The Abdus Salam
International Centre
for Theoretical Physics

2494-1

Workshop on High Performance Computing (HPC) Architecture and Applications in the ICTP

14 - 25 October 2013

Introduction to High-Performance Computing

Axel Kohlmeyer

ICTP, Trieste / Temple University
Philadelphia
USA

Introduction to High-Performance Computing

Dr. Axel Kohlmeyer

Sr. Scientific Computing Expert
ICTP, Trieste

Associate Dean for Scientific Computing
Temple University, Philadelphia

http://sites.google.com/site/akohlmey/

The Abdus Salam International Centre for Theoretical Physics

Why use Computers in Science?

- Use complex theories without a closed solution:
  solve equations or problems that can only be solved numerically, i.e. by inserting numbers into expressions and analyzing the results
- Do “impossible” experiments:
  study (virtual) experiments, where the boundary conditions are inaccessible or not controllable
- Benchmark correctness of models and theories:
  the better a model/theory reproduces known experimental results, the better its predictions

What is High-Performance Computing (HPC)?

- Definition depends on individual person
  > HPC is when I care how fast I get an answer
- Thus HPC can happen on:
  - A workstation, desktop, laptop, smartphone!
  - A supercomputer
  - A (Linux) cluster ← Our main topic
  - A grid or a cloud
  - Cyberinfrastructure = any combination of the above
- HPC also means High-Productivity Computing

The Abdus Salam International Centre for Theoretical Physics

Parallel Workstation

- Most desktops today are parallel workstations => multi-core processors
- Running Linux OS allows running programming like on a traditional Unix workstation
- All processor cores have access to all memory
  - Uniform memory access (UMA):
    1 memory pool for cores, same access speed for any memory address, but bandwidth is shared
  - Non-uniform memory access (NUMA):
    multiple pools, access speed depends on “distance”

An HPC Cluster is...

- A cluster needs:
  - Several computers, nodes, hardware similar to a workstation, but in special cases for rack mounting
  - One or more networks (interconnects) for inter-node communication and accessing common resources
  - Software that allows the nodes to communicate with each other (usually via calls to an MPI library)
  - Software that reserves resources to individual users
- A cluster is: all of those components working together to form one parallel computing device

What is Grid Computing?

- Loosely coupled network of compute resources
- Needs a "middleware" for transparent access for inhomogeneous resources
- Modeled after power grid
  => share resources not needed right now
- Run a global authentication framework
  => Globus, Unicore, Condor, Boinc
- Run an application specific client
  => SETI@home, Folding@home

What is Cloud Computing?

- Simplified: “Grid computing made easy”
- Grid: use “job description” to match calculation request to a suitable available host, use “distinguished name” to uniquely identify users, opportunistic resource management
- Cloud: provide virtual server instance on shared resource as needed with custom OS image, commercialization (cloud service providers, dedicated or spare server resources), physical location flexible, distributed/replicated storage

What is Supercomputing (SC)?

- The most visible manifestation of HPC
- Programs run on the fastest and largest computers in the world (=> Top500 List)
- Desktop vs. Supercomputer in 2013 (peak):
  - Desktop/Workstation: 10s of GigaFLOP/s
  - High-end GPU: >1 TeraFLOP/s
  - Top 10 Supercomputer: 10s of PetaFLOP/s
- Sustained vs. Peak: x86 CPUs -> 90-95%, BlueGene -> ~85%, CPU+Accelerator -> ~65%

TOP 500 Linux Cluster / 2002

TOP500 Supercomputer sites: TOP500 List 11/2002 - Mozilla {Build ID: 2002101717}

File Edit View Go Bookmarks Tools Window Help Debug QA

Back Forward Reload Stop http://www.top500.org/dlist/2002/11/ Search Print

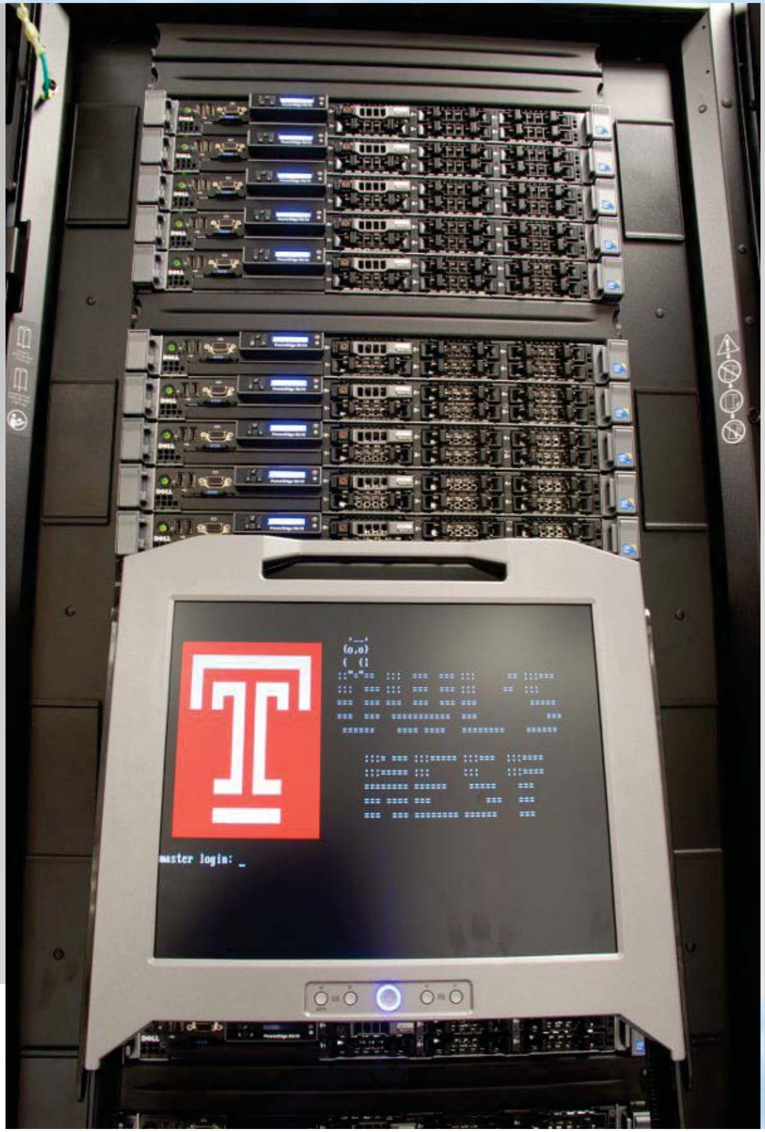
			USA/1999			MHz
298	**NEC** SX-5/32M2/ 32	**241.40** 256.00	**Bureau of Meteorology / CSIRO HPCCC** Australia/2000	**Research** Weather and Climate Research		**NEC Vector** SX5
299	**NEC** SX-5/32M2/ 32	**241.40** 256.00	**Meteorological Service of Canada (MSC)** Canada/1999	**Research** Weather and Climate Research		**NEC Vector** SX5
300	**NEC** SX-5/32H2/ 32	**241.40** 256.00	**National Research Institute for Metals** Japan/2000	**Research**		**NEC Vector** SX5
301	**IBM** LosLobos/ 512	**237.00** 375.00	**University of New Mexico** USA/2000	**Academic**	**150000** 20000	**NOW - Intel** NOW Cluster - Intel
302	**Megware** AMD AthlonMP1600+ - SCI 2D-Torus/ 128	**235.80** 358.40	**Theoretical Chemistry, Ruhr-University Bochum** Germany/2002	**Academic**	**103200** 17910	**NOW - AMD** NOW Cluster - AMD
303	**Cray Inc.** T3E1200/ 284	**235.00** 340.80	**Government** USA/1999	**Classified**		**T3E/T3D** T3E1200
304	**Megware** HELICS AMD 1.7 GHz - Myrinet/ 132	**234.80** 448.80	**Massey University Albany, IIMS & The Allan Wilson Centre** New Zealand/2002	**Academic**	**82080** 25000	**NOW - AMD** NOW Cluster - AMD
305	**IBM** pSeries 690 Turbo 1.3GHz GigEth/ 96	**234.00** 499.20	**Ahold** USA/2002	**Industry** Database		**IBM SP** pSeries 690 Turbo 1.3 GHz GigE
306	**IBM** pSeries 690 Turbo 1.3GHz GigEth/ 96	**234.00** 499.20	**BOUYGTEL** France/2002	**Industry** Telecomm		**IBM SP** pSeries 690 Turbo 1.3 GHz GigE
307	**IBM** pSeries 690 Turbo 1.3GHz GigEth/ 96	**234.00** 499.20	**Boeing** USA/2002	**Industry** Aerospace		**IBM SP** pSeries 690 Turbo 1.3 GHz GigE

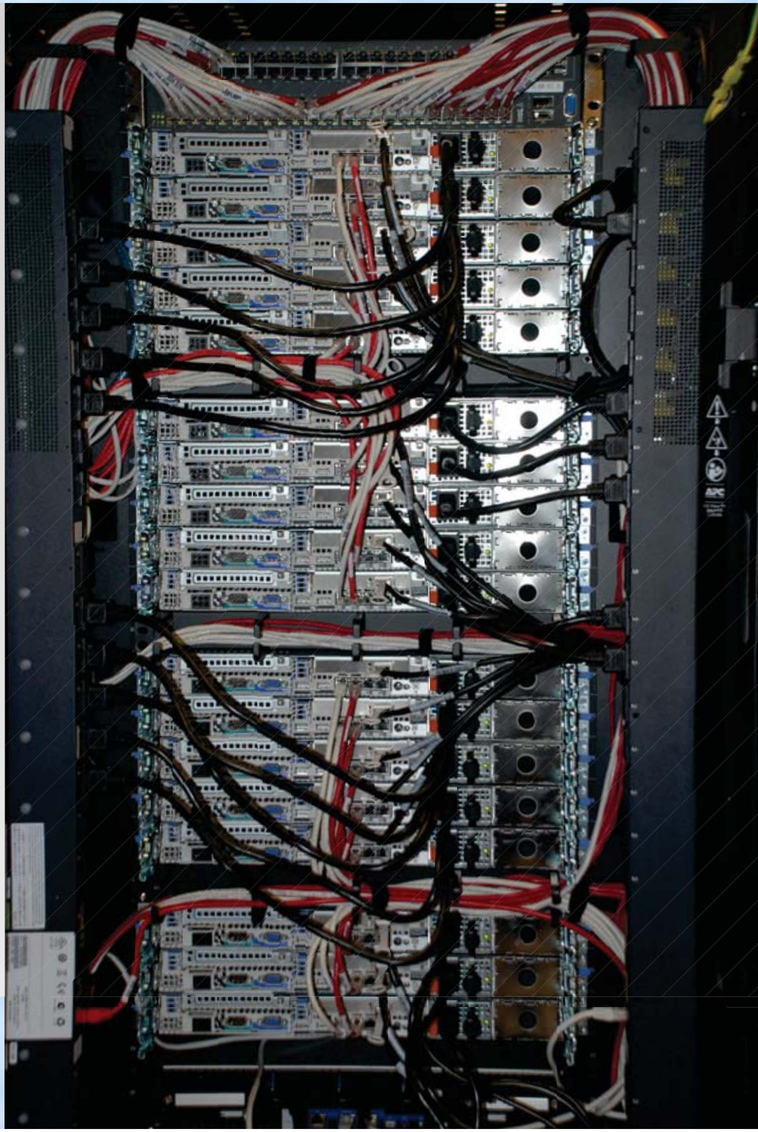
HPC Cluster 2002 / The Good

The Abdus Salam
International Centre
for Theoretical Physics

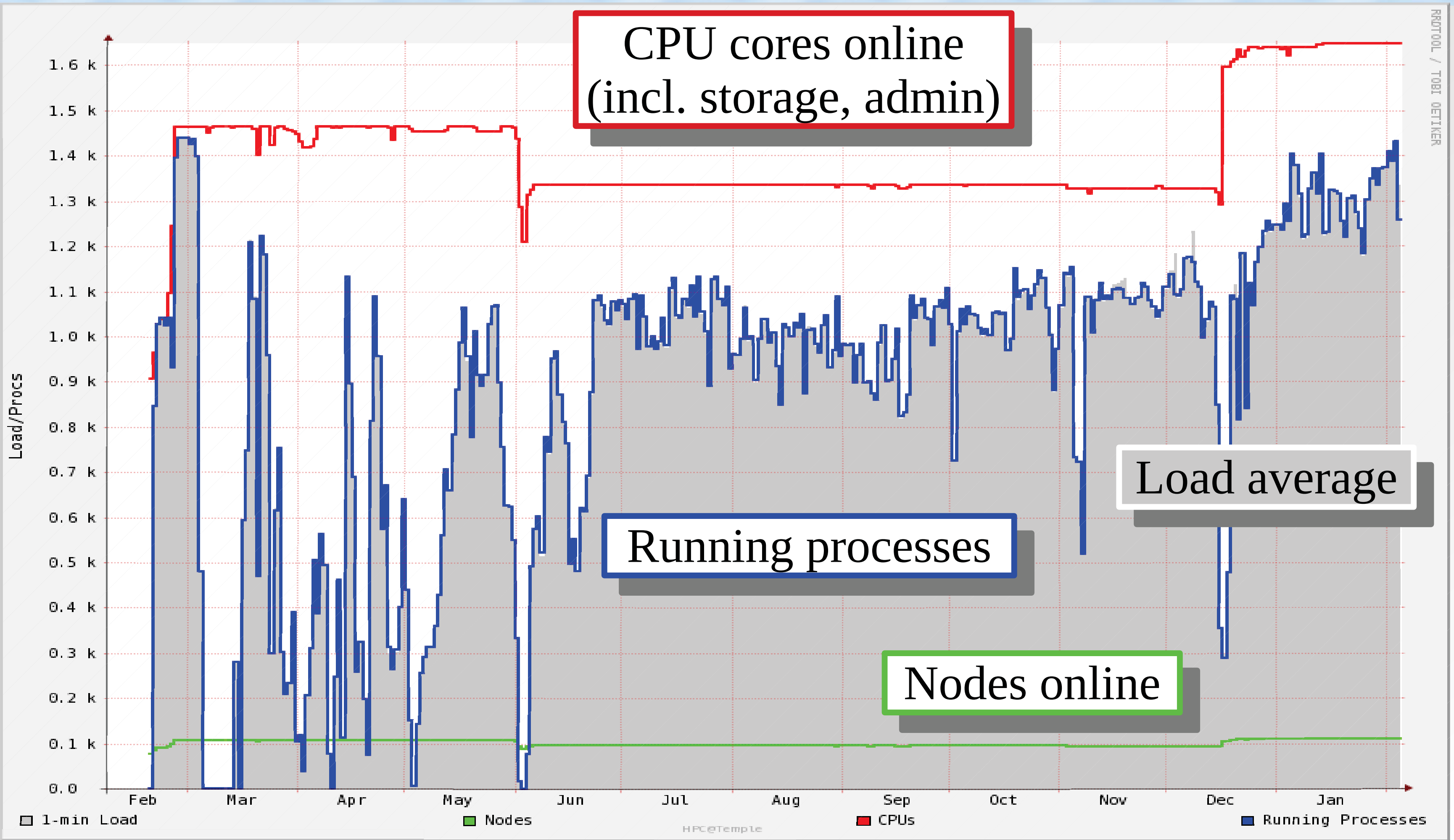
HPC Cluster in 2002 / The Ugly

HPC Cluster in 2012





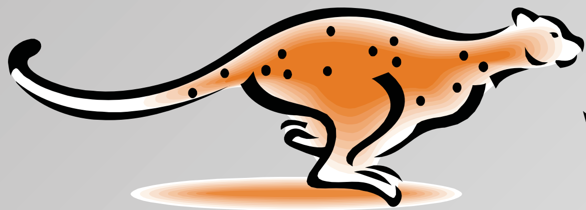
Cluster Deployment and Usage

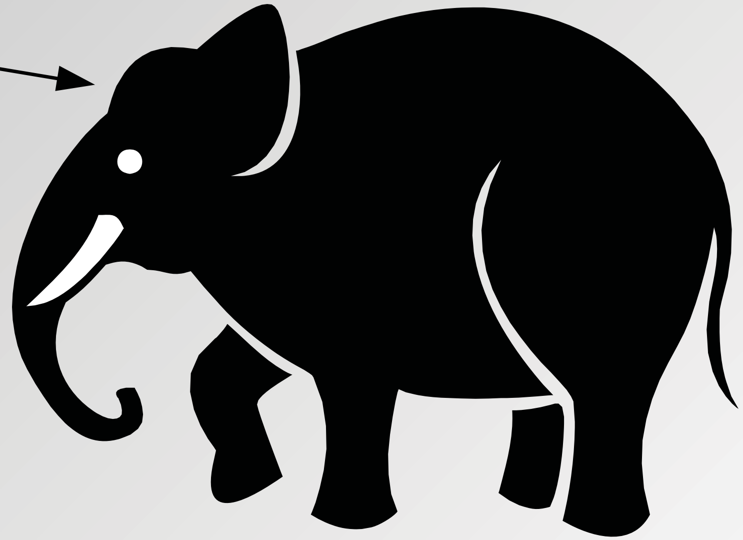


The Abdus Salam International Centre for Theoretical Physics

Who needs HPC?

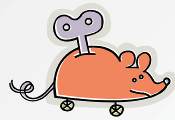
- My problem is big





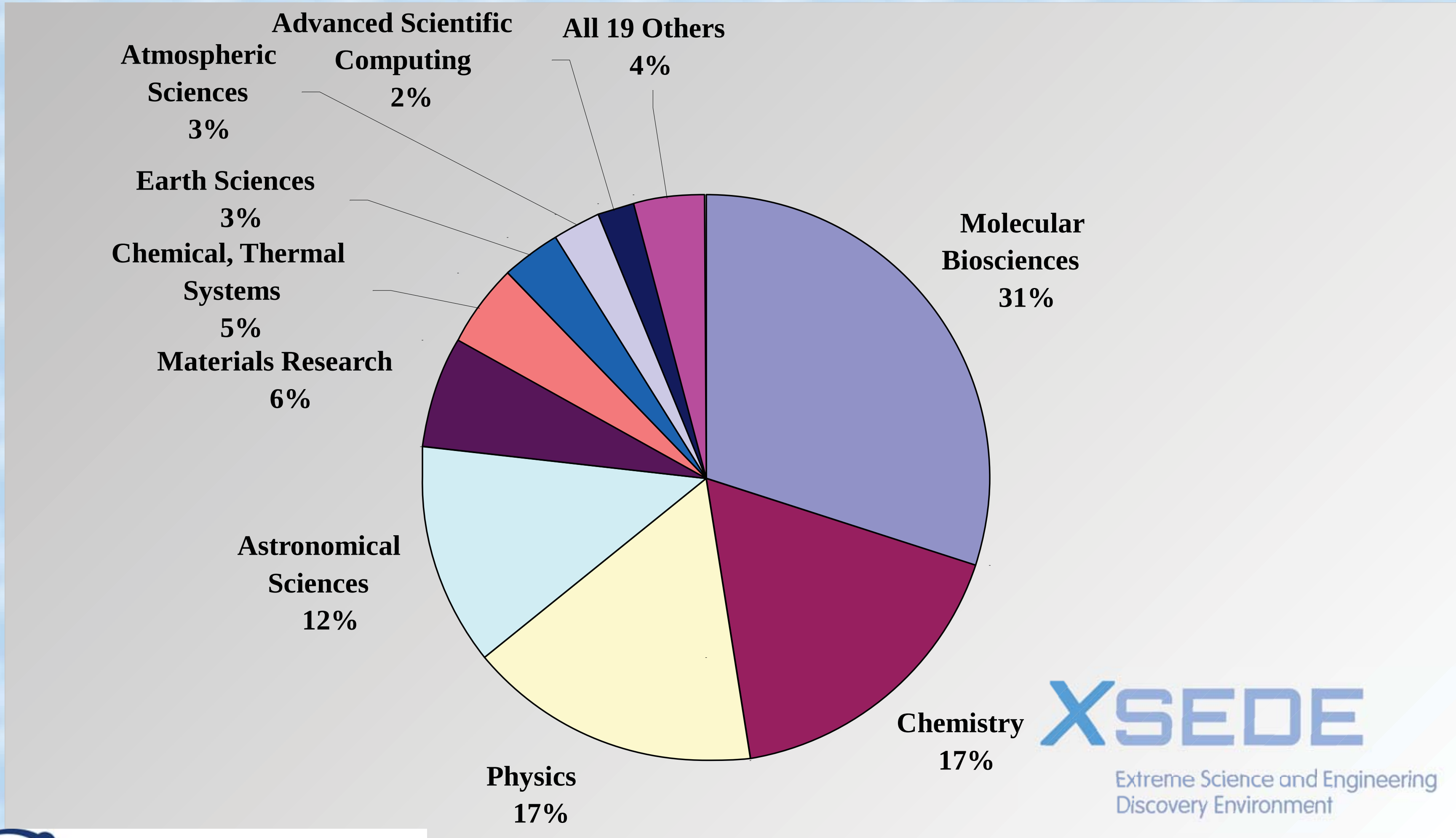
- My problem is complex

- My computer is too small and too slow



- My software is not efficient and/or not parallel

Research Disciplines in HPC



Why would HPC matter to you?

- Scientific computing is becoming more important in many research disciplines
- Problems become more complex, needs complex software and thus teams of researchers with diverse expertise (including expertise in deploying and operating clusters)
- HPC hardware is more complex, application performance depends on many factors
- Scientific (HPC) application development limited often limited by lack of training

HPC vs. Computer Science

- Most people in HPC are not computer scientists
- Software has to be correct (first) and (then) efficient; packages can be over 30 years "old"
- Technology is a mix of "high-end" & "stone age" (Extreme hardware, MPI, Fortran, C/C++)
- So what skills do I need to for HPC:
  - Common sense, cross-discipline perspective
  - Good understanding of calculus and (some) physics
  - Patience and creativity, ability to deal with "jargon"

The Abdus Salam International Centre for Theoretical Physics

HPC is a Pragmatic Discipline

- Raw performance is not always what matters: *how long does it take me to get an answer?*
- HPC is more like a craft than a science:
  => practical experience is most important
  => leveraging existing solutions is preferred over inventing new ones requiring rewrites
  => a good solution today is worth more than a better solution tomorrow
  => **but** a readable and maintainable solution is better than a complicated one

The Abdus Salam International Centre for Theoretical Physics

How to Get My Answers Faster?

- Work harder
  => get faster hardware (get more funding)
- Work smarter
  => use optimized algorithms (libraries!)
  => write faster code (adapt to match hardware)
  => trade convenience for performance
  (e.g. compiled program vs. script program)
- Delegate parts of the work
  => parallelize code, (grid/batch computing)
  => use accelerators (GPU/MIC CUDA/OpenCL)

What Determines Performance?

- How fast is my CPU?
- How fast can I move data around?
- How well can I split work into pieces?
  Very application specific:
  => never assume that a good solution for one problem is as good a solution for another
  => always run benchmarks to understand requirements of your applications and properties of your hardware
  => respect Amdahl's law

How Do We Measure Performance?

- For numerical operations: FLOP/s (or FLOPS) = Floating-Point Operations per second
- Theoretical maximum (**peak**) performance: clock rate x number of double precision addition and/or multiplications completed per clock
  => 2.5 Ghz x 4 FLOP/clock = 10 GigaFLOP/s
  => can never be reached (data load/store)
- Real (**sustained**) performance:
  => very application dependent
  => Top500 uses Linpack (linear algebra)

The Abdus Salam International Centre for Theoretical Physics

Fast and Slow Operations

- Fast (6): add, multiply (0.5 with pipelining)
- Medium (40): divide, modulus, sqrt()
- Slow (300): most transcendental functions
- Very slow (1000): power (x^y for real x <u>and</u> y)

Often only the fastest operations are pipelined, so code will be the fastest when using only add and multiply => linear algebra, tabulation
=> BLAS (= Basic Linear Algebra Subroutines)
plus LAPACK (Linear Algebra Package)

Software Optimization

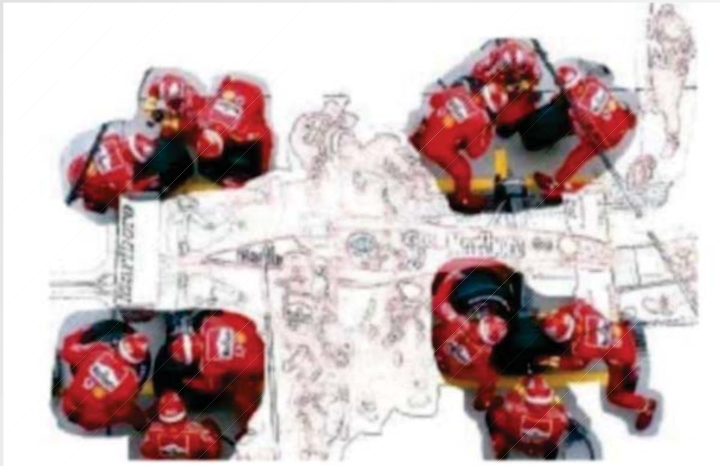
- Writing maximally efficient code is hard:
  => most of the time it will not be executed exactly as programmed, not even for assembly
- Maximally efficient code is not very portable:
  => cache sizes, pipeline depth, registers, instruction set will be different between CPUs
- Compilers are smart (but not too smart!) and can do the dirty work for us, but can get fooled

  => modular programming: generic code for most of the work plus well optimized kernels

A High-Performance Problem

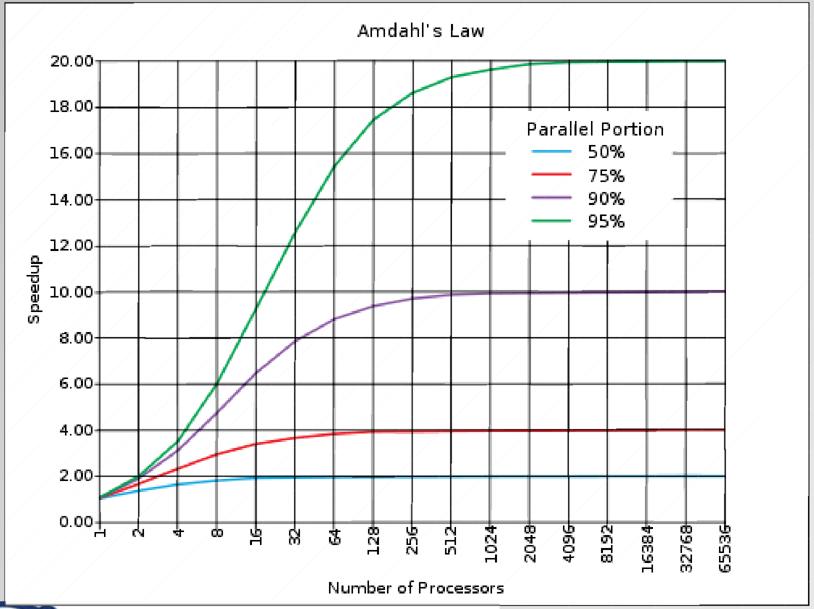
Two Types of Parallelism

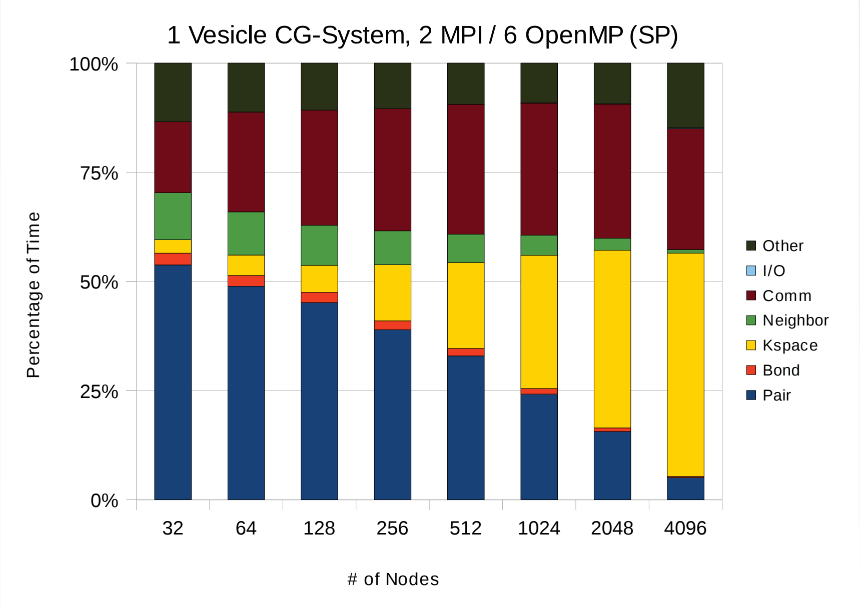
- Functional parallelism: different people are performing different tasks at the same time
- Data parallelism: different people are performing the same task, but on different equivalent and independent objects



Amdahl's Law vs. Real Life

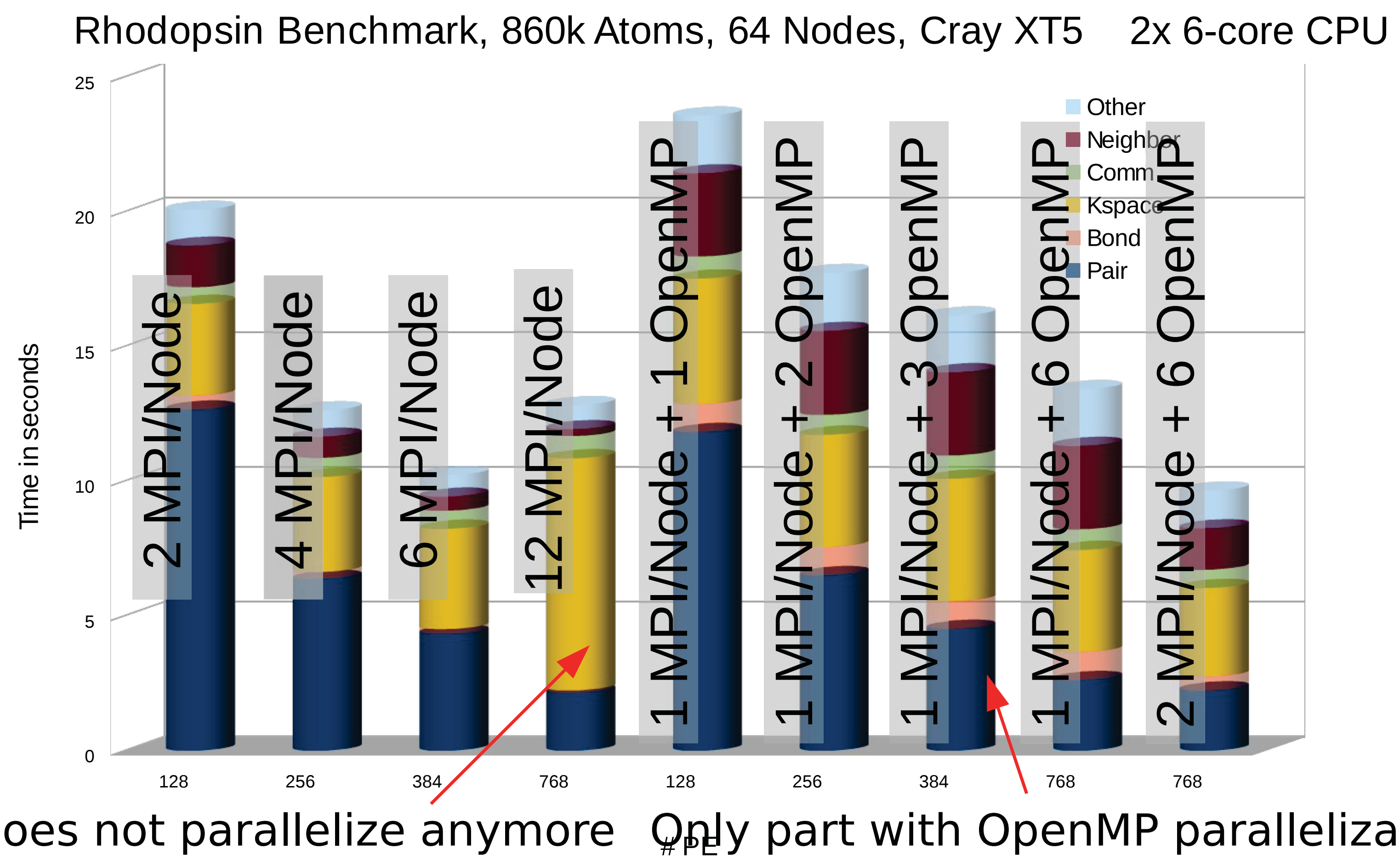
- The speedup of a parallel program is limited by the sequential fraction of the program.
- This assumes perfect parallel scaling without overhead
- **But:** different parts of a program parallelize differently well





The Abdus Salam International Centre for Theoretical Physics

Performance within an Application

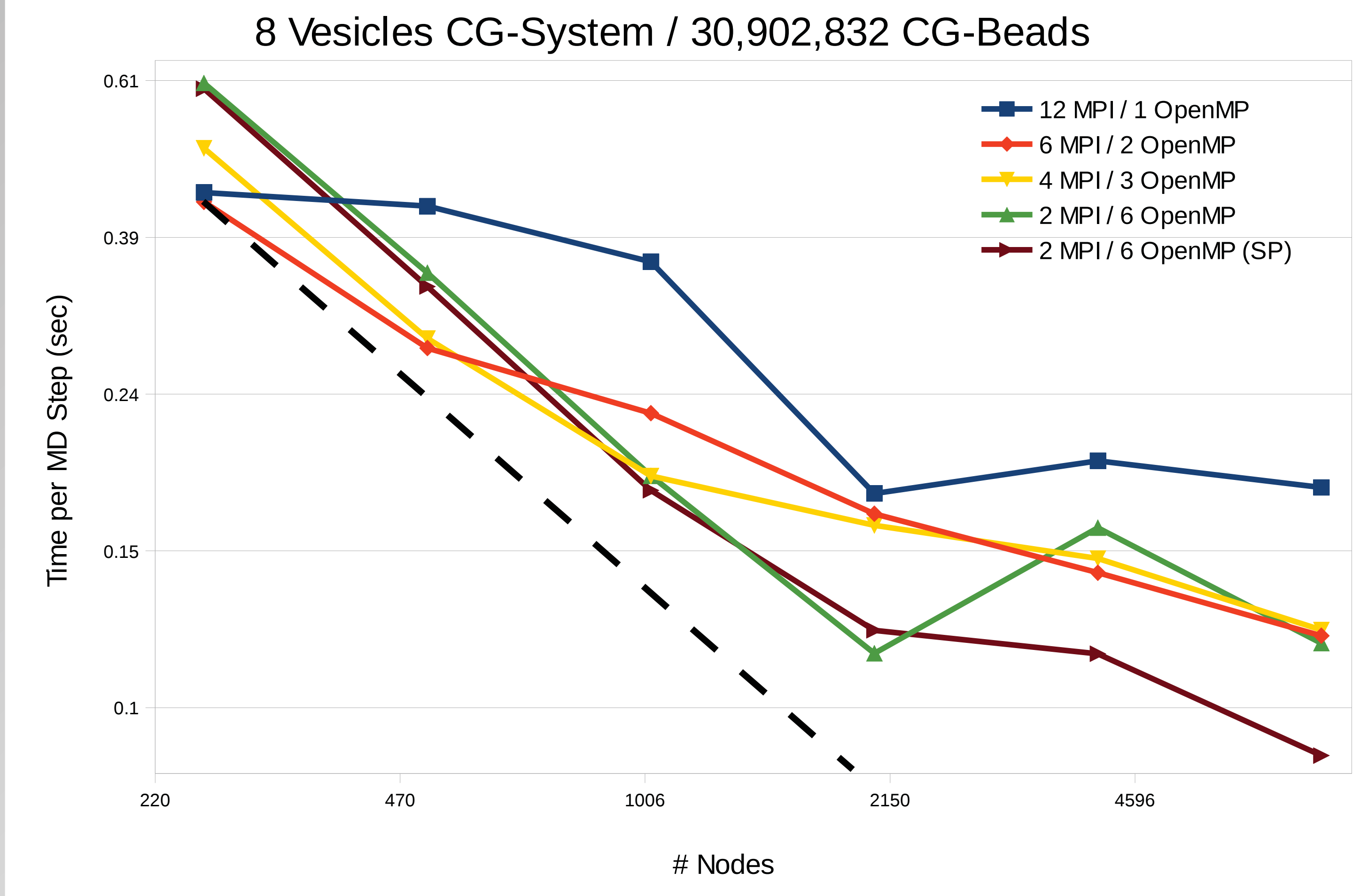


Does not parallelize anymore Only part with OpenMP parallelization

Performance of SC Applications

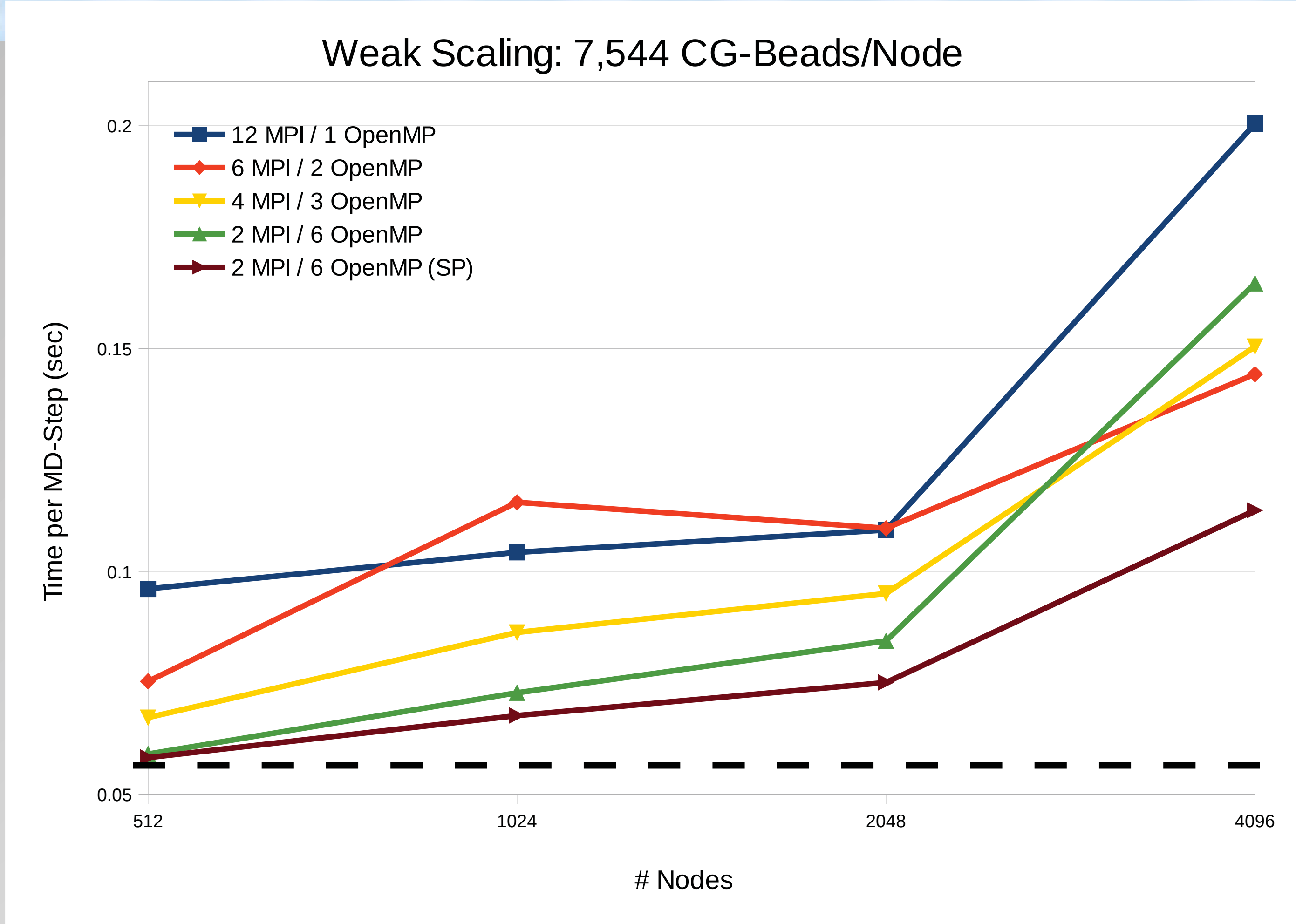
- Strong scaling: fixed data/problem set; measure speedup with more processors
- Weak scaling: data/problem set increases with more processors; measure if speed is same
- Linpack benchmark: weak scaling test, more efficient with more memory => 60-95% peak
- Climate modeling (WRF): strong scaling test, work distribution limited, needs high I/O and memory bandwidth, serial overhead: < 5% peak

Strong Scaling Graph



The Abdus Salam
International Centre
for Theoretical Physics

Weak Scaling Graph



How to Choose Hardware

- Benchmarks, benchmarks, benchmarks
- Performance is application specific:
  => learn how important applications behave
- Resource usage differs between applications:
  => learn which are the dominant users
- Extreme choices are high risk, look for balance
  => optimize common case, avoid bottlenecks
- Most clusters run a mix of job sizes
  => no need to have a homogeneous cluster

The Abdus Salam
International Centre
for Theoretical Physics

Interacting with Vendors

- Typically you interact with sales staff
  => they are selected for being good at selling
  => they do not run jobs or operate clusters
  => they don't know specifics of applications
  => their primary task is to make a sale
  => IT thrives on the promise of new hardware
  => most products are designed for businesses
  => service contracts are tailored for businesses
- You have to be specific about your needs; the better your knowledge, the better your position

Interacting with Scientists

- Scientists often do not (want to) know hardware
  => provide examples how to use applications
- Scientists get confused about technical jargon
  => offer simple choices, do not overconfigure
  => take care of common cases, not exceptions
- Scientists often struggle to describe problems
  => do not take everything literally
  => difficult to tell software from hardware issues
  => many HPC applications do not handle bad input gracefully for performance reasons

MPI versus OpenMP

- Which is “better”?
  => most parallel applications are written for MPI which has been around for longer
  => OpenMP is often added later to add fine grained parallelism to coarse MPI parallelism
  => MPI works best on distributed data and adapts to NUMA architectures (share nothing)
  => OpenMP avoids data replication and can thus improve cache utilization vs. MPI
  => on modern multi-socket, multi-core hardware you need to use both in concert

Accelerators / GPUs

- The sales pitch: accelerators are >100x faster
- The reality:
  - Typical comparison against a single CPU core
  - Accelerators require many work units
    => acceleration becomes more difficult on a cluster
  - Accelerators require significant code changes
    => not many scientific applications support them yet
  - Accelerators are subject to Amdahl's law
    => speedup is limited by the non-accelerated part
  - Accelerators benefit from small/simple kernels