with a peak computing power of 1Tflop. The greater is the computing power available and the larger is the parameter space which can be explored.

Periodic sources

Periodic gravitational waves are emitted in many processes involving single stars or binary systems.

For a single rotating deformed star the gravitational signal emitted is:

$$h(t) \propto h_0 \cos(2\Omega t)$$

Ω: angular velocity of the star

$$h_0 = \frac{4G}{c^4 r} I \Omega^2 e$$

I : momentum of inertia; e : ellipticity; r : distance

We are interested in periodic sources radiating in the Virgo sensitivity band.
These involve compact stars, namely NEUTRON STARS.

About 10^9 neutron stars are expected to exist in the Galaxy, but only ≈ 1000 have been detected, most as PULSARS.

We are interested in a *all-sky, all-frequency* search (*blind* search).
We do not know the exact spin-down law.
We account for this by a power expansion of the frequency around a constant value:

$$f(t) = f_0\left(1+\sum_{k=1} f_k \cdot t^k\right)$$

where f_k are the spin-down parameters. In general, we will consider only f_1, f_2 .

A huge parameter space must be explored: (a, d, f_0, f_1, f_2) .

For an all-sky search for frequencies below 200Hz and pulsar spin-down age t > 1000 years, the number of points in the parameter space is $\approx 10^{21}$ (for each frequency in the band) and, if we apply the matched filter

$8 \cdot 10^{14}$ *Tflops* are needed for 4 months of data!!!

Alternatives are necessary!

sub-optimal searches — *reduction of the parameter space*

The hierarchical procedure

Off-line analysis. It adapts the sensitivity to the available computing power.
The steps are:

1. construction of the short FFT data base (SFDB);
2. construction of the frequency-time peak map (through $|FFT|^2$);
3. data quality inspection;
4. incoherent search (Hough Transform) based, the first time, on the SFDB (HT);
5. construction of a more refined FFT data base, based on the estimated parameters of the candidates coming from step 3. (CS);
6. steps 4. and 5. are repeated until the full length of data is reached.

- Short FFT data base:

subsamplings of the detector output. The time duration T_{FFT} of the FFT is such that the frequency shift is lower than a frequency bin (40 min. @100 Hz). Division in different frequency bands.

Short computimg time but large occupation: e.g. $\approx$ 80GB/yr for the band 100-300 Hz.

- Data quality: identification of periods or bands of disturbance (noise is non-stationary).

Incoherent search: the Hough Transform

(collaboration with M.A. Papa & A. Sintes, AEI, Golm)

It starts from the frequency-time peak map (incoherent as it uses *periodograms*).

It is a transformation between the data set (peaks in the f-t map) and the set of parameters of the signal one is trying to identify.

For each peak (i.e a frequency f^*) one determines the locus of points on the sky where the source which produced that peak could be: it is given by

$$\cos(j) = \frac{c}{v} \cdot \frac{f^* - f_0}{f_0}$$

⇒ a circle on the celestial sphere, centered on the direction of the detector velocity vector and with radius J .

Due to the frequency discretization ($df = 1/T_{FFT}$), we a have an annulus.

Considering a set of peaks (in one or more periodograms) a partial Hough map is produced.

The amplitude of the minimum detectable signal is $\propto T_{FFT}^{-1/4}$.

Computing power: $CP \propto N_{FFT} \cdot N_{tot}$

N_{FFT} : number of periodograms; N_{tot} : number of points in the parameter space:

$$N_{tot} = N_f \cdot N_p \cdot N_{sd} \propto T_{FFT}^4 \Rightarrow CP \propto T_{FFT}^3$$

This means that in order to gain a factor of 2 in sensitivity, we must to increase T_{FFT} of a factor 16 and correspondingly the computing power increases of a factor $4 \cdot 10^3$.

The basic idea of the hierarchical procedure is to gain this factor of 2 considering a much smaller parameter space so that the corresponding computing power is of the same order of magnitude.

step	T_{FFT} [s]	c	N_{tot}	CP [Gf]
SFDB	$3.65 \cdot 10^3$	0.16		
HT1			$\approx 3 \cdot 10^{15}$	≈ 900
CS1	$5.84 \cdot 10^4$	0.32		
HT2			$\approx 10^{15}$	≈ 250
CS2	$9.34 \cdot 10^5$	0.64		
HT3			$\approx 10^{15}$	≈ 250
CS3	10^7	1		

CP estimations for $t = 10^3$ yr, $f_0 = 300$ Hz, $B = 200$ Hz

An overview of SCore 3.0 cluster system software

Y. Ishikawa
Real World Computing Partnership
Tsukuba

RWC
Real World Computing Partnership
RWCP
Funded by Ministry of International Trade and Industry, JAPAN

SCore 3.0, a system software for a seamless parallel and distributed computing environment, is now ready to use !

Real World Computing Partnership, RWCP, is pursuing research on a seamless parallel and distributed computing environment where the users are not aware of the underlying distributed environment, but use the network as if it were a single system. To realize such an environment, a new network architecture, communication libraries, an operating system, parallel programming languages, and parallelizing tools have been developed at RWCP.

The Parallel and Distributed System Software Laboratory (PDS Lab.) of RWCP developed RWC PC Clusters I and II, Alpha Cluster I, and their cluster system software called SCore for a first step toward the seamless parallel and distributed environment. The system has been demonstrated at SC'96, SC'97, and SC'98.

At SC'99, our new cluster and system software, called SCore Cluster I and SCore cluster system software version 3, are shown to demonstrate the research results of the second step toward the seamless parallel and distributed environment. This brochure describes an overview of the latest system.

Figure 1. SCore Cluster I

SCore Cluster I

The SCore Cluster I shown in Figure 1 is a research platform to develop the seamless parallel and distributed environment. The cluster is a heterogeneous cluster in a sense that two different processors, Intel Pentium and Compaq Alpha, are connected by different types of networks: Myrinet, a 100 Mbps Ethernet, and a gigabit Ethernet. The hardware specification is shown in Table 1.

Table 1. SCore Cluster I Specification

16 Intel Pentium Nodes
Intel Dual Pentium III 500 MHz
512 MB of main memory
9 GB of hard disk
16 Compaq Alpha Nodes
Compaq Alpha 21264 500 MHz
512 MB of main memory
9 GB of hard disk
Networks
Myrinet
Gigabit Ethernet
100 Mbps Ethernet

SCore 3.0 System Software

The SCore System is middleware implemented on top of Unix without any kernel modifications. Supported Unix kernels include Linux, NetBSD, and Sun OS. As shown in Figure 2, the SCore System software consists of a global operating system called SCore-D, a communication facility called PM, MPI implemented on PM called MPICH-SCore, and a multi-threaded programming language called MPC++. The distinguishing features of the SCore system software are summarized as follows:

Single System Image

Using SCore 3.0, users are not aware whether or not a system is a cluster of single/multi-processor computers or a cluster of clusters. A parallel application and an ordinary UNIX command may run by just specifying a computer node group of such a cluster. A Unix command runs in the SIMD execution style.

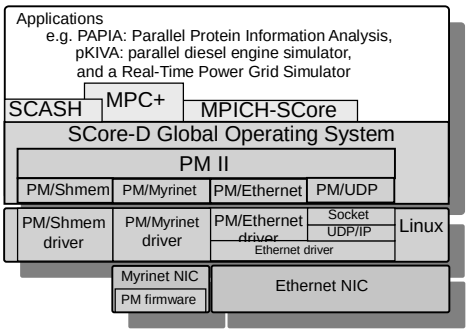


Figure 2. SCore 3.0 Software Architecture

Multiple Network Support

The PM high performance communication library is a dedicated communication library for cluster computing. PM was designed for the Myrinet network. It achieved more than 100 Mbytes of bandwidth and 15 micro second round trip time for 8 byte messages on Pentium Pro processors.

To adapt it to many types of networks, the PM II API was defined. Unlike other high-performance low-level communication libraries such as AM, FM, and BIP, PM II allows a program to communicate on different types of networks. The PM II drivers for Myrinet, Ethernet, and UDP have been implemented.

Seamless Programming Environment

The *Hmake* command, provided by SCore version 3, enables us to compile a program on heterogeneous computers. It generates binaries for underlying different types of execution environments such as Intel Pentium and Compaq Alpha processors. Using the MPC++ Multi-Threaded Template Library, a program runs on such a heterogeneous processor environment.

Heterogeneous Programming Language

The MPC++ *Multi-Threaded Template Library (MTTL)* specifies parallel description primitives implemented by the C++ template feature such as remote function/object invocation and global pointer. The MTTL runtime system of SCore version 3 supports both a heterogeneous and homogeneous execution environment at runtime.

If an MPC++ program contains remote function invocations whose arguments are only basic data types, the program runs on both the homogeneous and heterogeneous environment without any changes. If an object is passed to a remote function in the remote function invocation, marshalling and un-marshalling functions must be programmed. This drawback will be overcome using the MPC++ meta-level architecture which enables users to extend the C++ semantics or transform source to source.

Multiple Programming Paradigms

Unlike other cluster software, SCore not only supports the message passing paradigm, but also supports the shared memory parallel programming paradigm and the multi-threaded parallel programming paradigm.

Message passing paradigm

MPICH-SCore enables us to run an MPI program on a cluster of single processor computers, a cluster of multi-processor computers, or a cluster of clusters without any program modifications.

Shared memory paradigm

SCASH is a software distributed memory system implemented on top of the PM II API. SCASH is implemented on top of Linux without any kernel modification. It provides a multiple writer protocol using the release consistency model with both the page invalidation and the page update protocols. By integrating an OpenMP compiler, called Omni, developed at the Parallel and Distributed Performance Laboratory, the shared memory parallel programming paradigm will be supported.

Parallel Programming Support

Real-time process activity monitor

SCore-D allows us to watch parallel process activity in real-time as shown in Figure 3. Each bar represents the processor utilization on each node.

Deadlock detection

Since SCore-D knows the global status of a parallel process, i.e., each process status and communication buffer status, it can detect whether or not the parallel process is deadlocked.

Automatic debugger attachment

In most cluster systems, when a process dies, there is no chance to invoke a debugger interactively. SCore attaches the gdb debugger to the target parallel process when an exception signal is detected.

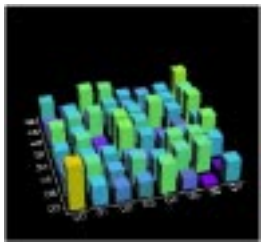


Figure 3. Real-Time Load Monitor

Fault Tolerance

Preemptive checkpoint

To enable a checkpoint function, the SCore checkpoint facility does not require any additional API and does not assume any parallel programming languages. The user parallel process image is stored to a local hard disk for checkpoint by a user specified interval.

Parallel process migration

Using the checkpoint function, a parallel process may migrate to another group of computers in SCore.

Flexible Job Scheduling

To utilize processor resources and to enable an interactive programming environment, the SCore-D global operating system multiplexes parallel processes[1] in processors' space and time domains simultaneously.

Gang scheduling

Parallel processes are gang-scheduled when multiplexed in the time domain.

Batch scheduling

Batch scheduling is implemented by setting an infinite value for the scheduling slice time.

Technology Transfer

RWCP transfers technologies developed at RWCP to industry. An example is a real-time power grid simulator developed by Mitsubishi Industrial Electronics Systems Laboratory and RWCP. It simulates in real time, various phenomena that might occur in an electric power transmission system.

If this simulator can be used for testing new protection and control devices before installation, it should help supply electricity in a more stable and efficient manner. Because real-time power grid simulators need to perform parallel processing every 50 microseconds, they have had to be built on very expensive and dedicated computers in the past. The system software running on a PC cluster developed by RWCP enabled us to achieve such a simulator using many inexpensive PCs. In this way, low latency and high bandwidth communication, realized by PM, enables us to develop applications using a low cost PC cluster that have needed dedicated parallel computers in the past.

SCore 2.4, the previous version of SCore, was released in the fall of 1998. More than 200 sites have obtained the software. The University of Bonn in Germany has developed the Parnass2 Cluster using the SCore software. They achieved a performance of 29.6 Gflops in the Linpack Benchmark. This performance ranks at 362 in the Supercomputer Top 500 issued in June 1999.

http://www.top500.org/lists/1999/06/top500.list.html

For more information, please visit the following URL:

http://www.rwcp.or.jp/lab/pdslab/

[1] A parallel process is a set of processes that are execution entities derived from a SPMD program.

RWC

History of RWC Cluster Computing Research

1995

RWC Workstation Cluster I (5 SS20s 75 MHz)

8 PM prototyped.

MPC++ MTTL developed.

12 SCore-D developed.

1996

RWC Workstation Cluster II (36 SS20s 75 MHz)

3

6 PM 1.0 achieves over 100 MB/s bandwidth on the prototype of RWC PC Cluster I.

9

RWC PC Cluster I (32 Pentiums 166 MHz)

10

11 Demonstration at Super Computing '96.

1997

SMP Cluster & COMPaS (4 & 8 Quad Pentium Pros 200 MHz)

2 MPICH-PM developed.

9 PM 1.2 Zero copy message transfer facility developed.

RWC PC Cluster II (64 Pentium Pros 200 MHz)

10 MPICH-PM supports zero copy transfer.

GigaE PM testbed (Essential PCI Gigabit Ethernet)

11 Demonstration at SC '97.

12

1998

RWC PC Cluster IIa for PAPIA (64 Pentium Pros 200 MHz)

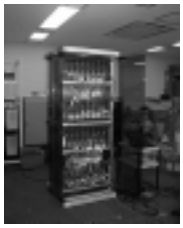
1 SCore Version 1.0 release

3 SCore Version 2.0 release

4 PAPIA WWW service started.

Expanded RWC PC Cluster II (128 Pentium Pros 200 MHz)

5 Real-Time Power Grid Simulator prototyped.



RWC Alpha Cluster I (32 Alpha 21164s 500 MHz)

6 SCore Version 2.2 release

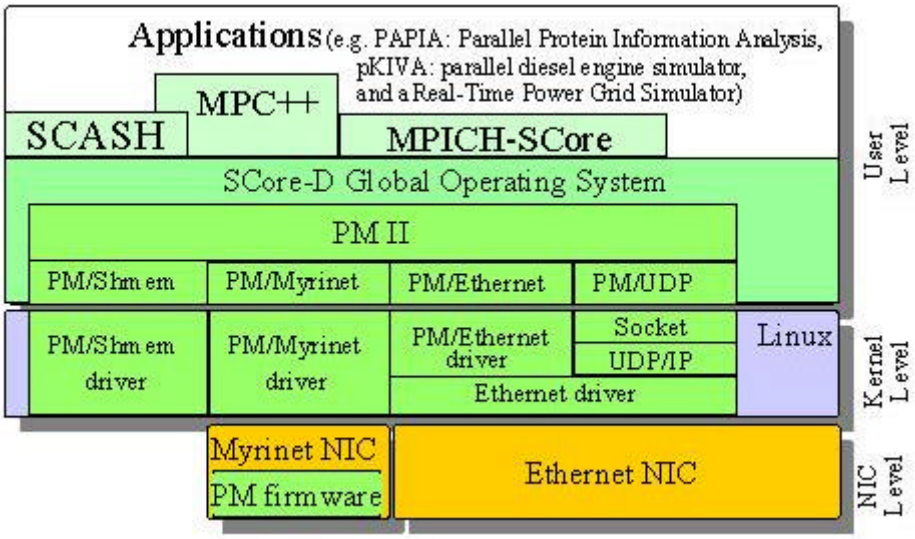
7 MPICH-PM/CLUMP developed.

8 SCASH developed.

9 GigaE PM developed.

Parallel and Distributed System Software Laboratory

Software Architecture



As shown in the above diagram, the SCore Cluster System Software consists of the following components:

- **PM II**
  PM II is a low level communication library for cluster computing. The PM II API (Application Program Interface) is carefully designed so that many types of networks and shared memory are accessed in a uniform way for cluster computing. PM II drivers for Myrinet, Ethernet, UDP, Shmem have been implemented.

- **SCore-D**
  A user-level global operating system to utilize cluster resources, such as processors, networks. Various PM network devices are managed and utilized by SCore-D. SMP clusters, heterogeneous clusters as well as homogeneous clusters are supported. SCore-D provides a flexible parallel job scheduling scheme. User parallel jobs are time-shared and space-shared simultaneously and effectively. Preemptive consistent checkpointing is provided by SCore-D, and any user programs can be checkpointed by only specifying a checkpoint interval time.

- **SCASH**
  A software DSM (Distributed Shared Memory) system using PM II. It employs the Lazy Release Consistency model with both update and invalidate protocols.

- **MPICH-SCore**
  MPICH-SCore, the former MPICH-PM/CLUMP, is based on the MPICH MPI library using PM II for interprocess communication. MPICH-SCore is a high-performance MPI on workstation, PC, and SMP clusters. For more information, see the MPICH-SCore Manual and Documents page.

- **MPC++**
  A multi-threaded C++ using the C++ template feature. It provides synchronous/asynchronous remote function invocation facilities, synchronization structures, global pointer, and other such functions. For more information, see the MPC++ Level0: Multi-Threaded Template Library page.

  MPC++ also supports a meta-level architecture to achieve an extensible and modifiable programming language system. The architecture makes it possible to incorporate new

optimizers into the compiler. Library designers are able to provide an optimizer specific to their class/template library in the library header file. The library user may use such a high performance library by including the header file. The MPC++ meta-level architecture is not included in this release. SCore-D is written in MPC++.

- **SCOUT**
  The `scout`(1) program provides a SIMD-style remote UNIX shell environment. Any UNIX command can be executed on cluster hosts in parallel.

In SCore 3.1, the following servers are run to maintain the system:

- **Cluster Database Server**
  All the cluster information required by the SCore Cluster System Software are maintained by the `scoreboard`(8) server.

- **Compute Host Lock Server**
  The Compute Host Lock Server (`msgbserv`(8)) provides a locking facility that prevents other users from using cluster hosts. The `msgb` command allows cluster users to browse the current status of a cluster. Users of parallel application programs are assumed to lock hosts via the `msgbserv` server. This locking management is done automatically by user programs when they are linked with the SCore runtime library.

- **System Information Server**
  The `scbcast`(8) server obtains system information generated by the SCore-D operating system and distributes it to clients. `scbcast` must be invoked before starting the SCore-D operating system and clients.

See the following sections for an introduction to the available environments:

- The SCore Single-User Environment
- The SCore Multi-User Environment

Modeling biological systems by computer simulations: molecular dynamics and (hybrid) Monte Carlo

G. La Penna, S. Letardi, V. Minicozzi, S. Morante, G.C. Rossi, V. Salina, Z. Aiudi, M. Freda, A. Amadei, A. Di Nola
Department of Physics and INFN
University of Roma "Tor Vergata"

After discussing the notion of modelling for the description of biological systems, as opposed to the well established reductionistic strategy developed in world of micro-physics, we will try to make a case for the use of numerical simulations in the study of such complex systems as the biological ones. In particular in this talk we will illustrate the point by presenting the results of a large-scale numerical investigation of structural and thermodynamic properties of a model of cell membrane, simulated as a bilayer of phospho-lipid molecules which are in turn described by a simple atomistic approximation. The study was performed by carrying out extensive Molecular Dynamics simulations (i.e. simulations in the microcanonical (NVE) ensemble) of systems of various sizes (from 2 x 18 to 2 x 256 molecules), over a fairly large set of temperatures and densities, both in vacuum and in presence of (small amounts of) water, using parallel platforms and more standard serial computers. Depending on the dimension of the system, its dynamics was followed for physical times that go from hundred picoseconds for the largest systems to few nanoseconds for the smallest ones. The bilayer appears to be stable even in the absence of water and neglecting Coulomb interactions in a wide temperature-density region. The magnitude of the region of physical parameters that was explored has allowed to analyze in detail the phase diagram of the system in the temperature-density plane and to expose structural changes, which may be interpreted as the system passing from a crystal to a gel phase and from the latter to an isotropic phase. Possible extensions of the philosophy of this approach to the study of the molecular bases of what we may suggestively call "local recognition mechanisms", i.e. the microscopic mechanisms which govern processes such as protein docking, receptor-epitope interactions,.etc., are briefly discussed.

PLAN OF THE TALK

COMPUTER SIMULATIONS AND MODELLING OF BIOLOGICAL SYSTEMS

- WHY A NUMERICAL APPROACH
- WHAT KIND OF SYSTEMS
  - BILAYERS, MONOLAYERS, ...
  - MICELLES IN A SOLVENT
- MODELS
  - ATOMIC DESCRIPTION
  - WHAT CAN WE LEARN / EXPERIMENTS
- ALGORITHMS AND MACHINES
  - { MD, MC, HMC / CAR-PARRINELLO (?) } — { SCALAR : PC's, ∝-STATIONS / PARALLEL: CRAY, APE }
- PROBLEMS
  - FINITE SIZE SCALING
  - N^2 - COMPLEXITY
- WHAT'S NEXT
  - BILAYER + SMALL MOLECULES (DRUG)
  - RECOGNITION MECHANISMS
  - TUNING LJ PARAMETERS BY PRESSURE
  - ...
- CONCLUSIONS

BIOLOGY
PHYSICS } THE VIEW POINT OF A (THEORETICAL) PHYSICIST

* COMPARE THE SITUATION IN THE

STUDY OF BIOLOGICAL SYSTEMS

- COMPLEX STRUCTURES GOVERNED BY UNKNOWN MACRO-LAWS
- POWERFUL AND CHEAP EXPERIMENTAL TECHNIQUES
- HUGE AMOUNT OF DATA
- INADEQUATE MODELS; POOR UNDERSTANDING OF THE "MICRO-TO-MACRO" TRANSITION

* AND AT THE OTHER EXTREME

ELEMENTARY PARTICLE PHYSICS

- (SUPPOSEDLY) "SIMPLE" SYSTEMS GOVERNED BY COMPLICATED KNOWN MICRO-LAWS
- VERY EXPENSIVE EXPERIMENTS
- VERY FEW NEW EXPERIMENTAL DATA (TO DAY)
- RATHER GOOD MODELS (ALMOST THEORIES)

PHYSICS (UNTILL VERY RECENTLY) HAS ALWAYS FOUND ITS WAY BY PROGRESSIVELY MOVING TOWARDS MORE AND MORE ELEMENTARY STRUCTURES

GAS → ATOMS → NUCLEONS → QUARKS → ?...

RADICAL REDUCTIONISM:

FUNDAMENTAL LAWS GOVERN ELEMENTARY OBJECTS

THIS ATTITUDE HAS BEEN VERY FRUITFUL IN THE "PARADIGMATIC" HIGH ENERGY PHYSICS DEVELOPMENT, BUT NOT QUITE IN OTHER EMERGING FIELDS OF INVESTIGATION

- DYNAMICAL SYSTEMS { WEATHER FORECASTING; CATALYTIC REACTIONS; FLUIDODYNAMICS (TURBULENCE)

PARADIGMA: NON-LINEARITY * CHAOS

- DISORDERED SYSTEMS GLASSES

PARADIGMA: FRUSTRATION - REDUNDANCY

- BIOLOGICAL SYSTEMS

PARADIGMA: ???

6

CONCERNING BIOLOGICAL SYSTEMS
UNDER THE HEADINGS OF (NUMEROSITY +) VARIETY
WE CAN FILL UP THE FOLLOWING LIST

1) PROTEINS
- $10^2 - 10^3$ amino-acids
- 10^4 atoms
- 20^{100} different sequences
- (only $\sim 10^8$ expressed: proteins)

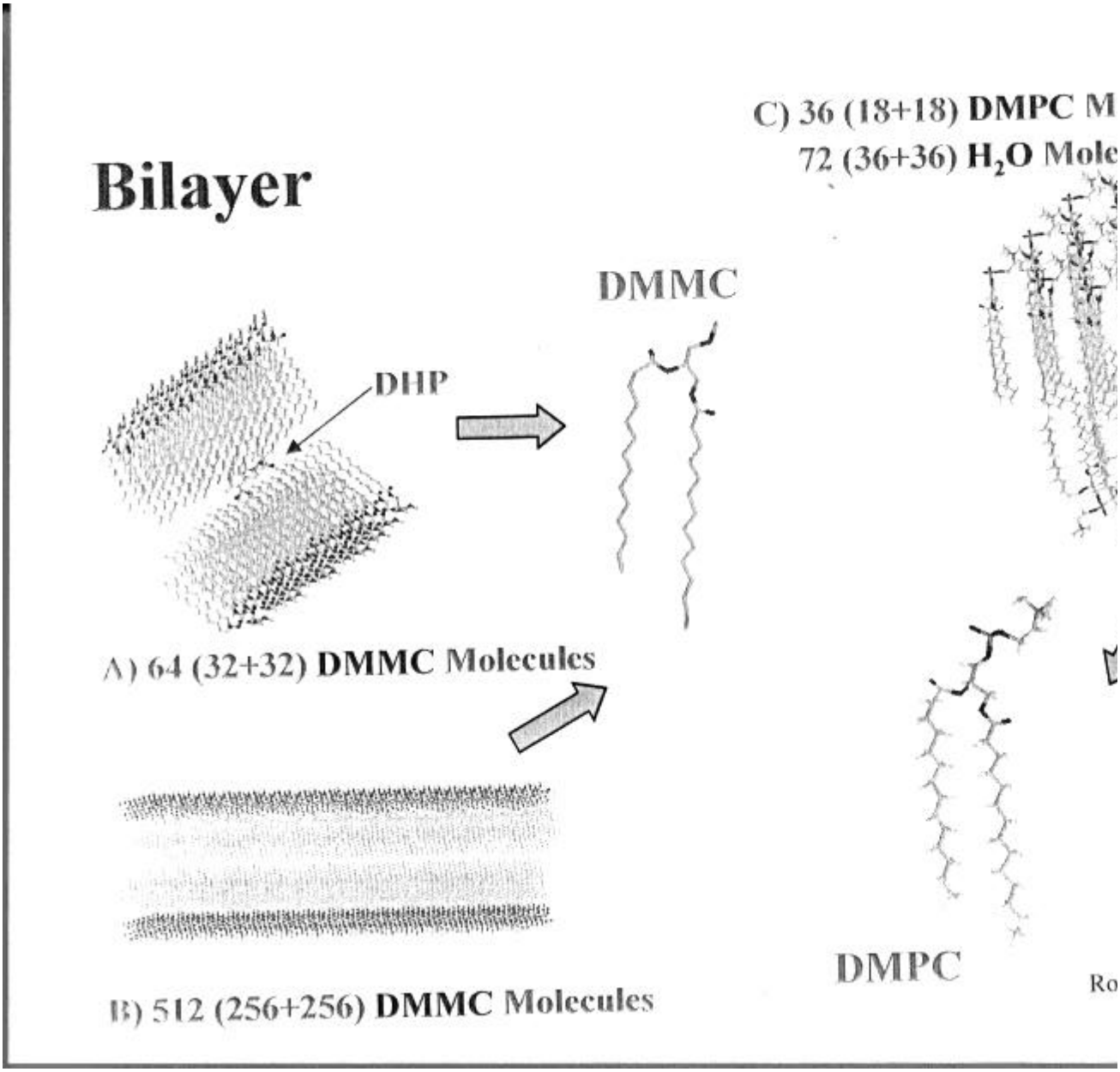
2) IMMUNE SYSTEM
- 10^6 actual repertoire of Ab's
- 10^7 available repertoire
- 10^8 lymphocytes

3) BRAIN
- 10^{10} neurons
- $10^{13} - 10^{14}$ synapses

4) GENOMA
- 10^9 bases (A C G T)
- 4^{10^9} possible "genomas"
- (only 10^{60} expressed? EIGEN)
  1 mut/sec

IT IS NOT AT ALL CLEAR WHICH ONE OF THE ABOVE SYSTEMS IS THE MOST COMPLEX ONE.

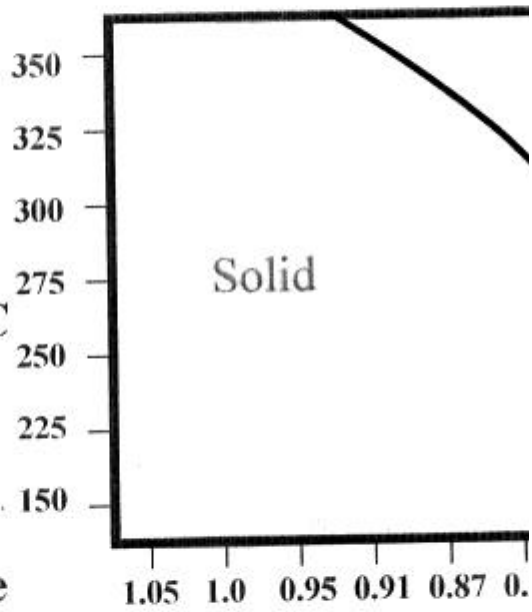
- MODEL VS FUNDAMENTAL LAWS
- EXPERIMENTS VS REPRODUCIBILITY


Bilayer
DMMC
DHP
A) 64 (32+32) DMMC Molecules
B) 512 (256+256) DMMC Molecules
C) 36 (18+18) DMPC M
72 (36+36) H2O Mole
DMPC

Bilayer

DMMC

- Stable even in absence of water for T<325 K and $\rho/\rho_0 > 0.79$
- Agreement between DMMC & DMPC systems
- Clearly visible Solid-Gel transition
- Gel-Liquid transition (finite size effect, water)
- Qualitative agreement with experimental results



R

CAN WE HOPE TO STUDY QUANTITATIVELY THE PHYSICAL INTERACTIONS UNDERLYING THE DIFFERENT LEVELS OF THE BIOLOGICAL "RECOGNITION MECHANISMS"?

AFTER ALL, EVERYTHING IS (QUANTUM) ELECTRODYNAMICS

- A FIRST STEP

  - CROSSING POINTS IN A FOLDED PROTEIN

STATISTICAL STUDY	NUMERICAL STUDY
SWEEP THROUGH DATA BANKS LOOKING FOR PREFERRED CROSS-MATCHING AMONG AMINO-ACIDS	USE SOME KIND OF SIMULATED TEMPERING IN ORDER TO STUDY THE ENERGY LANDSCAPE OF AMINO-ACIDS CROSSING POINTS

- A MORE AMBITIOUS TASK

  - APPLY A SIMILAR PHILOSOPHY TO THE CASE OF EPITOPE-RECEPTOR PAIRS

The Grid project and the LHC computing challenge

P. Capiluppi
University of Bologna and INFN

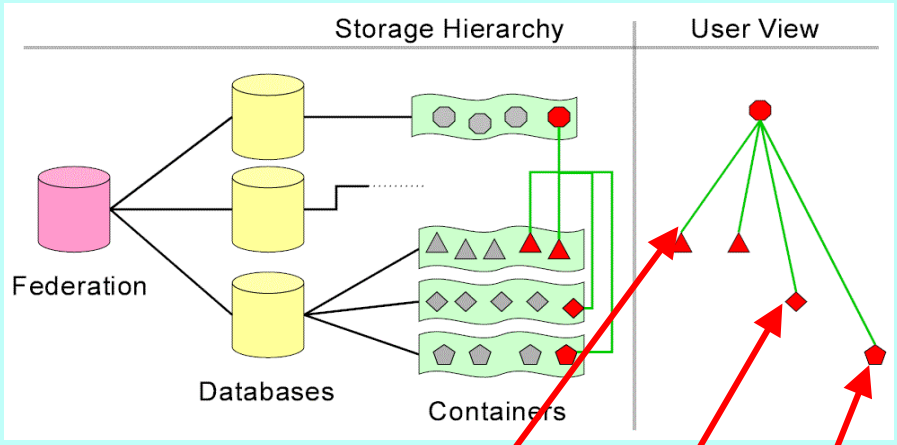
LHC Challenges: Scale

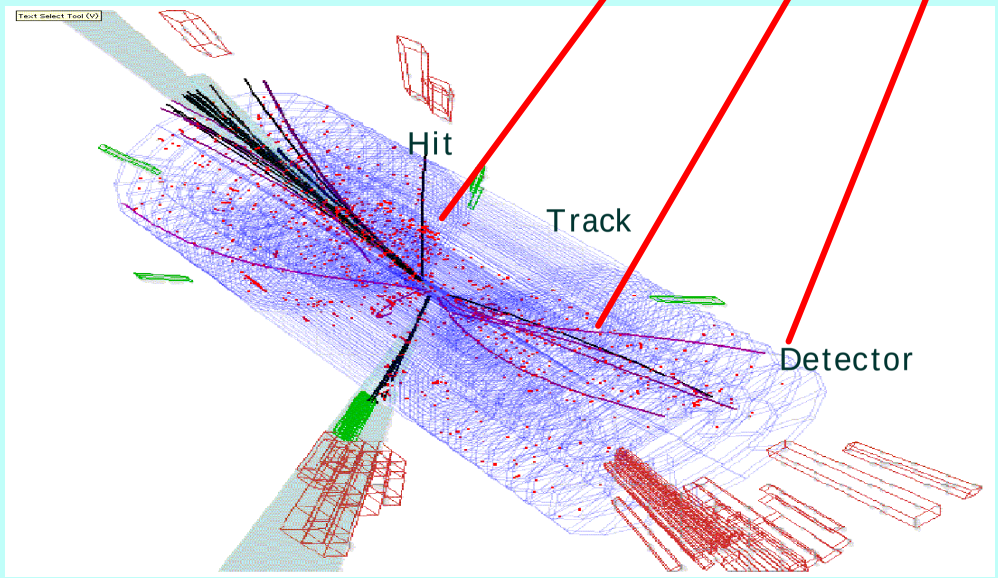
- Data written to tape — ~5 Petabytes/Year and UP (1 PB = 10^9 MBytes)
- Processing capacity — 1000 - TIPS and UP (1 TIPS = 10^6 MIPS)
- Typical networks — 0.5 - Few Gbps Per Link
- Lifetime of experiment — 2-3 Decades
- Users — ~ 5000 physicists
- Software developers — ~ 300 (Four Experiments)
- Events collected — 1 Billion/year/Experiment
- Event mean dimension — 1 Mbyte

- 0.1 to 1 Exabyte (10^{18} Bytes)

(~2010) (~2020 ?) Total for the LHC

GIOD Project Summary





➡GIOD has

- Constructed a Terabyte-scale set of fully simulated CMS events and used these to create a large OO database
- Learned how to create large database federations
- Completed the “100” (to 170) Mbyte/sec CMS Milestone
- Developed prototype reconstruction and analysis codes, and Java 3D OO visualization demonstrators, that work **seamlessly** with persistent objects over networks
- Deployed facilities and database federations as useful testbeds for Computing Model studies

MONARC: Common LHC Project

➡ Models Of Networked Analysis At Regional Centers

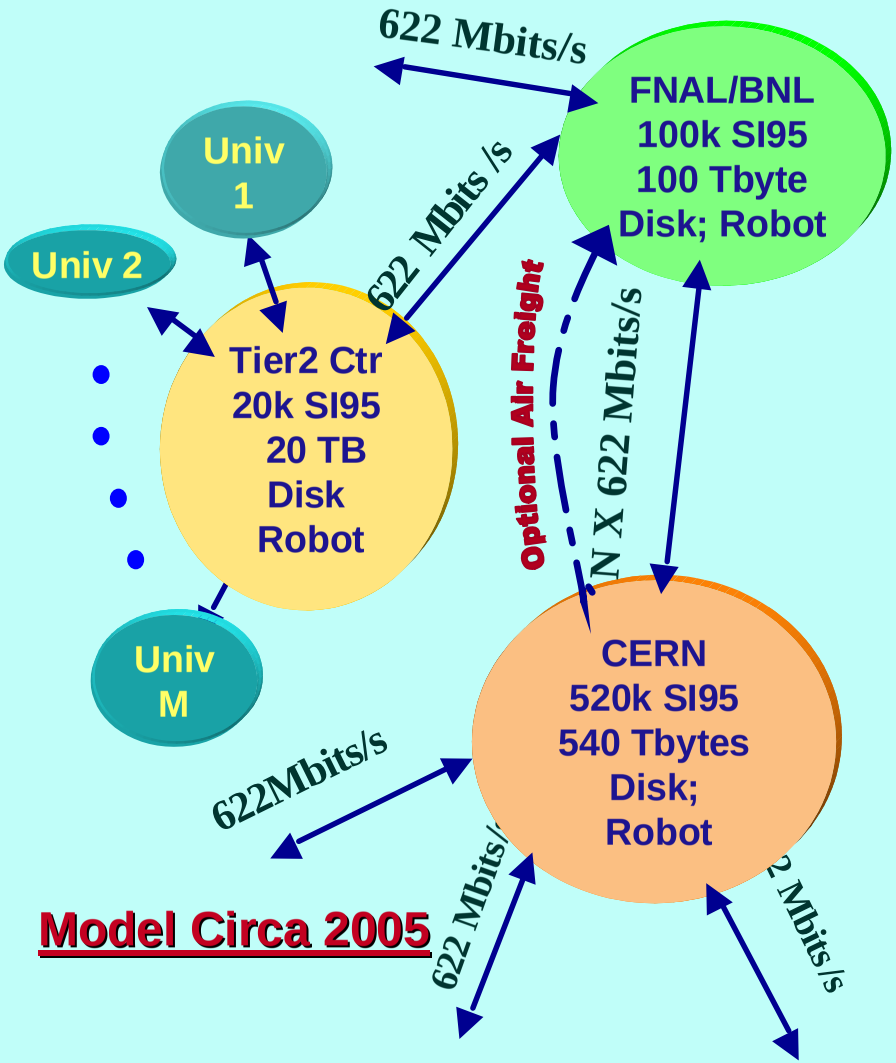
Caltech, CERN, Columbia, FNAL, Heidelberg, Helsinki, INFN, IN2P3, KEK, Marseilles, MPI Munich, Orsay, Oxford, Tufts

PROJECT GOALS

- Develop "Baseline Models"
- Specify the main parameters characterizing the Model's performance: throughputs, latencies
- Verify resource requirement baselines: (computing, data handling, networks)

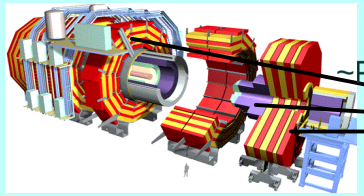
➡ TECHNICAL GOALS

- Define the Analysis Process
- Define RC Architectures and Services
- Provide Guidelines for the final Models
- Provide a Simulation Toolset for Further Model studies



Analysis Model Schema

Hierarchy of Resources (Tier0, Tier1, Tier2, ...)



~PBytes/sec

Online System

~100 MBytes/sec

1 TIPS = 25,000 SpecInt95

PC (1999) = ~15 SpecInt95

Offline Farm
~20 TIPS

~100 MBytes/sec

Bunch crossing per 25 nsecs.
100 triggers per second
Event is ~1 MByte in size

~622 Mbits/sec
or Air Freight

Tier 0

CERN Computer Center >20 TIPS

Tier 1

France Regional Center

Germany Regional Center

Italy Regional Center

Fermilab ~4 TIPS

~2.4 Gbits/sec

Tier 2

Tier2 Center ~1 TIPS

~622 Mbits/sec

Tier 3

Institute ~0.25TIPS

Physics data cache

100 - 1000 Mbits/sec

Tier 4

Workstations

Physicists work on analysis "channels".

Each institute has ~10 physicists working on one or more channels

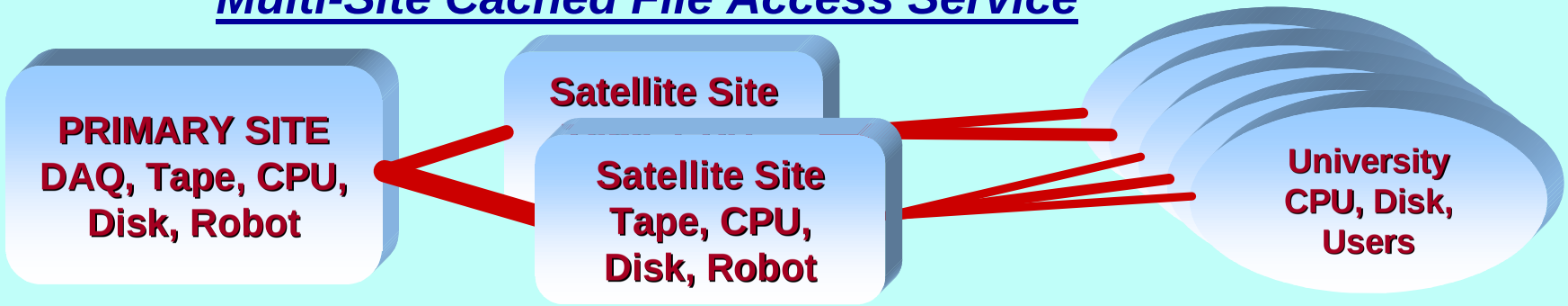
Data for these channels should be cached by the institute server

The Particle Physics Data Grid (PPDG)

DoE/NGI Next Generation Internet Project
ANL, BNL, Caltech, FNAL, JLAB, LBNL,
SDSC, SLAC, U.Wisc/CS

- First Year Goal: Optimized cached read access to 1-10 Gbytes, drawn from a total data set of 10-100+ Tbytes; ***SRMs***

Multi-Site Cached File Access Service



PPDG Report Soon; Prospect of Year 2 and 3 Funding

The Grid as advanced solution for WAN distributed computing

A. Ghiselli
INFN - CNAF

Computing clusters as simple composite elements(SCE) of grid

- SCE is a collection of basic elements (computer systems, networks, storage elements) aggregated with software and (dedicated) hardware. Building blocks for WAN grids or local grids.
- DATAGRID project: “capability to build and effectively manage large general-purpose data intensive computer clusters constructed from low-cost commodity components”.

from network to grid

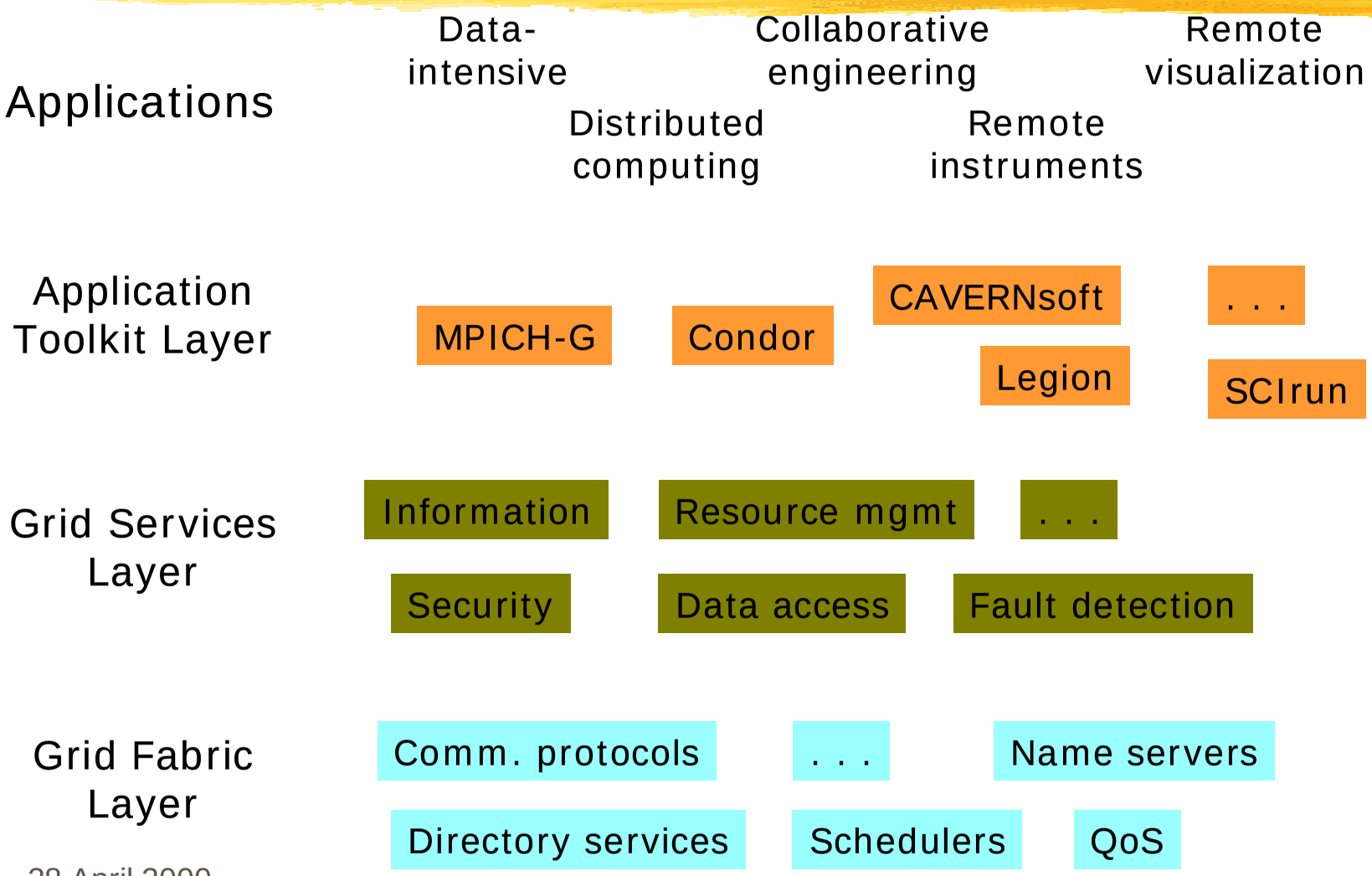
- Building blocks are there: grid is a network of computing systems and services
  - to increase delivered computation to the grid users:
    - Experimental scientists (radio astronomy, HEP...)
    - Distributed organizations (like INFN)
    - Tele-medicine
    - To applications having only episodic requirements for substantial computational resources (medical diagnosis....

Creating a Usable Grid :

Grid Services (“Middleware”)

- Standard grid services that
  - Provide “transparent” access to a wide range of resources (including networks)
  - Address interdomain issues of security, policy, etc.
  - Permit application-level management and monitoring of end-to-end performance
- Middleware-level and higher-level APIs and tools targeted at application programmers
  - Map between application and network views

Grid Services Architecture



Summary cont.

- “Grids” enable applications not supported by today’s commodity technologies
- Focus on high data rates, integrated computation, tight real-time coupling
- Demanding middleware requirements; Globus is one effort addressing these
- Large testbeds and projects are being created, deploying standard services/middleware
- Advanced application experiments underway, nationally and internationally

List of Participants

Roberto Alfieri	INFN-Parma
Giuseppe Aliberti	Department of Physics, University of L'Aquila
Roberto Aloisio	University of L'Aquila
Lidia Arcipiani	Enea
Enrico Becchetti	INFN-Perugia
Maurizio Benfatto	INFN-LNF
Christoph Best	John von Neumann Institute/DESY (Juelich)
Gianfranco Bilardi	Department of Computer Science, University of Padova
Vitaliano Chiarella	INFN-LNF
Giuseppe Ciaccio	CASPUR - Universiity of' Roma "La Sapienza" and DISI - University of Genova
Roberto Cucchi	INFN-CNAF
Barbara Curro' Dossi	ECT*, Trento
Roberto De Pietri	INFN-Parma
Paolo De Riso	INFN-RomaII
Giuseppe Di Carlo	INFN-LNF
Vincenzo Di Martino	Caspur, Roma
Francesco Di Renzo	INFN-Parma
Luca Di Sarra	University of L'Aquila
Andrea Donati	INFN-LNGS
Norbert Eicker	Department of Physics, University of Wuppertal
Stefano Fantoni	SISSA, Trieste
Angelo Galante	University of L'Aquila
Fabrizio Gentile	INFN-Perugia
Luigi Genoni	Secure NAT inc
Antonia Ghiselli	INFN-CNAF
Steven Gottlieb	Department of Physics, Indiana University Bloomington
Aurelio Federico Grillo	INFN-Lngs
Yutaka Ishikawa	Real World Computing Partnership, Tsukuba
Karl Jansen	CERN-TH Division
Sandor Katz	Eotvos University, Budapest
Sejong Kim	Department of Physics, Sejong University, R.of Korea
Edwin Laermann	Department of Physics, University of Bielefeld
Alessandro La Piana	ECT*, Trento
Thomas Lippert	Department of Physics, University of Wuppertal
Maria-Paola Lombardo	INFN-LNGS
Alessandro Lonardo	INFN -Roma I
Reinhard Lueling	Department of Computer Science,University of Paderborn
Agostino Mathis	HPCN-ENEA
Mirco Mazzucato	INFN-Padova
Andrea Michelotti	INFN -RomaI
Velia Minicozzi	Department of Physics, Uniiversity of RomaII Tor Vergata"
Burkhard Monien	Department of Computer Science, University of Paderborn
Silvia Morante	Department of Physics, University of RomaII "Tor Vergata"
Elisabetta Pace	INFN-LNF
Cristiano Palomba	Department of Physics, University of Roma I "La Sapienza"
Emanuele Panizzi	Department of Electric Engineering, University of L' Aquila and INFN-Roma I
Sandra Parlati	INFN-Lngs
Norbert Paschedag	DESY-Zeuthen
Francesco Pederiva	Department of Physics and INFN, University of Trento
Roberto Petronzio	Department of Physics and INFN, University of Roma "Tor Vergata"
Federico Rapuano	INFN-Roma1
Fulvio Ricci	INFN-Roma1

Vittorio Rosato	HPCN-ENEA
Davide Rossetti	INFN-Roma1
Giancarlo Rossi	Department of Physics and INFN, University of Roma "Tor Vergata"
Pietro Rossi	HPCN-ENEA
Giuseppe Sacco	INFN-Roma1
Alan Scheinine	CRS4
Klaus Schilling	Department of Physics, University of Wuppertal and NIC, DESY/Juelich
Wolfram Schroers	Department of Physics, University of Wuppertal
Roberto Scimia	University of Perugia
Hubert Simma	DESY-Zeuthen
Raffaele Tripiccione	INFN-Pisa
Peer Ueberholz	Department of Physics, University of Wuppertal
Giuseppe Ugolotti	Nice-Italy
Franco Valentinotti	Department of Material Science, University of Milano Bicocca
Marco Verdecchia	Department of Physics, University of L' Aquila and PSTd'A
Andrea Vicere'	INFN-Pisa
Guido Visconti	Department of Physics, University of L' Aquila and PSTd'A
Thomas Warschko	Department of Informatics, University of Karlsruhe