# Introduction to high performance computing


World's fastest supercomputer: Titan Cray XK7 – Oak Ridge National Laboratory – 20 peta(10^15) FLOPS (floating point operations per second)

Image source:http://www.olcf.ornl.gov/titan
top500.org

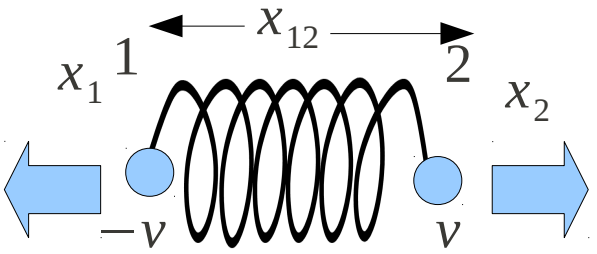
# Introduction

- Difference between normal computing and high performance computing (HPC)
- The hardware architecture overview
- Few problems of interest in HPC
- The software requirement
- Major super computers according to the speed are listed in websites www.top500.org
- Where to look for information

# A simple example using Newton's equation of motion

Consider a system that have two particles and a single spring connecting them – the equation of motion is

$$m\frac{d^2x_1}{dt^2} = -\frac{dV(x_{12})}{dx_1} \qquad m\frac{d^2x_2}{dt^2} = -\frac{dV(x_{12})}{dx_2}$$




Or we may write equation of motion as

$$m\frac{d^2x_1}{dt^2} = -F_{12} \qquad m\frac{d^2x_2}{dt^2} = F_{12}$$

The forces are now equal and opposite, by third law of motion

In general to solve this set of equation the following steps may be used

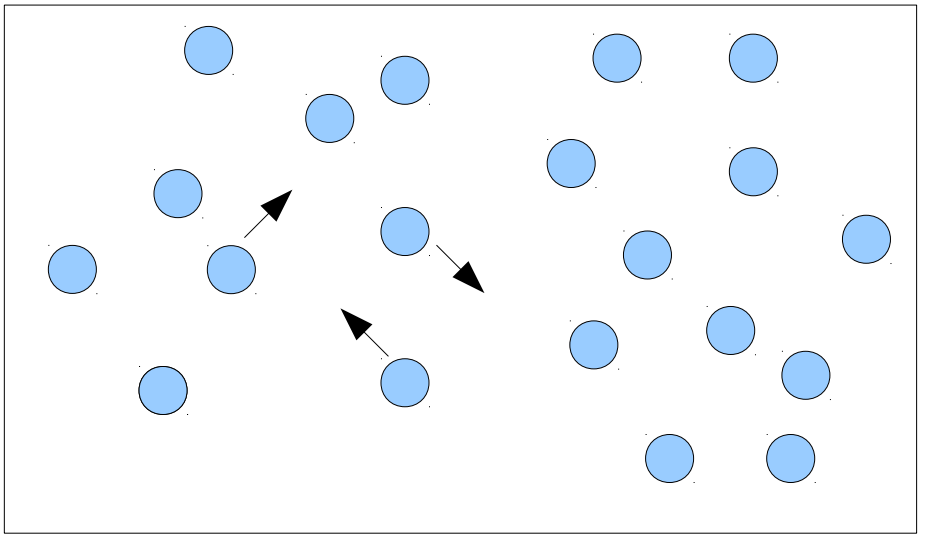
1) A model or potential that describe inter particle interaction

2) Calculation of accurate forces

3) An algorithm to integrate equation of motion

# Now consider the problem of collection of molecule that obey Newton's equation of motion

example:n particles of a liquid



Newton's equation for N body

$$m\frac{d^2x_1}{dt^2}=F_{12}+F_{13}+....+F_{1n}=F_1$$

$$m\frac{d^2x_2}{dt^2}=F_{21}+F_{23}+....+F_{2n}=F_2$$

$$m\frac{d^2x_n}{dt^2}=F_{n1}+F_{n2}+....+F_{n-1\ n}=F_n$$

For simulation of a real system it is required to simulate at least Avogadro number of molecules/atom $n\simeq 10^{23}$

Next we express the differential equation in the discrete form, express this in terms of ultra short time steps – there are many equivalent methods; one popular method is Verlet algorithm

$$x_i(t+\Delta t)=x_i(t)+\Delta t\, v_i(t)+\frac{1}{2}(\Delta t)^2\frac{F_i}{m_i}$$

$$v_i(t+\Delta t)=v_i(t)+\frac{1}{2}(\Delta t)\frac{(F_i(t)+F(t+\Delta t))}{m_i}$$

See: http://en.wikipedia.org/wiki/Verlet_integration

The points we are going to address is

We need properties of liquids of considerable time scale say a millisecond to make realistic properties $t \simeq 10^{-3}$

Considering very low mass of a single molecule the integration take place in femtosecond $\Delta t \simeq 10^{-15}$

If the total time is divided into number of steps $m \Delta t = t, \qquad m = 10^{12}$

The force calculation requires operations of the order $n \frac{(n-1)}{2} \simeq n^2$

For Avogadro number of particles the number of calculations is $\simeq (10^{23})^2$

This force calculations has to be repeated for m steps $\simeq (10^{23})^2 \times 10^{12} = 10^{58}$

The required computational capacity is beyond the reach of any known super computers at present $10^{58} / (20 \times 10^{15}) \simeq 10^{41}\, seconds \simeq 10^{34}\, years$

This problem can be approached with following approximations

First reduce the number of particles to a smaller size – say atoms $n \simeq 10^5$

Now change time required to upto nano-seconds so m reduced to $m \Delta t = t, \qquad m = 10^6$

Now the calculations can be completed by $\simeq \frac{(10^5)^2 \times 10^5}{2 \times 10^{15}} = 0.5\, seconds$

**Now for simulation of the manybody system use the algorithm given below.**

1) Define variables to store, position velocity, new force and old force, time step of integration etc.

2) Initialize the positions & velocities to start with meaningful values

Set mass, distance between the particles, the initial velocities, and the integration step

3) Now set up a loop (for loop) to perform the integration,

Inside this loop do a repeat on following steps

i) Calculate inter particle (between) forces

ii) Perform move (reassign position and velocity) using velocity Verlet algorithm

iii) Repeat the procedure until desired time/number $(m)$ of steps

$$t = m\Delta t$$

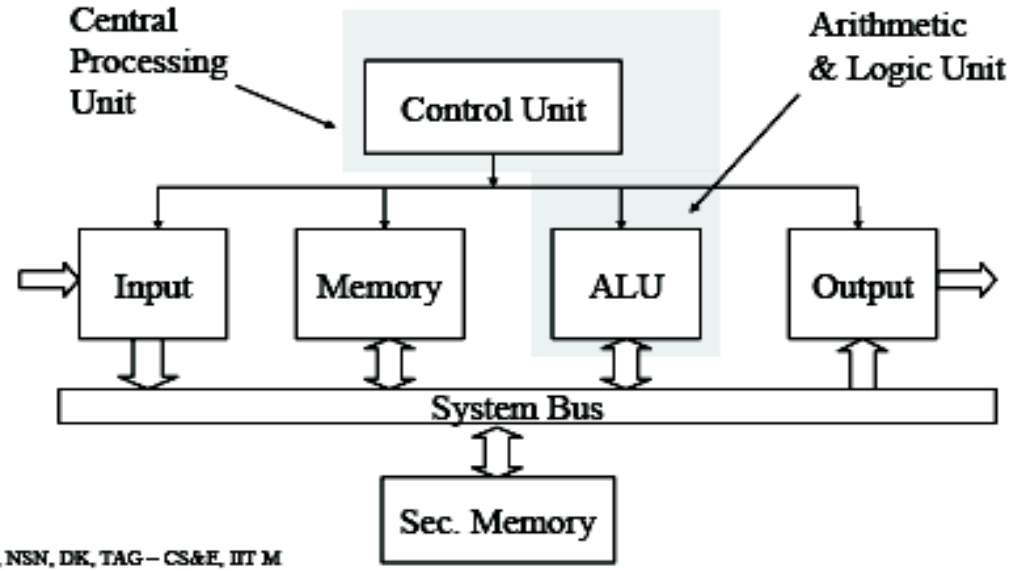
$$x_i(t+\Delta t) = x_i(t) + \Delta t\, v_i(t) + \frac{1}{2}(\Delta t)^2 \frac{F_i}{m_i}$$

$$v_i(t+\Delta t) = v_i(t) + \frac{1}{2}(\Delta t)\frac{(F_i(t) + F(t+\Delta t))}{m}$$

# Basic building blocks of a computing cluster - nodes

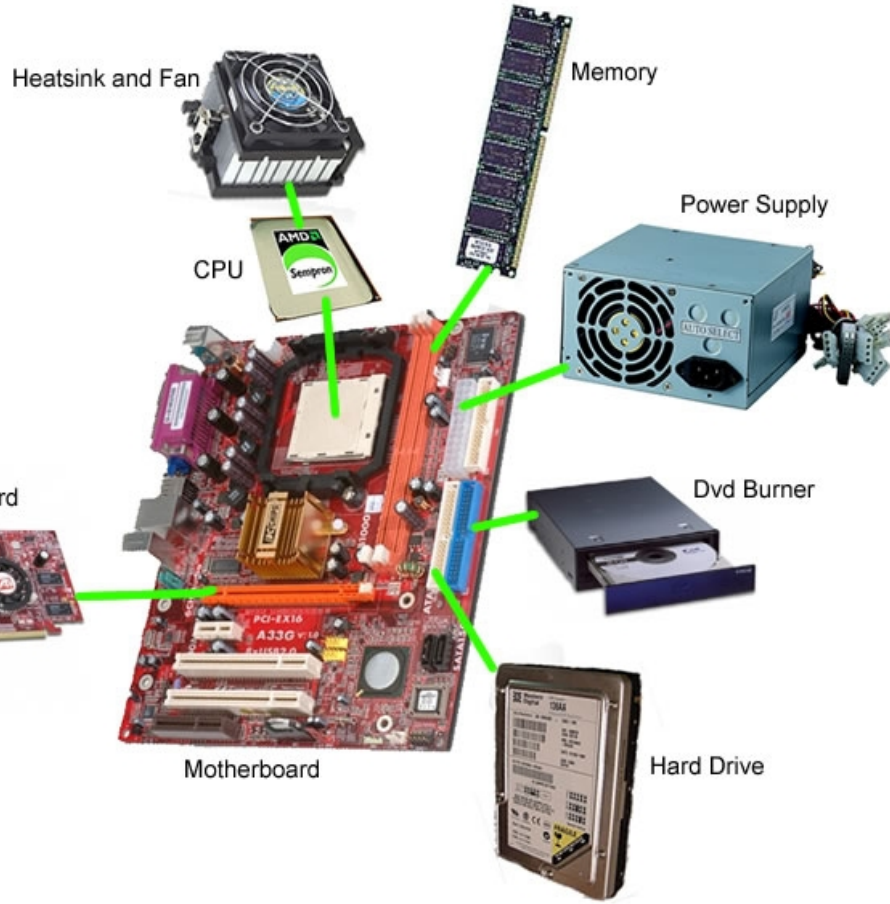
## Building Blocks (Computer Architecture)

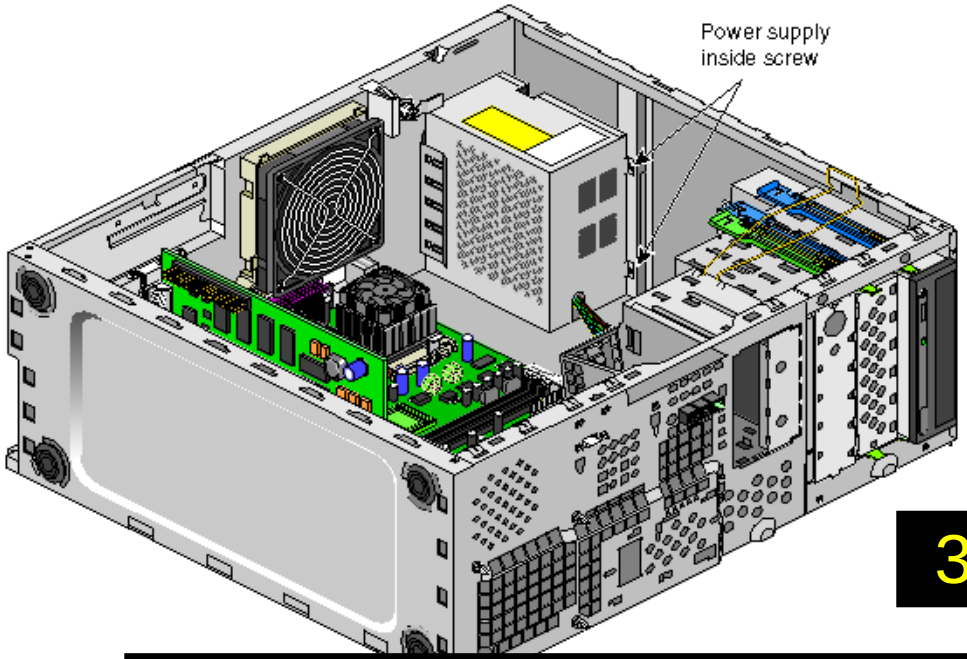
1




PSK, NSN, DK, TAG – CS&E, IIT M

2



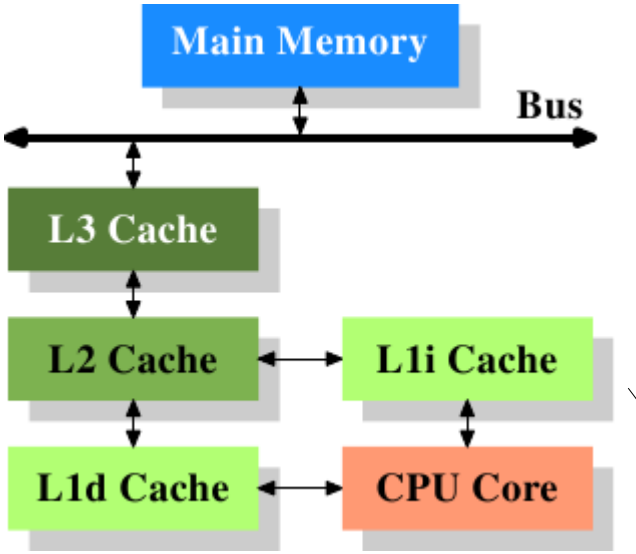



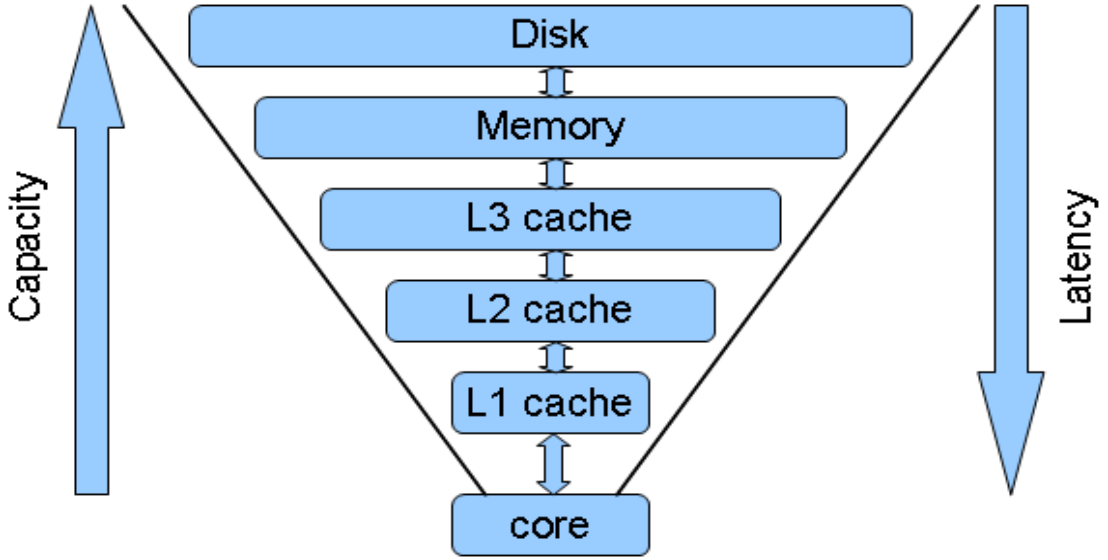

3

http://zweb.cs.utexas.edu/users/cs329e/chsu1/project/component.html

Speeds and Speed breakers in a computer

Maximum speed at which a computer operates is the clock speed of the computer




instructions level 1 cache

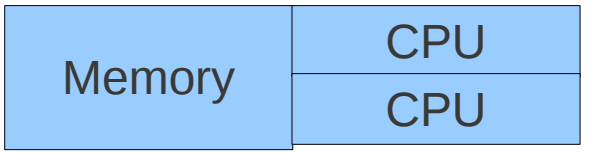
data level 1 cache

Eg: Intel® Core™ i7-3960X Processor Extreme Edition

Clock speed 3.3 Ghz Memory speed 1600 MHz  Cache – 15 Mb – number of cores – 6

Slowest device is the harddrive


Schematic representation of basic model of computer with single core




Schematic representation of basic model of computer with two core

A fastest running algorithm is the one whose binary has most frequently executed part reside on L1 cache of the processor.

See:http://www.linuxshowcase.org/2000/2000papers/papers/sears/sears_html/

But modern processors have more than one core with in a single processor. If the algorithm has parts that are independent then independent parts can be processed simultaneously

In the algorithm discussed for simulating Newtons equation of motion the independent steps are the calculation of forces

In a multi-core processor the simplest to way to improve the processing speed is to allow all cores to compute forces in parallel as this is an N^2 operation.

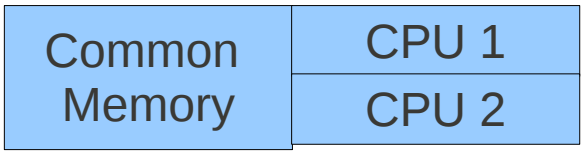
This parallelization can be achieved in single computer using **openMP**

http://openmp.org/wp/ http://en.wikipedia.org/wiki/OpenMP

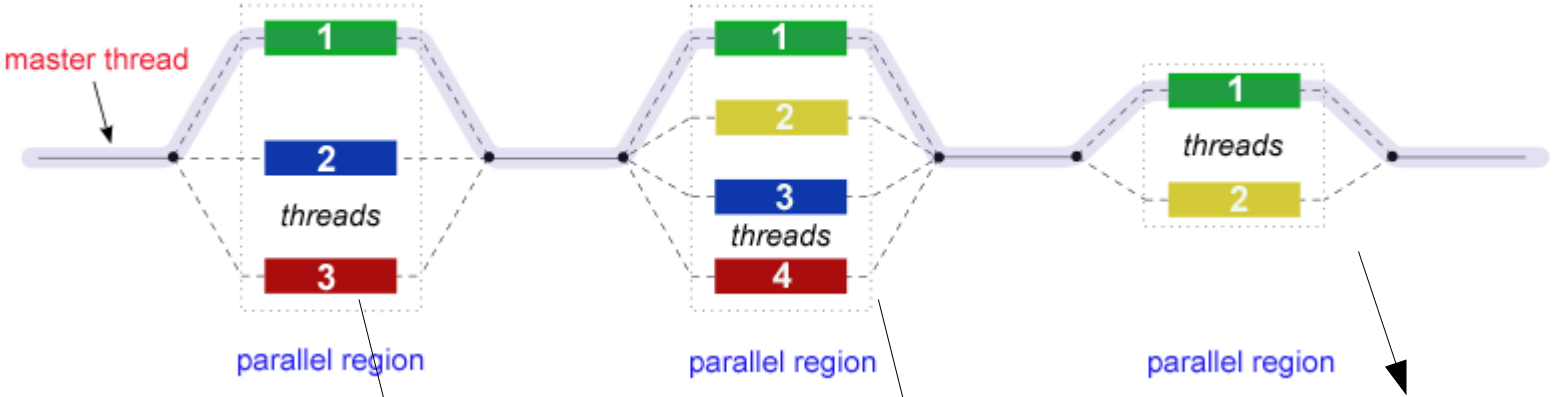
What is OpenMP?

In is a shared memory application programming interface that supports programing in Fortran, C, C++, it is implemented in most of the processor architectures and operating systems such as Windows, Linux, various UNIX version, Mac OSX

Architecture for which OpenMP is designed




This implemented with in any programing language using compiler directives, library routines, and environment variables




Eg: updating positions
And velocities

Eg: Calculation of forces

Eg: Initialization of velocities

This program can be compiled in gcc

#gcc --openmp <programname.c>

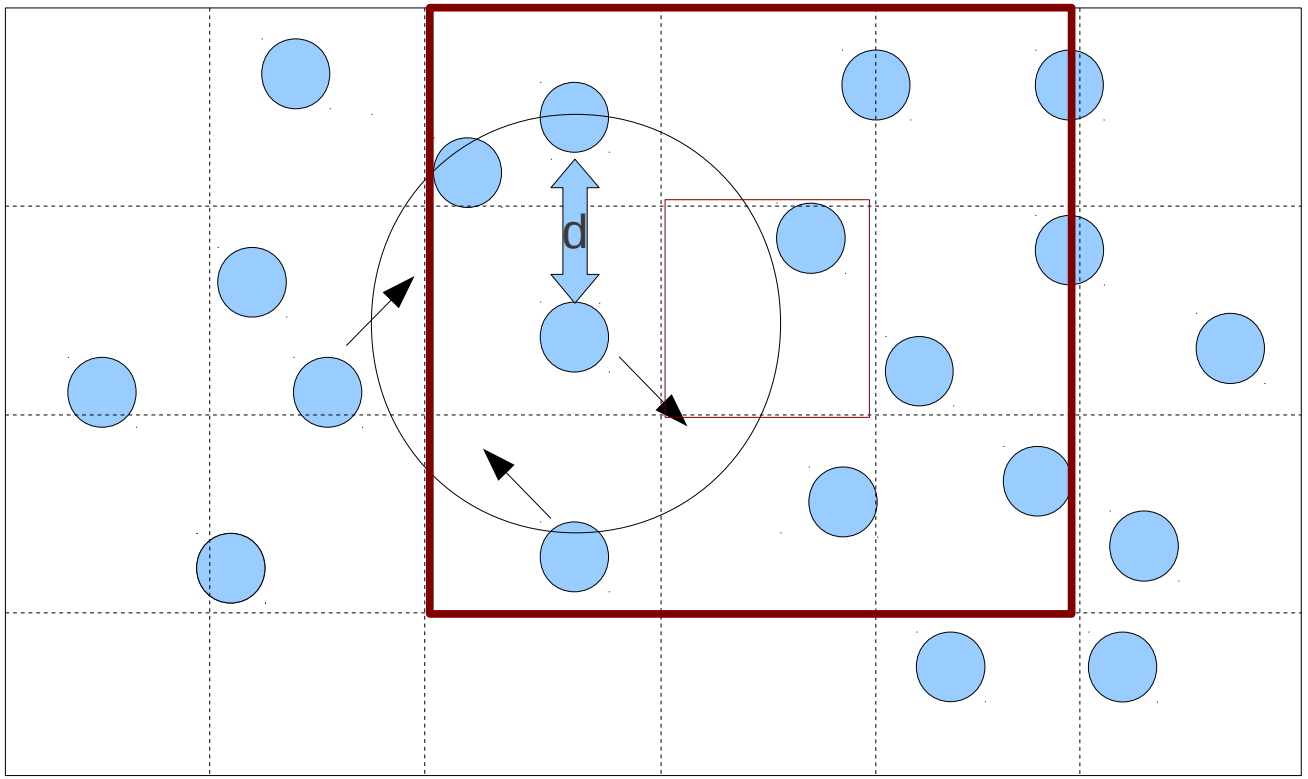
#./a.out

Hello, world.
Hello, world.
Hello, world.
Hello, world.

```
#include <stdio.h>
#include <omp.h>

int main(void)
{
  #pragma omp parallel
    printf("Hello, world.\n");
  return 0;
}
```

Compiler directive




## Speed-up algorithm:  using grids

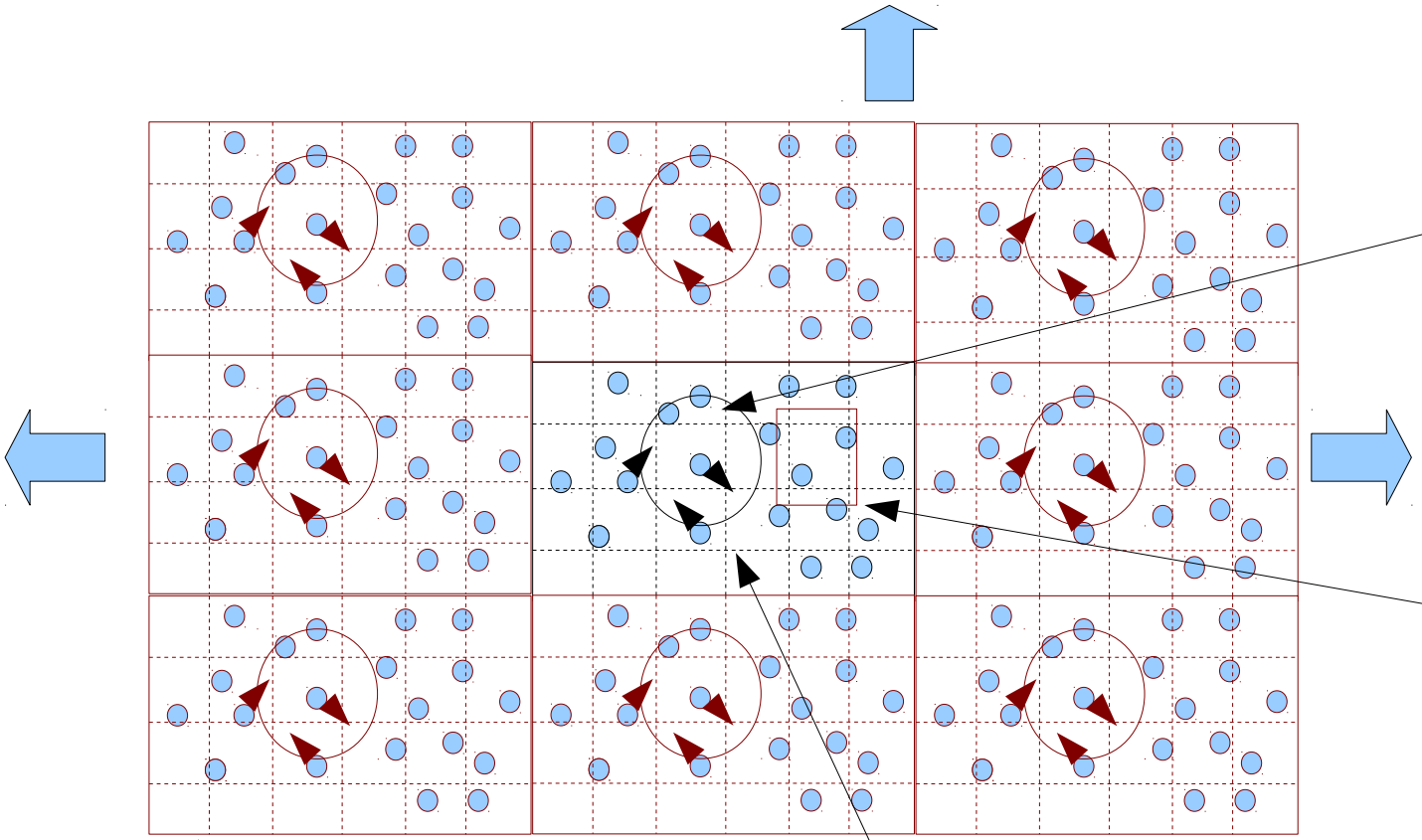
Define a circle  of interaction for a particle and cut-off the interaction potential beyond this point the errors in the calculation  is less than the numerical precision limit of the computers

Now divide the entire system into cells such that only neighboring cells interact

Assume that there are l particles in such cluster that have a central box and its neighbors -

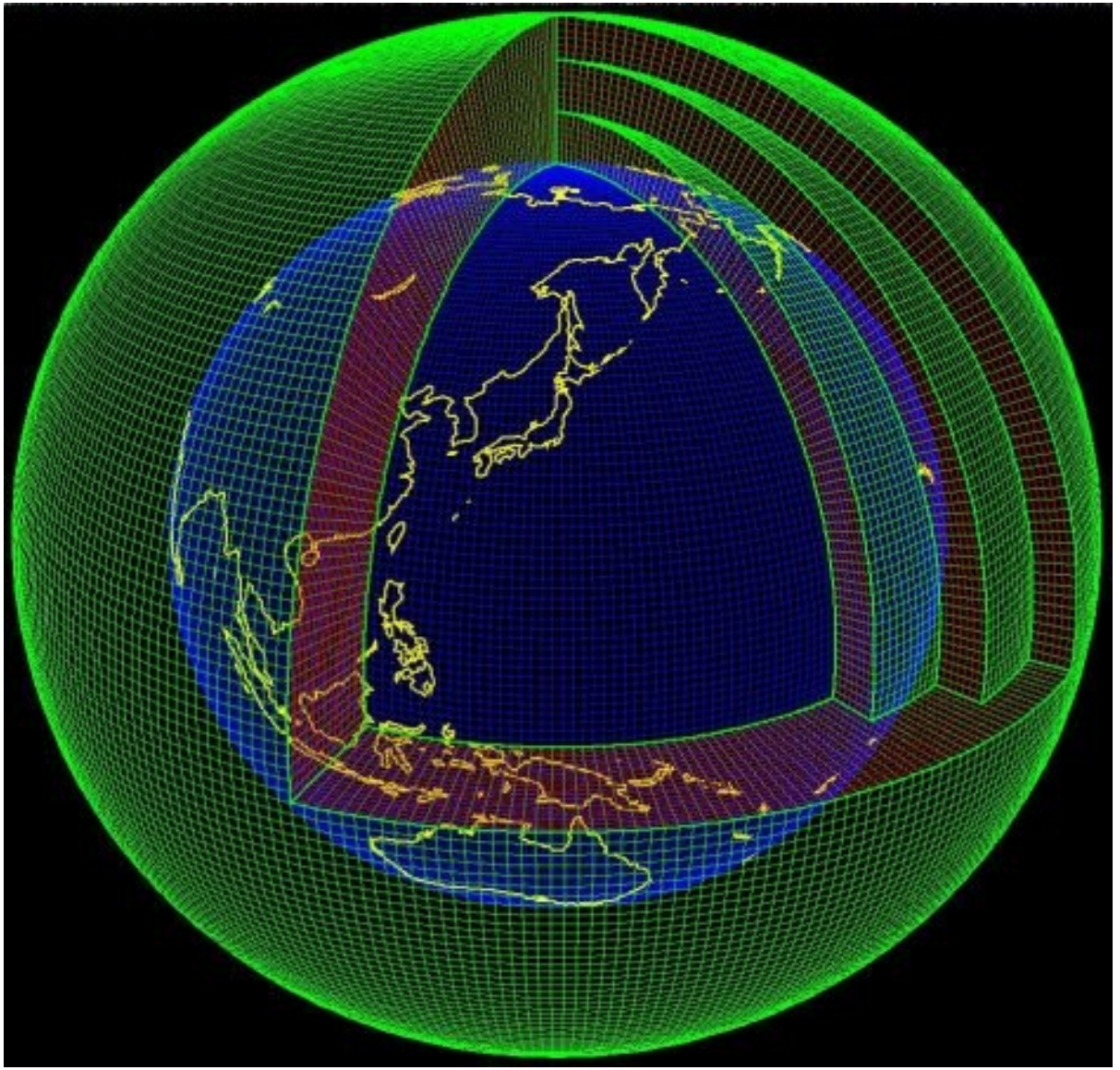
Now the problem is reduced to complexity of Nxl^2 – where N is the number of cells into which system is divided

We need more approximations to solve this problem with our accessible resources. Now the problem with reduced number of particles can represent infinite system by use of periodic boundary conditions




For simulation of reasonable large system we have to go beyond shared memory model -where multiple computers participate in solving a problem

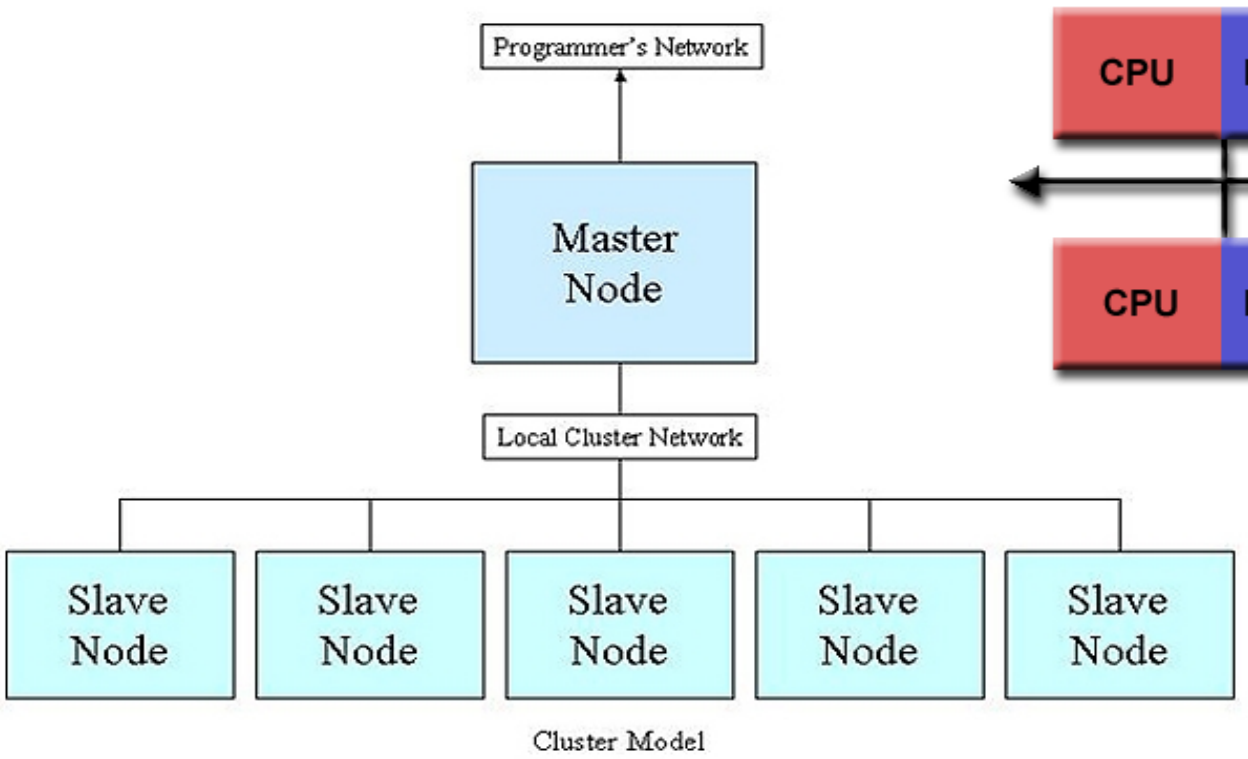
# Weather modeling – another area where problem is divided into cells



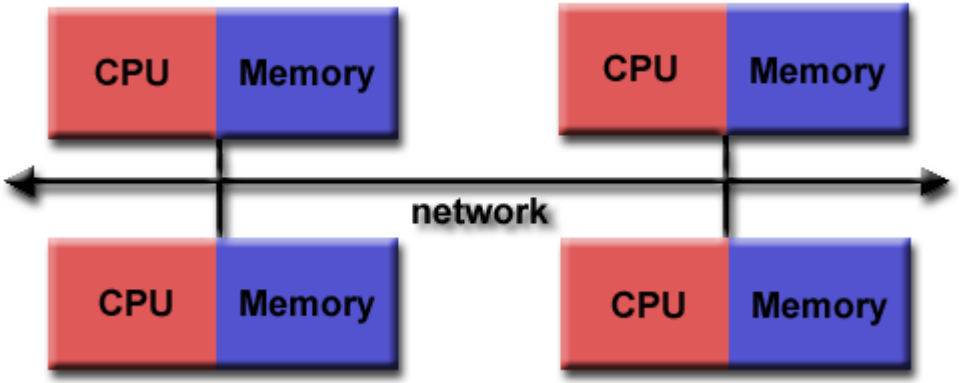


http://www.jma.go.jp/jma/indexe.html

# Distributed memory model: A diagrammatic representation of a computing cluster



Cluster Model




The inter-connect between them are by ethernet, fiber optic network interface etc.

In this system memory of the one node is accessible to operating system loaded to that particular node.

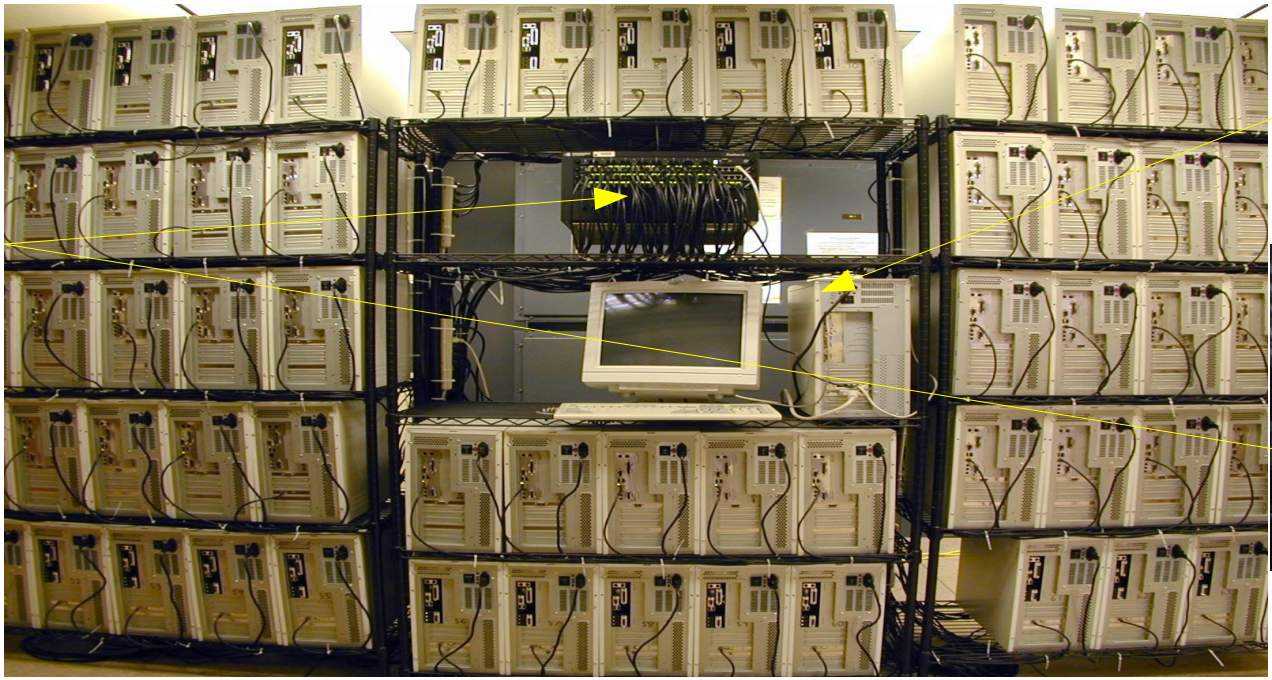
There specific hardware and software requirements that will allow transfer of data and the parallel run programs on multiple nodes

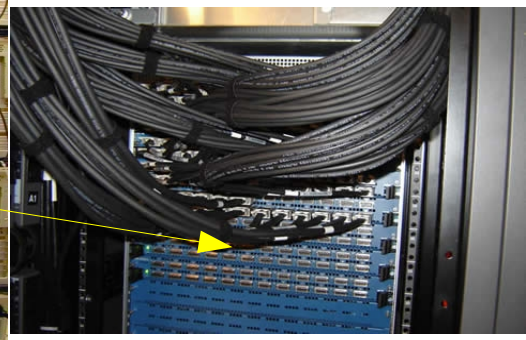
# Beowulf cluster is simplest of such cluster made of desktop machines

workstations/PC's

Master node

Network Switches





A cluster made of Personal computers - NASA 128-processor Beowulf cluster: A cluster built from 64 ordinary PC's.

There is no general  definition for the term Beowulf cluster in general:-

A Beowulf cluster consists of many workstations as client nodes, that have single node as head/master node, and all nodes are connected via network – the user interface is through the head node. Users submit jobs to client nodes from head/master node.

The common operating system found in such clusters are various Linux distributions



http://en.wikipedia.org/wiki/Beowulf_cluster

http://www.cse.mtu.edu/cseri.html

# Operating system on a cluster

- Single/Multi processor operating system

  Linux/Unix/Windows/MacOS/FreeBSD etc. are popular examples
- The jobs and load on each node is balanced by stand alone the operating system. But the job are supervised by instruction from the master node

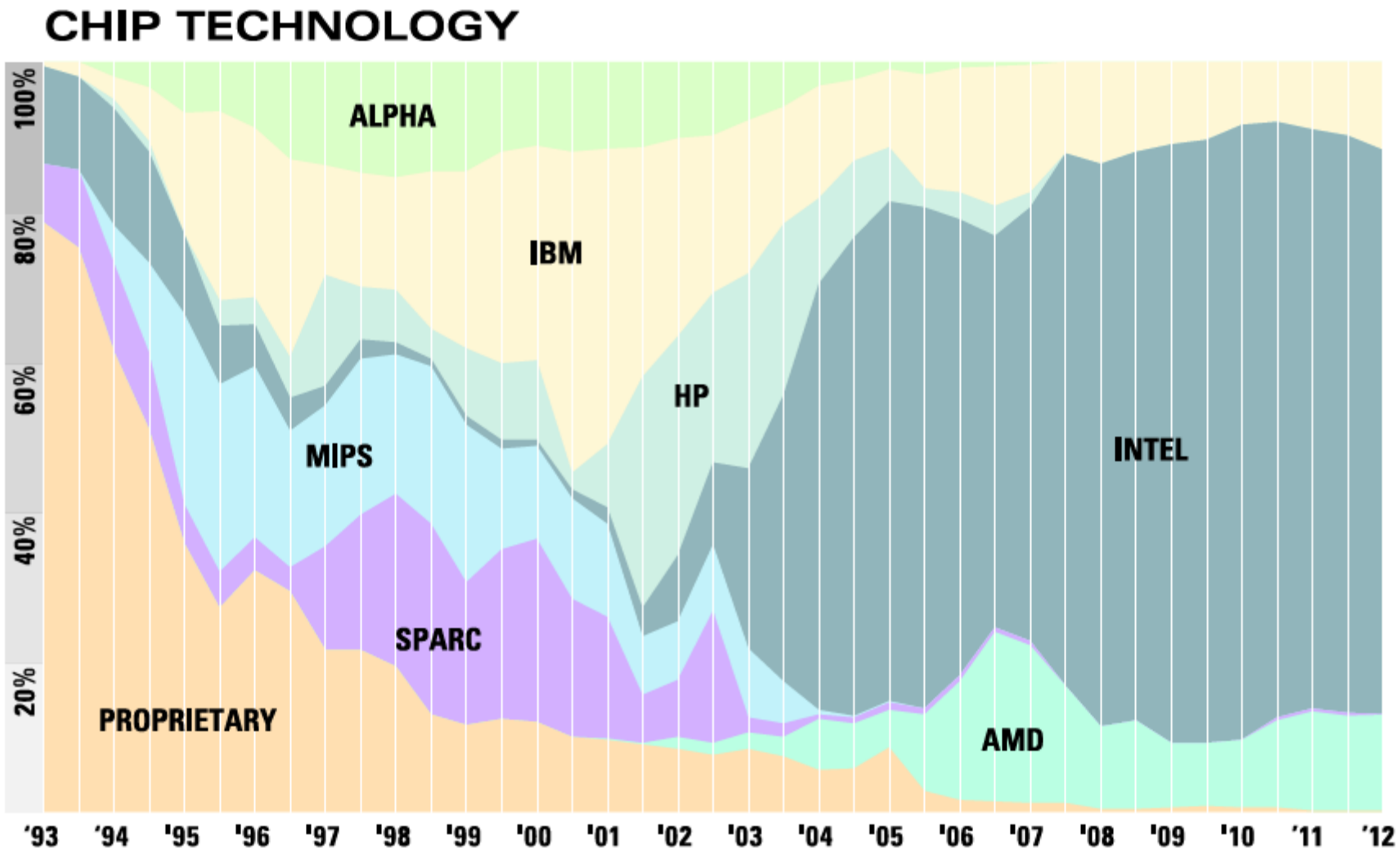
  Examples of job schedulers are OpenPBS/Torque(NASA), PBS Pro (Altire), SunGridEngine(SUN/Oracle), LoadLeveler (IBM) etc.
- The nodes contain softwares that run on this system – they are either locally installed – or on a shared partition from the master node

  Example: NFS (Network File system )
- It is desirable that all user information is common in all nodes

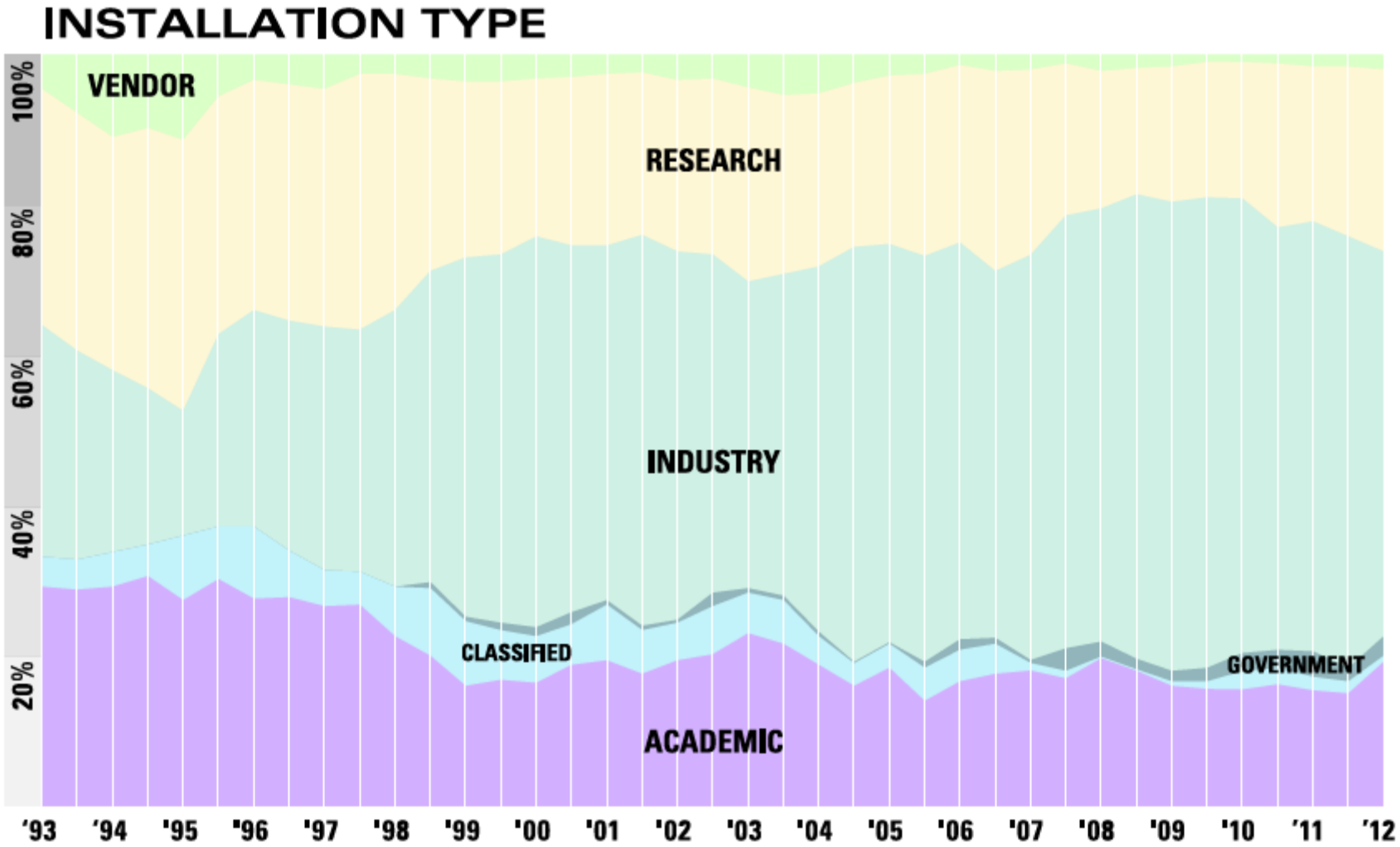
  Example NIS (Network Information system in Linux/Unix/MacOS)

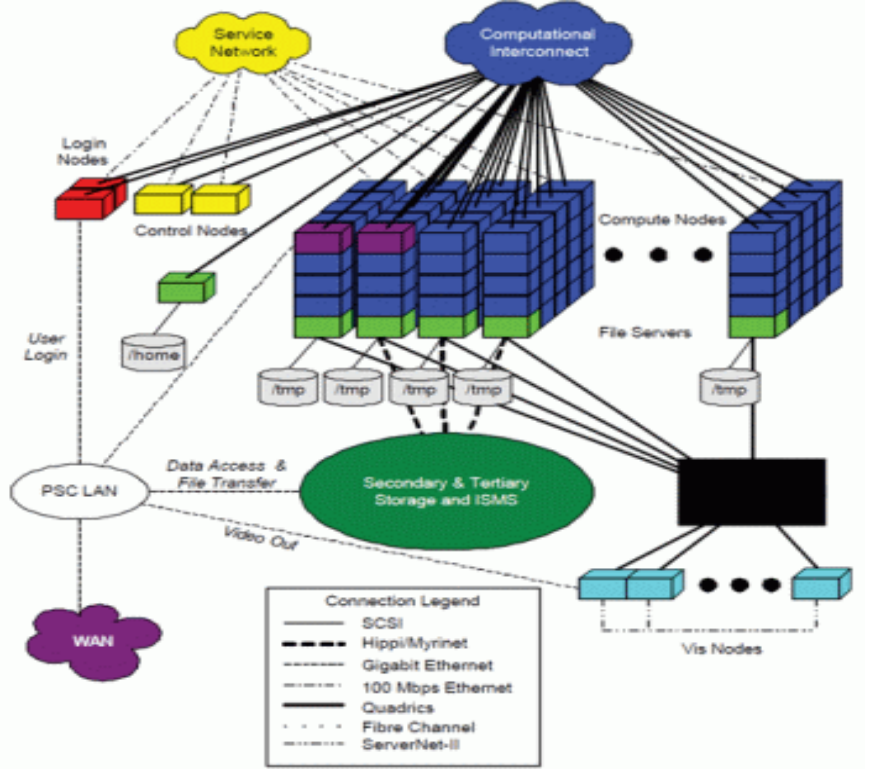
- Most popular operating system that loaded and listed in top500.org world's fastest super computers are variants of Linux

# Top 500 by Microprocessor



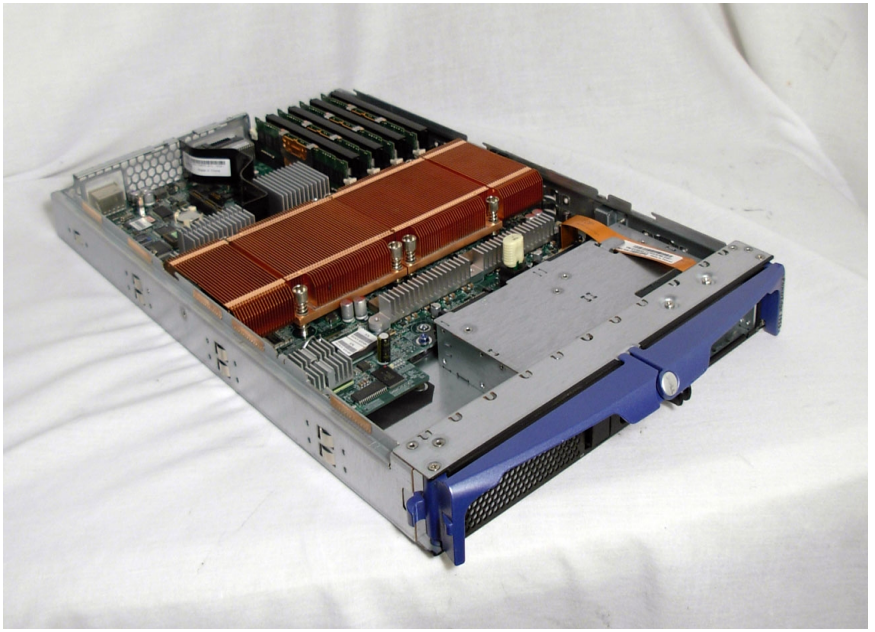
# Top 500 by Installation Type






- A more complex model of a computing cluster is given here
- A specialized high performance computing cluster is identical to a simple Beowulf model except there are more optimized hardware for HPC
- The interconnects involves fiber optic channel that can transfer data at the rate of hundreds of gigabit per second
- High speed data transfer to harddrive is achieved by implementation of parallel file system http://en.wikipedia.org/wiki/Parallel_file_system

# Hardware organization on an HPC cluster

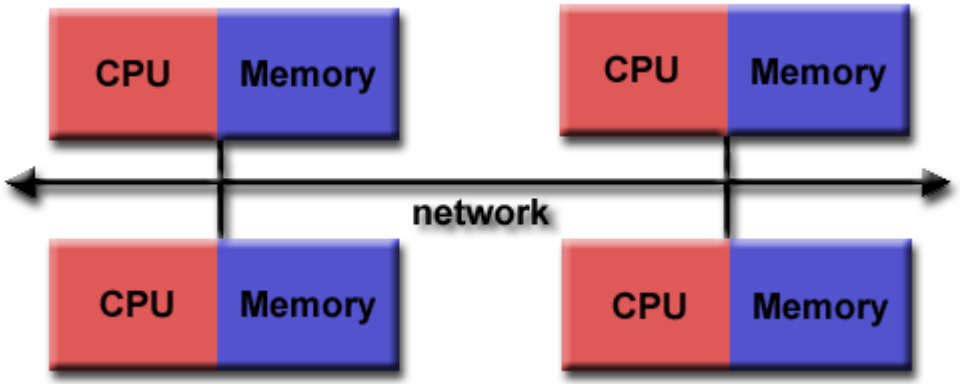


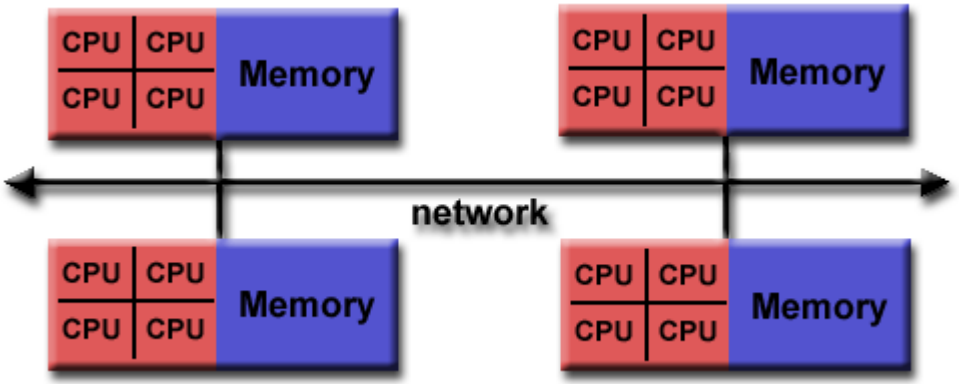
Single blade server

Assembly of multiple
Blade servers

Mounting of assembly of
Blade servers

# In all the cluster shown the parallelism can be achieved use of message passing libraries (MPI)




MPI is originally designed for running on multiple single processor machines

Modern standards MPI support multinode multi processor/core machines

MPI is set of library specifications for message passing between different parts of a parallel program.

MPI standards are implemented by different vendors – there are commercial as well as free implementations, popular ones include MPICH2, OpenMPI etc.



https://computing.llnl.gov/tutorials/mpi/#What

# Bottlenecks of parallel computing

- All algorithms cannot be made parallel due to the inter dependency of steps in the algorithms
- The parallelization cannot be very general as best options of parallelization depends also on the hardware architecture
- Parallelization require human intervention, some of the algorithms has to completely redesigned to run in parallel.
- Most of the users of parallel softwares are unaware of the bottlenecks of computing – some algorithms on increase of threads start loosing its efficiency