The Abdus Salam International Centre for Theoretical Physics

United Nations Educational, Scientific and Cultural Organization

IAEA International Atomic Energy Agency

2068-13

Advanced School in High Performance and GRID Computing - Concepts and Applications

30 November - 11 December, 2009

Parallel Computing with Linux; the Cluster Approach

S. Cozzini

CNR-INFM Democritos
Trieste
Italy

Advanced School in
High Performance
and GRID Computing:
concepts and applications

Linux cluster approach to parallel computing

Stefano Cozzini

Democrito and SISSA/eLAB – Trieste

Agenda

- Parallel Performance:
  - Does it really deserve to build clusters ?
- Linux clusters once again
- Hardware bricks for Linux Clusters
- Software stack on Linux Clusters
- How/where/when to choose a Linux Cluster ?

Parallel performance: Speedup

- The *speedup* of a parallel application is

$$Speedup(p) = Time(1)/Time(p)$$

- Where
  - Time(1) = execution time for a single processor
  - Time(p) = execution time using p parallel processors
- If Speedup(p) = p we have *perfect speedup* (also called *linear scaling*)

Absolute speedup

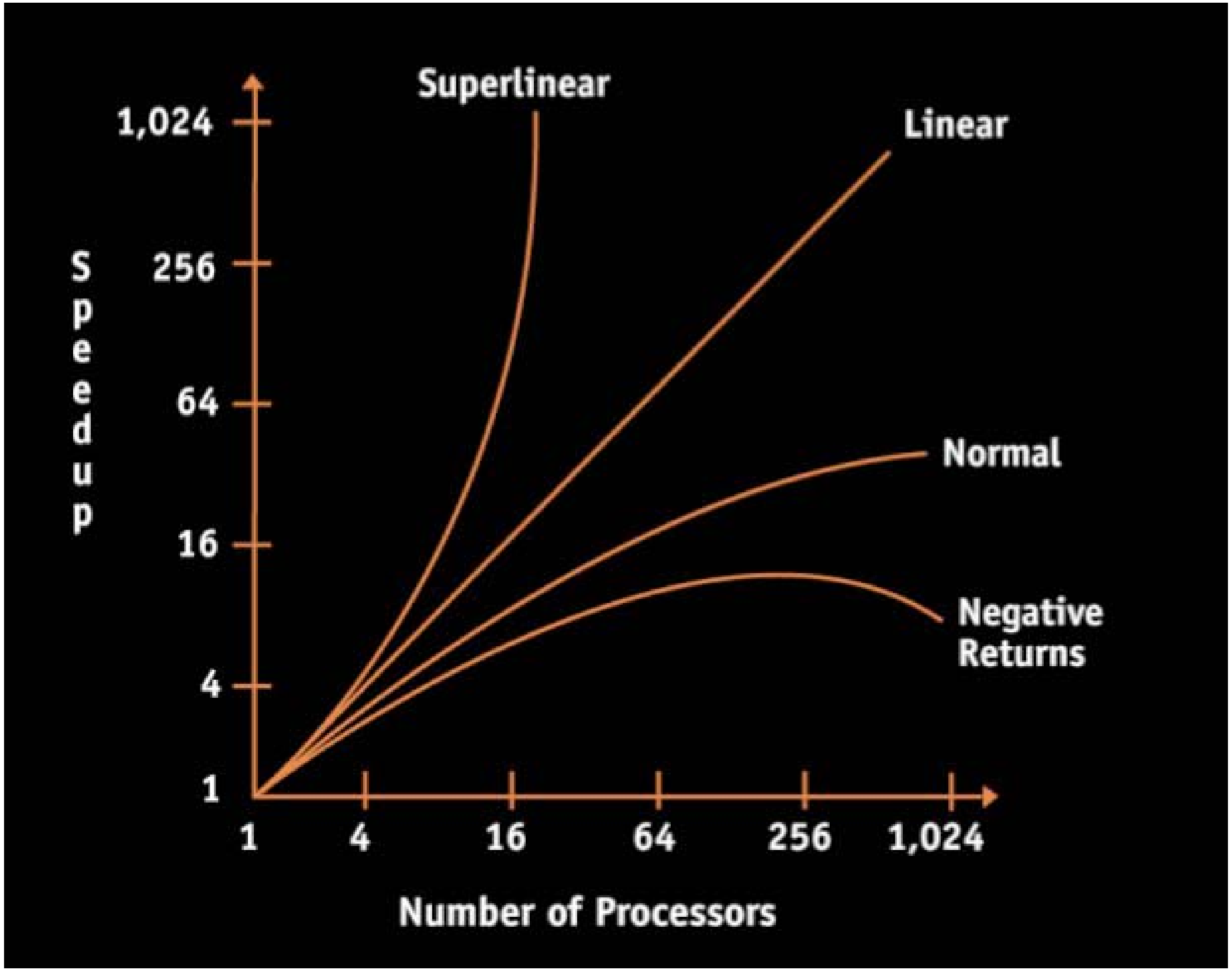
- Speedup compares an application with itself on one and on p processors
- more useful to compare:

  - The execution time of the best serial application on 1 processor

Versus

  - The execution time of best parallel algorithm on p processors

Speed-up



Superlinear
Linear
Normal
Negative
Returns
Speedup
1,024
256
64
16
4
1
1
4
16
64
256
1,024
Number of Processors

Superlinear Speedup

Question: can we find "*superlinear*" speedup, that is

$$\text{Speedup}(p) > p \quad ?$$

- Choosing a bad "baseline" for T(1)
  - Old serial code has not been updated with optimizations
- Shrinking the problem size per processor
  - May allow it to fit in small fast memory (cache)
  - Total time decreased because memory optimization tricks can be played.

Question

- Algorithm A and algorithm B solve in parallel the same problem
- We know that on 64 core:
  - Program A gets a speedup of 50
  - Program B gets a speedup of 4
- Which one do you choose ?

  A. program A

  B. program B

  C. None of the above

Answer

- None of the above as for as I do not know the

  overall execution time

  of them !

- What if A is sequentially 1000 time slower than B ?
- Always use the best sequential algorithm for computing speedup (absolute speedup)
- And the best compiler as well !

Limit to speedup

- All parallel programs contain:
  - Parallel sections
  - Serial sections
- Serial sections limits the speed-up:
  - Lack of perfect parallelism in the application or algorithm
  - Imperfect load balancing (some processors have more work)
  - Cost of communication
  - Cost of contention for resources, e.g., memory bus, I/O
  - Synchronization time
- Understanding why an application is not scaling linearly will help finding ways improving the applications performance on parallel computers.

Amdahl's Law

- Let s be the fraction of work done serially,
- So (1-s) [=P] is fraction done in parallel
- What is the maximum speedup for N processors?

$$speedup = \frac{1}{(1-P) + \frac{P}{N}} \Rightarrow \lim_{N \to \infty} \frac{1}{1-P}$$

Even if the parallel part speeds up perfectly, we may be limited by the sequential portion of code.

Amdahl's law(2)

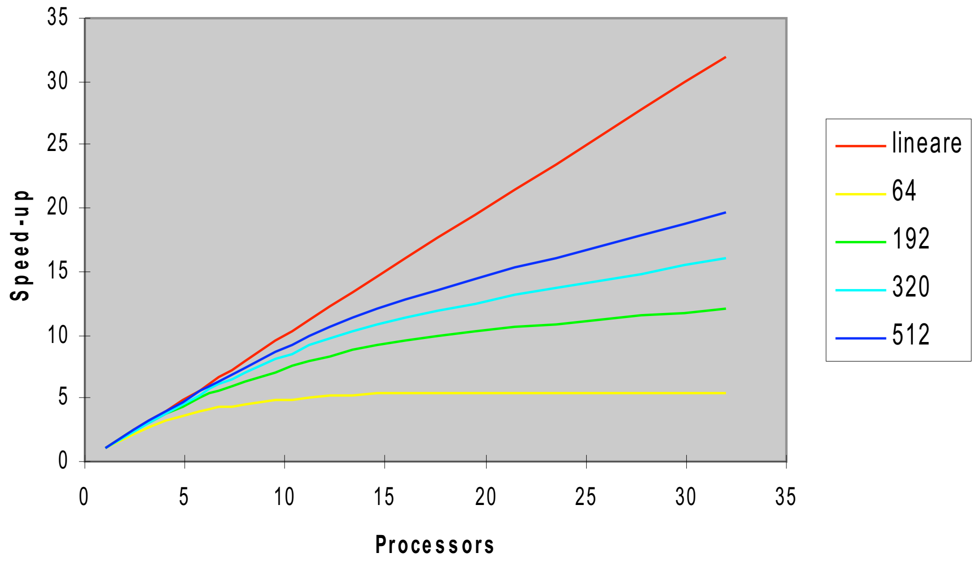
- Which fraction of serial code (parallel overhead) is it allowed ?

>	2	4	8	32	64	256	512	1024
5%	1.91	3.48	5.93	12.55	15.42	18.62	19.28	19.63
2%	1.94	3.67	6.61	16.58	22.15	29.60	31.35	32.31
1%	1.99	3.88	7.48	24.43	39.29	72.11	83.80	91.18

What about Scalability ???

Problem scaling..

- Amdahl’s Law is relevant only if serial fraction is independent of problem size, which is rarely true
- Fortunately “The proportion of the computations that are sequential (non parallel) normally decreases as the problem size increases ” (a.k.a. Gustafon’s Law)



Take home message..

- We can build parallel machine if we have large enough problem to solve on the top if it..

Let us now talk about linux clusters..

Hg1 cluster

What is a Beowulf Clusters ?

- Subject: Re: [Beowulf] about concept of beowulf clusters
  Date: Thu, 24 Feb 2005 19:41:22 -0500 (EST)
  From: Donald Becker <becker@scyld.com>
- The Beowulf definition is commodity machines connected by a private cluster network running an open source software infrastructure for scalable performance computing
- this means:

commodity machines: exclude custom built hardware e.g. an SP6 cluster is not a Beowulf cluster

connected by a cluster network: These machines are dedicated to being a cluster, at least temporarily. This excludes Netwok of workstations and others .

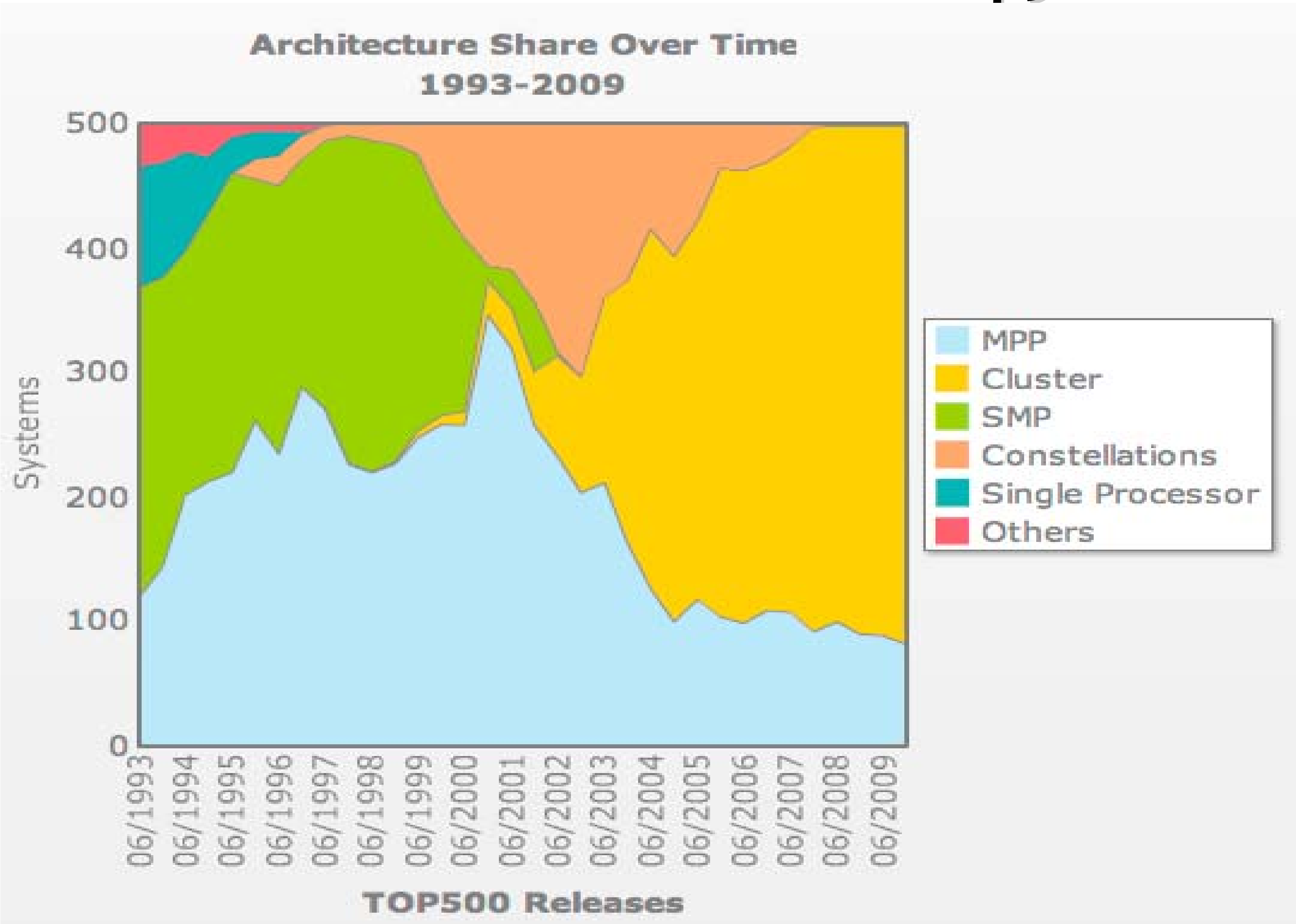
running an open source infrastructure The core elements of the system are open source and verifiable

for scalable performance computing The goal is to scale up performance over many dimension. Ideally a cluster incrementally scales both up and down, rather than being a fixed size.

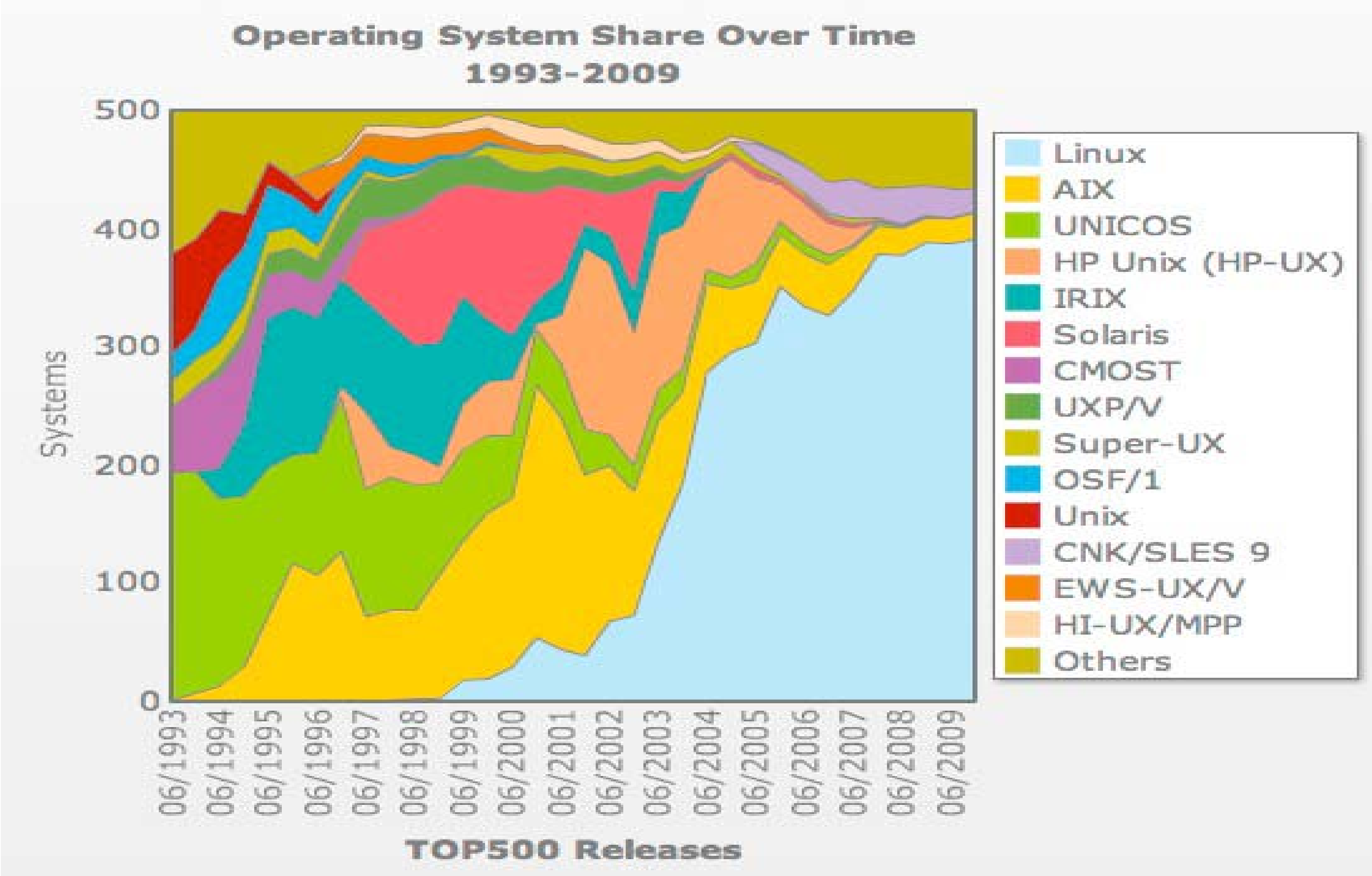
The Linux Cluster revolution in HPC

- The adoption of clusters, virtually exploded since the introduction of the first Beowulf cluster in 1994.
- The attraction lies
  - in the (potentially) low cost of both hardware and software
  - the control that builders/users have over their system.
- The problem lies:
  - you should be an expert to build and run efficiently your clusters
  - not always the problem you have fit into a cluster solution (even if this is cheap!)

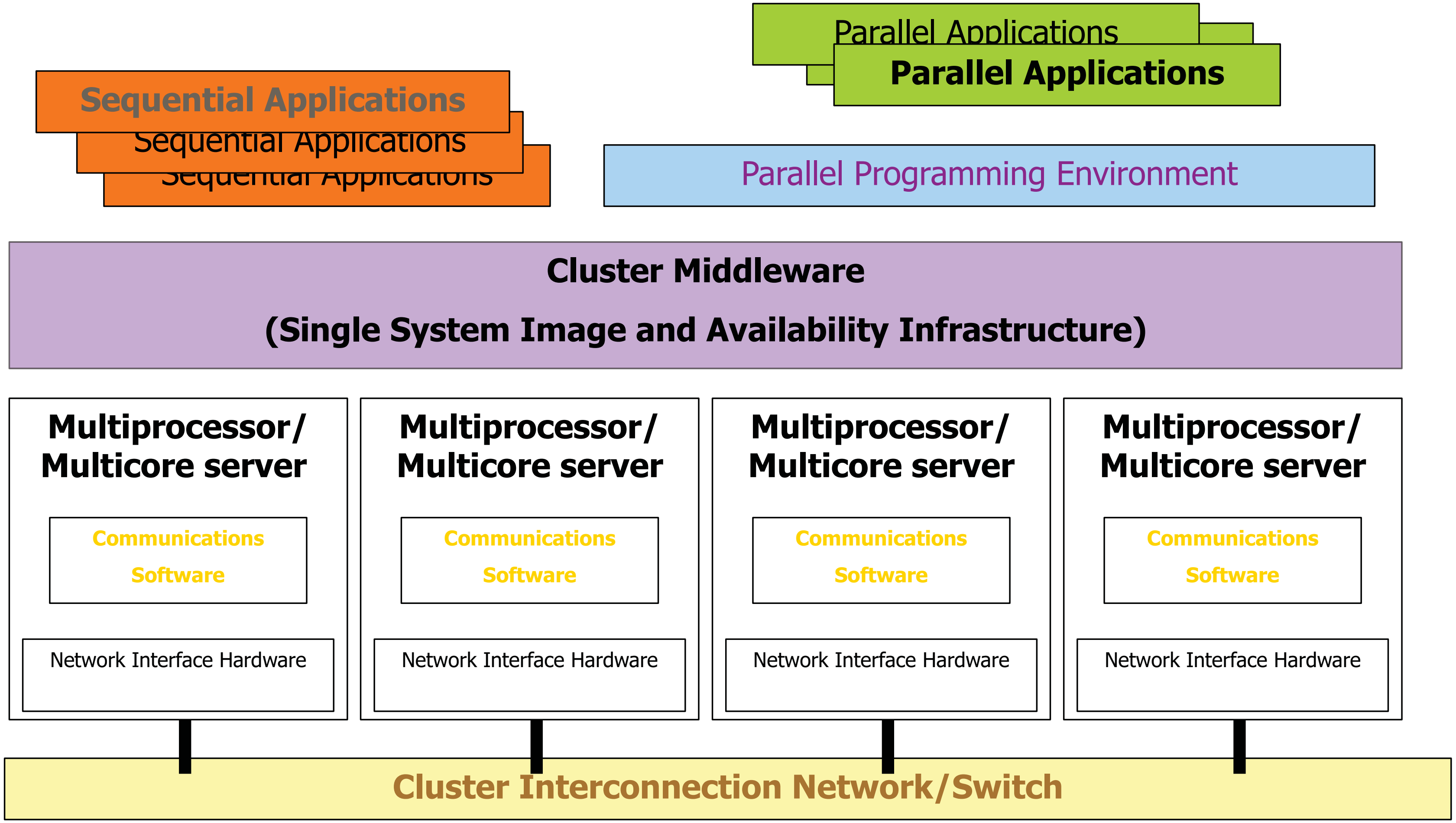
Evolution of cluster architectures in top500



Evolution of operating system in top500

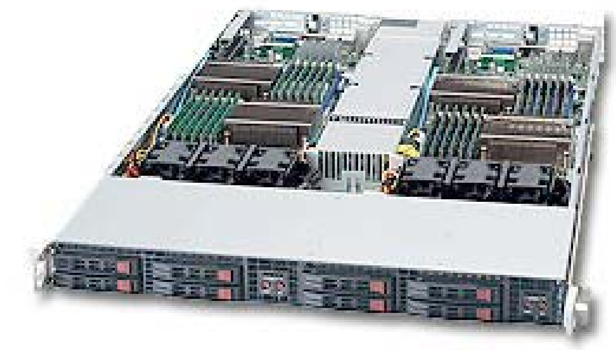
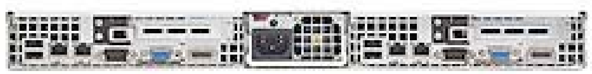


Cluster Computer Architecture



Hardware bricks: Computational nodes:

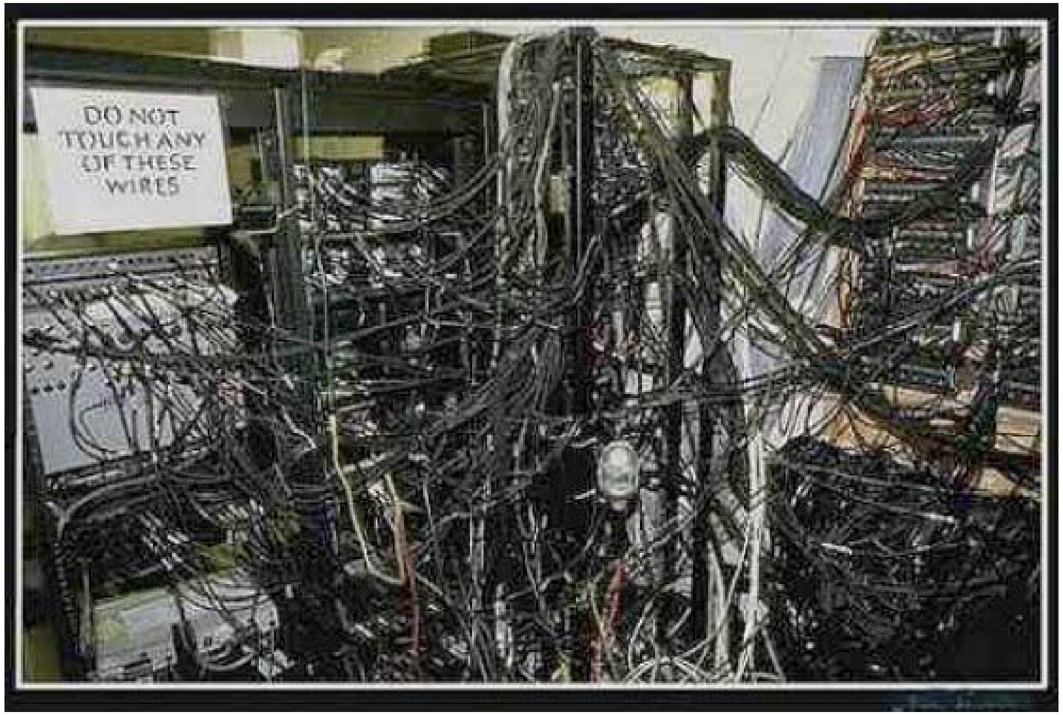
- Discussed yesterday
- To remember:
  - Multiprocessor/multicore architectures:
- Sissa cluster:

- Zebra: Intel Xeon E5420 2.5Ghz processors: 2 sockets 4 core =8
- iblade: AMD 275 2.2Ghz processor: 2 sockets 2 core= 4

About network for clusters

- The characteristics of the network cannot be ignored
  - Topology
    - Diameter
    - Bisection bandwidth
  - Performance
    - Latency
    - Link bandwidth



Interconnect Topologies

- ***Bus***
  - Nodes share a "party line".
  - Not very common any more, except between processors and memory inside a host.
- ***Hypercube***–SGI Origin and Altix
  - Nodes are vertices on an n-dimensional hypercube.
- ***Mesh***–Cray T3D/E and XT-3/4/5, IBM BlueGene
  - A 1D mesh with wrap-around at the edges is called a ***ring***.
  - A 2D (or more) mesh with wrap-around at the edges is called a ***torus***.
- ***Switched***–Ethernet, Infiniband, Myrinet,
  - Nodes are connected to a concentrator called a switch.
  - Multiple switches may be connected hierarchically (i.e. as a tree) or in any of the above topologies.

Interconnect Characteristic

- ***Latency:*** Initialization time before data can be sent
- ***Per-link Peak Bandwidth:*** Maximum data transmission rate (varies with packet size)
- ***Diameter***: Maximum number of hops to get between most distantly connected nodes.
  - Hypercube networks have best diameter, at most log 2(N) for N nodes.
-

Which networks for Linux Cluster ?

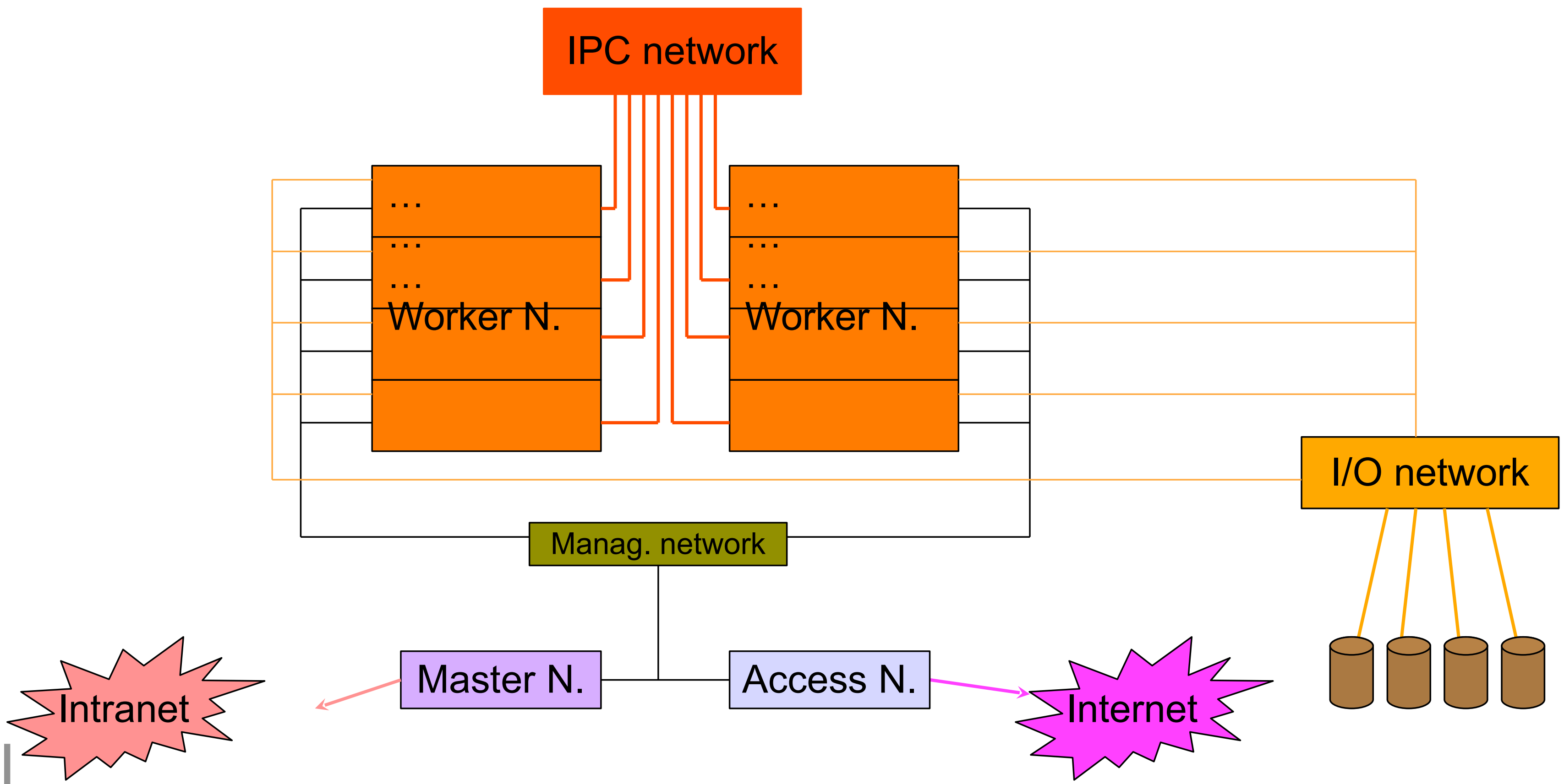
- Commodity
  - Gigabit Ethernet
- Pro:
  - Always available/easy to manage
  - Bandwidth could be aggregate easily (trunking/bonding)
- Cons:
  - High latency

- High Speed Network
  - Infiniband
- Pro:
  - Excellent performance: low latency high BW
- Cons:
  - Complex to manage
  - Expensive

Network Clusters classification

- HIGH SPEED NETWORK
  - parallel computation
    - low latency /high bandwidth
    - Usual choices: Myrinet / Infiniband...
- I/O NETWORK
  - I/O requests (NFS and/or parallel FS)
    - latency not fundamental/ good bandwidth
    - GIGABIT is ok
- Management network
  - management traffic
    - any standard network (fast ethernet OK)

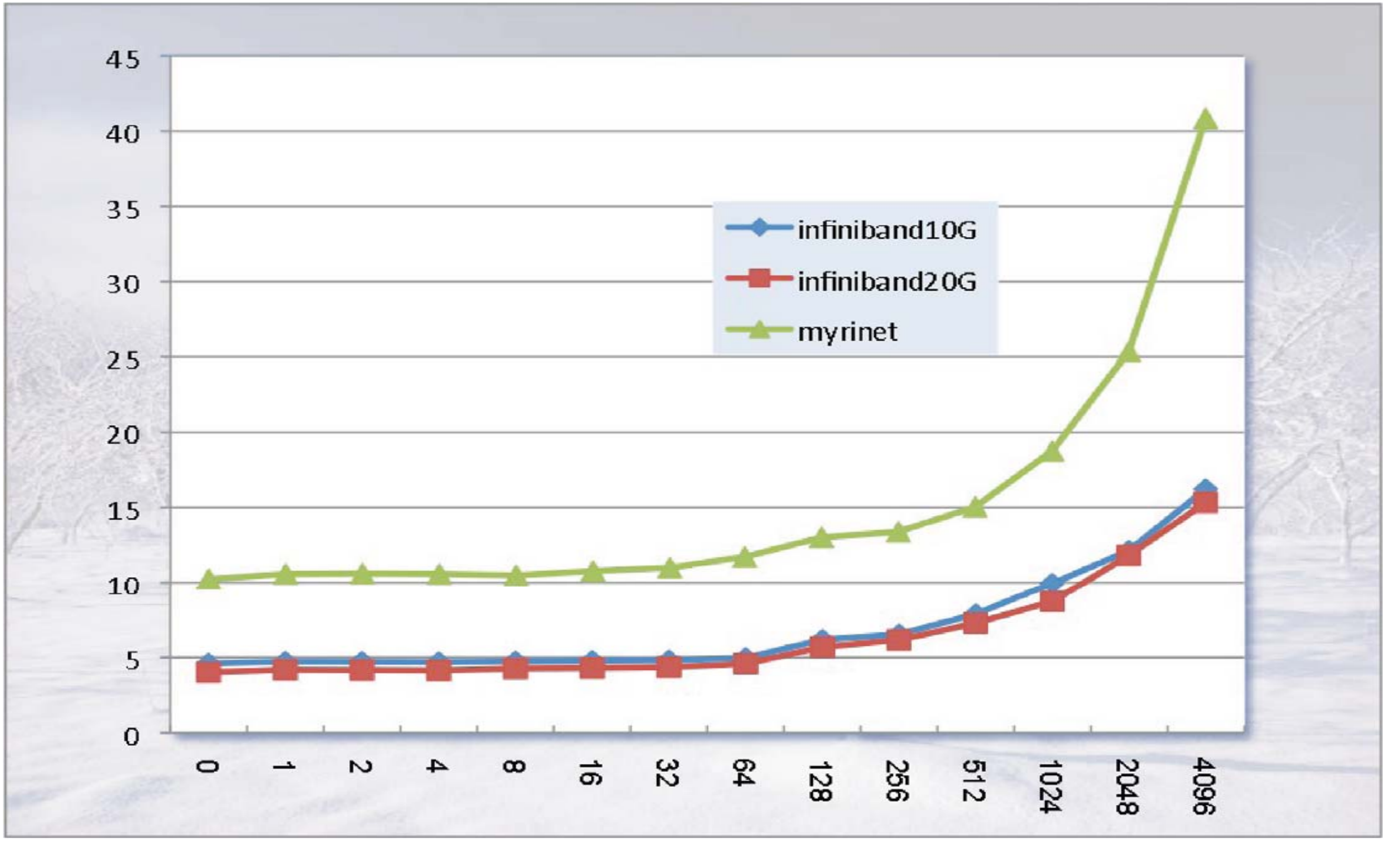
HPC cluster logical structure



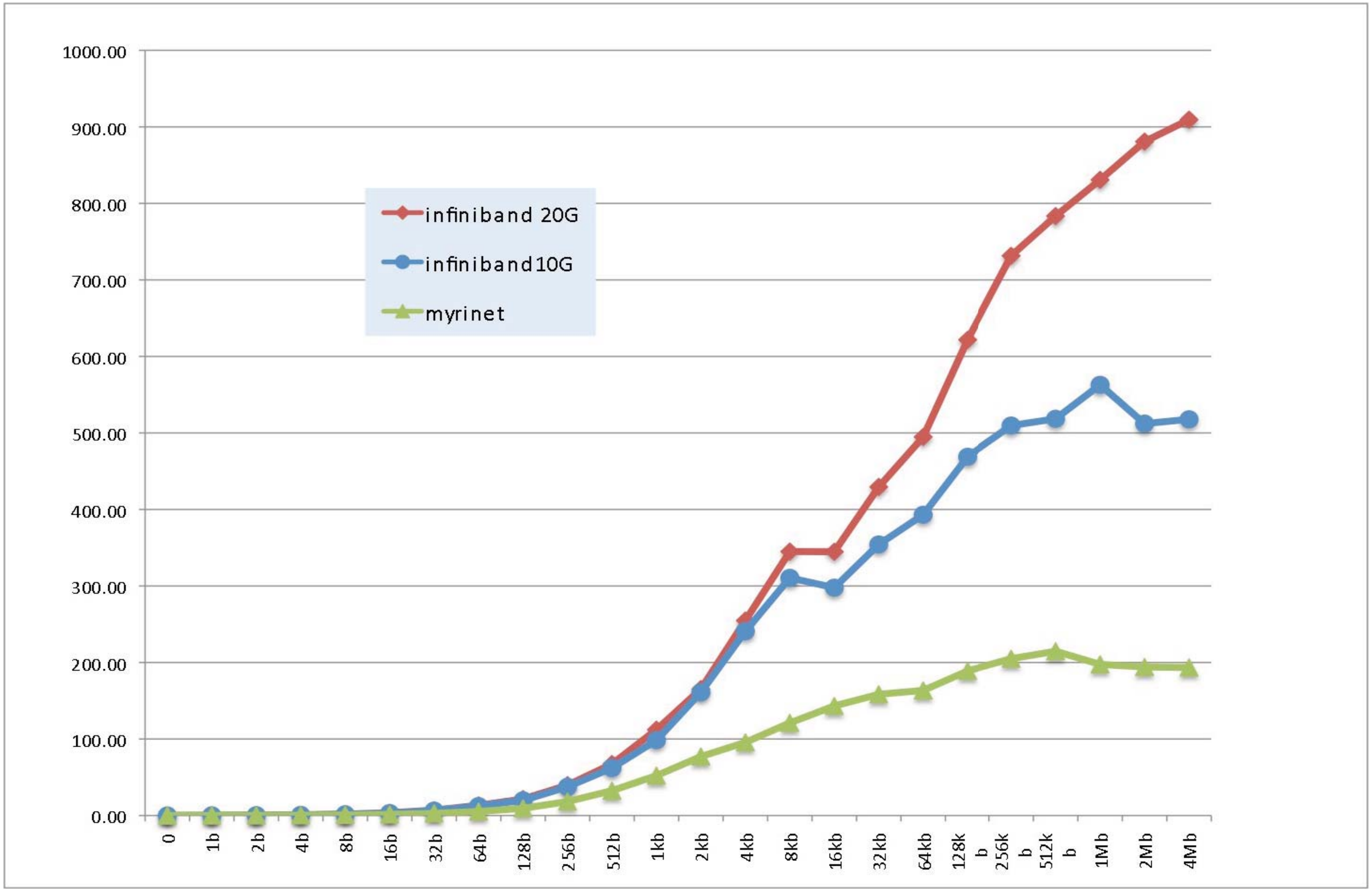
Interconnect Characteristics:

- Latency: Initialization time before data can be sent
  - How much does it take to open the channel ?
- Per-link Peak Bandwidth: Maximum data transmission rate (varies with packet size)
  - How wide is my channel ?
- Bisection Bandwidth: Bandwidth available if one half of nodes try communicating with the other half simultaneously.
- How to measure them ?
  - IMB benchmark : it will be use later in the lab..

Sissa cluster: latency



Sissa number: bandwidth

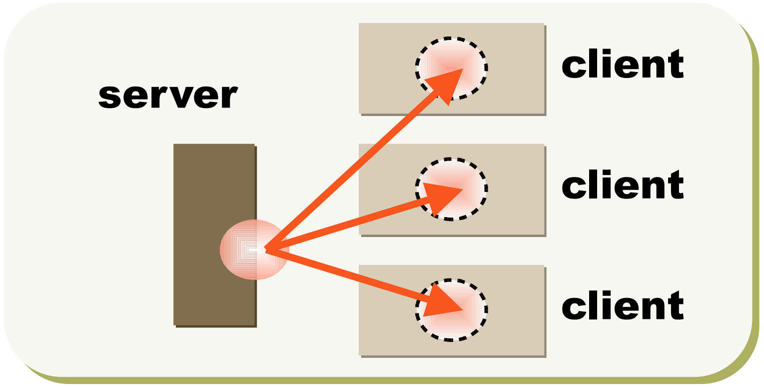


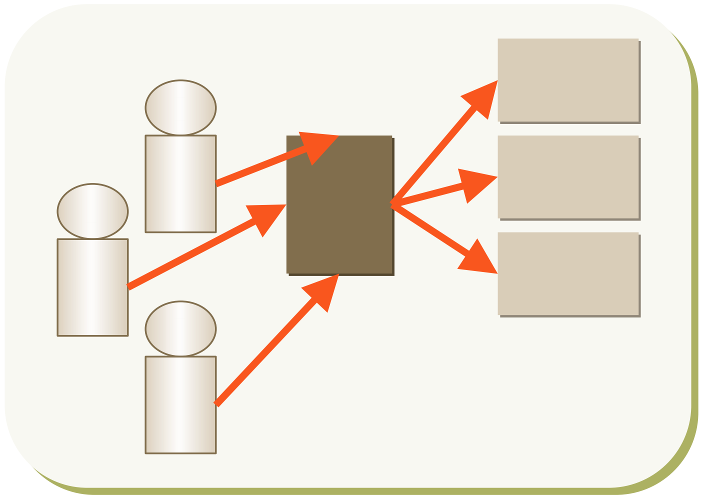
high speed network considerations

- In general the compute/communication ratio in a parallel program remains fairly constant.
- So as the computational power increases the network speed must also be increased.
- As multi-core processors proliferate, it is increasingly common to have 4, 8, or even 16 MPI processes sharing the same network device.
- Contention for the interconnect device can have a significant impact on performance.

Cluster middleware: beowulf approach

- Administration software:
  - NFS
  - user accounts
  - NTP
  - Monitoring tools
- Resource management and scheduling software (LRMS)
  - Process distribution
  - Load balance
  - Job scheduling of multiple tasks

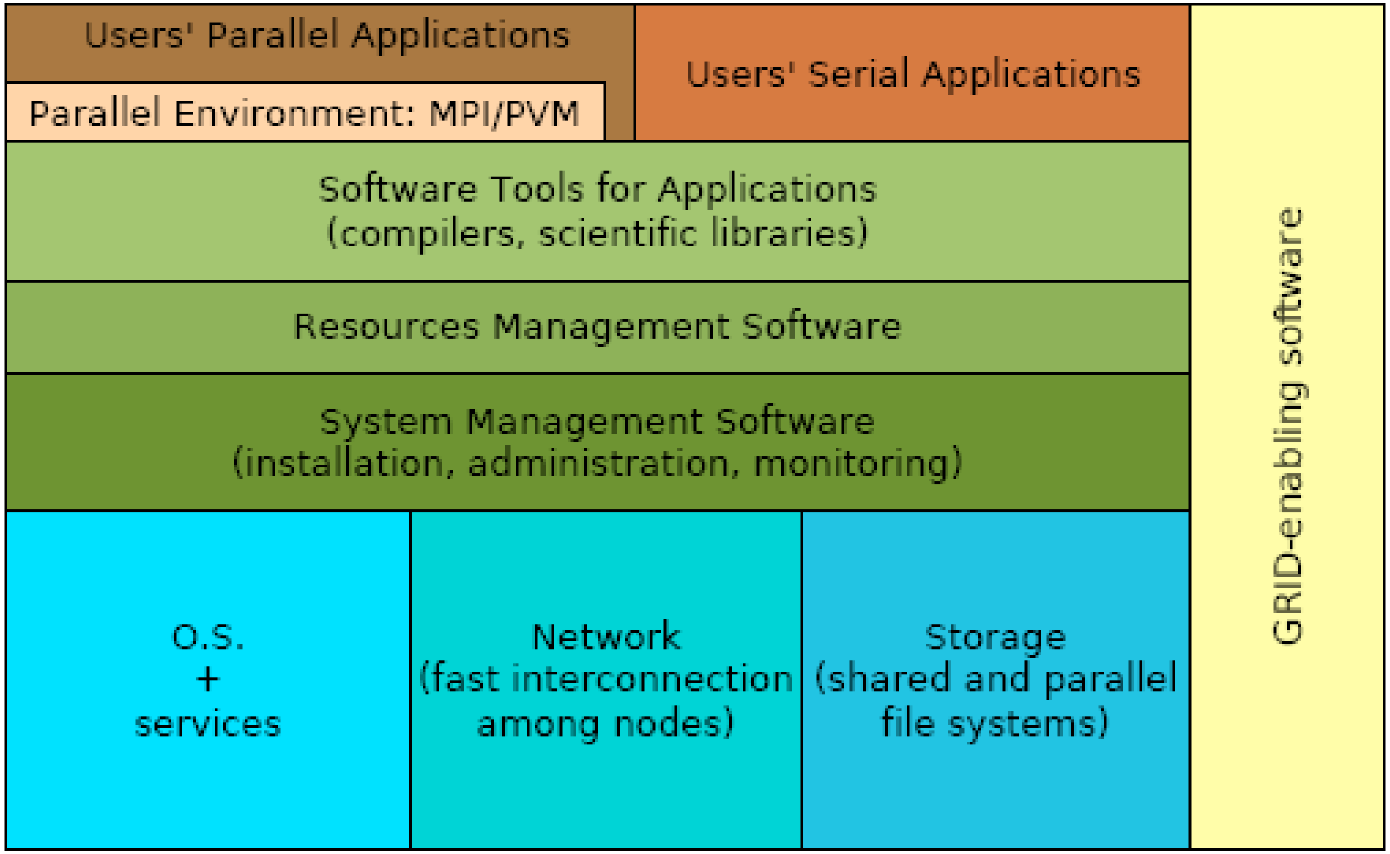




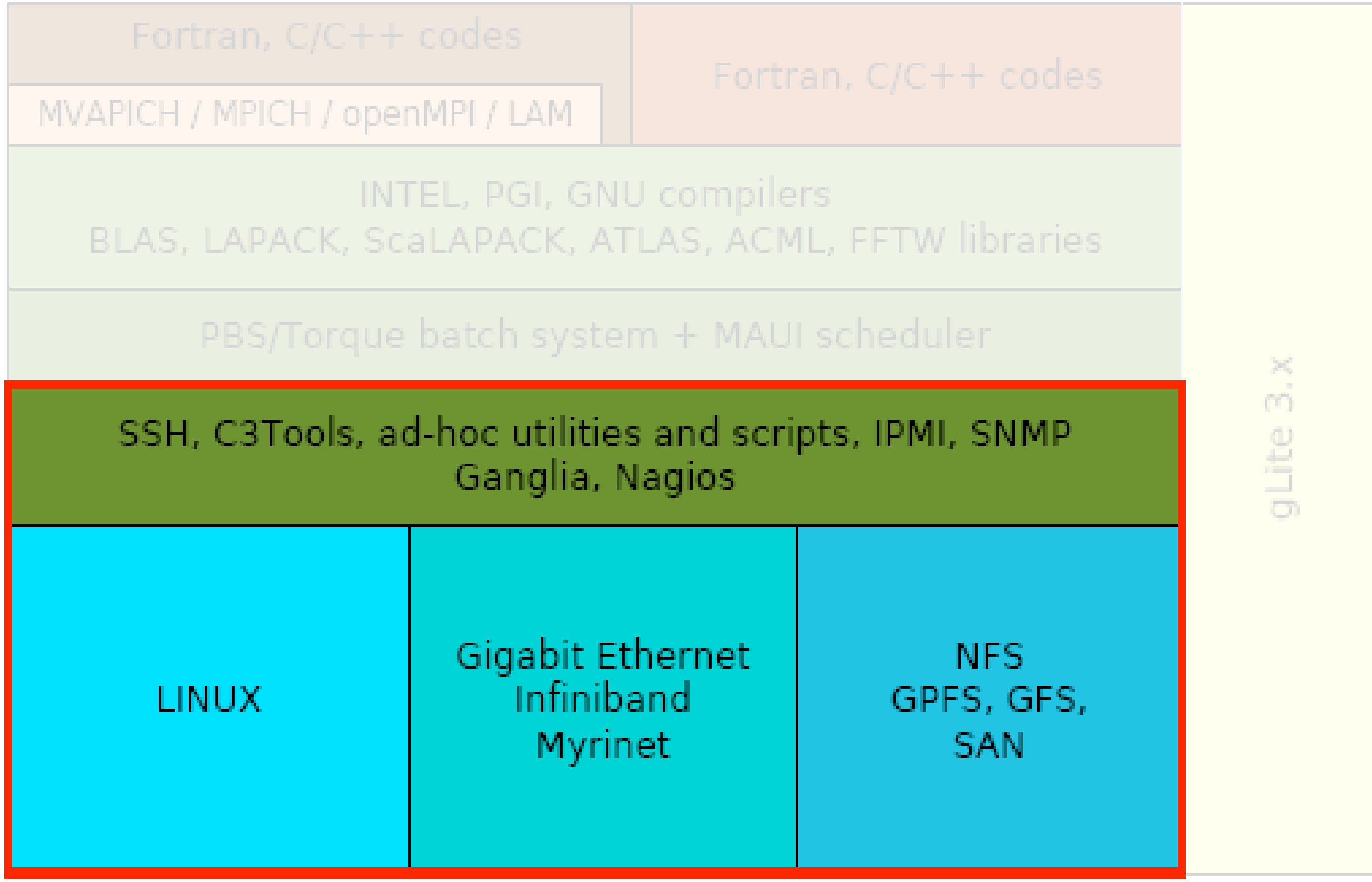
Cluster Management Toolkits

- Are generally made of an ensemble of already available software packages thought for specific tasks, but configured to operate together, plus some add-ons.
- Sometimes limited by rigid and not customizable configurations, often bound to some specific LINUX distribution and version. May depend on vendors' hardware.
- Free and Open
  - NPACI Rocks
  - xCAT (eXtreme Cluster Administration Toolkit)
  - EPICO : Enhanced Package for Installation and Configuration (locally developed v0.1)
- Commercial
  - Scyld Beowulf
  - IBM, HP, SUN and other vendors' Management Software...

Linuux Cluster: the software stacks



Linux Cluster: the sys. Adm. stacks



Cluster Pro&Cons

- Pro:
  - Price/performance when compared with a dedicated parallel supercomputer
  - Great opportunity for low budget institution
  - Flexibility: many ad hoc solution for different problems..
  - Open Technology
    - What you learn in this business can be used everywhere..
- Cons:
  - It is hard to build and operate medium and large cluster
    - Large collection of software that are not "talk to each other"
  - Lot of expertise needed (no plug and play yet)
  - How to use cluster power efficiently

Which cluster do I need ?

- Which applications ?
  - Parallel
    - Tightly coupled
    - Loosely coupled
  - Serial
    - Memory / I/O requirements
- Which user's community ?
  - Large /Small
  - Homogeneous /heterogeneous

Which computational nodes ?

- AMD vs Intel ?
  - price/performance ratio..
- Important parameters:
  - Remote management tools available
  - Form factor (1U / blade / desktop ?)
  - Power consumption ?
  - Scalability
- Budget consideration

Which network ?

- Difficult choice:
  - Which kind of cluster (HTC or HPC) ?
  - Which kind of application ?
    - Serial/Parallel
    - Parallel loosely coupled / tightly coupled ?
    - Latency or bandwidth dominated ?
  - Budget considerations
  - I/O considerations

To close..

Understand your computational problem before buying/building a cluster !
Run your own benchmarks before buying/building a cluster !