Abstract

The main thema of this talk will be an introduction to high-performance computing for the several aspects of scientific computing. The recent situation and developments both in hardware and software at the HPC will be introduced. Some general and conceptual ideas and informations about the methodologies, approaches, parallel programming tools and utilities/libraries will be presented and be exemplified wherever it is necessary. Introductory level programming examples for distibuted memory programming (Message Passing Interface, MPI) and shared memory programming (OpenMP) paradigms will be explained. It is also aimed to discuss parallel matrix diagonalization techniques as an application for scientific computing. A central problem in quantum chemistry is to describe the motion of N electrons in the field of M fixed nuclear point charges and brings the necessity of the diagonilization of a possibly large Hamiltonian matrix to obtain eigenvalues and eigenvectors. This results as $O(N^3)$ - $O(N^4)$ scaling behavior of run time and memory requirements with the studied system sizes N for for *ab initio* methods. Parallel eigensolvers will be introduced as the one of the possible solutions and the effects of the some computational parameters on the scaling behavior for a case study of tight-binding molecular dynamics simulation of carbon nanotube will be presented.

A brief survey of scientific/parallel computing/programming and parallel eigensolvers

April 9, 2008 Bilkent, Ankara

Dr. Cem Özdoğan

Department of Computer Engineering, Çankaya University,

06530 Ankara, Turkey E-mail:ozdogan@cankaya.edu.tr

About Myself

- Ph.D. from Physics, METU
- Currently Assistant Professor @ Cankaya University
- Responsibilities/Interests:
  - Lecturing: Numerical Compututations, Programming (Parallel, Systems, ..)
  - Construction & Administration of Cluster @ Cankaya,
  - Conduct research on computational chemistry
    - Tight-Binding Molecular Dynamics Simulations (conventional, linear scaling, parallel)
    - Ab-Initio Calculations (Gaussian, Vasp)
    - Specifically on carbon and boron nano structures
    - Generally systems in material science

Aims of This Presentation

- An introduction to high-performance computing.
- Computational Science Or Scientific Computing or just High Performance Computing.
- Available software packages and serial/parallel programming tools and utilities/libraries
  - Compilers & Debugger tools.
  - Utility Libraries.
  - Scientific Packages (Quantum Chemistry, Molecular Dynamics, Visualization).
- Parallel Computing.
- Parallel Eigensolvers.

Scientific Computing I

- Used to solve problems with
  - time complexity: Handle bigger problems
  - space complexity: The performance of a particular program may be improved by execution on a large machine

- for/to
  - Explanation; analyze the data obtained from experiments.
  - Estimation; computer experiments, predict results of experiments not performed yet,
  - Propose; model phenomena,
  - Validation; proving of previously proposed models,
  - Number Crunching!

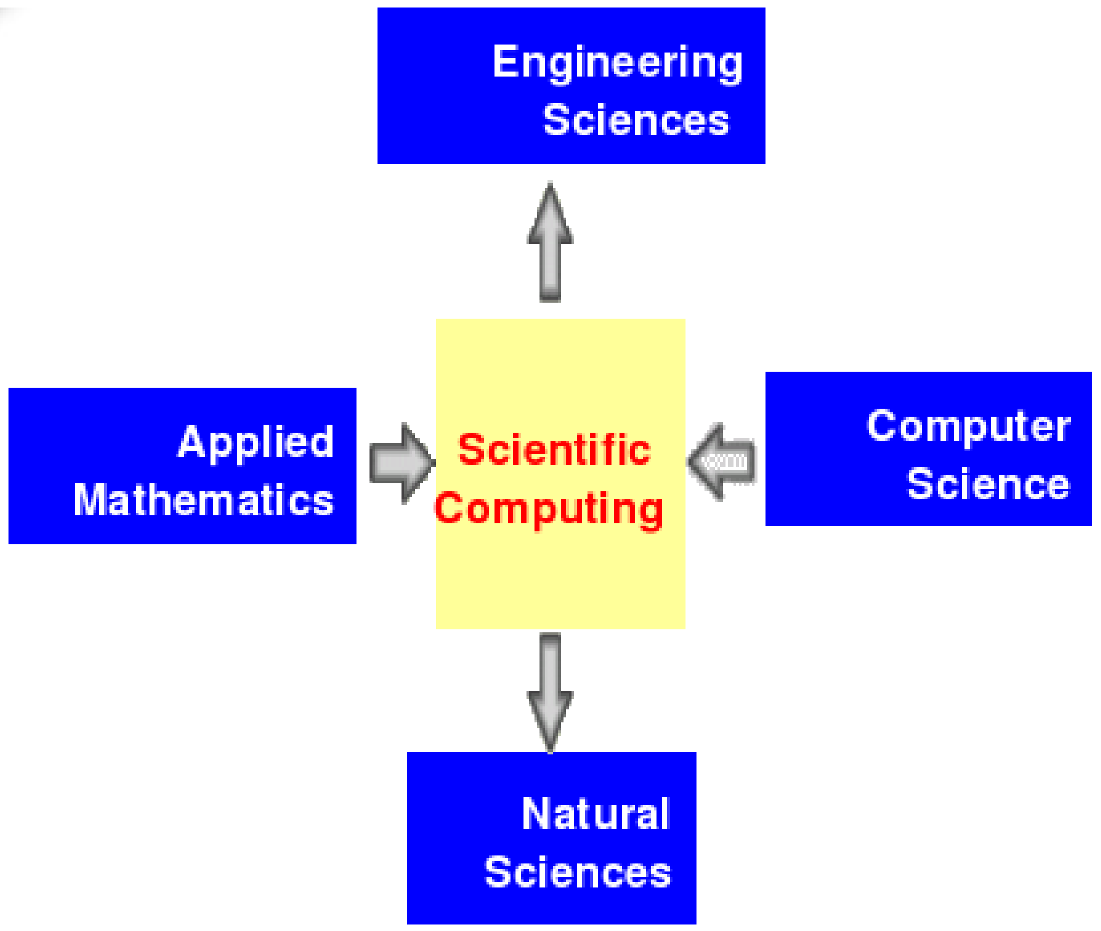
Scientific Computing II - wikipedia

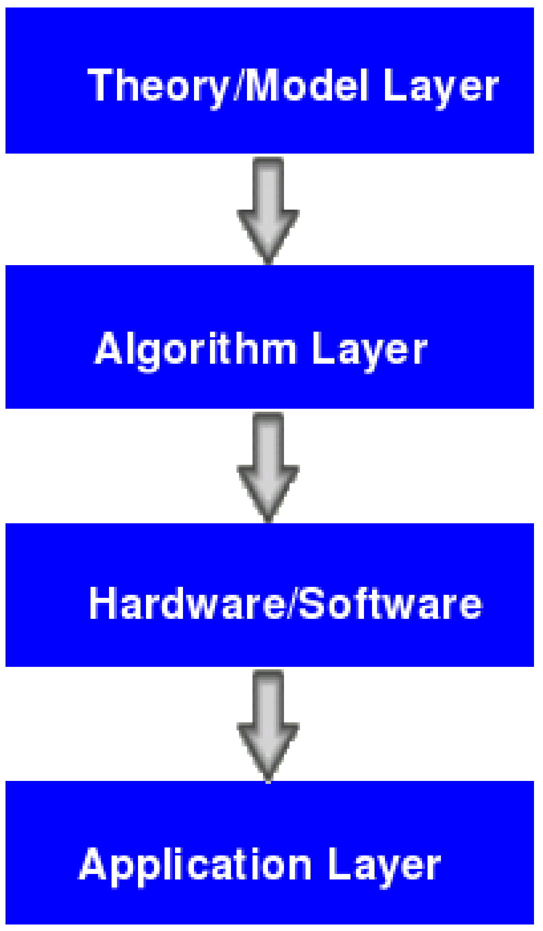
- Computational science (or scientific computing) is the field of study concerned with constructing mathematical models and numerical solution techniques and using computers to analyze and solve scientific, social scientific and engineering problems.
- In practical use, it is typically the application of computer simulation and other forms of computation to problems in various scientific disciplines.
- The field is distinct from computer science (the mathematical study of computation, computers and information processing).

Scientific Computing III - wikipedia

- Scientists and engineers develop computer programs, application software, that model systems being studied and run these programs with various sets of input parameters.
- Typically, these models require massive amounts of calculations (usually floating-point) and are often executed on supercomputers or distributed computing platforms.
- Numerical analysis is an important underpinning for techniques used in computational science.

Scientific Computing IV





Supercomputing I

- **Hardware:** Servers, storage, file systems, etc. The hardware structure or architecture determines to a large extent what the possibilities and impossibilities are in *speeding up* a computer system beyond the performance of a single CPU.
- **Compilers:** Another important factor that is considered in combination with the hardware is the capability of compilers to generate efficient code to be executed on the given hardware platform.
- Supercomputing is now High Performance Computing (HPC).
- Classifying computers by architecture; Number of instruction streams (“processors”) and data streams (“inputs”): Flynn’s taxonomy.

Supercomputing II

- HPC:
  - Large capability computers (fast CPUs). Performance is obtained by large numbers of these together working in parallel.
  - Massive memory. High I/O, enormous (fast & large) data storage.
  - High bandwith and low latency networks.
  - Specifically parallelized codes (MPI, OpenMP). MPI (Message Passing Interface) : the assembly language of parallel programming.
- The majority of systems look like minor variations on the same theme: clusters of RISC(EPIC)-based Symmetric Multi-Processing (SMP) nodes which in turn are connected by a fast network.

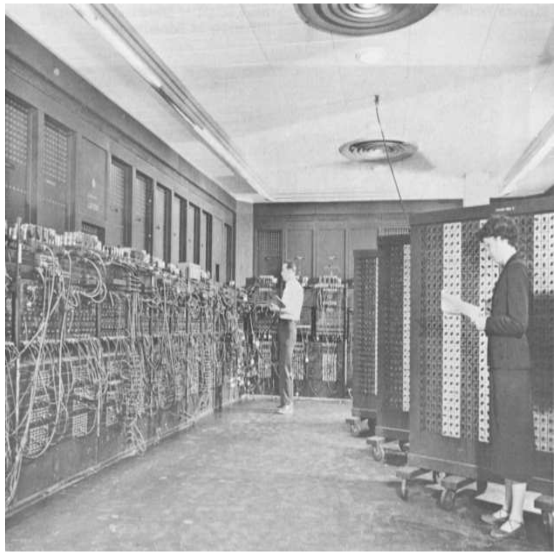
Supercomputing III

- Measure of performance:
  - MFLOPS (Million FLoating point Operations Per Second).
  - For example, an Intel Itanium 2 at 1.5 GHz can complete 4 floating point operations per cycle or a theoretical peak performance of 6 GFlop/s.
  - A clock cycle being defined as the basic internal unit of time for the system.
- The mismatch of communication vs. computation speed vs. memory speed
  - computational-bound & communication-bound.
  - memory-bound problems; can exhibit superlinear speedup when run on multiple processors compare to single processor.

Supercomputing IV

- Highest capacity communication networks (Myrinet, 10 GigE, InfiniBand, etc.)
  - 100 Mb/s Ethernet or Gigabit Ethernet, has the drawback of a high latency ($\cong 50 - 100\mu$ s)
  - Myri-10G, supports a 10 Gbit/s data rate, low latency
  - InfiniBand, the serial connection's signalling rate is 2.5 gigabit per second (Gbit/s) in each direction per connection.
    - Single, double, and quad data rates carry 2, 4, or 8 Gbit/s respectively.
    - InfiniBand QDR with 12X Quad (QDR) is expected as productions systems during 2008.

Some HPC Systems - ENIAC

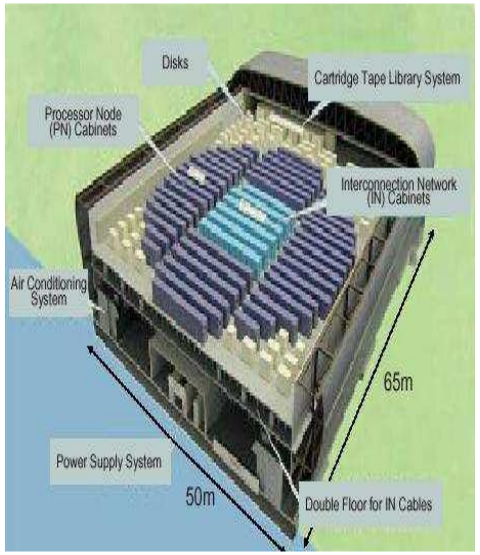


- ENIAC, short for Electronic Numerical Integrator And Computer.
- The first general purpose programmable electronic computer completed in 1945 at the University of Pennsylvania.
- Hydrogen bomb design.

Some HPC Systems - CRAY

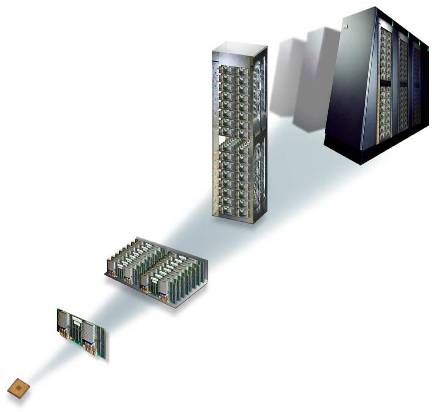
- The Cray 1 was installed in 1976 (Los Alamos). The first “supercomputer”.
- In 1993, Cray Research offered its first massively parallel processing (MPP) system, the Cray T3D supercomputer.
- In 2004, the Cray XD1 system designed around a direct-connect processor approach to massively parallel processing that directly links together processors, alleviating *memory contention* and *interconnect bottlenecks* found in cluster and SMP systems.

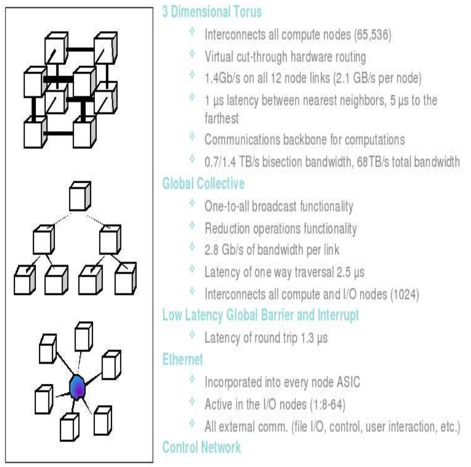
Some HPC Systems-EarthSimulator


Disks
Cartridge Tape Library System
Processor Node (PN) Cabinets
Interconnection Network (IN) Cabinets
Air Conditioning System
65m
Power Supply System
50m
Double Floor for IN Cables

- Japanese supercomputer simulates Earth for global climate models to evaluate the effects of global warming and problems in solid earth geophysics.. The fastest supercomputer in the world from 2002 to 2004.
- NEC SX architecture, 640 nodes, each node with 8 *vector* processors (8 Gflop/s peak per processor), 2 ns cycle time, 16GB shared memory. 10 TB memory. 700 TB disk space. 1.6 PB mass store
- Area of computer = 4 tennis courts, 3 floors.

Some HPC Systems - Blue Gene/L





3 Dimensional Torus

- Interconnects all compute nodes (65,536)
- Virtual cut-through hardware routing
- 1.4Gb/s on all 12 node links (2.1 GB/s per node)
- 1 µs latency between nearest neighbors, 5 µs to the farthest
- Communications backbone for computations
- 0.7/1.4 TB/s bisection bandwidth, 68TB/s total bandwidth

Global Collective

- One-to-all broadcast functionality
- Reduction operations functionality
- 2.8 Gb/s of bandwidth per link
- Latency of one way traversal 2.5 µs
- Interconnects all compute and I/O nodes (1024)

Low Latency Global Barrier and Interrupt

- Latency of round trip 1.3 µs

Ethernet

- Incorporated into every node ASIC
- Active in the I/O nodes (1:8-64)
- All external comm. (file I/O, control, user interaction, etc.)

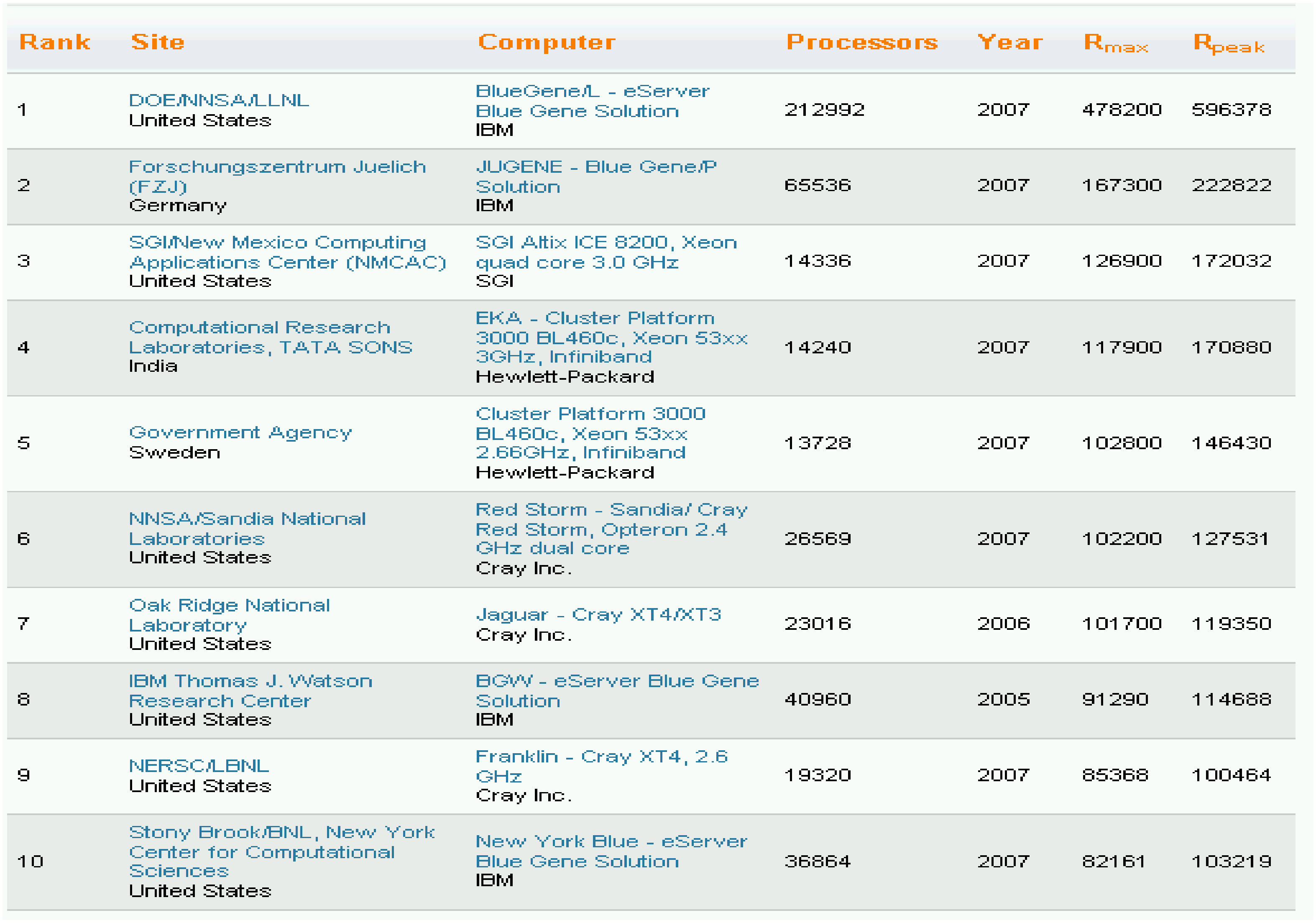
Control Network

- The Blue Gene/L machine was designed to reach operating speeds in the PFLOPS (petaFLOPS) range,
- Built in collaboration with the Department of Energy’s NNSA/Lawrence Livermore National Laboratory in California.
- LLNL system has a peak speed of 596 Teraflops. Blue Gene systems occupy the #1 and a total of 4 of the top 10 positions in the TOP500 supercomputer list (Nov 2007).
- Trading the speed of processors for lower power consumption.

top500.org

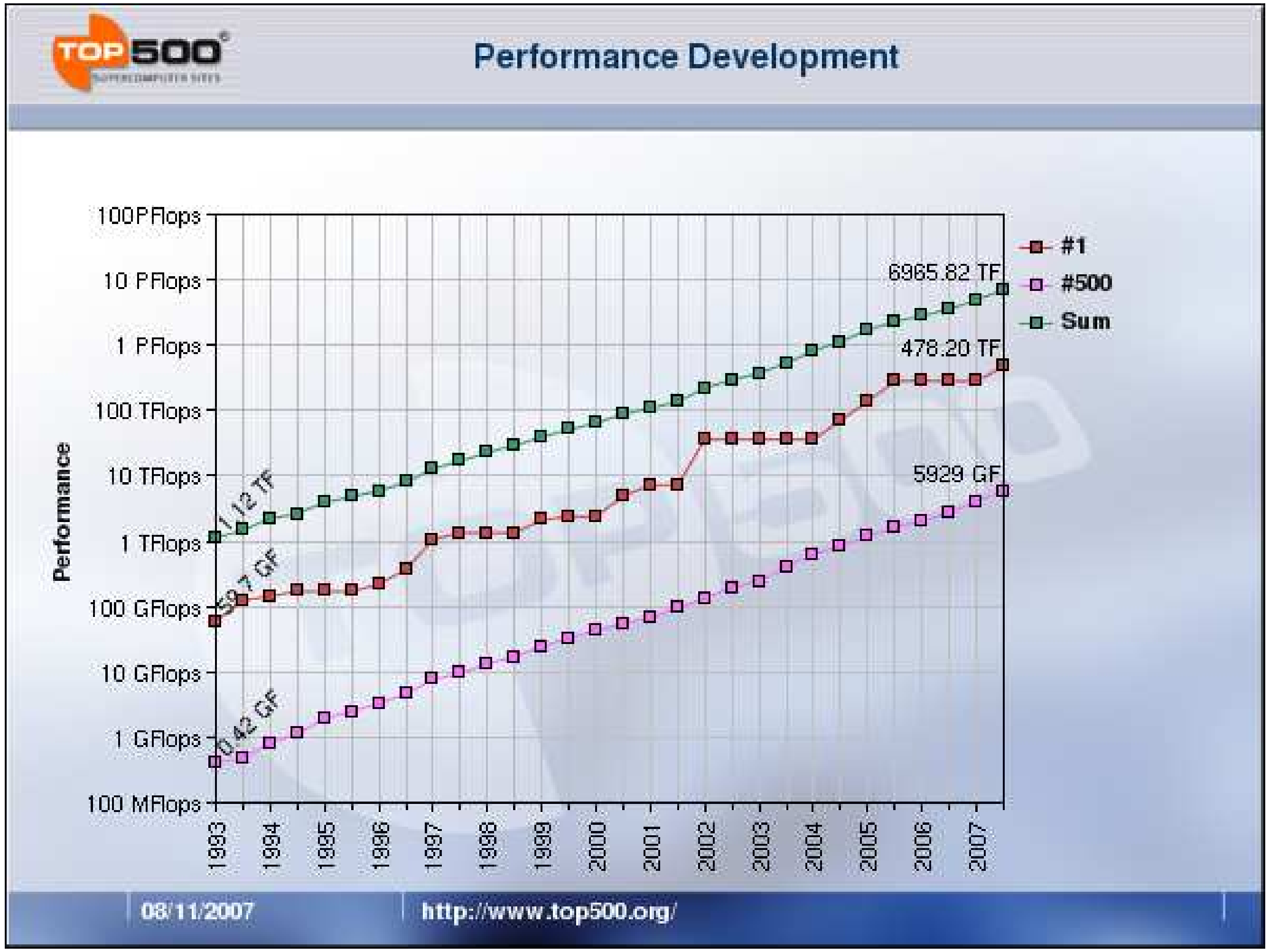
- A list of the 500 most powerful computer systems over the world.
- Established in June 1993, compiled twice a year (June & November).
- Using LINPACK Benchmark code (solving linear algebra equation aX=b).
- Organized by world-wide HPC experts, computational scientists, manufacturers, and the Internet community.

Top500 List - November 2007 (1-10)



Rank	Site	Computer	Processors	Year	R_{max}	R_{peak}
1	DOE/NNSA/LLNL United States	BlueGene/L - eServer Blue Gene Solution IBM	212992	2007	478200	596378
2	Forschungszentrum Juelich (FZJ) Germany	JUGENE - Blue Gene/P Solution IBM	65536	2007	167300	222822
3	SGI/New Mexico Computing Applications Center (NMCAC) United States	SGI Altix ICE 8200, Xeon quad core 3.0 GHz SGI	14336	2007	126900	172032
4	Computational Research Laboratories, TATA SONS India	EKA - Cluster Platform 3000 BL460c, Xeon 53xx 3GHz, Infiniband Hewlett-Packard	14240	2007	117900	170880
5	Government Agency Sweden	Cluster Platform 3000 BL460c, Xeon 53xx 2.66GHz, Infiniband Hewlett-Packard	13728	2007	102800	146430
6	NNSA/Sandia National Laboratories United States	Red Storm - Sandia/ Cray Red Storm, Opteron 2.4 GHz dual core Cray Inc.	26569	2007	102200	127531
7	Oak Ridge National Laboratory United States	Jaguar - Cray XT4/XT3 Cray Inc.	23016	2006	101700	119350
8	IBM Thomas J. Watson Research Center United States	BGW - eServer Blue Gene Solution IBM	40960	2005	91290	114688
9	NERSC/LBNL United States	Franklin - Cray XT4, 2.6 GHz Cray Inc.	19320	2007	85368	100464
10	Stony Brook/BNL, New York Center for Computational Sciences United States	New York Blue - eServer Blue Gene Solution IBM	36864	2007	82161	103219

Top500 - Performance Devel.



Top500 - Details

- Number of Processors
- Operating System
- Operating System Family
- Inter Connection
- Inter Connection Family
- Processor Generation
- Processor Family

Compilers

- FORTRAN 77/90/95
  - ifort/ifc - Intel Fortran Compilers
  - g77(f77)/gfortran -GNU project Fortran 77 compiler/Fortran 95 compiler
  - pgf77/pgf90 - Portland Group Fortran 77 compiler/Fortran 95 compiler
  - f77/f90/f95 - Absoft ProFortran Compilers
  - pghpf -Portland Group High Performance Fortran Compiler
- C/C++
  - icc - Intel C/C++ Compilers
  - gcc/g++(c++) - GNU project C compiler/C++ compiler
  - pgcc/pgCC - Portland Group C compiler/C++ compiler
- Parallel (mpich/Lam)
  - mpif77, mpif90, mpicc, mpiCC

Profilers & Debuggers

- Intel: idb, gprof
- PGI: pgdebug, pgprof, gprof SMP machines
- GNU: gdb, gprof
- Absoft: fx, xfx, gprof
- Debugging of parallel programs remain a challenging research area. Debugging is easier for MPI paradigm than shared memory paradigm (even if it is hard to believe)

Utilities & Libraries

- Mathematic Libraries
  - IMSL, NAG, etc.
- Scientific Computing
  - Linear Algebra, BLAS, ATLAS, EISPACK, LAPACK,SCALAPACK
  - Fast Fourier Transform, FFTW
  - The GNU Scientific Library, GSL
  - Utility Libraries, netCDF, PETSc, etc.
- Parallel Computing
  - OpenMP, PVM, MPI (MPICH, LAM/MPI, MPICH-GM)

Quantum Chemistry

- ABINIT yes(MPI) ADF PVM Cerius2 MPI GAMESS MPI Gaussian OpenMP MacroModel - MOLFDIR - Molpro MPI NWChem MPI MaterialStudio MPI CPMD MPI ACES2 - VASP ?

Molecular Dynamics

- Amber MPI) NAMD/VMD MPI Gromacs MPI InsightII - MacroModel - PMEMD MPI Quanta MPI Sybyl - CHARMM MPI TINKER - O -

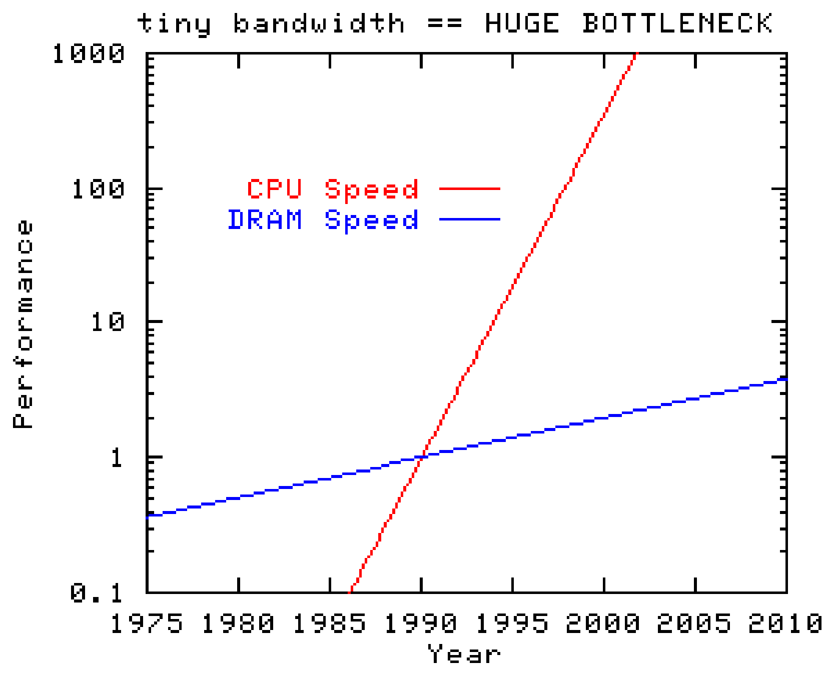
Molecular & Scientific Visualization

- AVS AVS- Express Cerius2 DINO ECCE GaussView GRASP InsightII - MOIL-VIEW MOLDEN MOLKEL MOLMOL MOLSCRIPT MOLSTAR MOVIEMOL NBOview QUANTA RASMOL RASTER3D SPARTAN SPACK SYBYL VMD Xtal/View XMGR GRACE

Parallel Computing

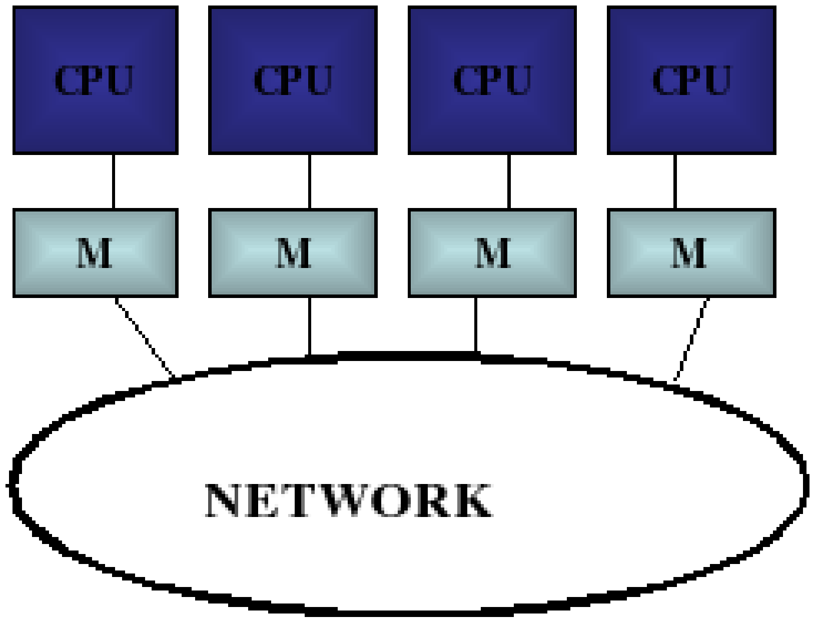
- Decomposition of a task into smaller tasks to be performed simultaneously, i.e. in parallel
- Parallelism is achieved by:
  - Break up tasks into smaller tasks
  - Assign smaller tasks to workers to work on simultaneously
  - Co-ordinate the workers
- Parallel Processing: Concurrent use of multiple processors to process data.
  - Running the same program on many processors.
  - Running different programs on the processor (not preferred).
  - Running many programs on each processor (not preferred).

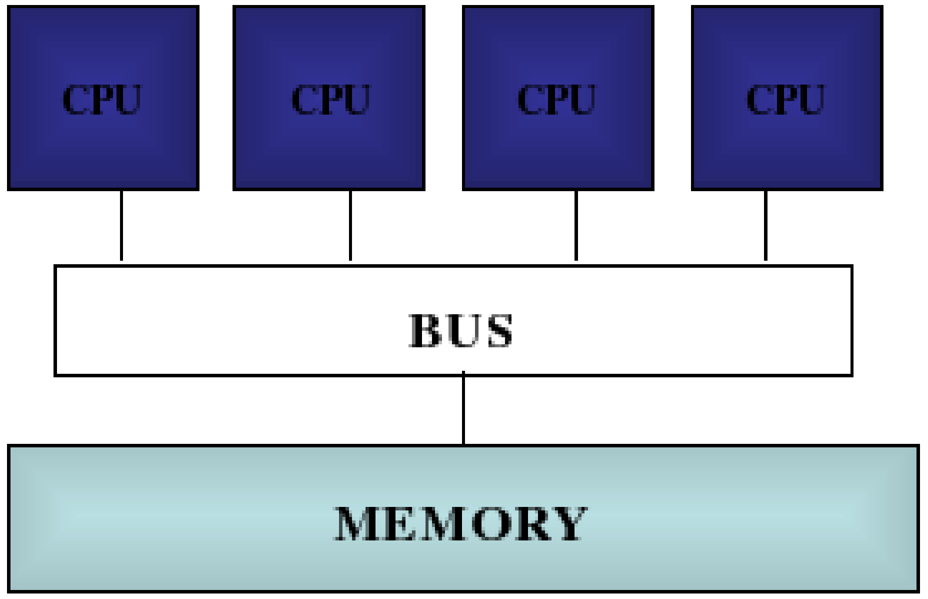
The Memory Problem



- Processor speed vs Memory speed
  - Hierarchal memory
  - Streaming data
  - Out of order execution
  - Compiler optimization

Memory Types I





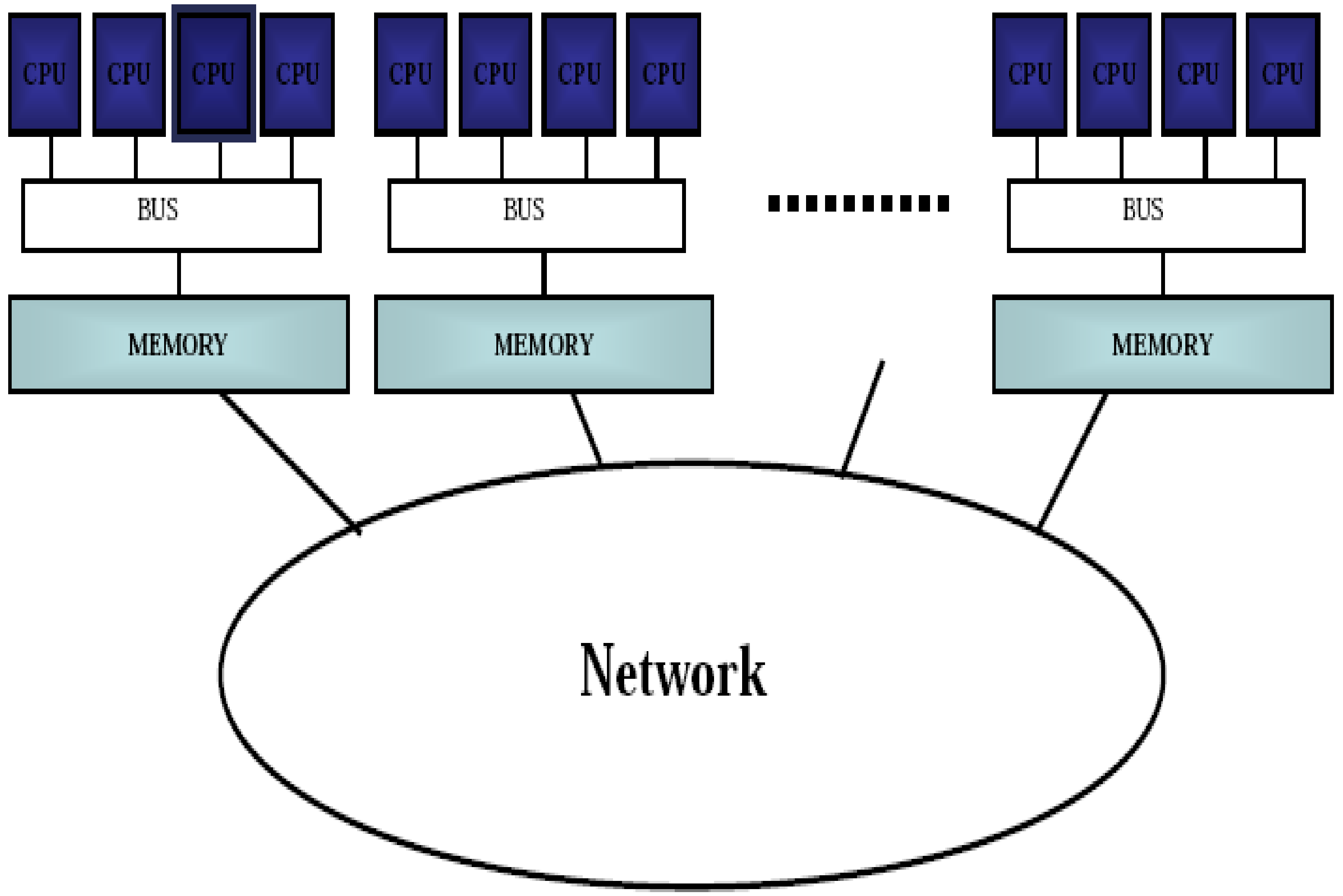
Memory Types II - Distibuted (DM)

- **Distributed**. Each CPU has its own associated memory.
  - The CPUs are connected by some network and may exchange data between their respective memories when required.
  - In contrast to shared memory machines *the user must be aware of the location of the data* in the local memories.
  - Must do message passing to exchange data between processors explicitly when needed..
  - The communication between processors is much slower than in SM-MIMD systems.
  - Moreover, the access to data that are not in the local memory belonging to a particular processor have to be obtained from non-local memory (or memories). This is again on most systems very slow as compared to local data access.

Memory Types III - Shared (SM)

- **Shared.** Shared memory systems have multiple CPUs all of which *share the same address space*.
  - This means that the knowledge of where data is stored is of no concern to the user as there is only one memory accessed by all CPUs on an equal basis.
  - Methods of memory access : Bus and Crossbar.
  - The bandwidth problem that obsseses shared-memory systems.
  - The speed of the memory is another critical issue.

Memory Types IV - Hybrid



Memory Types V - Programming

- Multiple instruction, multiple data (MIMD) model of parallel computers classified by their memory model. Shared memory. Distributed memory. Hybrid (Shared and Distributed memory)
- The "message passing" model has becomes so much accepted that PVM and MPI have been adopted by virtually all major vendors of DM-MIMD systems and even on SM-MIMD systems for compatibility reasons.
  - PVM - standing for Parallel Virtual Machine
  - MPI - standing for Message Passing Interface
- For SM-MIMD systems, OpenMP should be mentioned that can be used to parallelise Fortran and C(++) programs by inserting comment directives (Fortran 77/90/95) or pragmas (C/C++) into the code.
  - OpenMP has quickly been adopted by the major vendors and has become a well established standard for shared

OpenMP

- An Application Program Interface (API) ; explicitly direct multi-threaded, shared memory parallelism
  - Compiler Directives
  - Runtime Library Routines
  - Environment Variables
- Open specifications for Multi Processing via collaborative work between interested parties from the hardware and software industry, government and academia.
- Portable & Standardized
- The API is specified for C/C++ and Fortran
- Multiple platforms have been implemented including most Unix platforms and Windows NT
- Jointly defined and endorsed by a group of major computer hardware and software vendors

OpenMP - Example I

- In this simple example, the master thread forks a parallel region.
- All threads in the team obtain their unique thread number and print it.
- The master thread only prints the total number of threads. Two OpenMP library routines are used to obtain the number of threads and each thread's number.

```
export PGI=/usr/local/pgi
export PATH=$PGI/linux86/6.2/bin:$PATH
export MANPATH=$MANPATH:$PGI/linux86/6.2/man
export LD_LIBRARY_PATH=/usr/local/pgi/linux86/6.2/liblf:
/usr/local/pgi/linux86/6.2/lib:$LD_LIBRARY_PATH
export OMP_NUM_THREADS=4 (This line is optional.)
pgcc -mp -o omp_hello omp_hello.c
```

OpenMP - Example II

```
#include <omp.h>
#include <stdio.h>
#include <stdlib.h>
int main (int argc, char *argv[]) {
int nthreads, tid;
/* Fork a team of threads giving them their own copies of variables */
#pragma omp parallel private(nthreads, tid)
  {
  /* Obtain thread number */
  tid = omp_get_thread_num();
  printf("Hello World from thread = %d\n", tid);
  /* Only master thread does this */
  if (tid == 0)
    {
    nthreads = omp_get_num_threads();
    printf("Number of threads = %d\n", nthreads);
    }
  }  /* All threads join master thread and disband */
}
```

MPI I

- Parallelization scheme for distributed memory. Parallel programs consist of cooperating processes, each with its own memory. Standard (specification).
  - MPI 1: Traditional message-passing
  - MPI 2: Remote memory, parallel I/O, and dynamic processes
- Processes send data to one another as messages. Message can be passed around among compute processes
- Many implementations (almost each vendor has one)
  - MPICH and LAM/MPI from public domain most widely used
  - GLOBUS MPI for grid computing

MPI II

- Usually the same program is run on multiple processors
- The 6 basic calls in MPI are (in FORTRAN):
  - MPI_INIT(ierr)
  - MPI_COMM_RANK(MPI_COMM_WORLD, myid, ierr)
  - MPI_COMM_SIZE(MPI_COMM_WORLD, numprocs, ierr)
  - MPI_Send(buffer, count,MPI_INTEGER,destination, tag,MPI_COMM_WORLD, ierr)
  - MPI_Recv(buffer, count, MPI_INTEGER,source,tag, MPI_COMM_WORLD, status,ierr)
  - MPI_FINALIZE(ierr)

MPI - Example I

- Program takes data from process zero and sends it to all of the other processes by sending it in a ring.
- That is, process *i* should receive the data and send it to process *i+1*, until the last process is reached.
- Assume that the data consists of a single integer. Process zero reads the data from the user.

MPI - Example II

```
#include <stdio.h>
#include "mpi.h"
int main( argc, argv )
int argc;
char **argv;
{
    int rank, value, size;
    MPI_Status status;
    MPI_Init( &argc, &argv );
    MPI_Comm_rank( MPI_COMM_WORLD, &rank );
    MPI_Comm_size( MPI_COMM_WORLD, &size );
    do {
if (rank == 0) {
    scanf( "%d", &value );
    MPI_Send( &value, 1, MPI_INT, rank + 1, 0, MPI_COMM_WORLD );
}
else {
    MPI_Recv( &value, 1, MPI_INT, rank - 1, 0, MPI_COMM_WORLD,
      &status );
    if (rank < size - 1)
MPI_Send( &value, 1, MPI_INT, rank + 1, 0, MPI_COMM_WORLD );
}
printf( "Process %d got %d\n", rank, value );
    } while (value >= 0);
    MPI_Finalize( );
    return 0;
}
```

What is A Grid?

- Web: Uniform access to documents
- Grid: Flexible, high-performance access to resources and services for distributed communities

References I

- http://its.unc.edu/hpc/training/scientific/
- ENIAC
- CRAY
- The Earth Simulator Center
- Blue Gene /L
- http://www.top500.org
- Overview of Recent Supercomputers by Aad J. van der Steen and Jack J. Dongarra.
- http://www.beowulf.org
- http://www.linuxhpc.org
- http://www.supercluster.org/
- Task Force on Cluster Computing
- A. Geist, A. Beguelin, J. Dongarra, R. Manchek, W. Jaing,

References II

- W. Gropp, S. Huss-Ledermann, A. Lumsdaine, E. Lusk, B. Nitzberg, W. Saphir, M. Snir, MPI: The Complete Reference, Vol. 2, The MPI Extensions, MIT Press, Boston, 1998.
- OpenMP
- http://www-unix.mcs.anl.gov/mpi/index.htm
- http://www-unix.mcs.anl.gov/mpi/mpich/
- http://www.lam-mpi.org/
- http://www.mpi-forum.org
- http://www-unix.mcs.anl.gov/mpi/tutorial/learning.html

References III

- Intel Compilers
- Gnu Compilers
- Portland Compilers
- Absoft Compilers