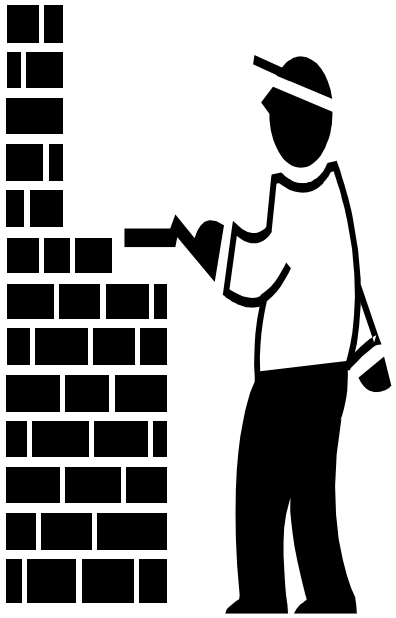
Understanding, Provisioning and Enabling High Performance Computing Machines

Brajesh Pande

Computer Centre IIT Kanpur

High Performance Computing An Analogy

If one person can build a wall in 500000!!! days. How can we build the wall faster?



High Performance Computing An Analogy

Make person work faster – better resources

Provide tools for using resources

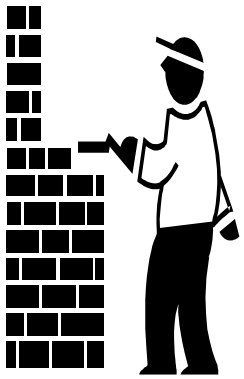
Organize work

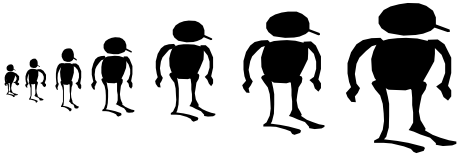
Make fetching resources faster

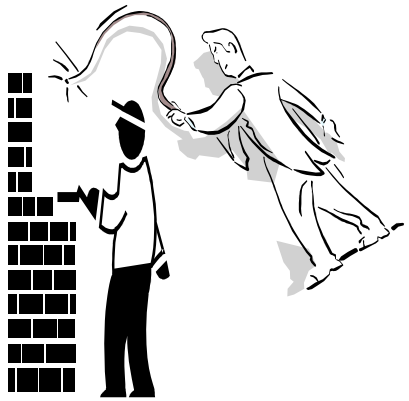
Put more persons (faster?!!!)

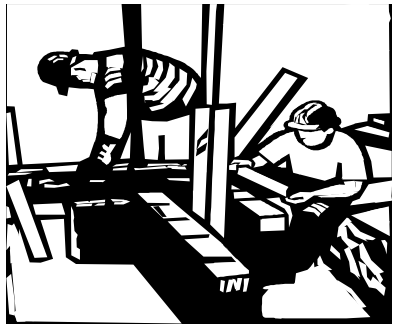
Distribute and manage tasks

Building more than one wall









Agenda

- The quest for speed (Why HPC)
  - Computer Basics
  - Faster Processors and Memory (work faster)
    - Single CPU limitations
  - Parallelism (do more parallely / more persons)
    - Fine Grained Parallelism – Pipelining and Pre-fetching
    - More processors – Coarse Grained Parallelism
    - More processors – Classifications
    - Memory Hierarchies and Core Memories (better resources)
    - Software Parallelism (tools for using the resources)
  - Network Technologies (speed of communication)
- HPC Cluster elements Management Issues and Issues of Scale (building more than one wall)

Why High Performance

- Why High Performance
  - High Performance means doing things faster
  - More Efficiency
  - More Profit
  - The ability to solve more difficult and complex problems in the same time (pushing the barriers)

The Quest For Speed (Why HPC)

- Applications require fast machines (super computers)
  - Computer generated imagery (movies, flight simulators etc)
  - Cat scans, MRI, Virtual Surgery, Virtual Worlds
  - Weather prediction, Price prediction
  - Data mining for usage patterns etc.
  - Bulk Disk Servers / Web Servers
  - Biological sciences / Molecular Simulations / Many body problems / aero dynamics etc
- Ever Increasing Needs for
  - Faster - processing power, network bandwidth, Input Output
  - Faster technology - more achievement- more expectation, Faster and faster machine – new technology

Computer Basics

- Electronic circuit that can recognize and execute simple instructions
  - Adding 2 numbers
  - Checking to see if the number is zero
  - Storing, Remembering and fetching numbers (data)
- The instructions together form the machine language
- Designers try to make these very simple and use clever combinations to realize the computer
- It is very difficult and tedious to work in the machine language let alone solve staggering problems.
- Solution??

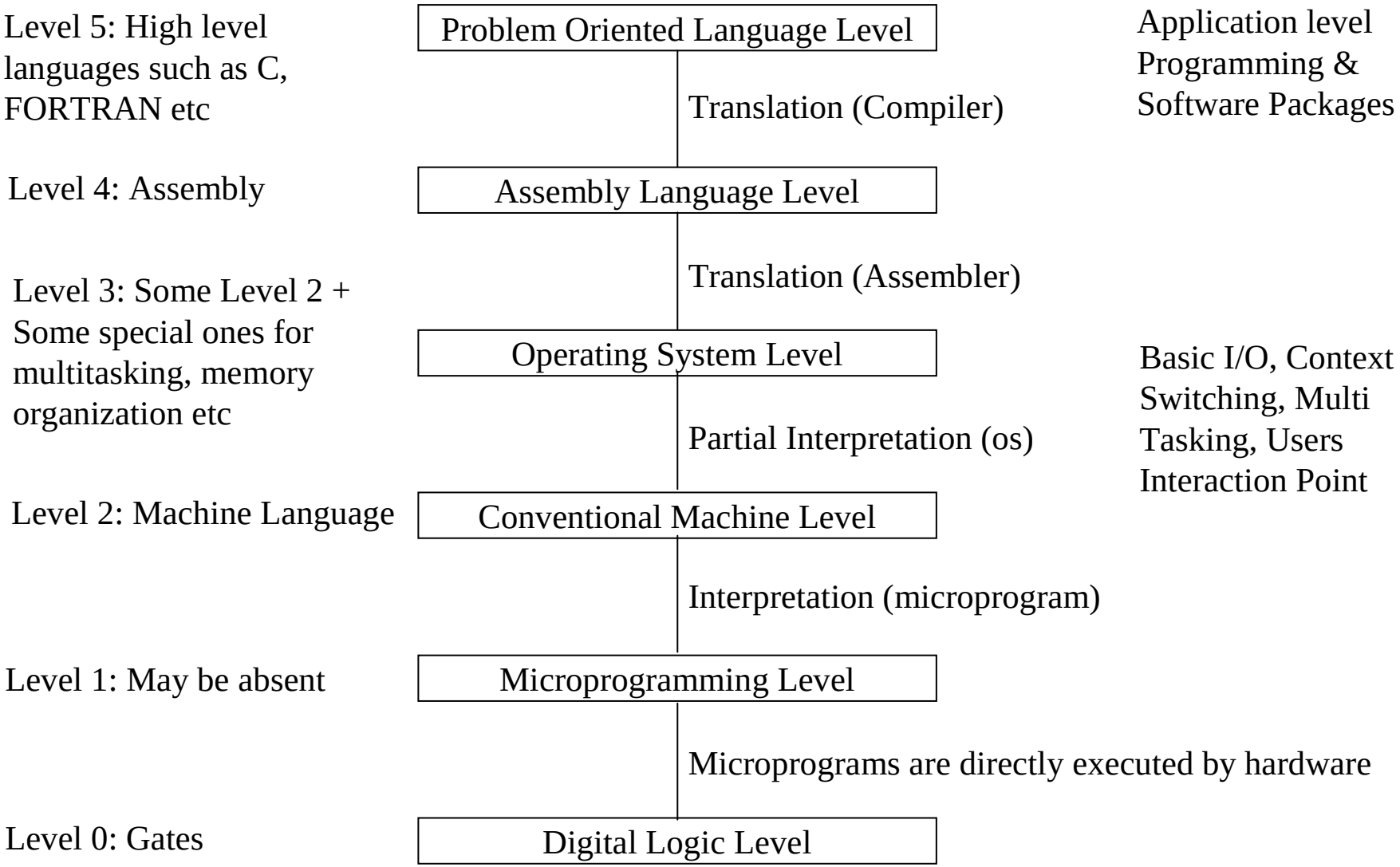
Computer Basics: Binary System and ASCII set

- Computers can only count with two numbers 0 & 1
  - 0 is low voltage and 1 is high voltage of some circuit
  - Language of computers consists of 0s and 1s

- Computers also differentiate between characters and numbers
- The most used characters & punctuations etc are mapped to numbers
- Thus in ASCII coding “a” is 97 “0” is 48 and “A” is 65 and “Z” is 90
- Capitals are different from normal characters
- So human language and computer language is defined. How to bridge the gap??

0	000	000
1	001	001
2	010	002
3	011	010
4	100	011
5	101	012
6	110	020
7	111	021

Computer Basics: Multilevel Machine

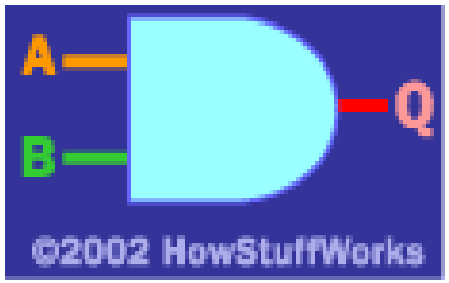


Tanenbaum Structured Computer Organization

Computer Basics: Gates

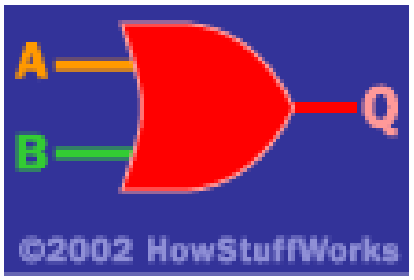
AND Gate

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



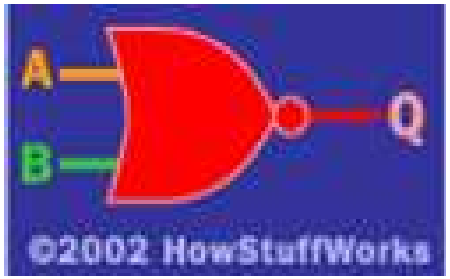
OR Gate

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



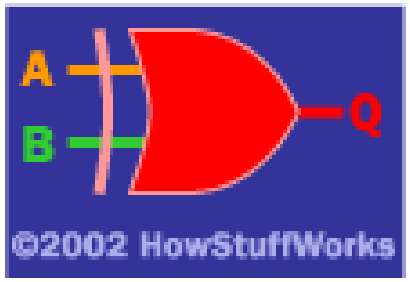
NOR Gate

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

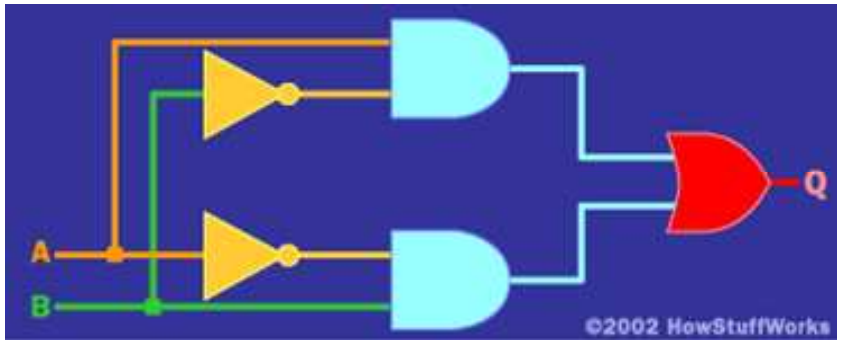


XOR Gate

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



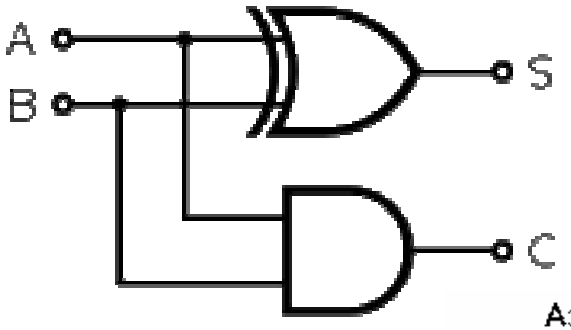
XOR ALTERNATIVE IMPLEMENTATION

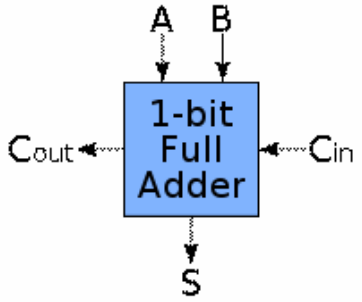


Computer Basics: ADDITION

1-bit Adder with Carry-Out

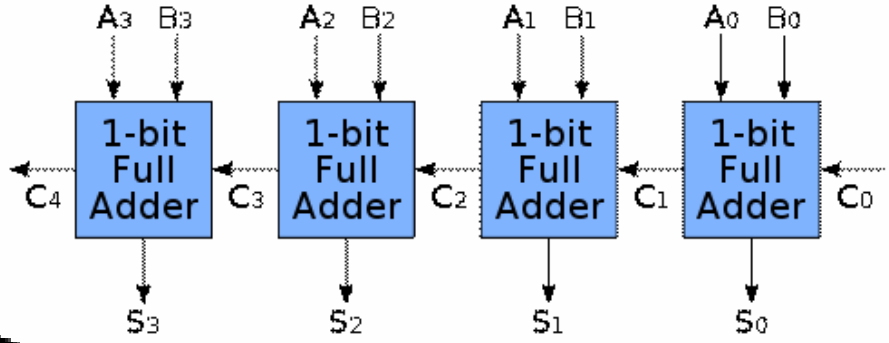
A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

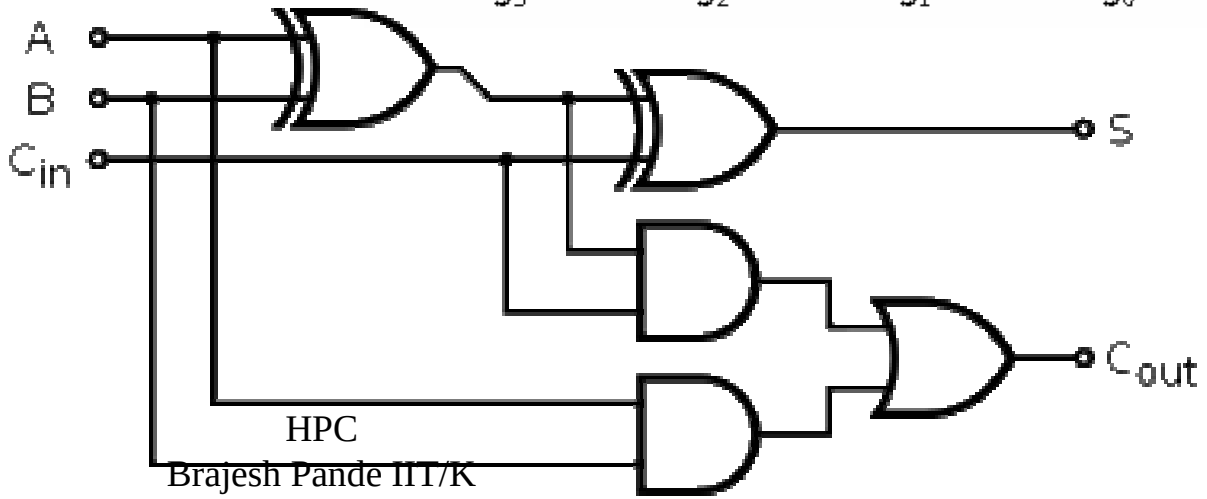




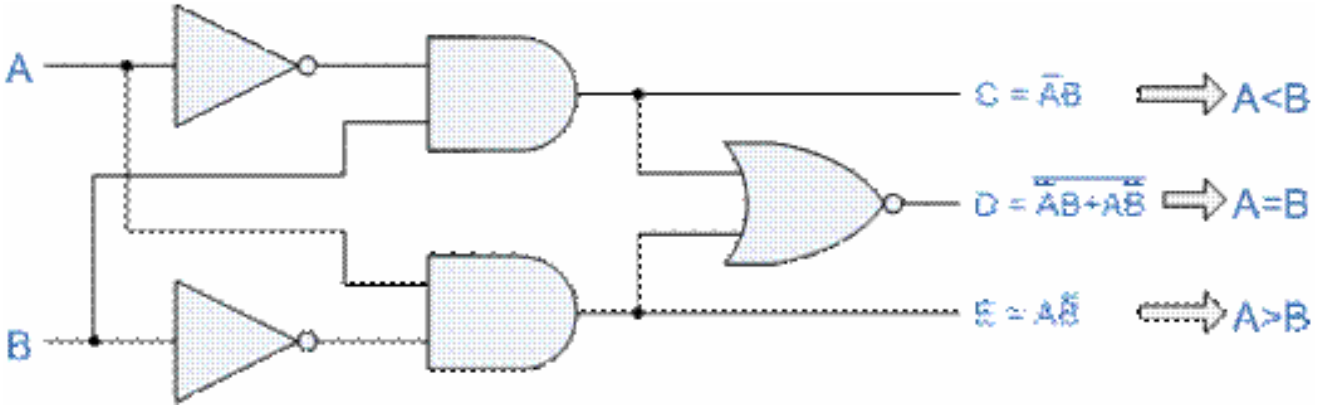
One-bit Full Adder with Carry-In and Carry-Out

C_{In}	A	B	S	C_{Out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



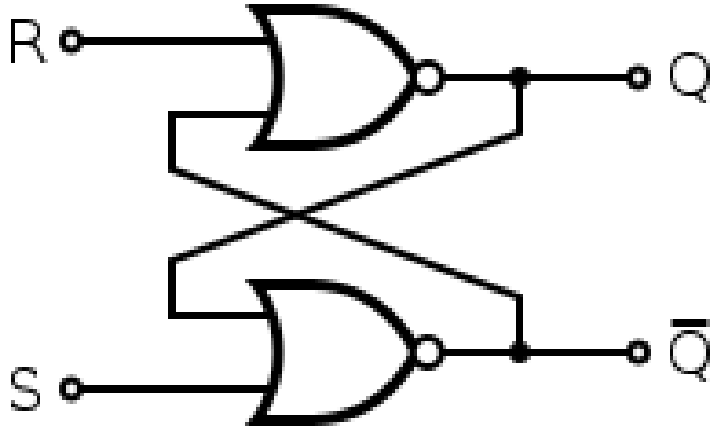


Computer Basics: Comparators

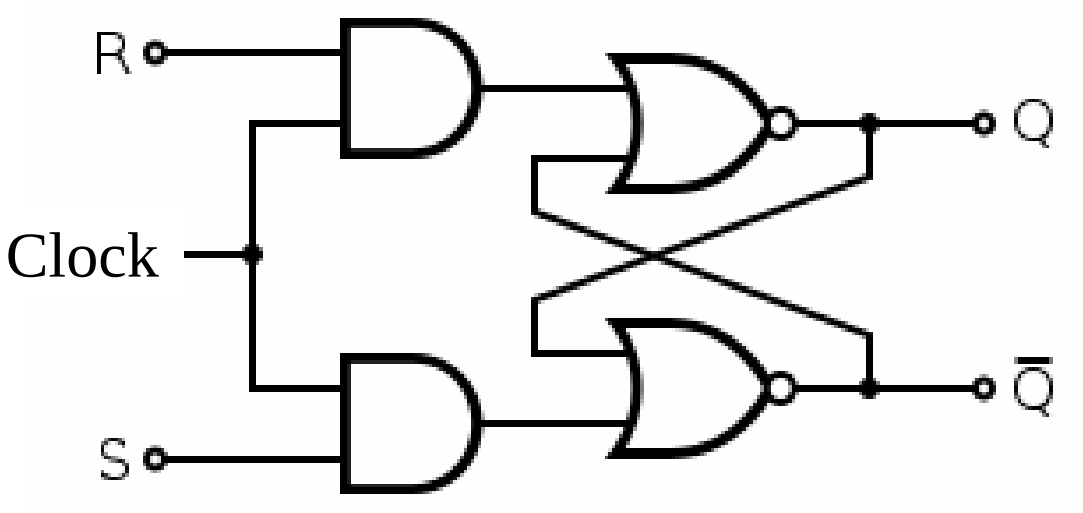


Computer Basics: Latches

S	R	Action
0	0	Keep state
0	1	Q = 0
1	0	Q = 1
1	1	Restricted combination



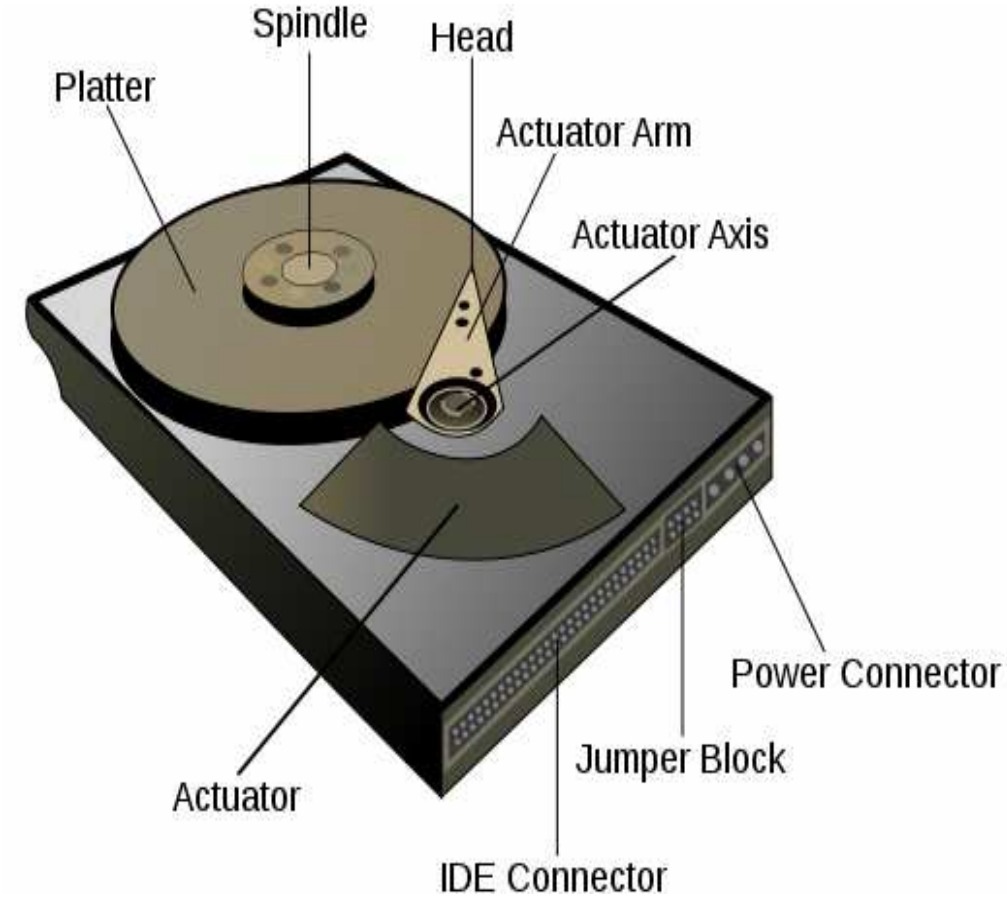
Computer Basics: Clocked Latches

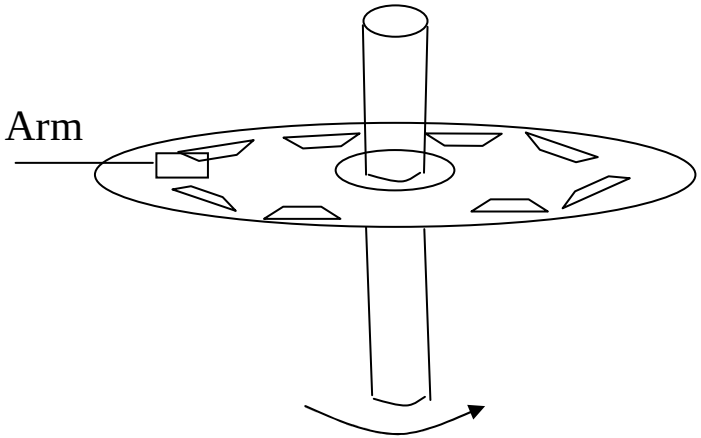


Electronic circuit that can recognize and execute simple instructions

- Adding 2 numbers
- Checking to see if the number is zero
- Storing, Remembering and fetching numbers (data)
- And can have some timing

Computer Basics: Disk Drives and Tapes





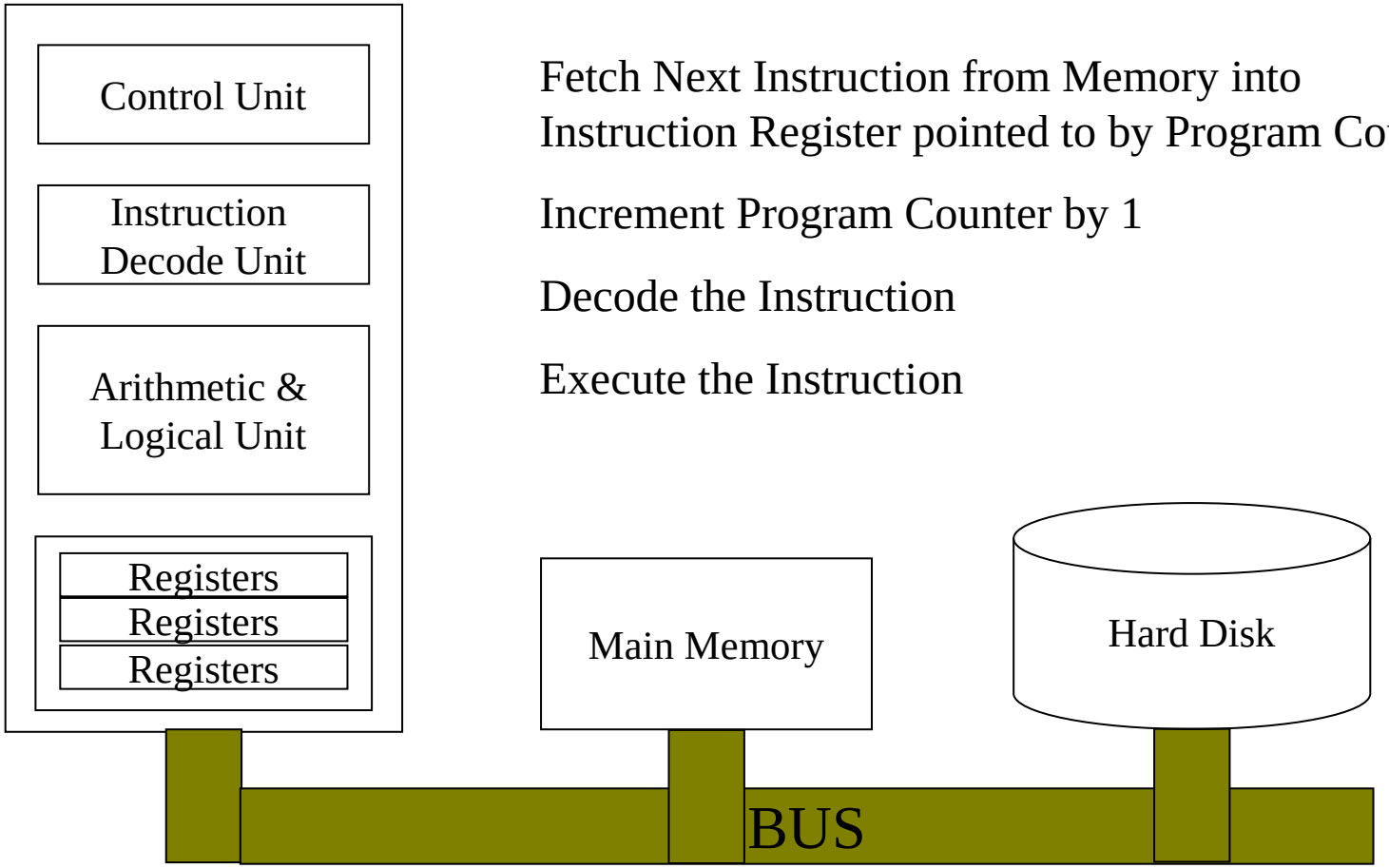
Tracks

Sectors

Seek Time between tracks

Latency Time between sectors

Computer Basics: A Simple Organization



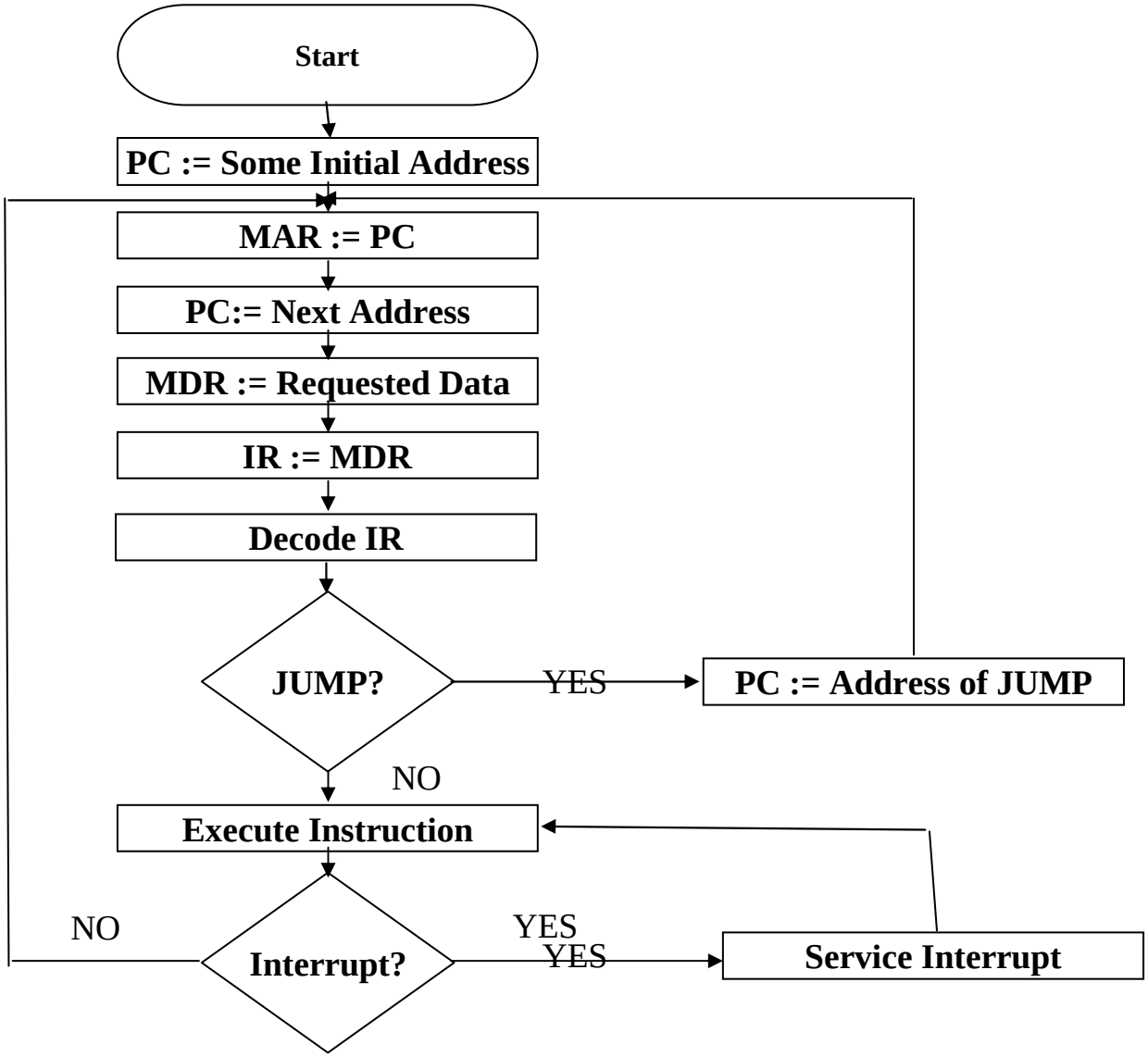
Fetch Next Instruction from Memory into Instruction Register pointed to by Program Counter

Increment Program Counter by 1

Decode the Instruction

Execute the Instruction

The Fetch and Execute Cycle



The Quest For Speed – CPU Speeds

Name	Date	Transistors	Microns	Clock Speed	Data Width	MIPS
8088	1974	6,000	6	2MHZ	8	0.33
8088	1979	29,000	3	5MHZ	16/8	0.64
80286	1982	134,000	1.5	6MHZ	16	6
80386	1985	275,000	1.5	16MHZ	32	5
80486	1989	1,200,000	1.5	25MHZ	32	20
Pentium	1993	3,100,000	0.8	60MHZ	64/32	100
Pentium-2	1997	7,500,000	0.35	233MHZ	64/32	~300
Pentium-3	1999	9,500,000	0.25	450MHZ	64/32	~510
Pentium-4	2000	42,000,000	0.18	1.5GHZ	64/32	~1700
Pentium-4(and more)	2004	125,000,000	0.09	3.6GHZ	64/32	~7000

The Quest For Speed – MEMORY / BUS Speeds

DEVICE	NAME	Rate Giga bytes
DRAM	Dynamic Random Access Memory	0.176
SDDRAM	Synchronous DRAM (clocked)	0.533
DDR 2,3,4 - SDRAM	Double Data Rate (uses both edges)	2.1 – 52.8

Data is transferred at the rate of the bus speeds

Assume a bus of 64 bits

Data transfer rate (DDR) = bus clock rate * 2 * 64 / 8

The Quest For Speed – BUS Speeds

While total memory has gone from a few MB to a few GB bus speeds have also increased

Bus	**Width (bits)**	**Bus Speed (MHZ)**	**Band Width (MBytes/Sec)**
8-bit ISA	8	8.3	7.9
16-bit ISA	16	8.3	15.9
EISA	32	8.3	31.8
VLB	32	33	127.2
PCI	32	33	127.2
64 bit PCI	64	66	508.6
AGP	32	66	254.3
AGP (x2 mode)	32	66*2	508.6
AGP (x4 mode)	32	66*4	1013.2
AGP8x 4000, PCIexpress 8-16000, QPInterconnect 25600 Hyper Transport 51200			

The Quest For Speed – BUS Speeds

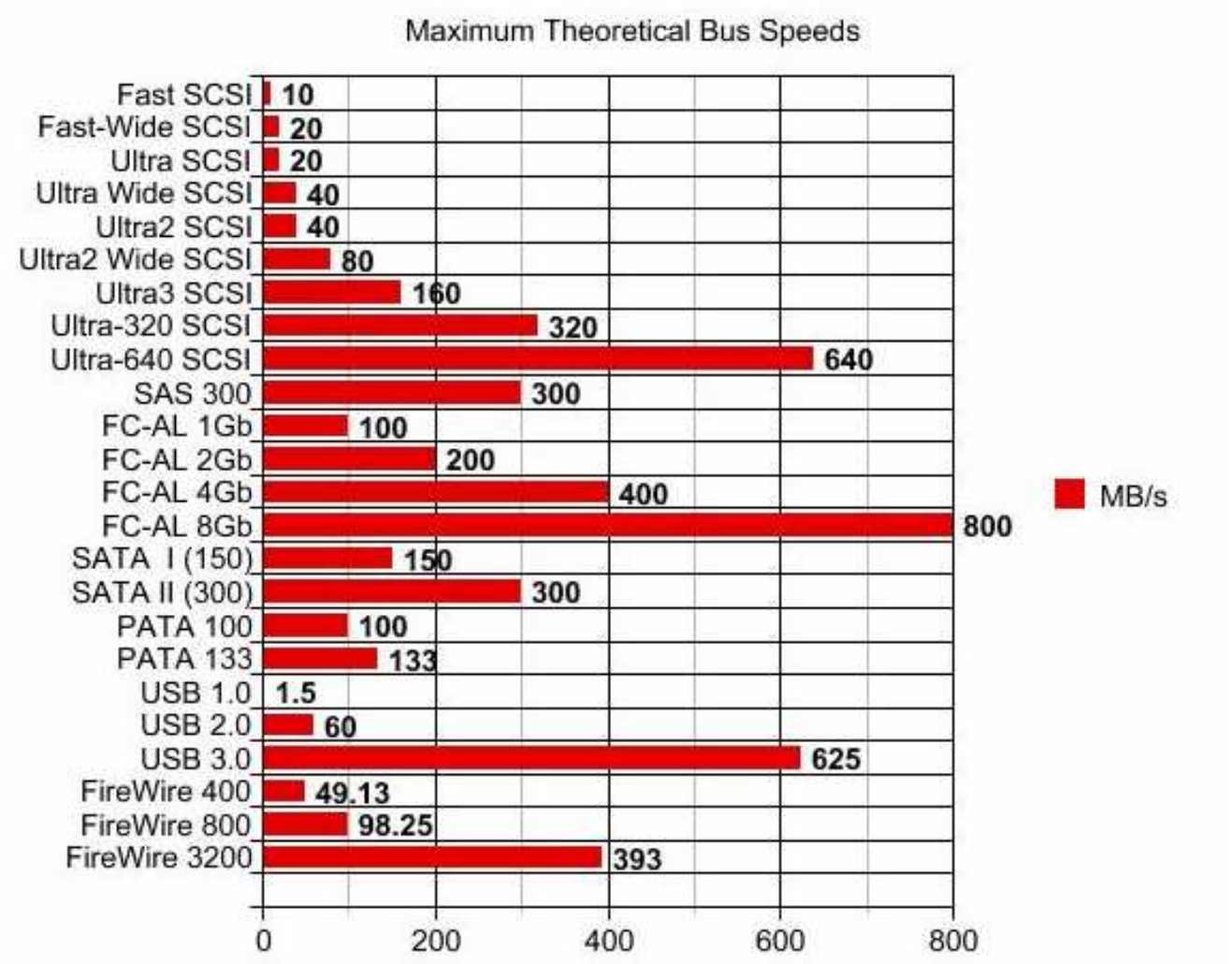
While total memory has gone from a few MB to a few GB bus speeds have also increased

Acronym	**Bus Name**	**Width (bits)**	**Bus Speed (MHZ)**	**Band Width (MB/Sec)**
8-bit ISA	Industry Standard Arch	8	8.3	7.9
16-bit ISA	-do-	16	8.3	15.9
EISA	Extended ISA	32	8.3	31.8
VLB	Vedio Electronic Standards Association. Local bus	32	33	127.2
PCI	Peripheral Component Interconnect	32	33	127.2
64 bit PCI	-do-	64	66	508.6
AGP	Accelerated Garaphics Port	32	66	254.3
AGP (x2 mode)	-do-	32	66*2	508.6
AGP (x4 mode)		32	66*4	1013.2
AGP8x 4000, PCIexpress 8-16000, QPInterconnect 25600 Hyper Transport 51200				

The Quest for Speed – Disk Speeds

Acronym	Meaning	Description
SASI	Shutgart Assoicates System Interface	SCSI Predecessor
SCSI	Small Computers System Interfac	Bus Oriented with concurrent operations (5 MBps 8 PAR) (10-640)
SAS	Serial Attached SCSI	SCSI Improvement Uses Serial Communication (300-1200 MBps)
ST-506	Segate Technology	Historical Seagate 5MBp/s
ST-412	Seagate Technology	Historical Minor over ST-506
ESDI	Enhanced Small Disk Interface	Historical Compatible with ST faster and more integrated (5, 15, 20)
ATA (PATA)	Advanced Technology Attachment	Successor to ST/ESDI with disk controller on device. No concurrency (150-300 MBps)
SATA	Serial ATA	Successor to ATA. Concurrent (150-300 MBps)
FC	Fiber Channel	Successor to SCSI (upto 800 MBps)

The Quest For Speed – Disk Speeds



The Quest For Speed – Single CPU Limitations

- CPU can execute only one instruction at a time
  - program execution is purely sequential
  - multiprogramming is possible thanks to time division
  - increasing performance means making the clock faster
- fundamental limit #1: smaller wires imply faster speed
- fundamental limit #2: heat dissipation
  - the smaller the system the more heat it generates
  - the smaller the system the harder it is to dissipate the heat
- Divide between Memory and CPU; the rest of the machine also becomes a bottle neck; most of memory is idle; busses are slow
- LIMIT OF SPEED UP with ONE CPU at any point of time
- How to Increase Speed???

Parallelism

- Parallelism - key to the performance of machines
- Parallelism is the quality that allows something to be done in parts that work independently.
  - Hardware parallelism
    - CPU level parallelism
    - System level parallelism
    - Memory hierarchies and bus connectivity
    - I/O parallelism
  - Software parallelism
    - Programming Environments
- Queuing Systems and Schedulers

CPU Level Fine Grained Parallelism Other..

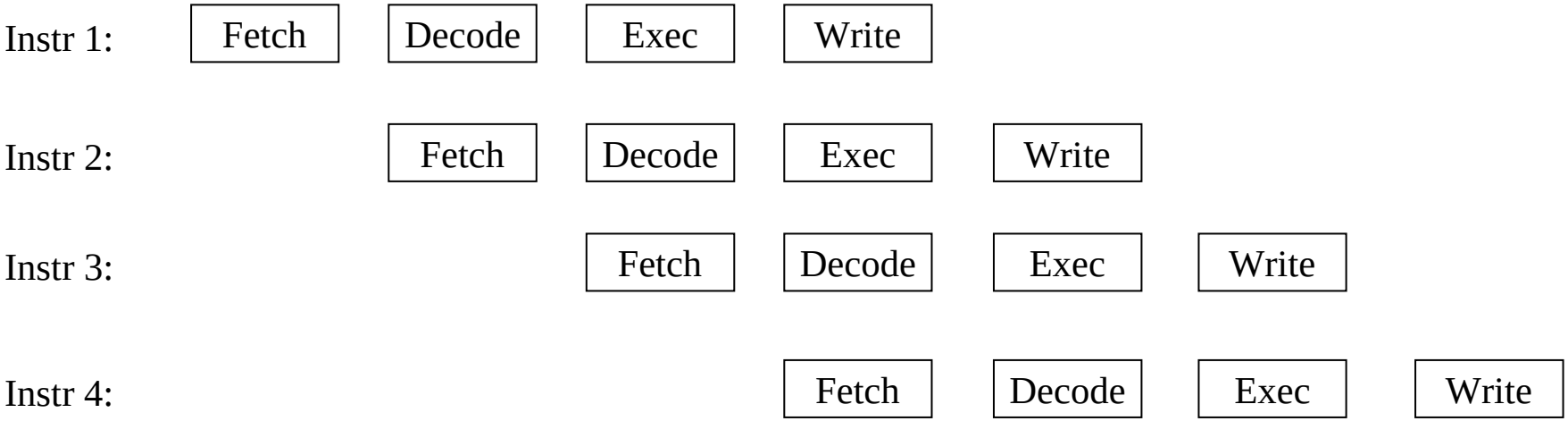
- Parallelizing Instructions

1. E=A+B
2. F=C+D
3. G=E*F
4. H=X*Y

1 and 2 and 4 are independent
3 is dependent on 1 and 2
1 2 and 4 can be executed in any order and also simultaneously

  - Out of order Execution
  - Pipelining
  - Register Renaming
  - Speculative Execution
  - Branch Prediction
  - SuperScalar Architecture

Pipelining – Example Interleaving Instructions



Pipelining - Example

- IF (A) then A1,A2,A3…..ELSE B1,B2,B3…
- Pipe line is a small area in the CPU where instructions next in line to be executed are stored
- Super scalar architecture pre-fetches both branches whereas normal architecture pre-fetches only one of the branches
  - The result of the branch is not known so Instructions are fetched to each execution unit and once the branch result is known then either results of first execution unit of the second are used
  - A1 B1 While the conditional is being evaluated
  - A2 B2 A1 A2 and A3 and B1 B2 and B3 are
  - A3 B3 fetched and await execution

CPU Level Parallelism – Fine Grained

- CISC (Complex Instruction Set Computer)
  - Variable length Instructions
  - Try to do many things in a single Instruction
  - Require more cycle
  - Try to reduce the number of instruction calls
  - A complex micro program can make it expensive
- RISC (Reduced Instruction Set Computer)
  - Small Set of simplified instructions
  - Same size of instructions
  - Require a single cycle
  - Can be pipelined

More CPUs - Coarse Grained Parallelism

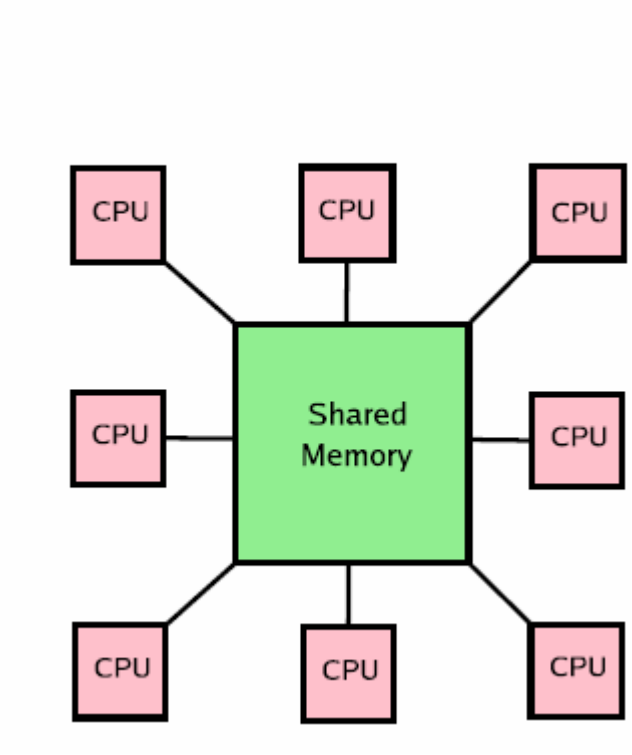
- Many CPUs running at normal speed (some defn)
- Speed up computations
  - at least those that can be parallelized
- Deal with heavier loads
  - different CPUs deal with different transactions, users
- Enormous range of systems:
  - single servers with 2, 4, 8, 16, and more CPUs
  - supercomputers and clusters 10 – 10000 and more CPUS
  - Internet wide computing, grid computing
  - Virtually no limit
  - Homogeneous as well as heterogeneous

More CPUs - Coarse Grained Parallelism

- A set of CPUs and available memory / memories can be connected and used in several ways
- Classification based on topology of connection and use SMP, MMP, CLUSTERS
- Classifications based on nature of instruction execution and data sets
  - Flynn's taxonomy
    - SISD Classical machine
    - SIMD Vector machines
    - MISD arguably none possibly pipelined machines, fault tolerant systems
    - MIMD Multi Processors
- Vector Processors
  - Provide Vector Operands for Instructions
  - Compilers convert loops to vector operations
  - 50-100 times more expensive
  - Special memory access is provided
  - Used in some legacy applications

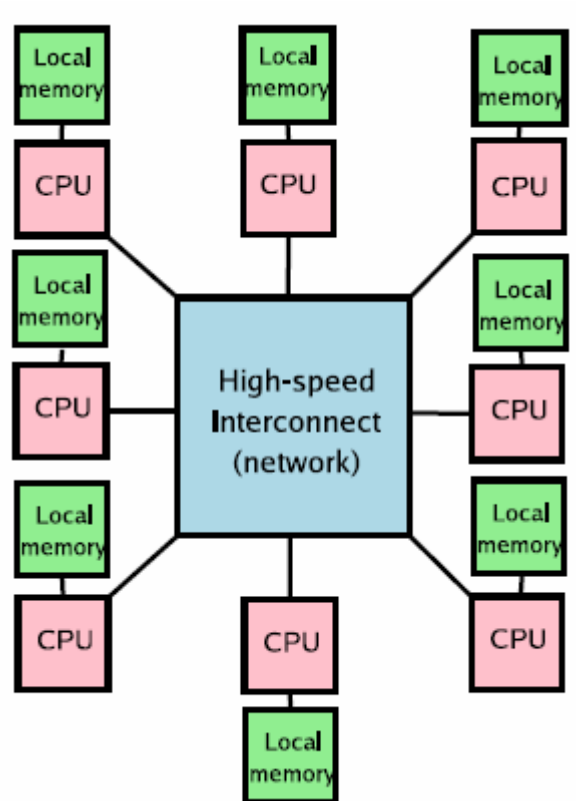
More CPUs - Coarse Grained Parallelism

- Symmetric Multi Processors or Shared Memory Processors when sharing OS SMP
- between 2 and few hundreds of CPUs
- all have access to the entire physical memory
- memory access time 10 – 50 nano seconds
- looks simple
- difficult to implement
- Limit on the number of CPUs on same board (heat problems)
- 10-20 times costlier



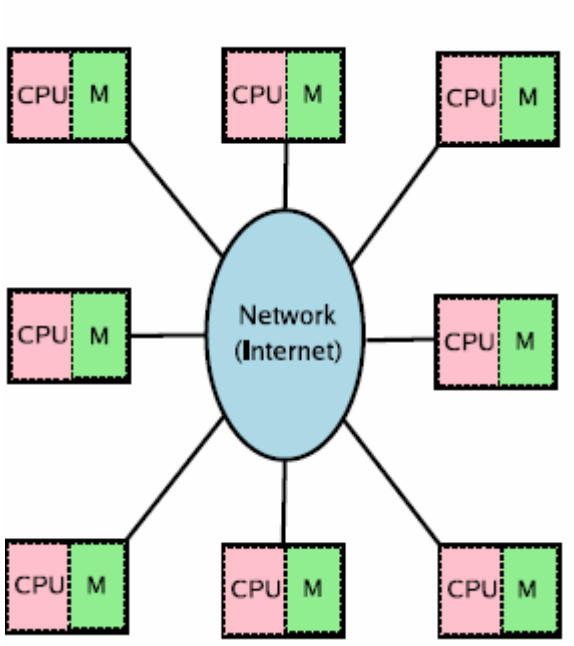
More CPUs - Coarse Grained Parallelism

- Message Passing Multi Processors (Massively Parll)
- 10-1000s of CPUs
  - each CPU has its own memory, inaccessible by others
- communicate by sending messages over the interconnect
- Latencies 1-50 micro secs
- much easier to build than shared-memory ones
- harder to program
- each node can be a shared memory multiprocessor
- Cost and speed of interconnect is important

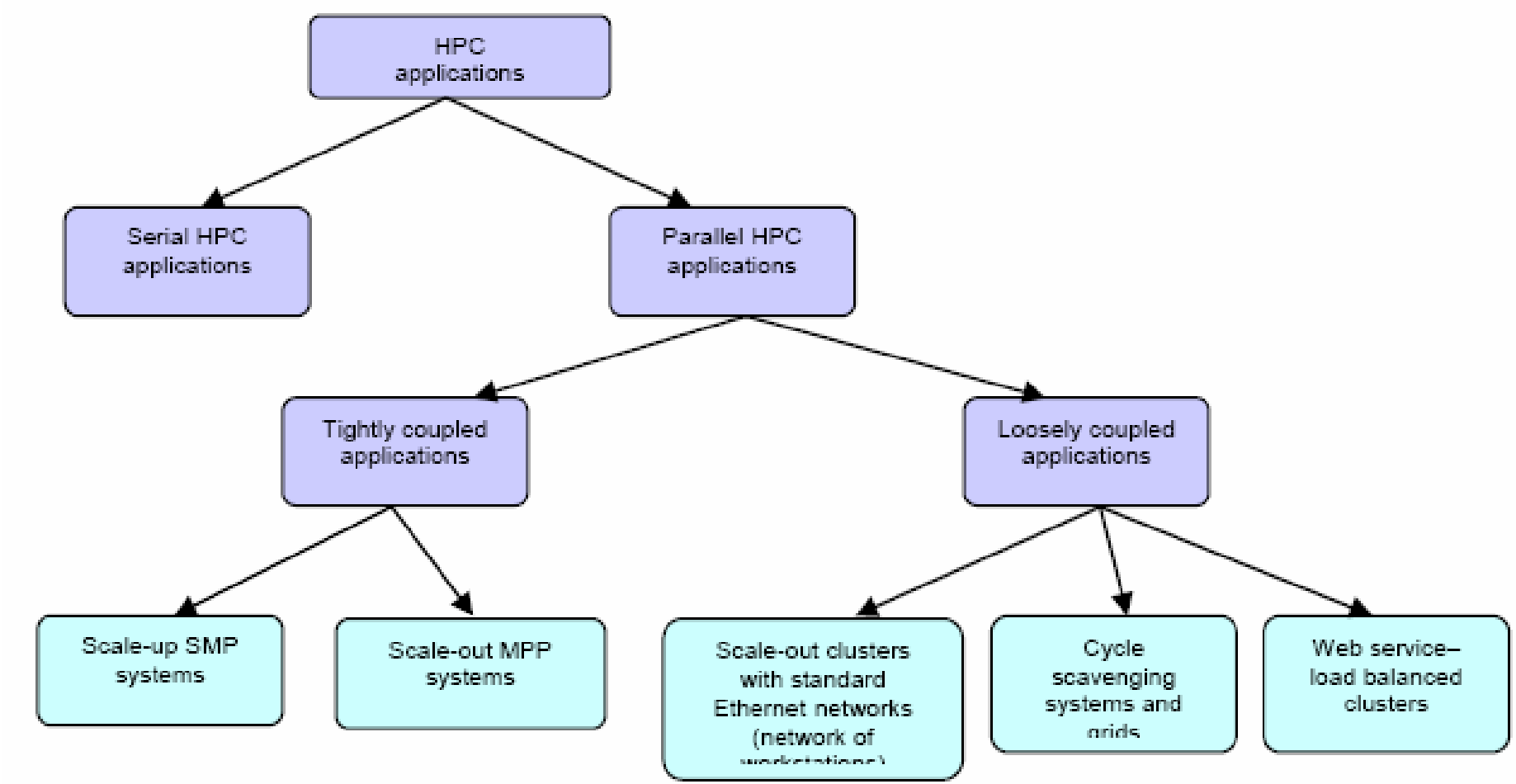


More CPUs - Coarse Grained Parallelism

- Distributed Multi Processors (Clusters)
- Potentially huge number of nodes
- each node is a complete system
- very similar to message-passing multicomputers
- but long latencies 1-50 milli-sec
- often no QoS (e.g. over Internet)
- loosely-coupled. vs. tightly-coupled
- Cheapest of the three

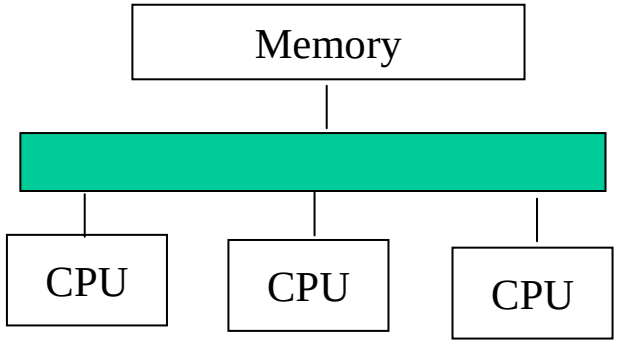


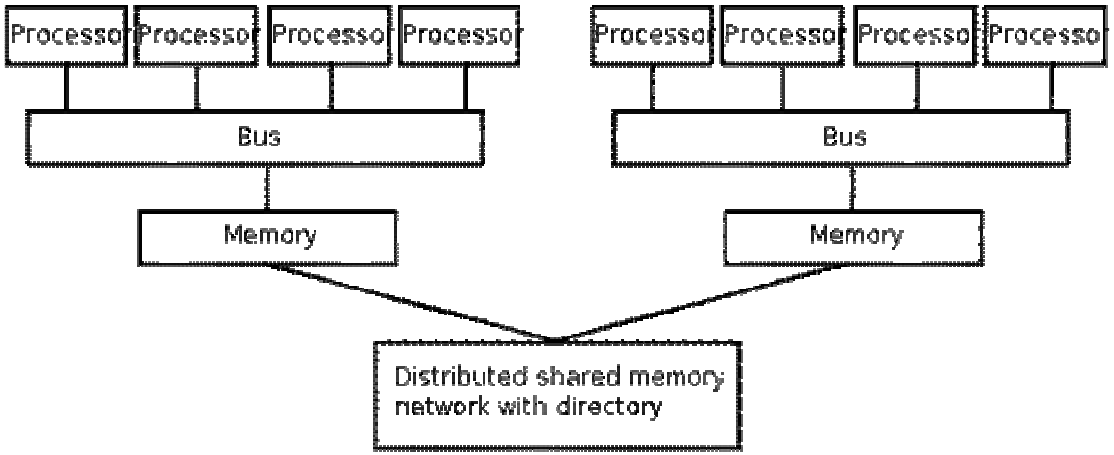
Classification of HPC Applications



More CPUs – Some Classifications

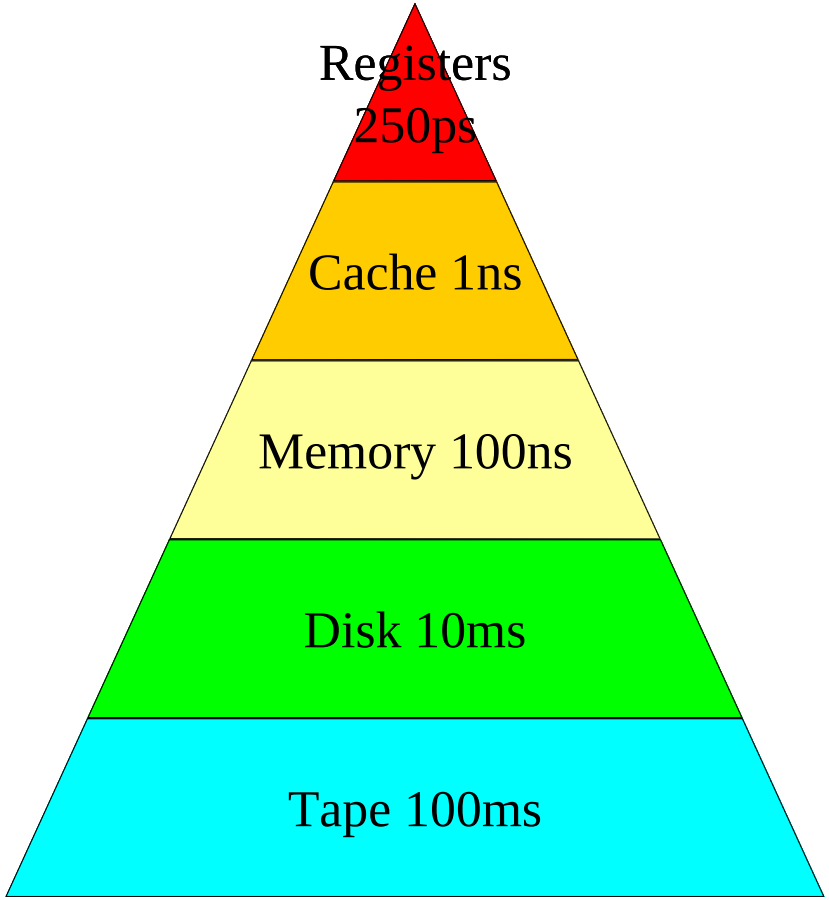
- Classification Based on Memory Access and Instruction Sets
- UMA
  - Uniform memory access machines
- NUMA
  - Non Uniform Memory Access
  - Local Fast Memory
  - Slower Shared Memory





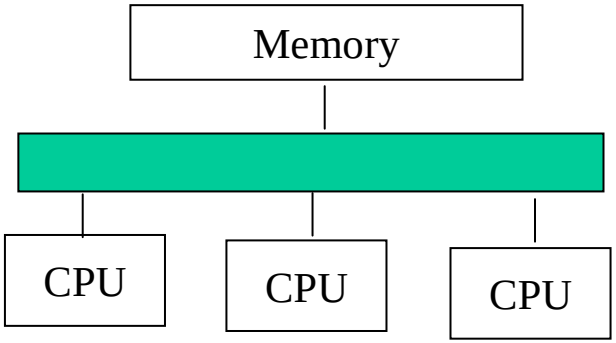
Memory Considerations

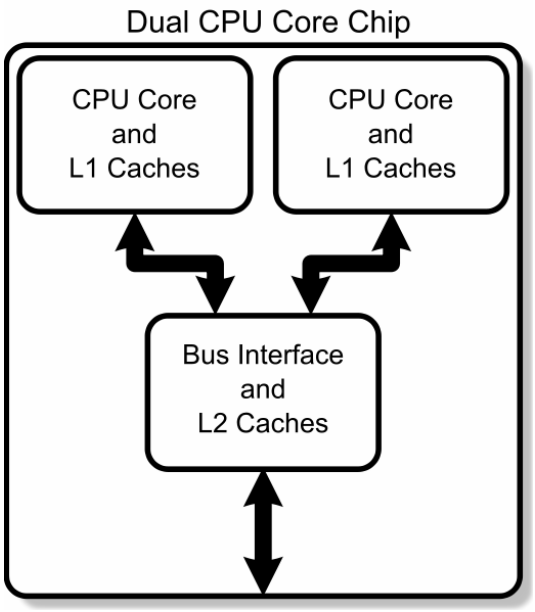
- Caching
- Smaller applications entire data fits in cache – efficient
- Larger Apps with high cache hit ratio also efficient
- Problem that do not fit in cache of one cpu but can fit in combined caches of n cpus in parallel can have super linear scale up
- Locality of reference
- Machines provide several levels of cache to optimize memory access



Dual Core / Multi Core Memories

- With more than one CPU there is a contention for the memory bus
- As there is only one path to the memory any of the CPU can demand the memory / register bus and some of the CPUs may have to wait before it can access the bus
- The overall system performance goes down
- The answer is to provide multiple paths to the memory as well as efficiently manage the local cache and memory of each CPU (Direct Connect)
- This is captured by Multi Core (Direct Connect) and PCI express (“Hubbed Connect”) architectures
- All devices can be connected point to point



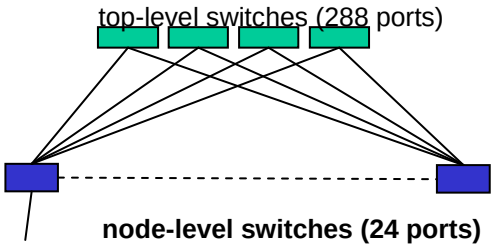


Threads vs Processes

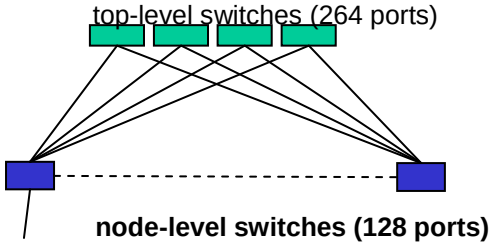
- Smallest unit of execution that can be scheduled by the CPU
- On a single CPU we have time division multipexing between processes and threas
  - Improves CPU utilization
- Threads exist within a processes
- Processes have a state but threads share the state of the parent and have only essential states to run (registers, scheduling policy priority, signal states etc
- Threads share variables and address space
- Processes interact by inter-process communication and are therefore slower
- Threads in a process switch context faster than processes

High Speed Network Inter Connects

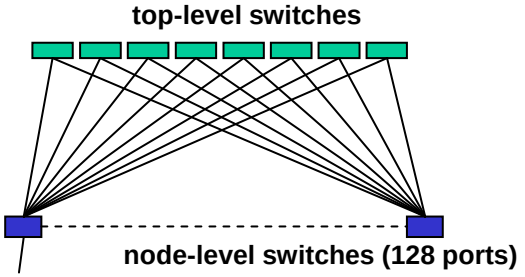
- Infiniband
  - Standard for fast HPC
  - IB 4x – speeds 40Gb/s SDR, DDR, QDR latency of 200, 150, 100 ns, 10,20,40 Gbs
  - Scalable topologies with federation of switches
- Myrinet
  - Speeds up to 800MB/s, <6μSec MPI latency
  - 16 port, 128 port, 256 port switches max 10Gbs
  - Scalable topologies with federation of switches
- Quadrics
  - Elan 4 – 800MB/s, <3μSec MPI latency
  - 8 port, 32 port, 64 port, 128 port switches
  - Latest 10Gbs 1.3 μSec latency
  - Scalable topologies with federation of switches
- GigE
  - 60-80MB/s, >40 μSec MPI latency



Connects to 12 nodes

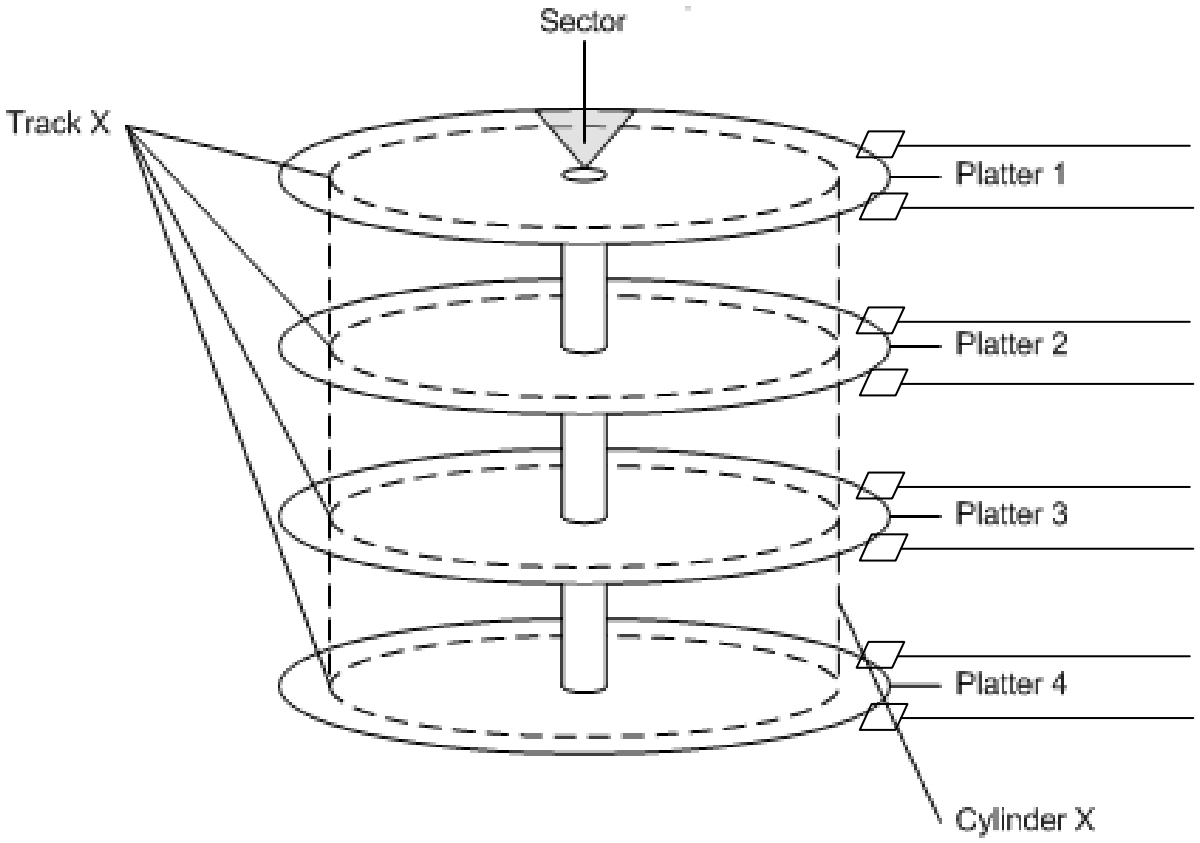


Connects to 64 nodes



Connects to 64 nodes

Fast File Systems



- Split data to span blocks from across many blocks
- At form factor of 1 inch limits this to 3
- Added parity for error correction
- If parity blocks are at the outer edge and more relevant data towards the centre then latency can be reduced
- More arms mean more power!

Fast File Systems RAID LEVELS

Redundant Array of Inexpensive DIsks

Striping

Mirroring

1 Parity Unit and Error Correction Make 3 4 5

Extra Parity Unit Makes 6

Hybrid RAID 1+0 0+1 5+1 etc

Raid	Description
0	Block Level Striping
1	Mirroring
2	Error Correction Hamming
3	Byte Level Striping Dedicated parity
4	Block Level Striping with Dedicated Parity
5	Block Level Striping With Distributed Parity
6	Block Level Striping with double distributed Parity

Parallel File Systems

- Store several copies of a data at the file system level
- Files can be fetched in parallel from different storage servers
- To map duplicate data to the store meta data servers are required
- Any request results in a consultation of the meta data servers and parts of the file are fetched from several sources

Software Parallelism

- Probably the toughest part
  - Mostly left to the user
  - Identify parts of programs that can be parallelized
  - Distribute the parallelizable parts and data so that communication is minimized
  - Gather the parallelizable parts and data
- Needs dependency analysis
- Requires multiprocessing (several control processes, independent stacks), threading (user space parallelism, requires cooperation among processes, same stack) and synchronization (locking etc)
- MPI, PVM, LAM, HPF, OPEN MP

Example OpenMP Program Fragament

- An OpenMP program:
  - always begins and ends like a serial program with a single thread of control, called the "master" thread,
  - but anywhere in the middle of the program the master may spawn additional parallel threads.
- Together the parallel threads accomplish the task that the single thread would have undertaken alone in a serial program.
- The section of program code in which parallel threads are spawned and run is called a parallel construct.
- In a shared-memory machine each parallel thread executes on its own processor. Yet they all share the same physical memory.

```
#include <omp.h> /* Here's the header file for OpenMP. */
……..
#pragma omp parallel for
for ( i = 0; i < arraySize; i++ ) {
    y[i] = sin( exp( cos( - exp( sin(x[i]) ) ) ) );
}
```

A Parallel Job Example MPICH

```
#include <iostream.h>
#include "mpi.h"
int main(int argc, char *argv[])
{
  MPI::Init(argc, argv);
  int rank = MPI::COMM_WORLD.Get_rank(); //rank of this process
  int size = MPI::COMM_WORLD.Get_size(); //num processes
  cout << "Hello World! I am " << rank << " of " << size << endl;
  MPI::Finalize();
}
MPI_Init(&argc, &argv);
int MPI_Comm_rank(MPI_Comm comm, int rank)
MPI_Finalize();

mpicxx
mpicc, mpif77 mpif90
mpirun –np 4 –mchinefile <somehostfile>
```

Cluster Components

- AMD Opteron, INTEL XEON
- OS – Linux in various flavors (Red Hat EL/AS)
- Interconnect – Myrinet, Infiniband,Gigabit
- Maintenance and monitoring – Sun Control Station, CMU Tool, NAGIOS, PXE boot
- Parallel Computing Environments – PVM, MPI, LAM
- Parallel File Systems – Lusture, GFS etc
- Libraries, Software, Compilers – NAG, PGI, CPMD, g77, GAUSSIAN, FLUENT etc
- Resource Manager – Sun Grid Engine, PBS PRO etc
- Power and AC

Heat In Data Centres

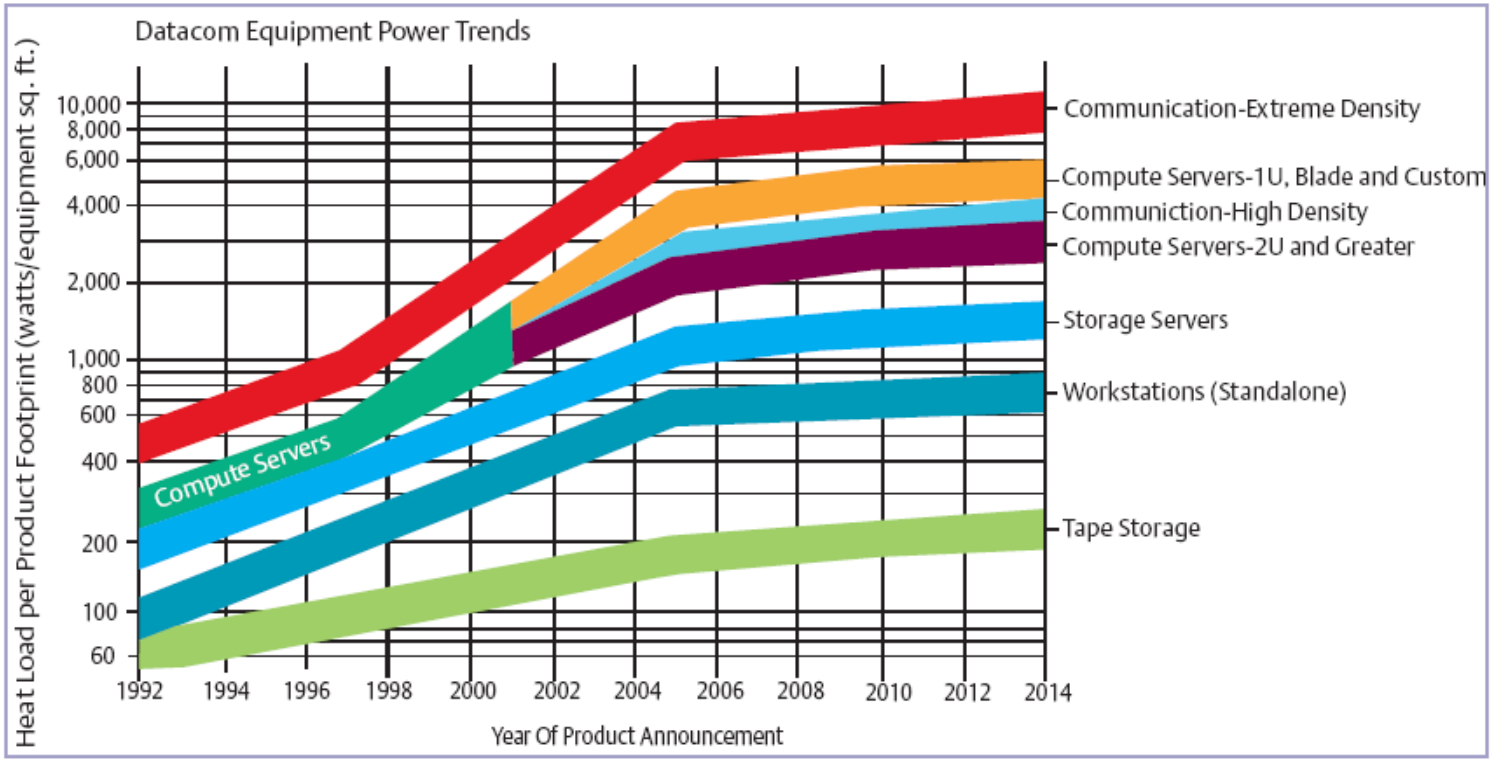


Figure 1. Equipment densities are rising even faster than once predicted.

Bottom Cooled Approach With Aisling

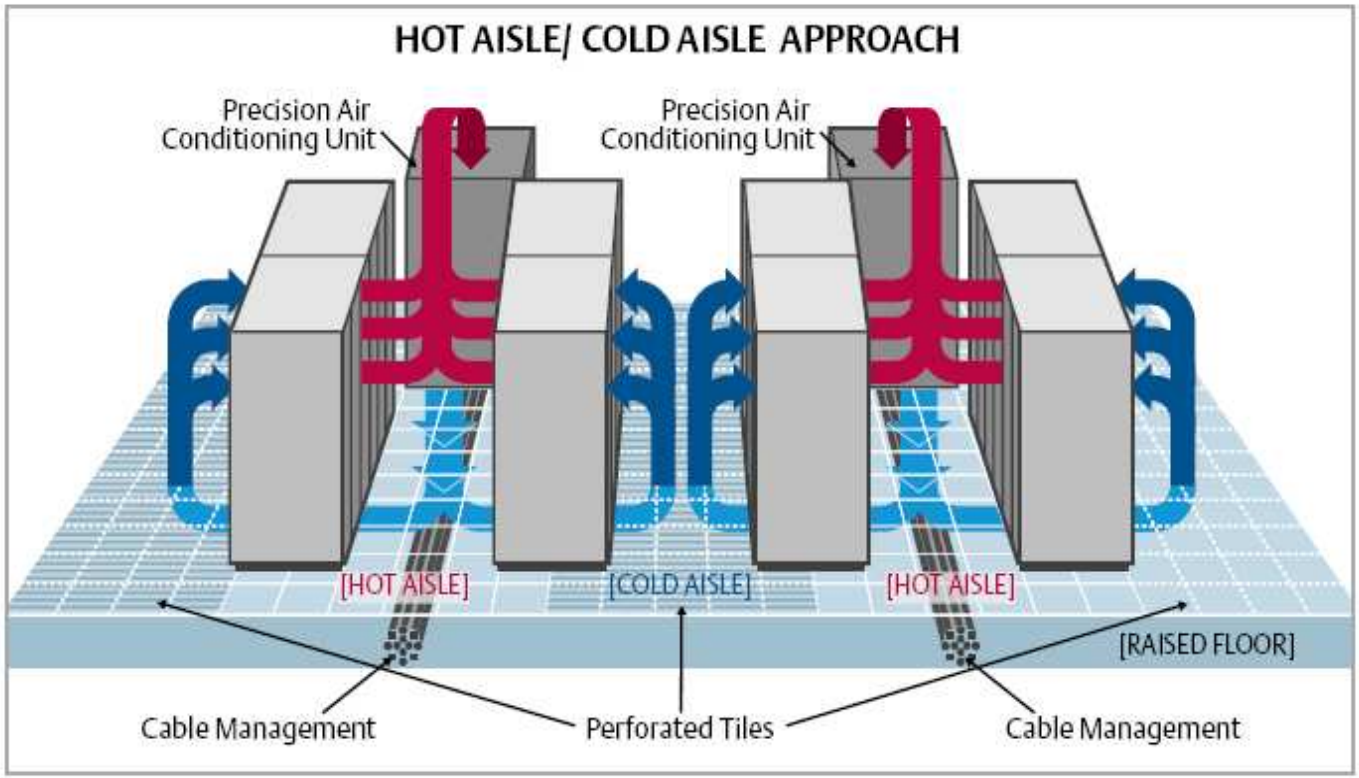
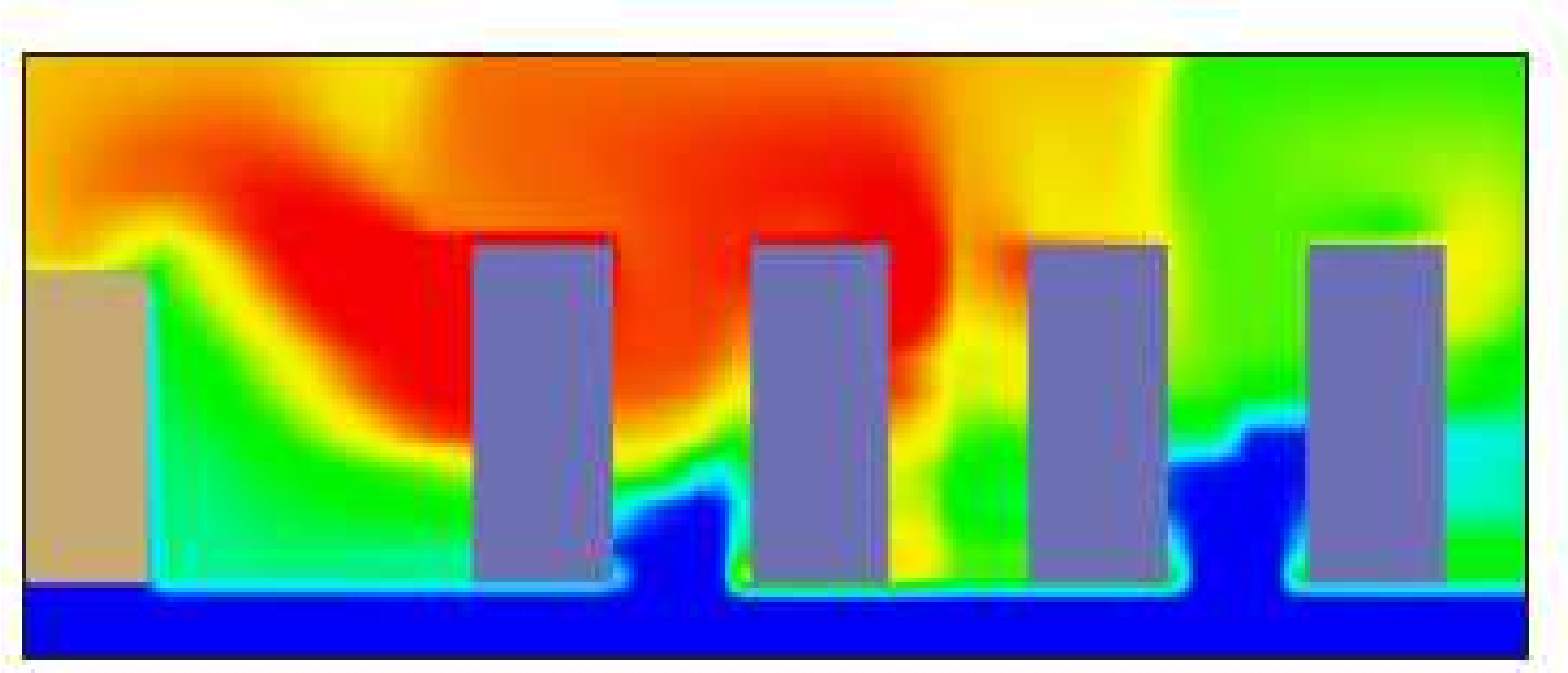
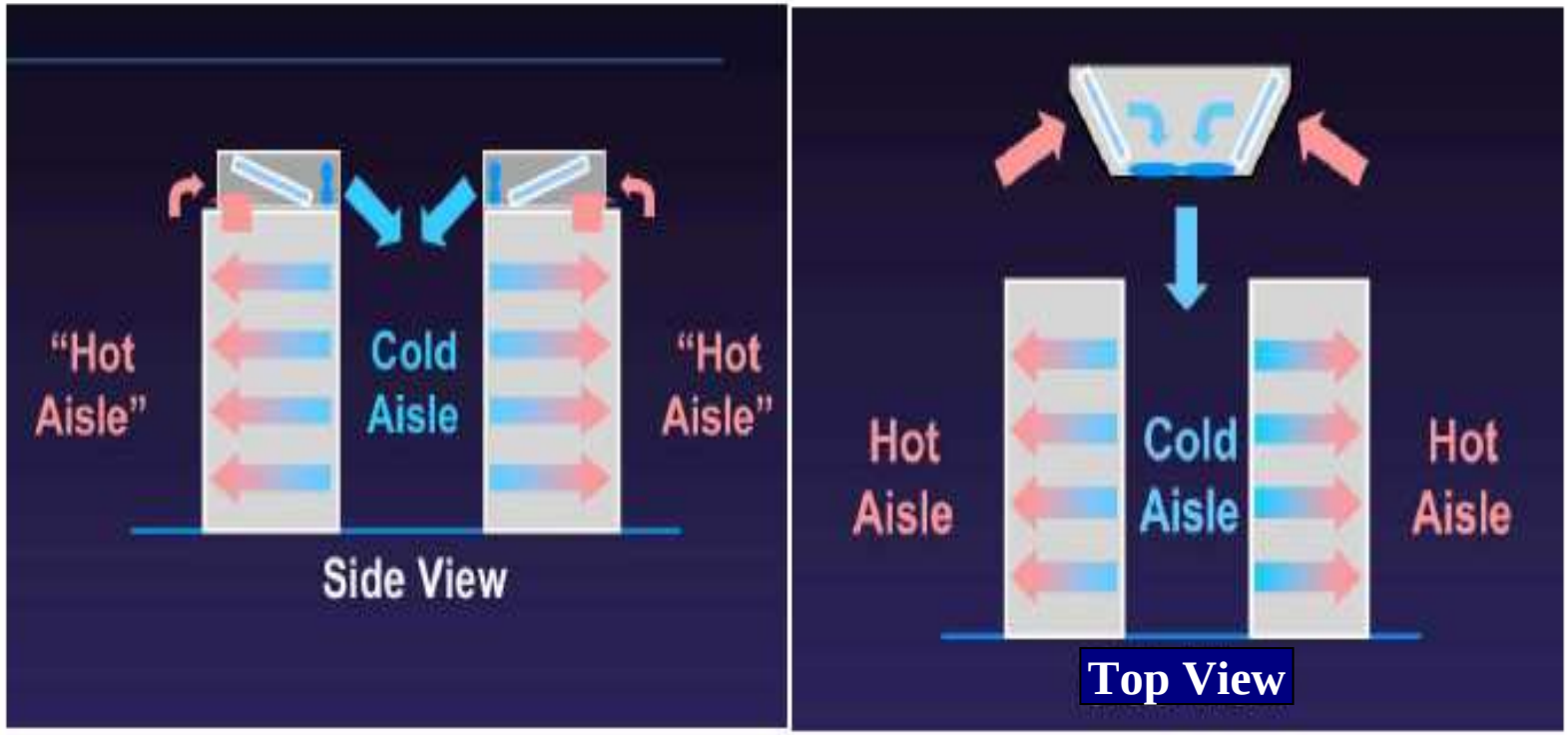


Figure 5. Racks arranged in a hot aisle/cold aisle configuration.

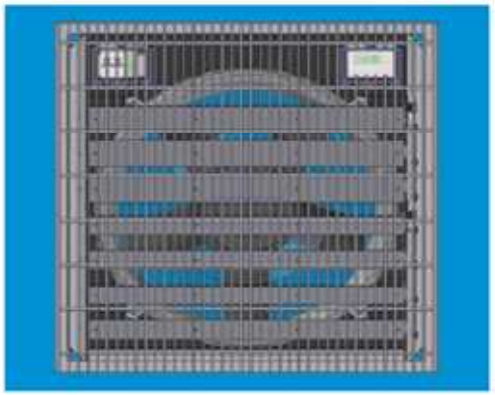
Extreme Heat Density Conditions

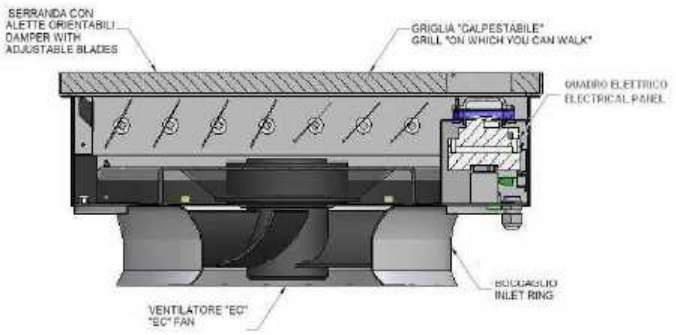


Managing Extreme Heat Top-Bottom Approach

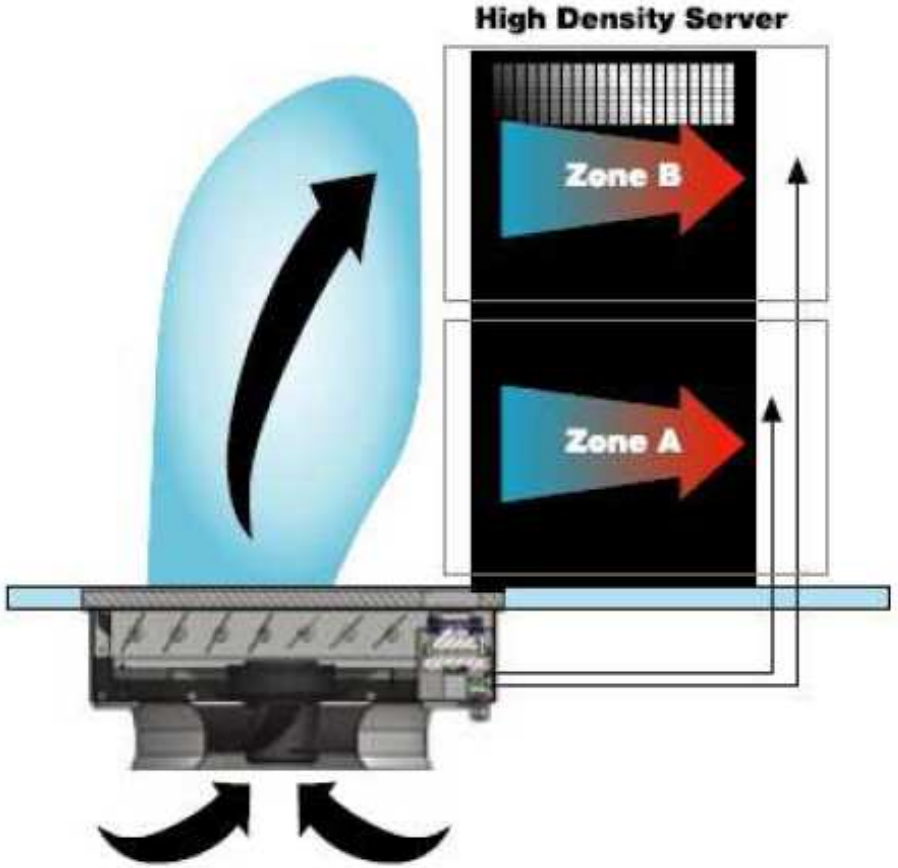


Managing Extreme Heat Bottom Active-Tile

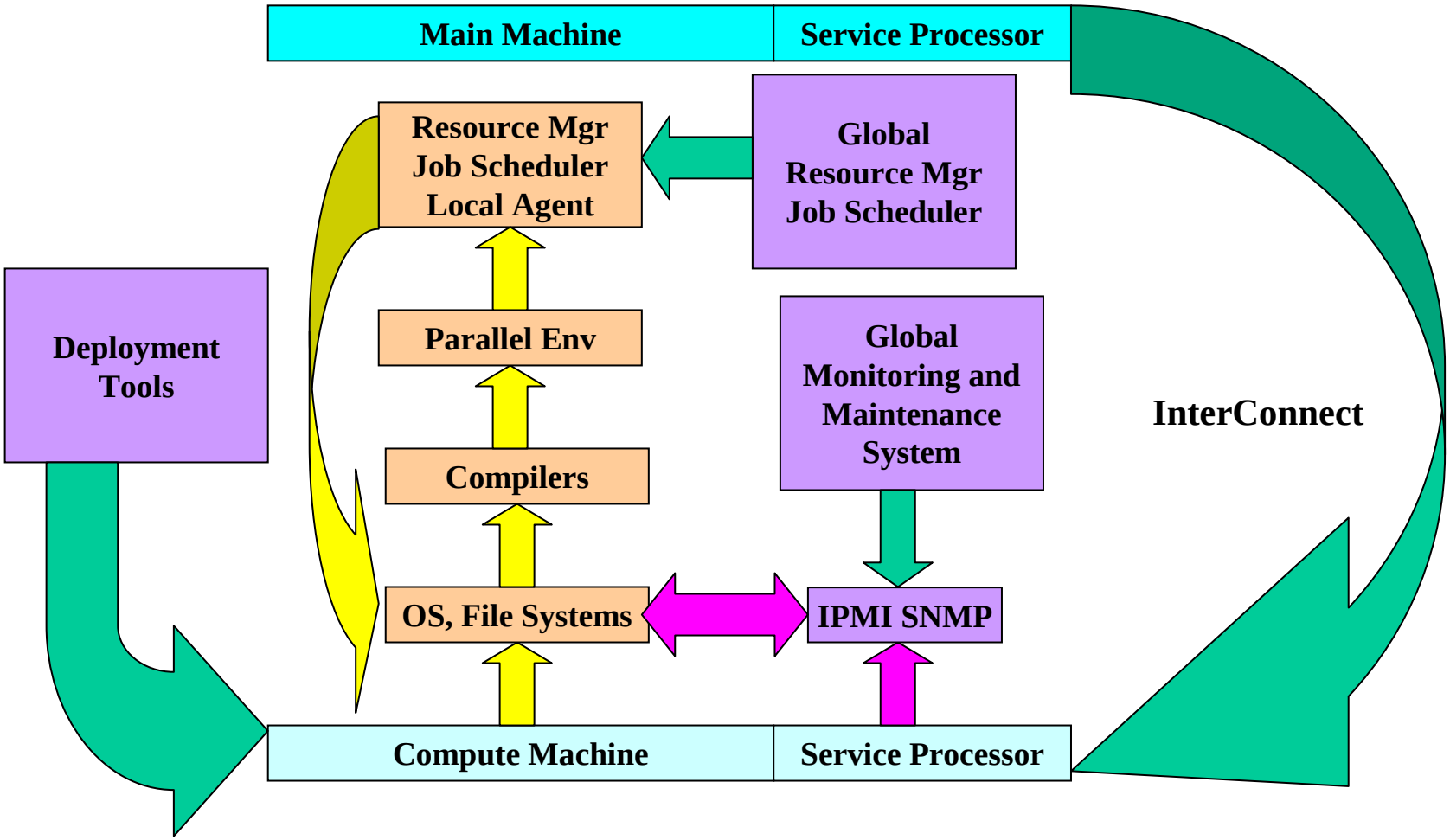


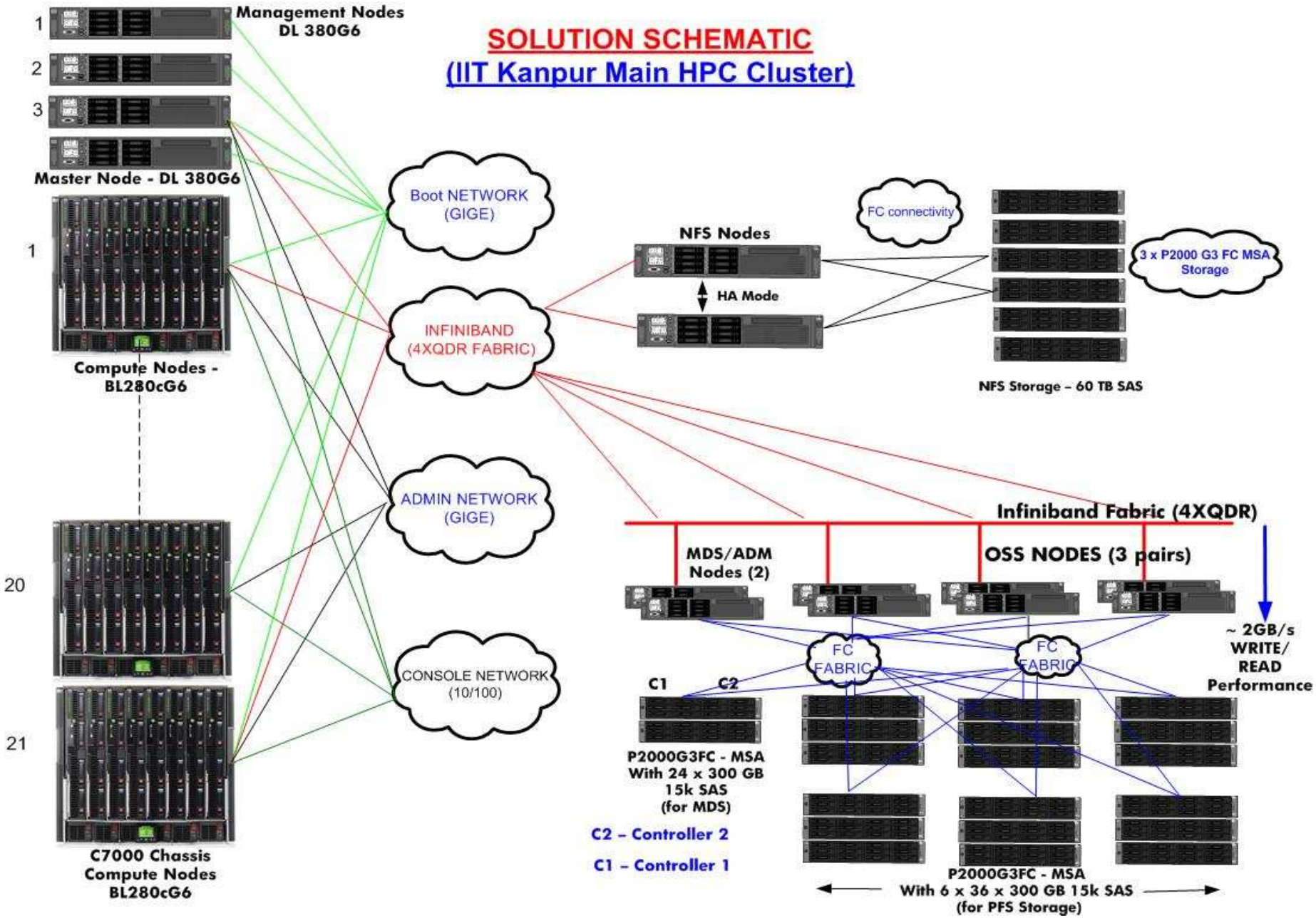


- *section of active floor module integrated in the floor –*



HPC Clusters An Architectural View




SOLUTION SCHEMATIC
(IIT Kanpur Main HPC Cluster)
1
2
3
Management Nodes
DL 380G6
Master Node - DL 380G6
1
Compute Nodes -
BL280cG6
20
21
C7000 Chassis
Compute Nodes
BL280cG6
Boot NETWORK
(GIGE)
INFINIBAND
(4XQDR FABRIC)
ADMIN NETWORK
(GIGE)
CONSOLE NETWORK
(10/100)
NFS Nodes
HA Mode
FC connectivity
3 x P2000 G3 FC MSA
Storage
NFS Storage – 60 TB SAS
Infiniband Fabric (4XQDR)
MDS/ADM
Nodes (2)
OSS NODES (3 pairs)
~ 2GB/s
WRITE/
READ
Performance
FC
FABRIC
FC
FABRIC
C1
C2
P2000G3FC - MSA
With 24 x 300 GB
15k SAS
(for MDS)
C2 – Controller 2
C1 – Controller 1
P2000G3FC - MSA
With 6 x 36 x 300 GB 15k SAS
(for PFS Storage)

More Definitions

- Shared Computing
  - refers to a collection of computers that share processing power in order to complete a specific task (car)
- Grid Computing
  - a computer network in which each computer's resources are shared with every other computer in the system. (tent)
- Cloud Computing
  - a system in which applications and storage "live" on the Web rather than on a user's computer. (hotmail) (varied OS)
- Utility Computing
  - a company offers specific services (such as data storage or increased processor power) for a metered cost to another company

Cluster Components

- Set of smp / non smp machines / nodes / blades
- Connected nodes (High speed interconnect)
- OS
- Deployment tools
- Maintenance and Monitoring Systems
- Parallel Computing Environments and Compilers
- Parallel File Systems
- Libraries, Software Packages
- **Resource Management Systems (Schedulers)**

Grid / Cluster Computing

- Grid - a collection of resources that are used for performing a task.
- Users view it as a large system with a few points of access.
- As a powerful distributed system.
- Usually treat it as a single computational resource.
- Resource manager takes up the task of submitting jobs to the grid.
- User does not bother where the job is being fired.

Grid Classification

- Cluster Grids
  - Set of hosts working together with a single point of access
  - Single Owner, Single Site, Single Organization
- Campus Grids
  - Shared heterogeneous nodes within a geographical boundary, usually an educational / corporate campus
  - Multiple Owners, Single Site, Single Organization
- Global Grids
  - Collection of campus grids that cross organizational boundaries, users can access resources far beyond what they can within their organization (Cost?)
  - Multiple Owners, Multiple Sites, Multiple Organizations.

Grid Engine (Resource Manager)

- The grid engine delivers computational power based on enterprise resource *policies* of the organization's
- Policies are usually targeted towards maximizing throughput and utilization
- The grid engine examines resources based on the policies
- It then allocates and delivers resources optimizing usage
- The grid engine software provides users with the means to submit tasks to the grid for transparent distribution of the associated workload

Grid Engine (Resource Manager)

- Users can submit batch jobs, interactive jobs, and parallel jobs to the grid.
- Supports dynamic scheduling, accounting and check pointing
- Provides tools for monitoring and controlling jobs.
- Accepts *jobs* from the outside world. Jobs are users' requests for computer resources.
- Puts jobs in a holding area until the jobs can be run.
- Sends jobs from the holding area to an execution device.
- Manages running jobs.
- Logs the record of job execution when the jobs are finished.

Grid Engine Components

- The engine has three main components
  - Hosts, Daemons, Queues
- Hosts
  - Master, Execution, Administration, Submit Host
- Daemons
  - sge_qmaster, sge_schedd, sge_execd
  - schedd decides the queue and priority
  - qmaster maintains info on hosts, queues, permissions and system loads
  - execd responsible for running and informing status to master

Grid Engine Components

- Queues
  - Container class for all jobs allowed to run on more than one host
  - Has attributes (like a parallel environment, amount of free tmp area, number of software licenses associated with it)
  - Jobs that require attributes are automatically dispatched to queues that can satisfy the required attributes
  - A cluster is a collection of hosts
  - A host has attributes including number of slots / processors specified for computation (slots need not be same as processors)
  - Hosts are associated with queues
- Grid Engine provides commands and interfaces to manipulate and configure the queues, the hosts and associated attributes

Some Queue Manipulation Commands

- qrsh
- qsh
- qtcsh
- qlogin
- qacct
- qdel
- qmod
- qsub
- qconf
- qstat

- qconf –ah –as <host>
- qconf –sul
- qconf –mconf
- qconf –mp <name_par_env>

Important output of some queue manipulation commands

qconf –sq reserved.q

- qname reserved.q
- hostlist host1 host2
- seq_no 3
- loadthreshold np_load_average=8.5
- pe_list mpichpar
- slots 2
- Userlist reservedusers
- shell /bin/csh
- prolog /tmp/myscript
- epilog /tmp/cleartmp
- s_cpu <some number>

qconf –spe mpichpar

- pe_name mpichpar
- slots 400
- user_lists NONE
- xuser_lists NONE
- start_proc_args /home/sgeadmin/mpi/startmpi.sh -catch_rsh $pe_hostfile
- stop_proc_args /home/sgeadmin/mpi/stopmpi.sh
- allocation_rule $round_robin

Seeing user groups with special priviledges

qconf –su reservedusers

- name reservedusers
- type ACL
- entries @ccce,sgeadmin,arkde,amaresh,narsimh\
  dharmkv,pssundar,vivkumar,pravir,vankates\
  bhanesh,aprataps,samrahul,mkv,janurag,santo,ramji

Submitting A Sequential Job

```
#!/bin/sh
#$ -N Sleeper
#$ -S /bin/sh
/bin/echo "Sleeping now at: `date` and `hostname`“
time=60
if [ $# -ge 1 ]; then
        time=$1
fi
sleep $time
echo Now it is: `date`
```

qsub -q <qname> <your_wrapped_job>

qsub -l tmpfree=5G -q seq.q <your_wrapped_job>

qsub -cwd -o /dev/null -e /dev/null myjob.sh

Submitting A Parallel Job

```
#!/bin/csh
#$ -N MPI_Job
#$ -pe mpich* 2-20
#$ -v MPIR_HOME=/opt/mpichdefault-1.2.6
echo "Got $NSLOTS slots.“
# enables $TMPDIR/rsh to catch rsh calls if available
set path=($TMPDIR $path)
$MPIR_HOME/bin/mpirun -np $NSLOTS –machinefile \
    $TMPDIR/machines -nolocal /somepath/a.out
```

qsub -pe mpichpar 10 -q par.q <your_wrapped_job>

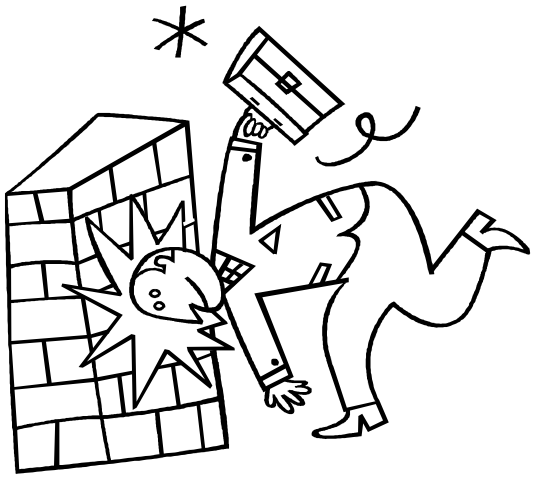
Further Study - Globus Tool Kit

- The Globus Toolkit. www.globus.org
- Community-based, open-architecture, open-source set of services and software libraries that support Grids and Grid applications.
- Issues of security, information discovery, resource management, data management, communication, fault detection, and portability.
- Globus Toolkit mechanisms are in use at hundreds of sites and by dozens of major Grid projects worldwide.
- Grid Security Infrastructure (GSI), supports single sign on, delegation, and credential mapping.

Concluding Thoughts

- HPC components require understanding several concepts
- CPU, MEMORY, DISKS, FILE SYSTEMS tec
- Interconnect of these make a NOW and other HPC systems
- Also when scale becomes large issues of management crop up
- Leveraging and management technologies
- Several Perspectives – HW, User, System Administrator
- Several Classification of Grids
  - Cluster, Campus, and Global Grids, cloud, utility computing
  - Future seems to be in term of global grids and compute farms
  - Issues of secure exchange and use of clusters will come up
- Open Grid Service Architecture - GLOBUS

Thank You



If the right number of people get together and work with common sense, the right tools and manage things well then reasonably large systems can be realized in reasonable time