PARADOXical Training
Institute of Physics Belgrade, 14 October 2011

Introduction to High Performance Computing

Antun Balaz
Institute of Physics Belgrade
http://www.scl.rs/

Academic and Educational Grid Initiative of Serbia
AEGIS

PRACE

HP-SEE

egi

gLite

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Agenda

- Introduction:
  - or why scientists need HPC & GRID
- Section 1: HPC concepts
- Section 2:
  - Parallel computing
  - Parallel machines
- Section 3: GRID concepts
- Conclusions
  - Computer infrastructure for everybody

Why computer simulations in science ?

- **numerically solve theories** which could not be solved otherwise (i.e. get numbers out of theories, much in the same way as experiments get numbers out of nature)
- **perform virtual experiments** when real experiments are not possible, or under conditions controlled in ways not possible in the lab
- **benchmark** the soundness of ideas and theories

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Examples of numerical simulations at SCL

- Ultra-cold quantum gases
  - Global and local properties of BEC
  - Parametric resonance
- Strongly correlated systems
  - Fractional quantum Hall effect
  - Bose-Hubbard
- Granular materials
  - Classical MD simulations of collective behavior

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

What resources are needed to perform them?

- Example 1: pure CPU
  - Monte Carlo for global properties of BEC
- Example 2: CPU and memory
  - Exact diagonalization for study of BEC and FQHE
  - Large number of particles in studies of granular material
- Example 3: CPU and storage
  - Single runs sometimes produce large (TBs) outputs, or may need to process large inputs
- Ad hoc, fast, powerful, reliable computational platform - in short HPC!

Agenda

- Introduction:
  - or why scientists need HPC & GRID
- **Section 1: HPC concepts**
- Section 2:
  - Parallel computing
  - Parallel machines
- Section 3: GRID concepts
- Conclusions
  - Computer infrastructure for everybody

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

What HPC stands for?

- High Performance Computing
- The term is most commonly associated with computing used for scientific research. [from Wikipedia]
- It involves not only hardware, but software and people as well!
- HPC encompasses a collection of powerful:
    - hardware systems
    - software tools
    - programming languages
    - parallel programming paradigms

which make previously unfeasible calculations possible

Scientific Computing Laboratory

14 Oct 2011

Only performance?

- High Throughput Computing
- High Availability Computing
- Capacity Computing
- Capability computing
- To reflect a greater focus on the productivity, rather than just the performance, of large-scale computing systems, many believe that HPC should now stand for High Productivity Computing

How to run applications faster ?

- There are 3 ways to improve performance:
  - Work Harder
  - Work Smarter
  - Get Help
- Analogy in computer science
  - Use faster hardware
  - Optimize algorithms and techniques used to solve computational tasks
  - Use multiple computers to solve a particular task
- All 3 strategies can be used simultaneously!

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Measures of performance

- How fast can I crunch numbers on my CPU?
- How fast can I move the data around?
  - from CPUs to memory
  - from CPUs to disk
  - from CPUs to/on different machines
- How much data can I store?

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

CPU crunching

- Number of floating point operations per second (flops, Mflops, Gflops…)
- Comparison of theoretical and sustained peak performance
- Theoretical peak performance:
  - determined by counting the number of floating-point additions and multiplications that can be completed during a period of time, usually the cycle time of the machine
  - IPC=Instruction per Cycle

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Peak performance of modern systems

- My laptop: 9.6 Gflops
  - 4 IPC x 2 cores x 2.4 GHz
- Cluster at SCL: 6.6 Tflops
  - 4 IPC x 712 cores x 2.33 GHz
- sp6.cineca.it: 100 Tflops
  - 5300 Power6 processors @ 4.7Ghz
- Cray Jaguar, ORNL: 2331 Tflops
  - Opteron Six Core 2.6 GHz
  - 224162 cores

Sustained performance

- What your application is actually able to get from the computer hardware

- Example from previous century:

Table 1: Benchmark Results for NERSC's 644-PE Cray T3E

Benchmark	System Performance	Single Processor Performance	% of Peak
Theoretical peak	580.0 Gflop/s	900 Mflop/s	100.0%
Linpack	444.2 Gflop/s	690 Mflop/s	76.6%
LSMS code (Locally Self-consistent Multiple Scattering), 1998 Gordon Bell Prize-winning application	256.0 Gflop/s	398 Mflop/s	44.1%
Average of seven major NERSC applications	67.0 Gflop/s	~104 Mflop/s	11.6%
NAS Parallel Benchmarks	29.6 Gflop/s	~46 Mflop/s	5.1%

Data movement

- bit/s transmitted
- among computers: networks
  - default (commodity): 1 Gb/s
    - 1000 Mb/s = 1 Gb/s
  - custom (high speed)
    - 10Gb/s, 20 Gb/s and now 40Gb/s
- within the computer:
  - CPU – Memory: thousands of Mb/s
    - 10 - 100 Gb/s
  - CPU - Disks: MByte/s
    - 50 ~ 100 MB/s up to 1000 MB/s

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Storage sizes

- Size of storage devices:
  - kbyte/Mbyte ----> caches / RAM
  - Gigabyte ----> RAM / hard disks
  - Terabyte ----> Disks / SAN
  - Petabyte ----> SAN / Tapes

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

HPC architecture

HPC architectures try to maximize performance simultaneously on all the three aspects (number crunching/ data access /data storage) by using many Processing Elements (CPUs) together to solve a given task

PARALLEL COMPUTING

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Agenda

- Introduction:
  - or why scientists need HPC & GRID
- Section 1: HPC concepts
- **Section 2:**
  - Parallel computing
  - Parallel machines
- Section 3: GRID concepts
- Conclusions
  - Computer infrastructure for everybody

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

What is parallel computing?

- Parallel computing is the simultaneous execution of the same task (split up and specially adapted) on multiple processors in order to obtain results faster
- The process of solving a problem usually can be divided into smaller tasks, which may be carried out simultaneously with some coordination

[from Wikipedia]

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

High performance problem example:

picture from http://www.f1nutter.co.uk/tech/pitstop.php

Scientific Computing Laboratory

14 Oct 2011

Analysis of the parallel solution

- Functional decomposition
  - different people are executing different tasks

- Domain decomposition
  - different people are solving the same global task but on smaller subset

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

HPC parallel computers

- The simplest and most useful way to classify modern parallel computers is by their memory model.

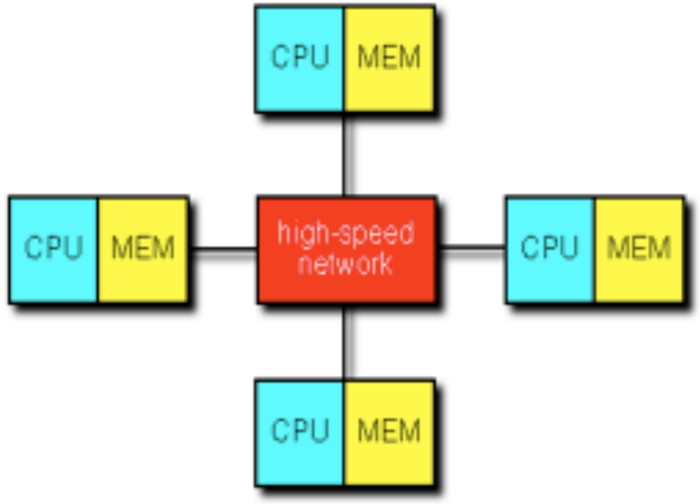
- How CPUs view and can access the available memory?
  - SHARED MEMORY
  - DISTRIBUTED MEMORY

SCIENTIFIC COMPUTING LABORATORY

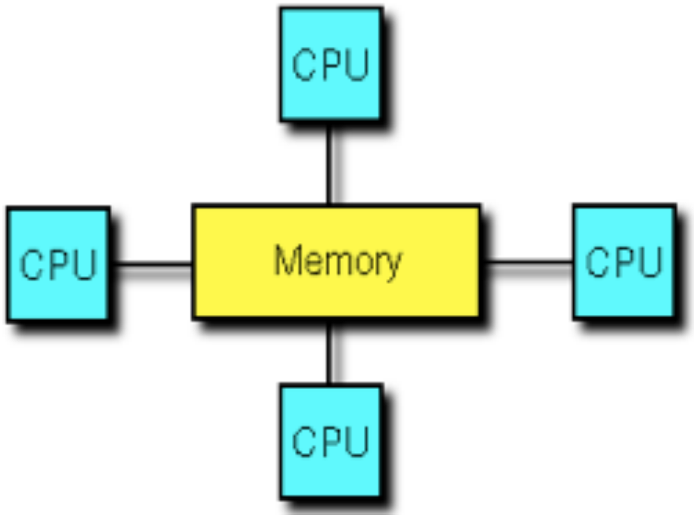
14 Oct 2011

Shared vs. Distributed

- Distributed Memory:
  - Each processor has its own local memory. Must do message passing to exchange data between processors.
  - Multi-computers



- Shared Memory
  - Single address space. All processors have access to a pool of shared memory.
  - Multi-processors

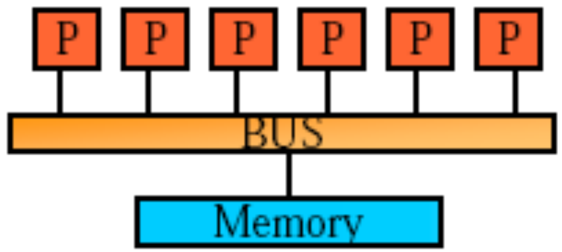


SCIENTIFIC COMPUTING LABORATORY

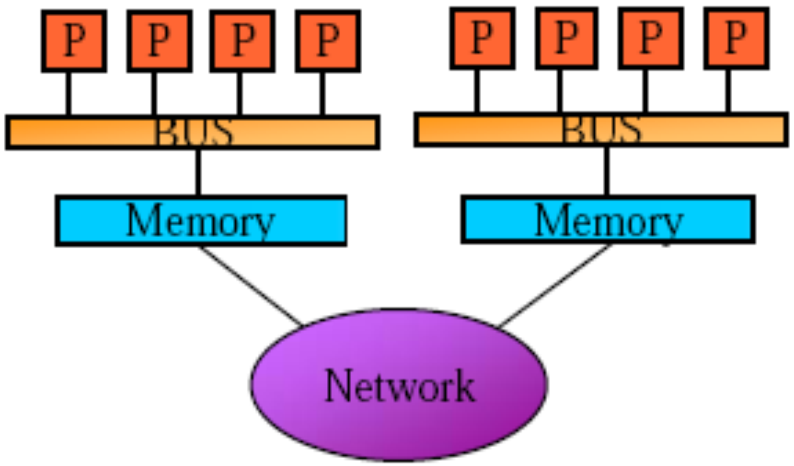
14 Oct 2011

Shared Memory: UMA vs. NUMA

- Uniform memory access (UMA): Each processor has uniform access to memory. Also known as symmetric multiprocessors (SMP).



- Non-uniform memory access (NUMA): Time for memory access depends on location of data. Local access is faster than non-local access.

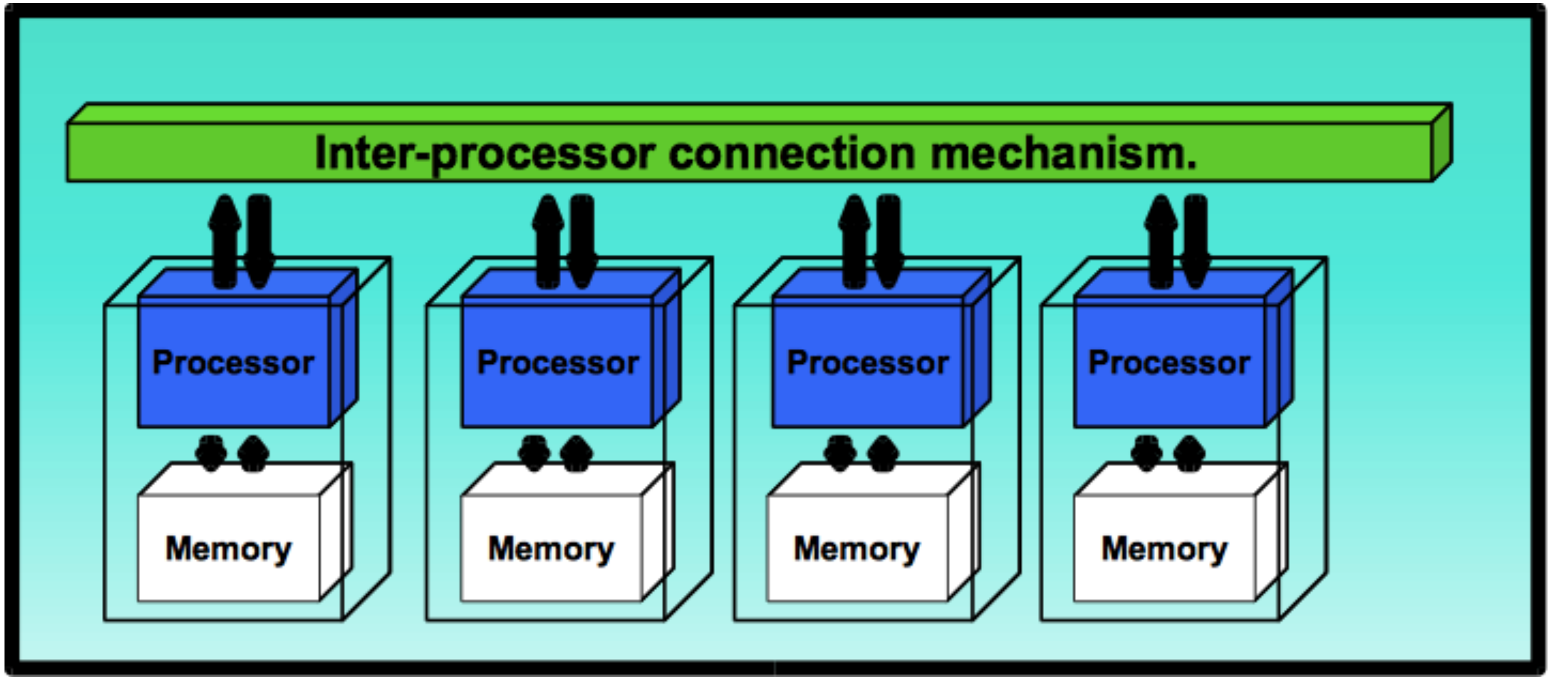


SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Clusters: distributed memory

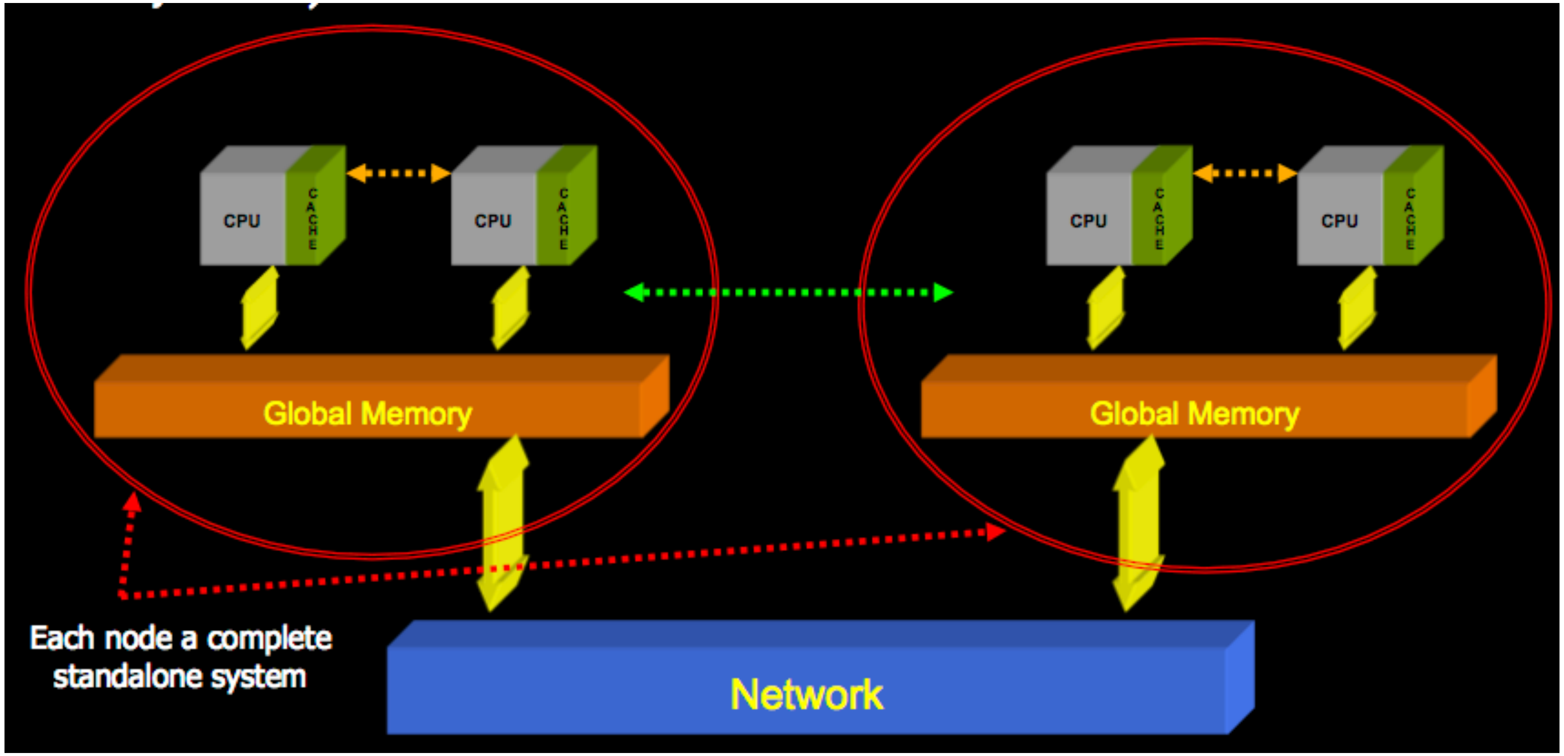
- Independent machines combined into a unified system through software and networking



14 Oct 2011

Hybrids

- Modern clusters have hybrid architecture
- Many-core CPUs make each node a small SMP system



SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Parallel Programming Paradigms

- Memory models determine programming paradigms

Parallel machines	
Distributed memory	Shared memory
Parallel paradigms	
Message passing	Data parallel
All processes could **directly access only their local memory.** Explicit messages are requested to access remote memory of different processors.	Single memory view. all processes (usually threads) could **directly access the whole memory.**

Scientific Computing Laboratory

14 Oct 2011

Parallel performance

- 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 a perfect speedup (also called linear scaling)

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Absolute performance

- Speedup compares performance of 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

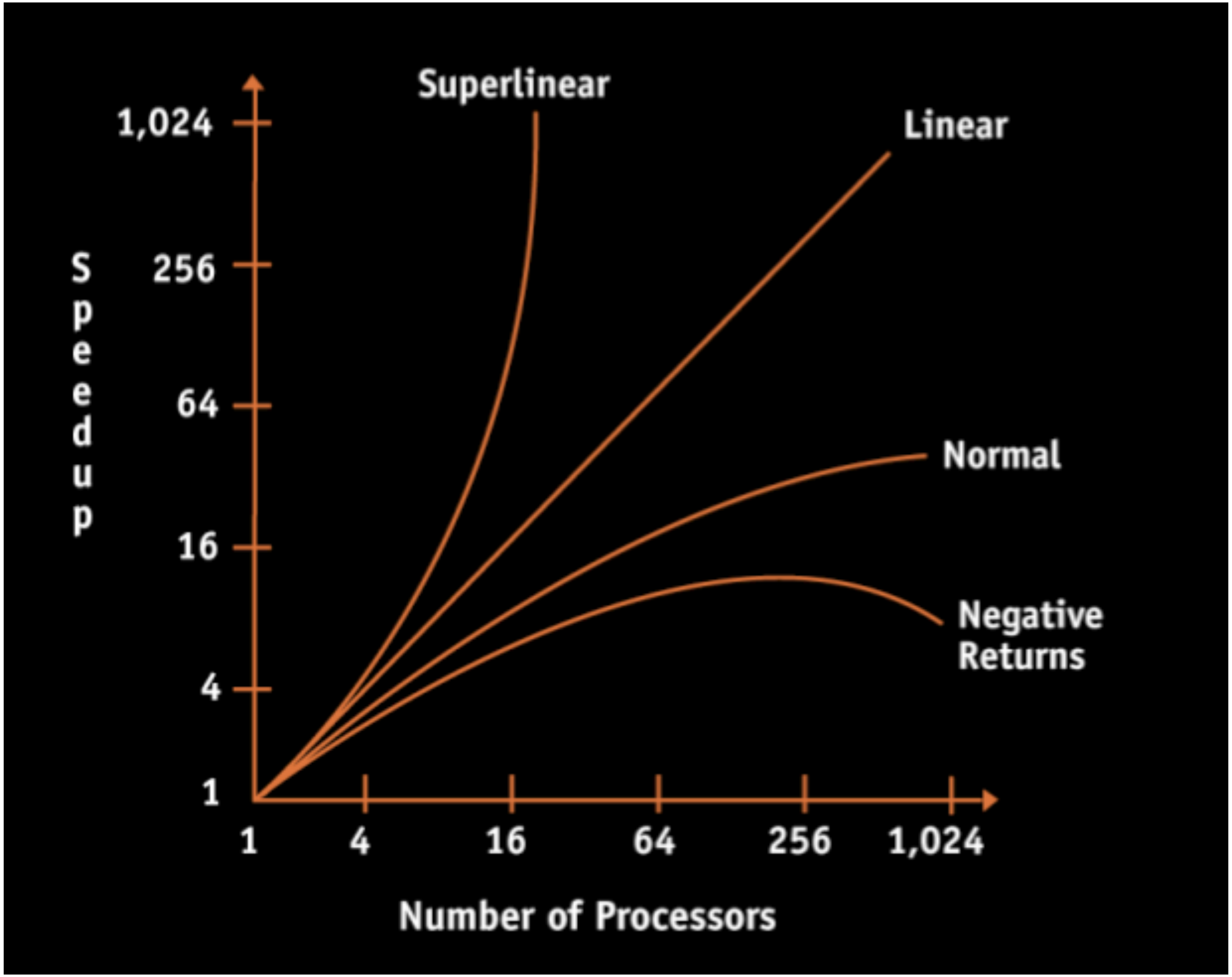
  vs.

  - The execution time of best parallel algorithm on p processors

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Speedup



SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Superlinear speedup?

- Can we find **superlinear** speedup, i.e.

  Speedup(p) > p
- Yes, we can:
  - 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.

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

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 ?
  - 1) program A
  - 2) program B
  - 3) None of the above

Scientific Computing Laboratory

14 Oct 2011

Answer

- None of the above, since we do not know the **overall execution time** of each 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 to produce the executable, for both serial and parallel version of the application!

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Limits to speedup

- All parallel programs contain:
  - Parallel sections
  - Serial sections
- Serial sections limit 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.

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Amdahl's law (1)

- Let *s* be the fraction in an application representing the work done serially
- Then, 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 scales perfectly, we may be limited by the sequential portion of the code!

Scientific Computing Laboratory

14 Oct 2011

Amdahl's law (2)

- The presence of a serial part of the code is quite limiting in practice:

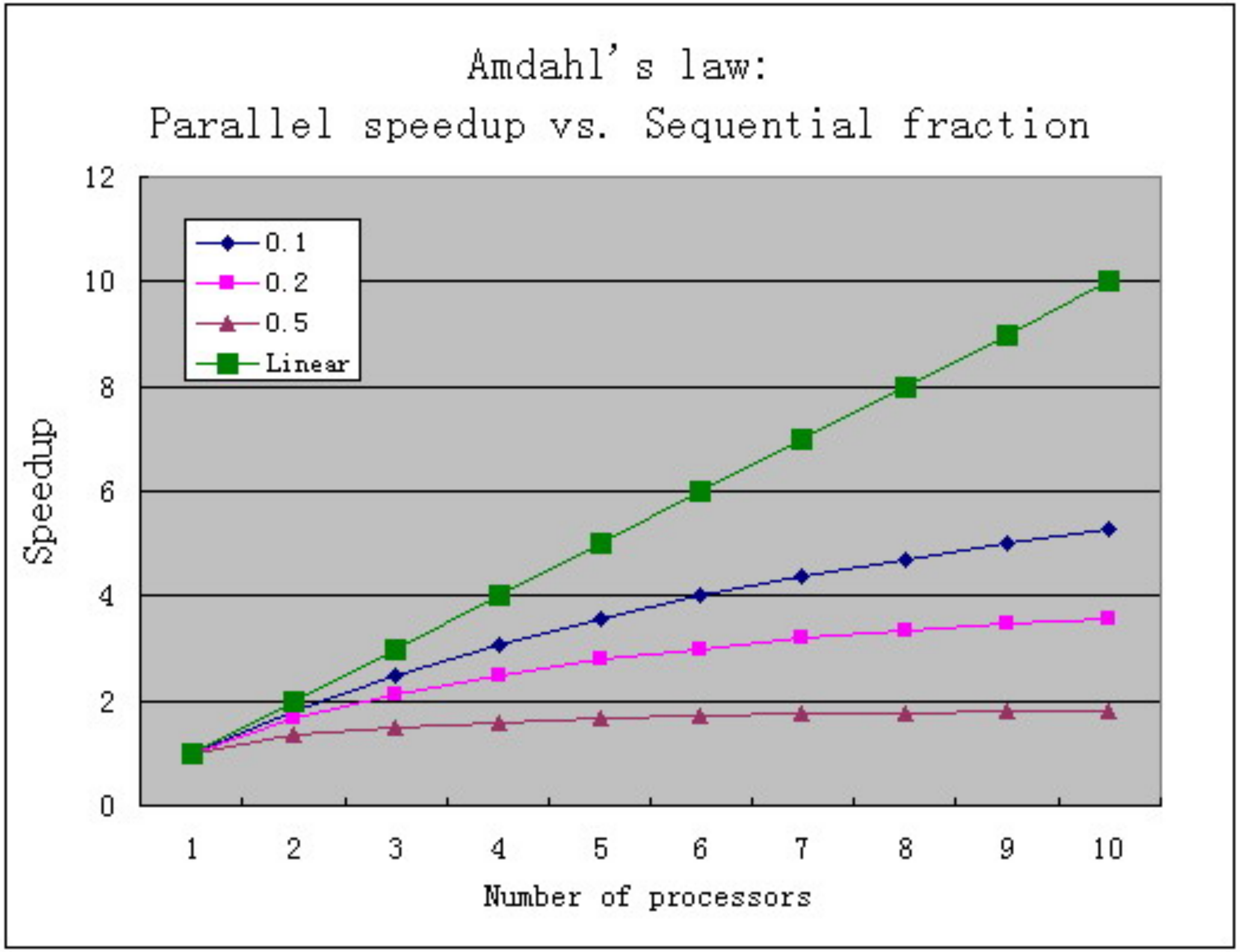
>	2	4	8	32	64	256	512	1024
5%	1.91	3.48	5.93	12.55	15.42	18.62	19.28	19.63
2%	1.94	3.67	6.61	16.58	22.15	29.60	31.35	32.31
1%	1.99	3.88	7.48	24.43	39.29	72.11	83.80	91.18

- Amdahl's Law is relevant only if serial fraction is independent of the problem size
- Fortunately, the proportion of the computations that are sequential (non parallel) usually decreases as the problem size increase (a.k.a. Gustafson's law)

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Effective parallel performance



Scientific Computing Laboratory

14 Oct 2011

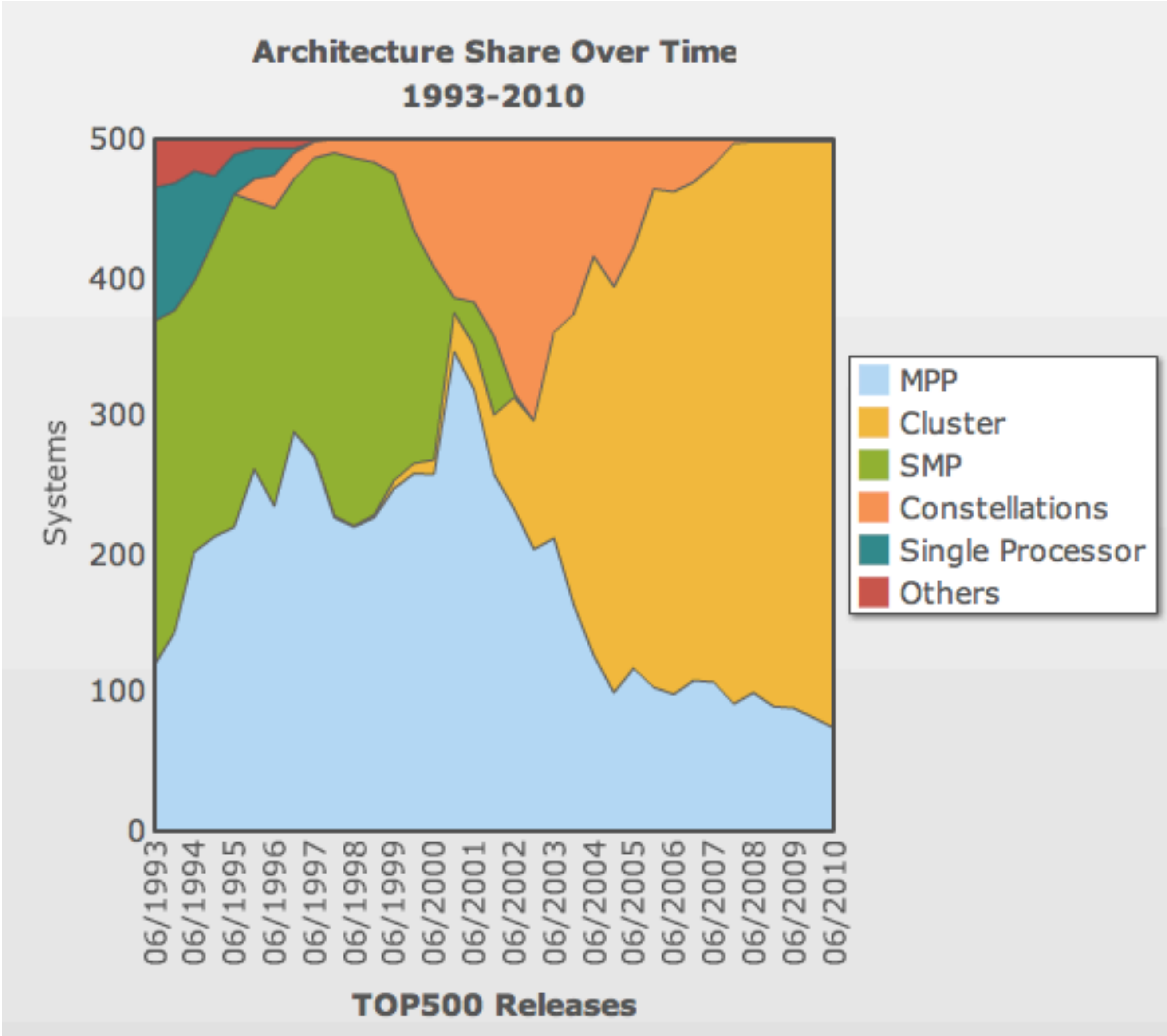
Linux cluster revolution

- The adoption of clusters virtually exploded since the introduction of the first Beowulf cluster in 1994
- The ingredients / attraction lies in:
  - low costs of both hardware and software
  - affordable interconnect technologies
  - the control that builders / users have over their own systems
- The problems:
  - you should be an expert to build and run efficiently your clusters
  - not always the problem you have fits into a cluster solution (even if this is cheap!)

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

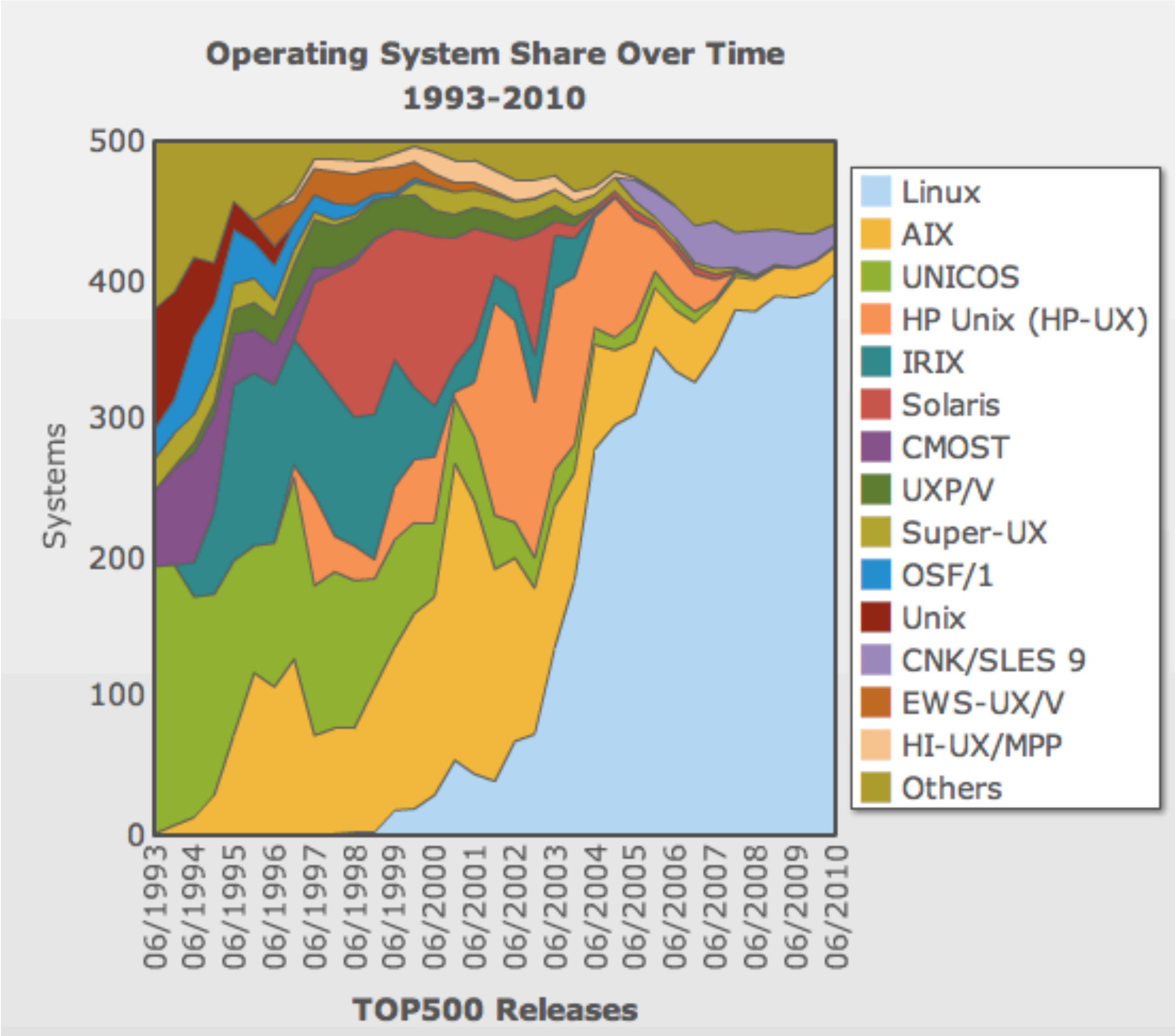
Clusters on Top500



SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Linux clusters on Top500



Scientific Computing Laboratory

14 Oct 2011

Agenda

- Introduction:
  - or why scientists need HPC & GRID
- Section 1: HPC concepts
- Section 2:
  - Parallel computing
  - Parallel machines
- **Section 3: GRID concepts**
- Conclusions
  - Computer infrastructure for everybody

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

GRID: cluster of clusters

- Motivation: When communication is close to free we should not be restricted to local resources when solving problems.
- A Grid Infrastructure built on top of the Internet and the Web to enable and exploit large scale sharing of resources
- It should provides **Scalable, Secure, Reliable** mechanisms for discovery and for remote access of resources.

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Resource sharing

- **Applications**: web services technology
- **CPU and Storage**: Grid computing, Cloud Computing, etc.
- **Data**: Data Grid, Virtual Observatory, Google Filesystem, etc.
- **Instruments**: Virtual Labs, collaboration tools, etc.

Scientific Computing Laboratory

14 Oct 2011

Agenda

- Introduction:
  - or why scientists need HPC & GRID
- Section 1: HPC concepts
- Section 2:
  - Parallel computing
  - Parallel machines
- Section 3: GRID concepts
- **Conclusions**
  - Computer infrastructure for everybody

Scientific Computing Laboratory

14 Oct 2011

Building a computational infrastructure

- Open source software + commodity (off the shelf) hardware provide now tools to build low cost HPC infrastructure based on clusters
- GRID infrastructures are just two clicks away and can provide large amounts of resources
- Planning of a computational infrastructure depends on your needs

Scientific Computing Laboratory

14 Oct 2011

Elements of a computational infrastructure

- Hardware
  - The basic bricks
- Software
  - To make hardware usable
- People
  - Installers / sys admins / planners / users etc..
- Problems to be solved
  - Any action in building such an infrastructure should be motivated by real needs

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

What infrastructure do you need?

- Applications
  - Parallel
    - Tightly coupled
    - Loosely coupled
    - Embarrassingly parallel
  - Serial
    - Memory / I/O requirements
- User community
  - Large /Small
  - Distributed or not?
  - Homogeneous /heterogeneous
- Budget considerations

Scientific Computing Laboratory

14 Oct 2011

HPC projects

- HP-SEE regional HPC project
- PRACE-1IP: European Tier-0 systems
- PRACE-2IP: European Tier-1 systems

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Conclusions (1)

- Modern scientific research need lots of computational resources provide by HPC/GRID infrastructures
- HPC means parallel computing
- GRID means pooling of geographically distributed resources
- HPC and GRID computing are not mutually exclusive but can be both used to address computational resources in a transparent way.
- The challenge is now to build your own computational infrastructure **driven by real needs**

SCIENTIFIC COMPUTING LABORATORY

14 Oct 2011

Conclusions (2)

- With access to commodity HPC hardware and a free OS such as Linux, entry-level HPC is within reach of even the most modest budget.
- It is relatively easy to join large Grid infrastructures that make available large amount of computational resources.

However:

- To fully exploit HPC/GRID one needs to obtain detailed knowledge of HPC/GRID architectures as well as master HPC/GRID development tools and advanced programming techniques.
- We hope to give some insight about these in this two week school!

Scientific Computing Laboratory

14 Oct 2011