Current Architectures for Parallel Processing

António Lira Fernandes
MICEI'0304
Universidade do Minho

Architectures for Parallel Processing

> "With the development of new kinds of equipment of greater capacity, and particularly of greater speed, it is almost certain that new methods will have to be developed in order to make the fullest use of this new equipment. It is necessary not only to design machines for the mathematics, but also to develop a new mathematics for the machines."
>
> Douglas Rayner Hartree, 1952

Outline

- Introduction
- Taxonomy
- Memory Models
- Bus/ Interconnected
- Programming Models
- Top500

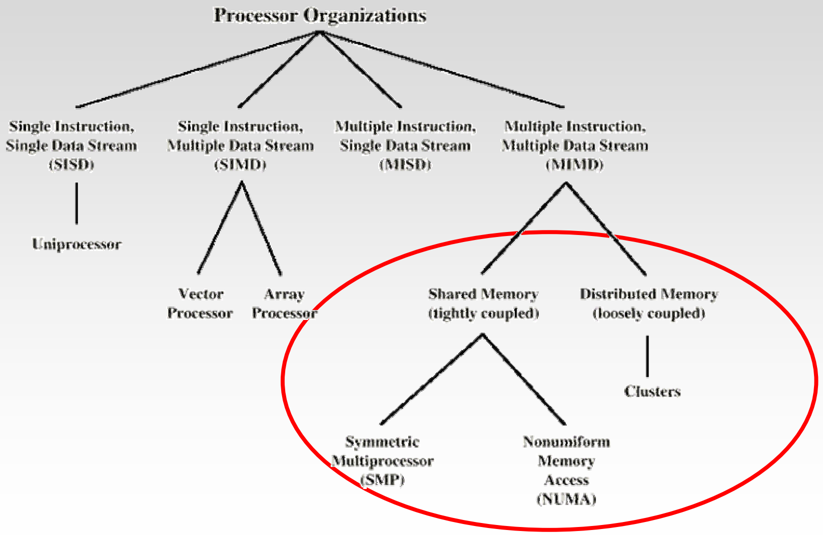
Parallel Computing - What is it?

- Parallel computing is when a program uses concurrency to either:
  - —decrease the runtime for the solution to a problem.
  - —Increase the size of the problem that can be solved.
- Parallel Computing gives you more performance to throw at your problems.

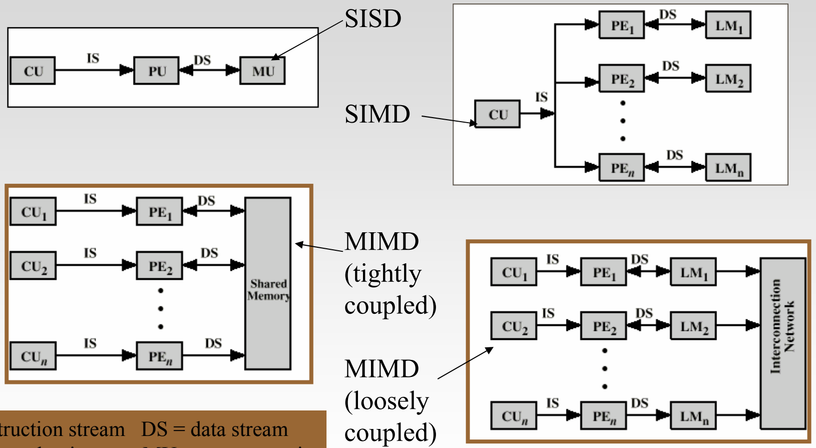
Current Parallel Approaches

- Single computers – multiple processing elements
  - tightly-coupled system (SMP & ccNUMA)
- Clusters
  - loosely-coupled system
- Fusion of the two
  - hybrid-coupled system (Super Clusters)

Taxonomy of Parallel Processor Architectures (Flynn)


Processor Organizations
Single Instruction, Single Data Stream (SISD)
Single Instruction, Multiple Data Stream (SIMD)
Multiple Instruction, Single Data Stream (MISD)
Multiple Instruction, Multiple Data Stream (MIMD)
Uniprocessor
Vector Processor
Array Processor
Shared Memory (tightly coupled)
Distributed Memory (loosely coupled)
Clusters
Symmetric Multiprocessor (SMP)
Nonuniform Memory Access (NUMA)

Four Architectures



SISD

SIMD

MIMD (tightly coupled)

MIMD (loosely coupled)

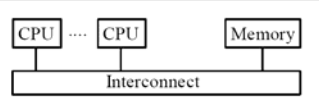
IS = instruction stream DS = data stream
CU = control unit MU = memory unit
PU = processing unit LM = local memory
PE = processor element

Outline

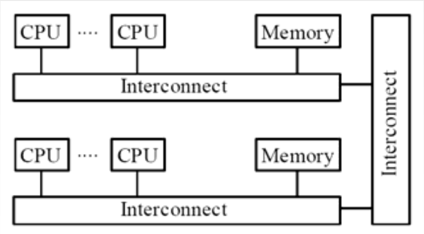
- Introduction
- Taxonomy
- Memory Models
  - Shared
  - Distributed
- Bus/ Interconnected
- Programming Models
- Top500

Memory Models

- Distributed memory
- Shared-memory
  - Uniform Memory Access (UMA)
  - Non-Uniform Memory Access (NUMA)
    - (distributed shared-memory)



UMA



NUMA

Memory Models

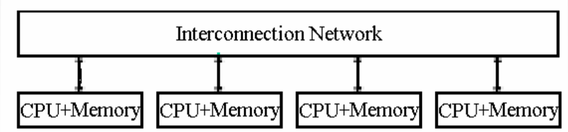
- Why NUMA architecture?
  - UMA system bus gets saturated (if too much traffic)
  - UMA crossbar gets too complex (too expensive)
  - UMA architecture does not scale beyond a certain level
- Typical NUMA problems
  - High synchronization costs (of subsystem interconnect)
  - High memory access latencies (some times not)
  - Might need memory sensitive strategies
    - loose shared-memory advantage

Outline

- Introduction
- Taxonomy
- Memory Models
  - Shared
  - Distributed
- Bus/ Interconnected
- Programming Models
- Top500

Memory Models

- Interconnected "von Neumann" computers by Ethernet, Myrinet, FDDI, ATM
- Distributed Memory, i.e. Summit Beowulf
- Heterogeneous mixture of processors
- Less Expensive
- LANs and WANs are also being used, but the communication costs are higher.



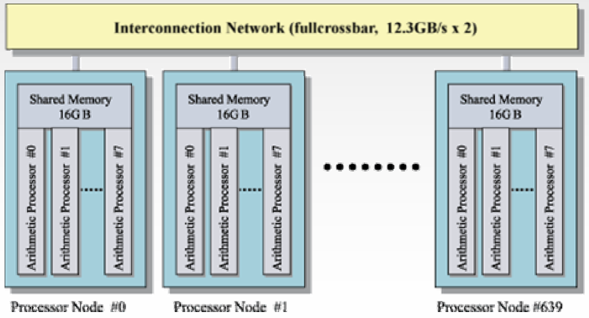
Cluster

Clusters Beowulf

- First cluster - Beowulf was developed in 1994 by Thomas Sterling and Don Becker, NASA researchers.
- Total performance: 60 Mflops.
- 16 nodos with the follow configuration:
    - —486DX4 100MHz (performance: 4,5 Mflops);
    - —256KB Cache;
    - —16MB RAM;
    - —HD 540MB;
    - —Ethernet network.

NUMA vs. cluster computing

- NUMA can be viewed as a very tightly coupled form of cluster computing.
- Using an cluster architecture a NUMA can be implemented entirely in software



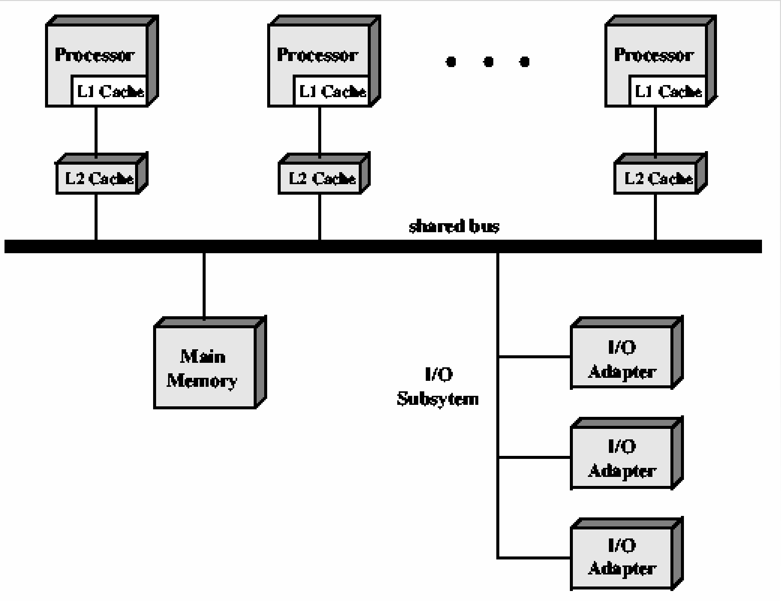
Outline

- Introduction
- Taxonomy
- Memory Models
- Bus/ Interconnected
    - —Bus
        - – Time shared or common bus
        - – Multiport memory
        - – Central control unit
    - —Interconnection
- Programming Models
- Top500

Time Shared Bus

- Simplest form
- Structure and interface similar to single processor system
- Following features provided
    - —Arbitration - any module can be temporary master
    - —Time sharing - if one module has the bus, others must wait and may have to suspend
- Now have multiple processors as well as multiple I/O modules

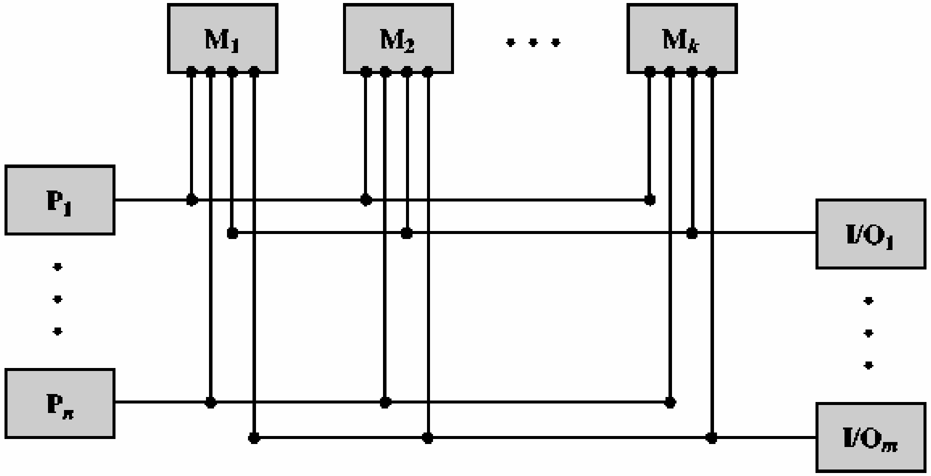
Shared Bus



Multiport Memory

- Direct independent access of memory modules by each processor
- Logic required to resolve conflicts
- Little or no modification to processors or modules required
- Advantages and Disadvantages...

Multiport Memory Diagram

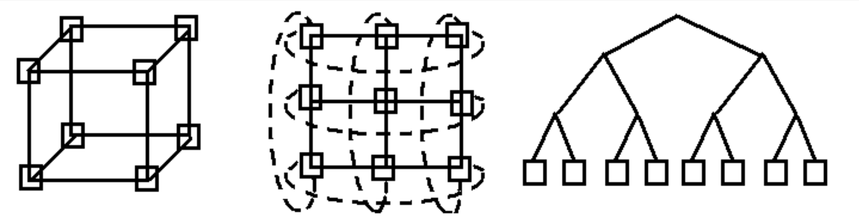


Outline

- Introduction
- Taxonomy
- Memory Models
- Bus/ Interconnected
  - —Bus
  - —Interconnection
    - – Static
    - – Dynamic
- Programming Models
- Top500

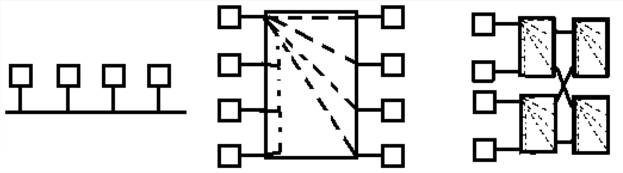
Static Interconnected

- Cube
- Mesh, Intel Paragon
- Tree, Thinking Machine CM-5

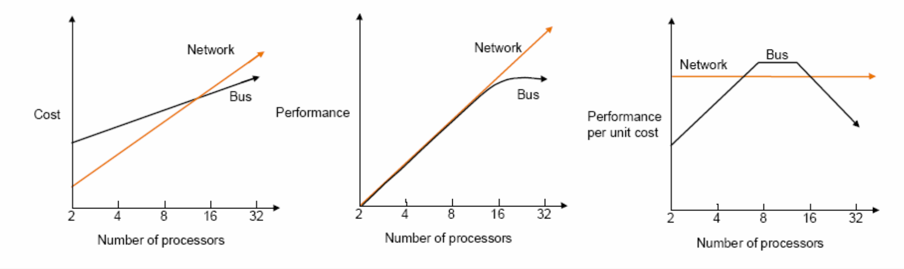


Dynamic Interconnected

- Paths are established as needed
  - —Bus based, SGI Power Challenge
  - —Crossbar
  - —Multistage Networks

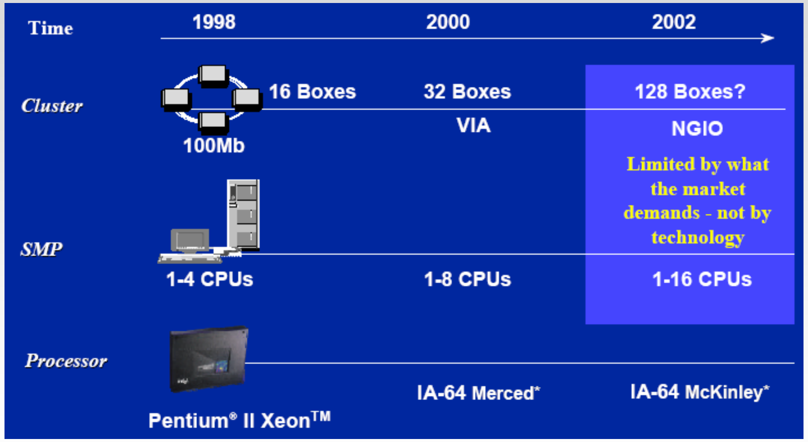


Bus vs Network



João Luís Sobral 2002

The Hardware is in great shape



Time	1998	2000	2002
Cluster	16 Boxes 100Mb	32 Boxes VIA	128 Boxes? NGIO Limited by what the market demands - not by technology
SMP	1-4 CPUs	1-8 CPUs	1-16 CPUs
Processor	Pentium® II Xeon™	IA-64 Merced*	IA-64 McKinley*

Tim Mattson

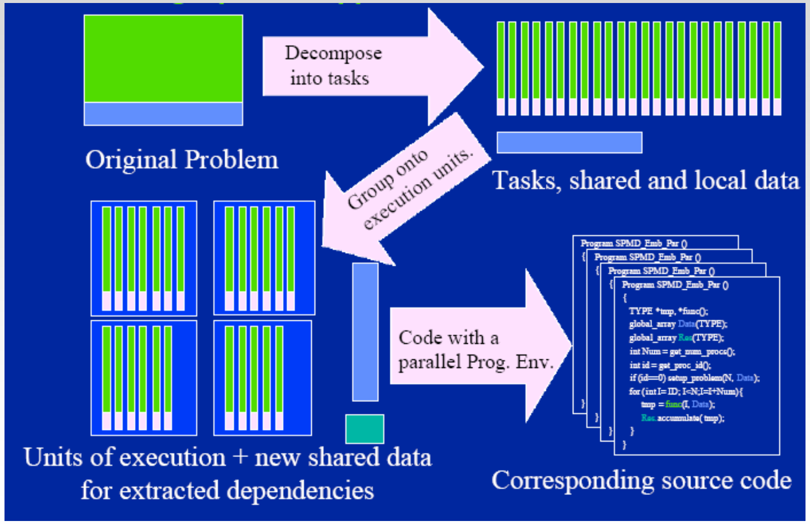
Outline

- Introduction
- Taxonomy
- Memory Models
- Bus/ Interconnected
- Programming Models
  - Message-passing (PVM, MPI)
  - Threading (OpenMP/threads)
- Top500

Programming Models

- Message-passing (PVM, MPI)
  - Individual processes exchange messages
  - Works on clusters and on parallel computers (topology transparent to user)
  - Manual – transform to parallel
- Threading (OpenMP/threads)
  - Efficient only on shared memory systems
  - One process (environment), multiple threads
  - Cheap, implicit communication
  - Different scheduling approaches
  - Limited (semi-) automatic – transform to parallel

Writing a parallel application


Decompose into tasks
Original Problem
Group onto execution units.
Tasks, shared and local data
Code with a parallel Prog. Env.
Units of execution + new shared data for extracted dependencies
Corresponding source code

```
Program SPMD_Emb_Par ()
{
  TYPE *tmp, *func();
  global_array Data(TYPE);
  global_array Res(TYPE);
  int Num = get_num_procs();
  int id = get_proc_id();
  if (id==0) setup_problem(N, Data);
  for (int I= ID; I<N;I=I+Num){
      tmp = func(I, Data);
      Res.accumulate( tmp);
  }
}
```

Outline

- Introduction
- Taxonomy
- Memory Models
- Bus/ Interconnected
- Programming Models
  - Message-passing (PVM, MPI)
  - Threading (OpenMP/threads)
- Top500

MPI

- MPI 1 (1994) and later MPI 2 (1997) is designed as a communication API for multi-processor computers.
- Passing messages between processes
- Implemented using a communication library of the vendor of the machine.
- Adds an abstraction level between the user and this vendor library, to guarantee the portability of the program code.
- Work on heterogeneous workstation clusters
- High-performance communication on large multi-processors
- Rich variety of communication mechanisms.

MPI

- Pros:
  - Very portable
  - Requires no special compiler
  - Requires no special hardware but can make use of high performance hardware
  - Very flexible - can handle just about any model of parallelism
  - No shared data! (You don’t have to worry about processes "treading on each other's data" by mistake.)
  - Can download free libraries (Linux PC)
  - Forces you to decomposing your problem.
- Cons:
  - All-or-nothing parallelism (difficult to incrementally transform to parallel the existing serial codes)
  - No shared data - Requires distributed data structures
  - Could be thought of assembler for parallel computing - you generally have to write more code
  - Partitioning operations on distributed arrays can be messy.

Outline

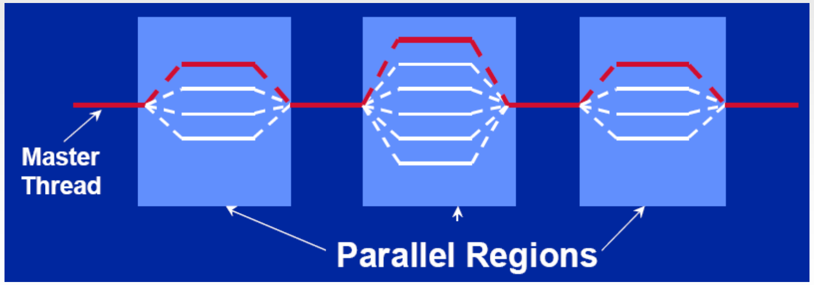
- Introduction
- Taxonomy
- Memory Models
- Bus/ Interconnected
- Programming Models
  - Message-passing (PVM, MPI)
  - Threading (OpenMP/threads)
- Top500

OpenMP

- Is an API for multithreaded applications.
  - A set of compiler directives, library routines and environment variables.
- Initiated specification (basic loop-based parallelism) in
  - Fortran (77 and up), C, and C++.
- Is fork-join model of parallel execution.
- Usually used to parallelize loops. (consuming loops)
- Threads communicate by sharing variables
- To control race conditions we use synchronization to protect data conflicts. (Synchronization is expensive so - change how data)
- Is available for a variety of platforms.

Fork-Join Parallelism:

- Master thread spawns a team of threads as needed.
- Parallelism is added incrementally: i.e. the sequential program evolves into a parallel program.



OpenMP

- Pros:
  - Incremental parallelism - can transform to parallel existing serial codes one bit at a time
  - Quite simple set of directives
  - Shared data
  - Partitioning operations on arrays is very simple.
- Cons:
  - Requires proprietary compilers
  - Requires shared memory multiprocessors
  - Shared data
  - Having to think about what data is shared and what data is private
  - Generally not as scalable (more synchronization points)
  - Not well-suited for non-trivial data structures like linked lists, trees etc

MPI vs OpenMP

- Pure MPI
  - Pro:
    - Portable to distributed and shared memory machines.
    - Scales beyond one node
    - No data placement problem
  - Con:
    - Difficult to develop and debug
    - High latency, low bandwidth
    - Explicit communication
    - Large granularity
    - Difficult load balancing
- Pure OpenMP
  - Pro:
    - Easy to implement parallelism
    - Low latency, high bandwidth
    - Implicit Communication
    - Coarse and fine granularity
    - Dynamic load balancing
  - Con:
    - Only on shared memory machines
    - Scale within one node
    - Possible data placement problem
    - No specific thread order

Why Hybrid

- Hybrid MPI/OpenMP paradigm is the software trend for clusters of SMP architectures.
- Elegant in concept and architecture:
  - using MPI across nodes
  - and OpenMP within nodes.
  - Good usage of shared memory system resource (memory, latency, and bandwidth).
- Avoids the extra communication overhead with MPI within node.
- OpenMP adds fine granularity (larger message sizes) and allows increased and/or dynamic load balancing.
- Some problems have two-level parallelism naturally.
- Some problems could only use restricted number of MPI tasks.
- Could have better scalability than both pure MPI and pure OpenMP.

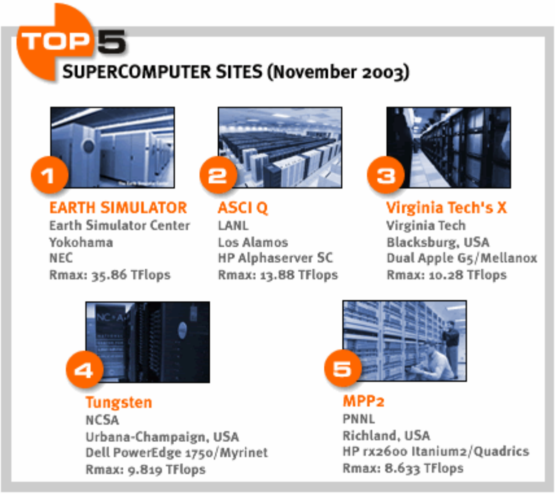
Outline

- Introduction
- Taxonomy
- Memory Models
- Bus/ Interconnected
- Programming Models
- Top500

Top 500

Rank	Site Country/Year	Computer / Processors Manufacturer	Computer Family Model	Inst. type Inst. Area	R_{max} R_{peak}	N_{max} n_{half}
1	Earth Simulator Center Japan/2002	**Earth-Simulator** / 5120 NEC	**NEC Vector** SX6	**Research**	35860 40960	1.0752e+06 266240
2	Los Alamos National Laboratory United States/2002	**ASCI Q - AlphaServer SC45, 1.25 GHz** / 8192 HP	**HP AlphaServer** Alpha-Server-Cluster	**Research**	13880 20480	633000 225000
3	Virginia Tech United States/2003	**1100 Dual 2.0 GHz Apple G5/Mellanox Infiniband 4X/Cisco GigE** / 2200 Self-made	**NOW - PowerPC** G5 Cluster	**Academic**	10280 17600	520000 152000
4	NCSA United States/2003	*Tungsten* **PowerEdge 1750, P4 Xeon 3.06 GHz, Myrinet** / 2500 Dell	**Dell Cluster** PowerEdge 1750, Myrinet	**Academic**	9819 15300	630000
5	Pacific Northwest National Laboratory United States/2003	*Mpp2* **Integrity rx2600 Itanium2 1.5 GHz, Quadrics** / 1936 HP	**HP Cluster** Integrity rx2600 Itanium2 Cluster	**Research**	8633 11616	835000 140000
6	Los Alamos National Laboratory United States/2003	*Lightning* **Opteron 2 GHz, Myrinet** / 2816 Linux Networx	**NOW - AMD** NOW Cluster - AMD - Myrinet	**Research**	8051 11264	761160 109208
7	Lawrence Livermore National Laboratory United States/2002	**MCR Linux Cluster Xeon 2.4 GHz - Quadrics** / 2304 Linux Networx/Quadrics	**NOW - Intel Pentium** NOW Cluster - Intel	**Research**	7634 11060	350000 75000
8	Lawrence Livermore National Laboratory United States/2000	**ASCI White, SP Power3 375 MHz** / 8192 IBM	**IBM SP** SP Power3 375 MHz high node	**Research**	7304 12288	640000
9	NERSC/LBNL United States/2002	*Seaborg* **SP Power3 375 MHz 16 way** / 6656 IBM	**IBM SP** SP Power3 375 MHz high node	**Research**	7304 9984	640000
10	Lawrence Livermore National Laboratory United States/2003	**xSeries Cluster Xeon 2.4 GHz - Quadrics** / 1920 IBM/Quadrics	**IBM Cluster** xSeries Cluster Xeon - Quadrics	**Research**	6586 9216	425000 90000
14	**Chinese Academy of Science** China/2003	**DeepComp 6800, Itanium2 1.3 GHz, QsNet** / 1024 Legend	**Legend** DeepComp 6800	**Academic**	4183 5324.8	491488
15	Commissariat a l'Energie Atomique (CEA) France/2001	**AlphaServer SC45, 1 GHz** / 2560 HP	**HP AlphaServer** Alpha-Server-Cluster	**Research**	3980 5120	360000 85000
16	HPCx United Kingdom/2002	**pSeries 690 Turbo 1.3GHz** / 1280 IBM	**IBM SP** SP Power4, Colony	**Academic**	3406 6656	317000

Top 500



TOP 5 SUPERCOMPUTER SITES (November 2003)

1 EARTH SIMULATOR
Earth Simulator Center
Yokohama
NEC
Rmax: 35.86 TFlops

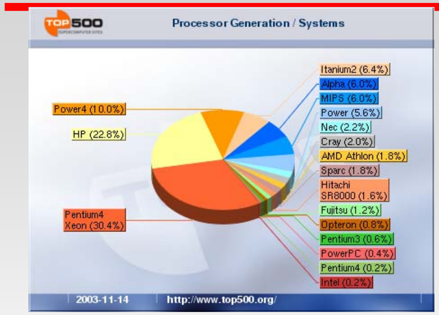
2 ASCI Q
LANL
Los Alamos
HP Alphaserver SC
Rmax: 13.88 TFlops

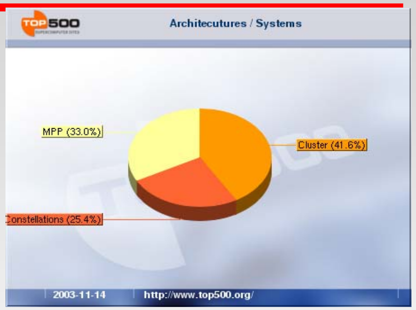
3 Virginia Tech's X
Virginia Tech
Blacksburg, USA
Dual Apple G5/Mellanox
Rmax: 10.28 TFlops

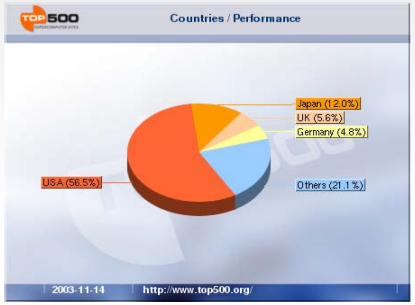
4 Tungsten
NCSA
Urbana-Champaign, USA
Dell PowerEdge 1750/Myrinet
Rmax: 9.819 TFlops

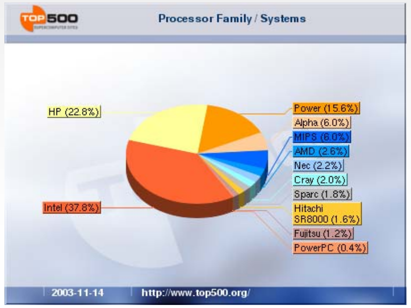
5 MPP2
PNNL
Richland, USA
HP rx2600 Itanium2/Quadrics
Rmax: 8.633 TFlops

Top 500



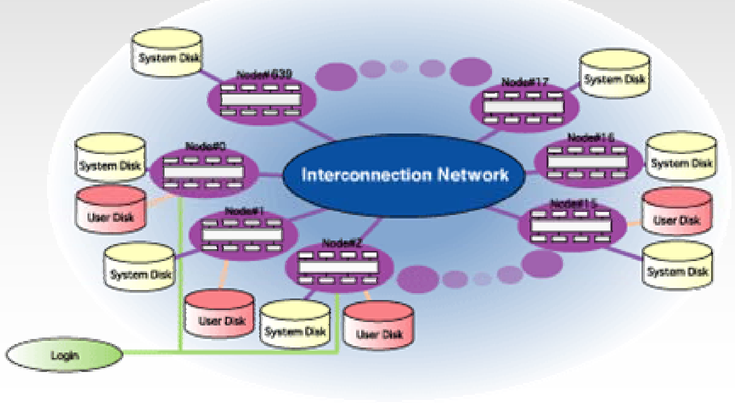






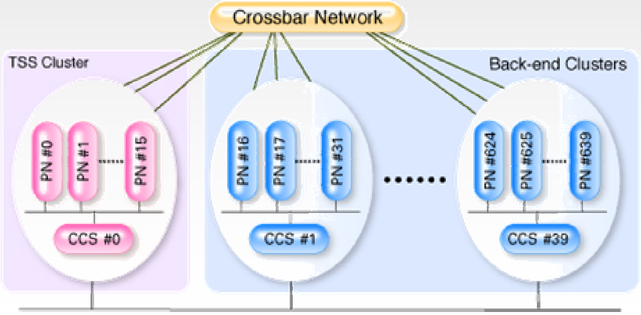
Earth Simulator

- Is a highly parallel vector supercomputer system
- Use distributed-memory
- 640 processor nodes (PNs)
- Connected by 640x640 single-stage crossbar switches



Earth Simulator

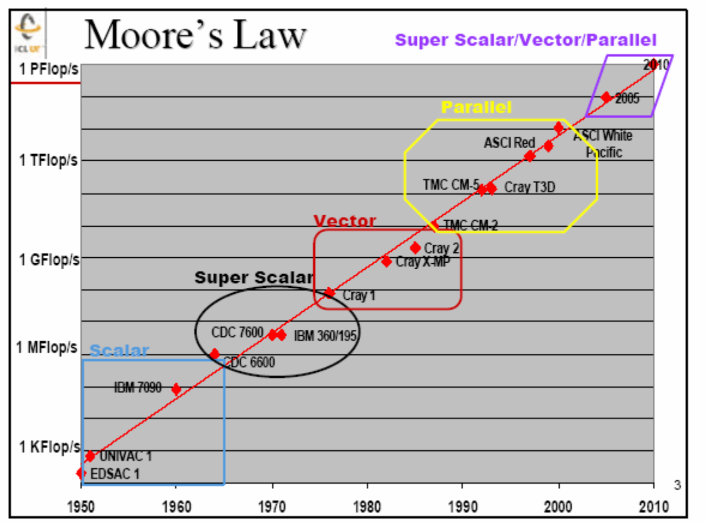
- Each PN is a system with a shared memory
  - 8 vector-type arithmetic processors (APs)
  - 16 GB main memory system (MS)
  - remote access control unit (RCU)
  - one I/O processor
  - peak performance of each AP: 8GFlops
- 5120 APs with 10 TB of main memory
- Theoretical performance: 40TFlops



Earth Simulator

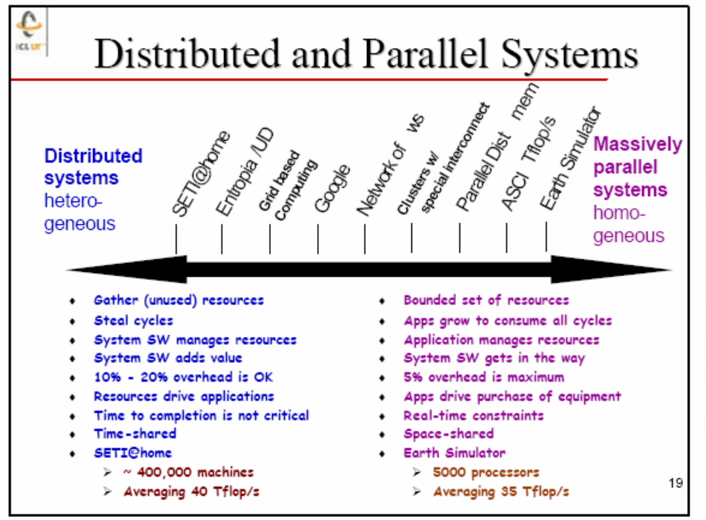
- MPI/ES is a message passing library based on the MPI-1 and MPI-2 standards
- Provides high-speed communication capability that fully exploits the features of Interconnection Network and shared memory.
- Can be used for both intra- and inter-node parallelization.
- An MPI process is assigned to an AP in the flat parallelization, or to a PN that contains microtasks or OpenMP threads in the hybrid parallelization.
- MPI/ES libraries are designed and optimized carefully to achieve highest performance of communication on the ES architecture in both of the parallelization manner.

Evolution



Jack Dongarra

Solutions



Distributed and Parallel Systems

Distributed systems hetero-geneous

SETI@home

Entropia /UD

Grid based Computing

Google

Network of ws

Clusters w/ special interconnect

Parallel Dist mem

ASCI Tflop/s

Earth Simulator

Massively parallel systems homo-geneous

- Gather (unused) resources
- Steal cycles
- System SW manages resources
- System SW adds value
- 10% - 20% overhead is OK
- Resources drive applications
- Time to completion is not critical
- Time-shared
- SETI@home
  - ~ 400,000 machines
  - Averaging 40 Tflop/s

- Bounded set of resources
- Apps grow to consume all cycles
- Application manages resources
- System SW gets in the way
- 5% overhead is maximum
- Apps drive purchase of equipment
- Real-time constraints
- Space-shared
- Earth Simulator
  - 5000 processors
  - Averaging 35 Tflop/s

Jack Dongarra

Final

“In respect of military method, we have, firstly, Measurement; secondly, Estimation of quantity; thirdly, Calculation; fourthly, Balancing of chances; fifthly, Victory.”

“Fighting with a large army under your command is nowise different from fighting with a small one: it is merely a question of instituting signs and signals. “

SUN TZU ON THE ART OF WAR