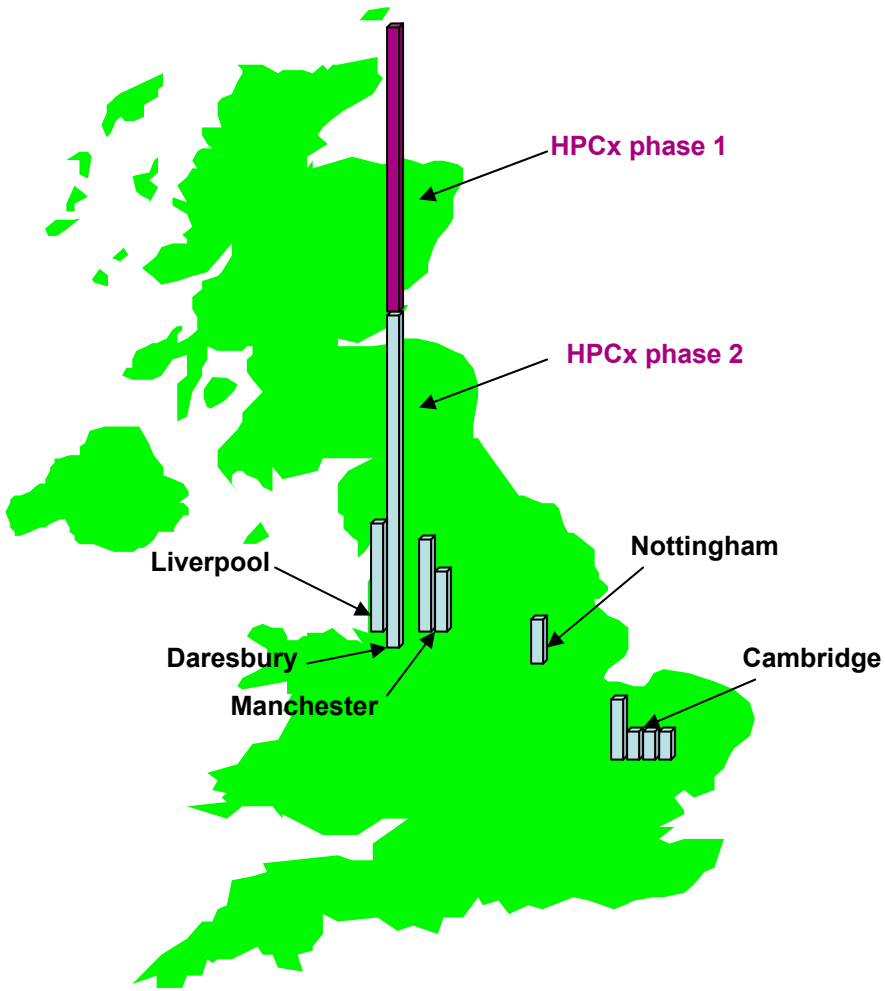
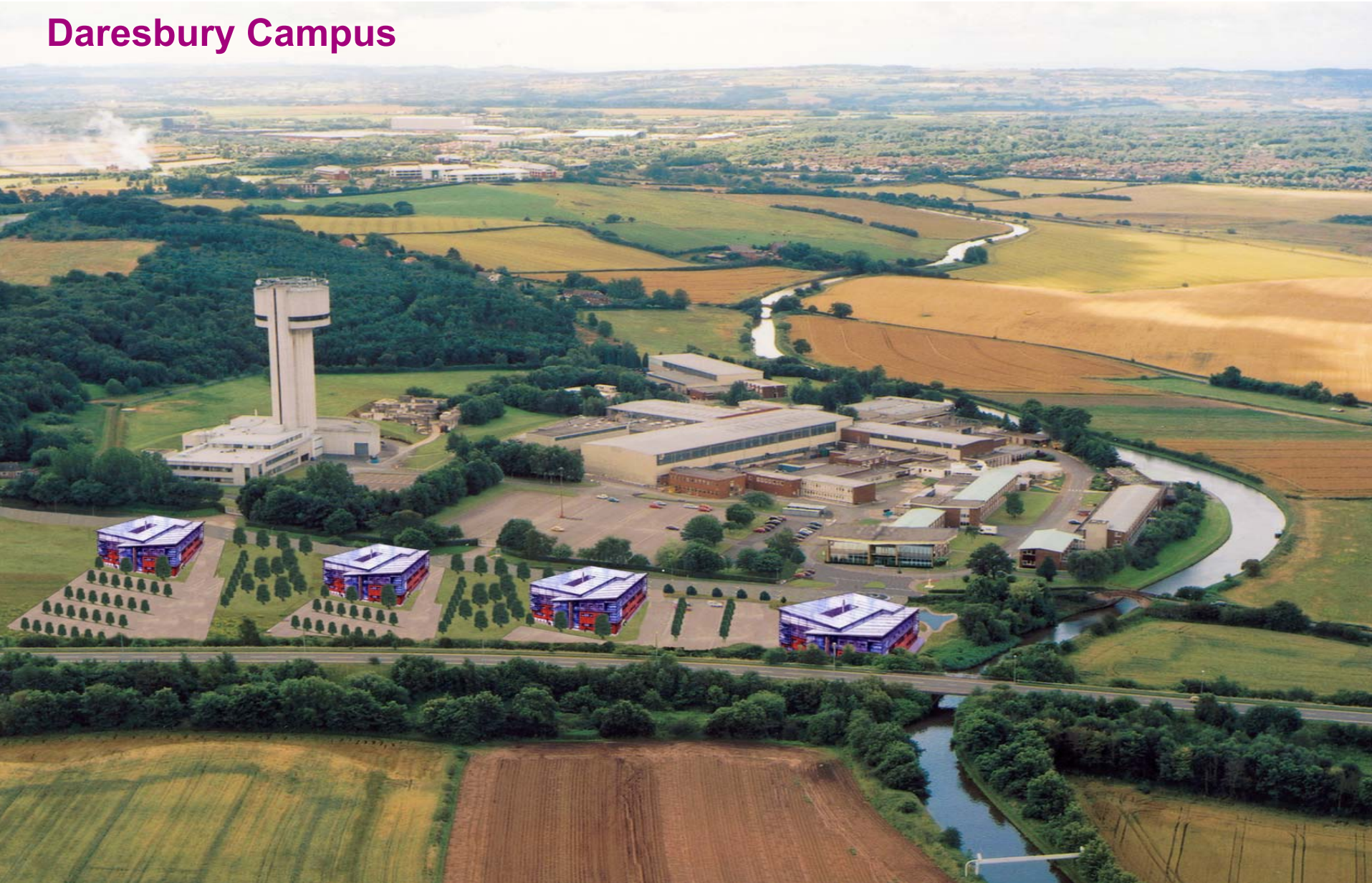
Parallel Algorithms on a cluster of PCs

Ian Bush
Computational Science & Engineering Department
Daresbury Laboratory
I.J.Bush@dl.ac.uk


HPCx phase 1
HPCx phase 2
Liverpool
Nottingham
Daresbury
Cambridge
Manchester

CSE Computational Science & Engineering Department

Daresbury Campus

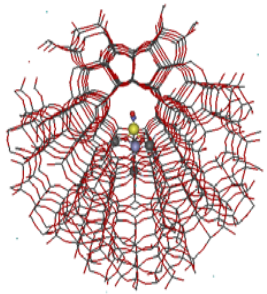


Computational Science and Engineering in a Nutshell

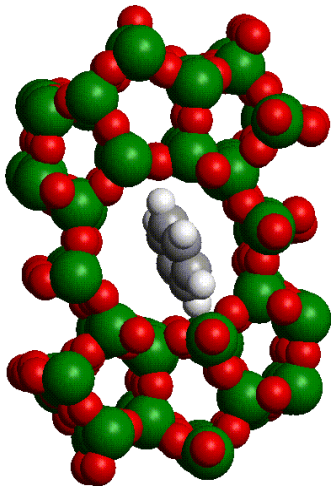
- Do theory, make mathematical models
- Write programs
- Use programs to do science
- Make programs available to university colleagues and support them
- Make programs run quickly on the fastest machines around
- Operate some of the fastest machines around - HPCx - as a facility for University researchers

We're solving the Schrodinger equation, Newton's equations, Navier-Stokes equations, etc for ever more complicated and realistic models.

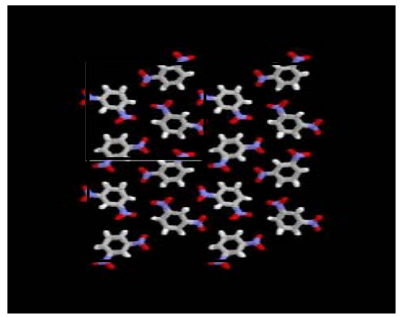
Materials Science and Condensed Matter



Modelling active sites catalysts using combine QM/MM methods to design more specific, more environmentally friendly, more active systems capable of working at lower temperatures and pressures.

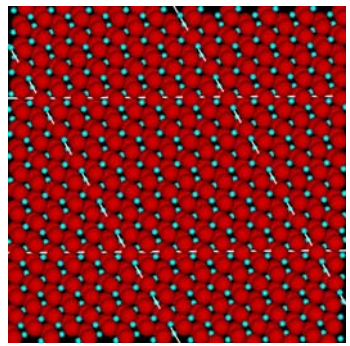


Slow diffusion!
Bluemoon method
Fixed and flexible framework
Reaction path found
Free energy profiles
MC method for D_0



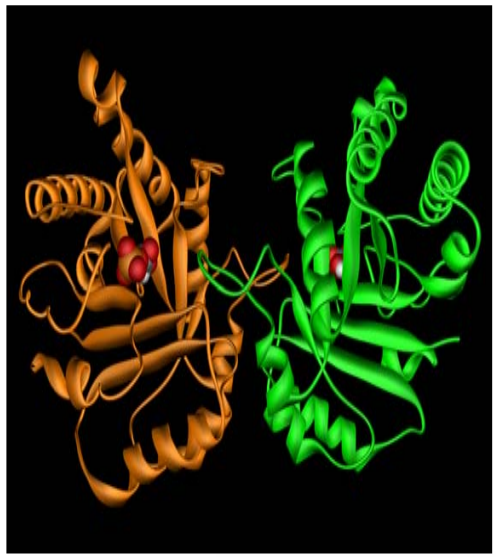
Accurate modelling of molecular interactions for crystal structure prediction

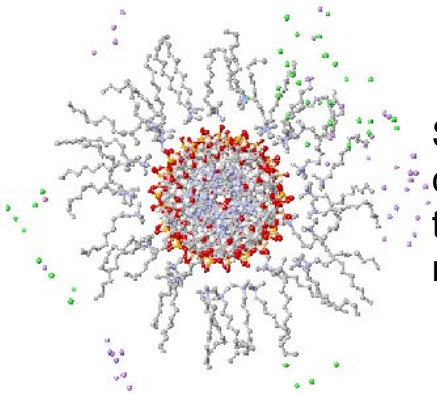
(5x5) Cr_2O_3 surface containing 700 atoms (14850 basis functions) per unit cell. Such calculations are essential in order to study the role in defects in determining the properties of real materials



Life Sciences

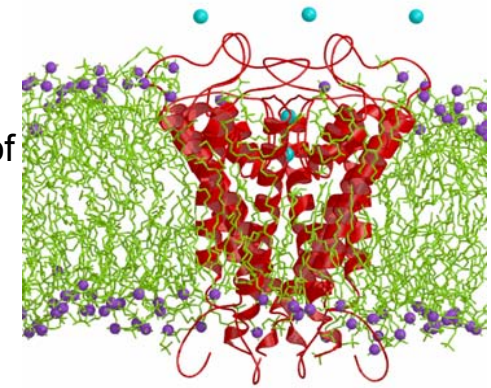
Modelling active sites of enzymes in solvents at room temperature - simulation of TIM at 300K to study mechanism of active site



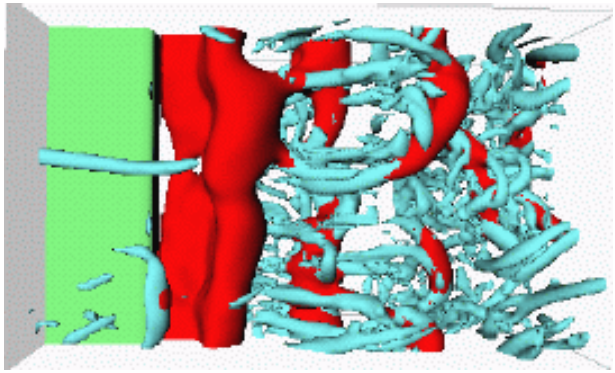


Simulations of liposomes coating DNA strands prior to transport across cell membranes

Virtual Outer Membrane - molecular dynamics simulations of transport channels through membranes require 2 million atoms to be modelled for 100 ns with multiple comparative runs to generate statistics (Mark Sansom)

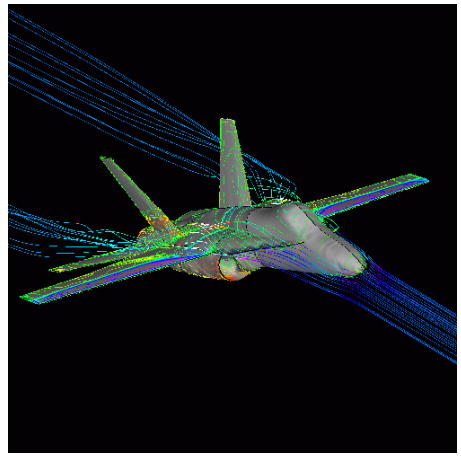


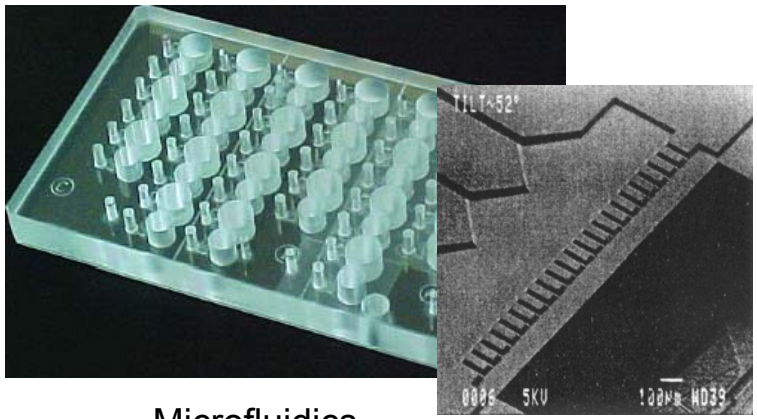
Computational Engineering



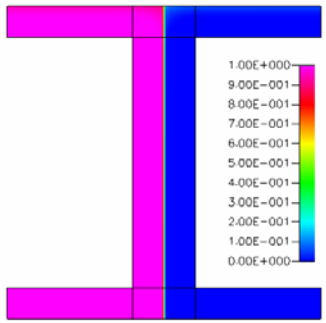
Modelling turbulence
(Neil Sandham)

Modelling complex geometries

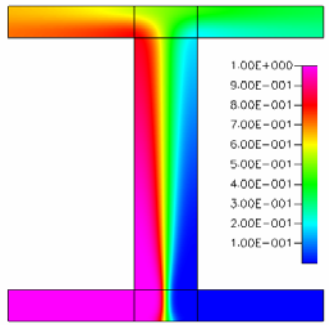




Microfluidics



Diffusion of red blood cells
$D=6.8\times10^{-10}$ cm^2/s



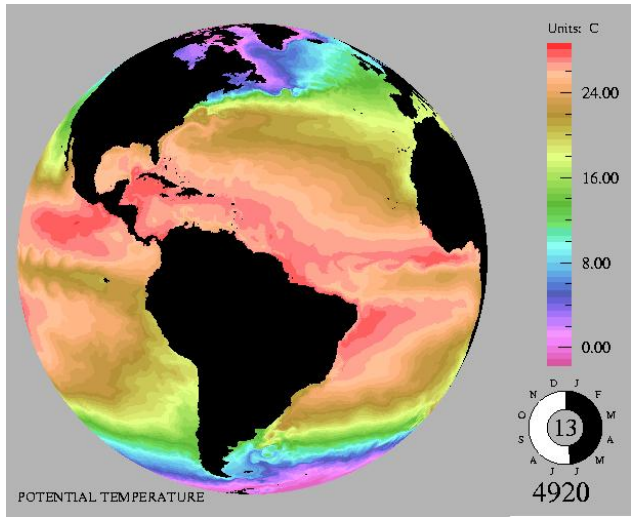
Diffusion of serum albumin
$D=6.5\times10^{-7}$ cm^2/s

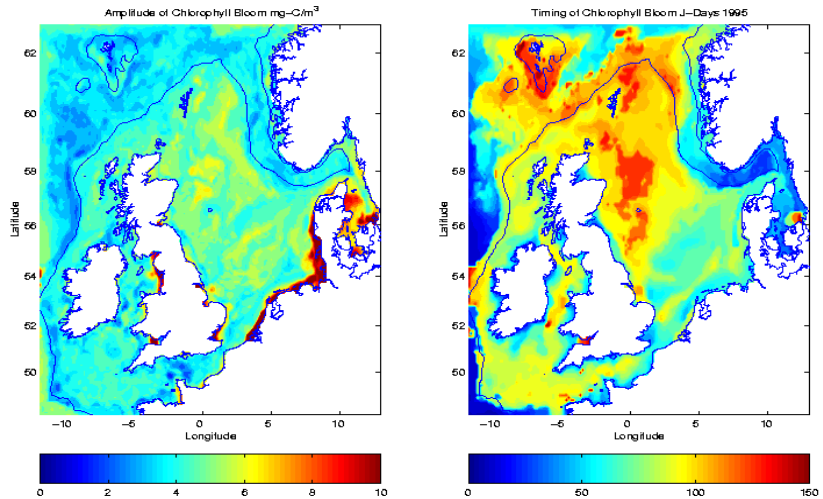
Ocean and Climate Models

- A 1/12º Ocean Model
  - **has 608 million grid cells**
  - **needs 60 Gbyte storage**
  - **needs 40 x 10^{15} floating point operations/model year**
  - **produces a 20 Gbyte data set every 3 model days**

- A comparative climate model with ocean, atmosphere and land sub-models needs about twice the resources.

- Greenhouse effect, raised CO_2 emissions, ozone depletion, storm and gulf stream variability, regional shelf edge models, biological sub-models





CCLRC

Some Application Codes

- GAMESS-UK: general purpose ab initio molecular electronic structure program for performing SCF-, MCSCF- and DFT-gradient calculations, together with a variety of techniques for post Hartree Fock calculations. Integrated QM/MM modelling. (140 user groups)
- DL_POLY: general purpose molecular dynamics code, replicated/ distributed data, NVT, NPT, NST, NVE thermodynamic ensembles, multiple time stepping and RESPA algorithms, Ewald summation for electro-statics. (350 users)
- CRYSTAL: 1D, 2D, 3D periodic Gaussian, Hartree-Fock, total energy, forces. (330 users)
- CASTEP: The Cambridge Serial Total Energy Package plane wave DFT calculation of the total energy, forces and stresses in a 3D-periodic system. (150 users)
- Plus lots of others with no names ….

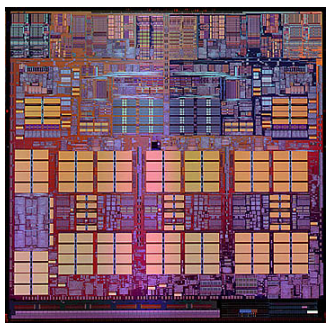
The HPC*x* Project

- Operated by University of Edinburgh and Daresbury
- First Tera-scale research computing facility in the UK
- Largest academic HEC facility in Europe; 2nd largest in the world
- Funded by EPSRC, NERC, BBSRC
- Located at Daresbury from October 2002
- IBM is the technology partner
  - Power4-based Regatta systems
  - Year 0: 3 Tflops sustained (on Linpack) - 1280 cpus
  - Year 2: 6 Tflops sustained - 1600 cpus
  - Year 4: 12 Tflops sustained

Vital Statistics of HPCx

Phase 1, Phase2 and possible Phase 3 configurations

	Phase 1	Phase 2	Phase 3
Start date	December 2002	April 2004	October 2006
Processors	1280 Power4 cpus	1600 Power4+ cpus	3072 Power4+ cpus
Peak speed	6.7 TFlops	10.9 TFlops	
Sustained speed	3.2 TFlops	6.2 TFlops	12 Tflops
Switch	Colony	Federation	Federation
Memory	1.28 TBytes	1.6 TBytes	3.072 TBytes
Disk	18 TBytes	36 TBytes	72 TBytes
Storage	35 TBytes	70 TBytes	140 TBytes
Power consumption	0.4 MWatts	0.5 MWatts	1 MWatt



Power 4 (aka Regatta H) chip

HPC*x* is a Really Big System

- In 1980, the faster supercomputer in the world was the Cray 1 (at Daresbury).
- In 2003, HPC*x* is faster than the Cray 1 by a factor of 45,578.
- Applying Moore's Law between 1980 and 2003 gives a factor of 2048.
- HPC*x* beats Moore's Law because it is ***parallel*** - many CPUs joined together.
- Programming a parallel computer for efficiency and speed (ie capability computing) is hard, because the ***traffic of data between the CPUs*** has to be managed carefully.
- A measure of parallel efficiency is scaling: does your program continue to run more quickly if you use more CPUs?
- *Not many real programs scale well enough ….*

But it's NOT all about 1000+ processor systems !

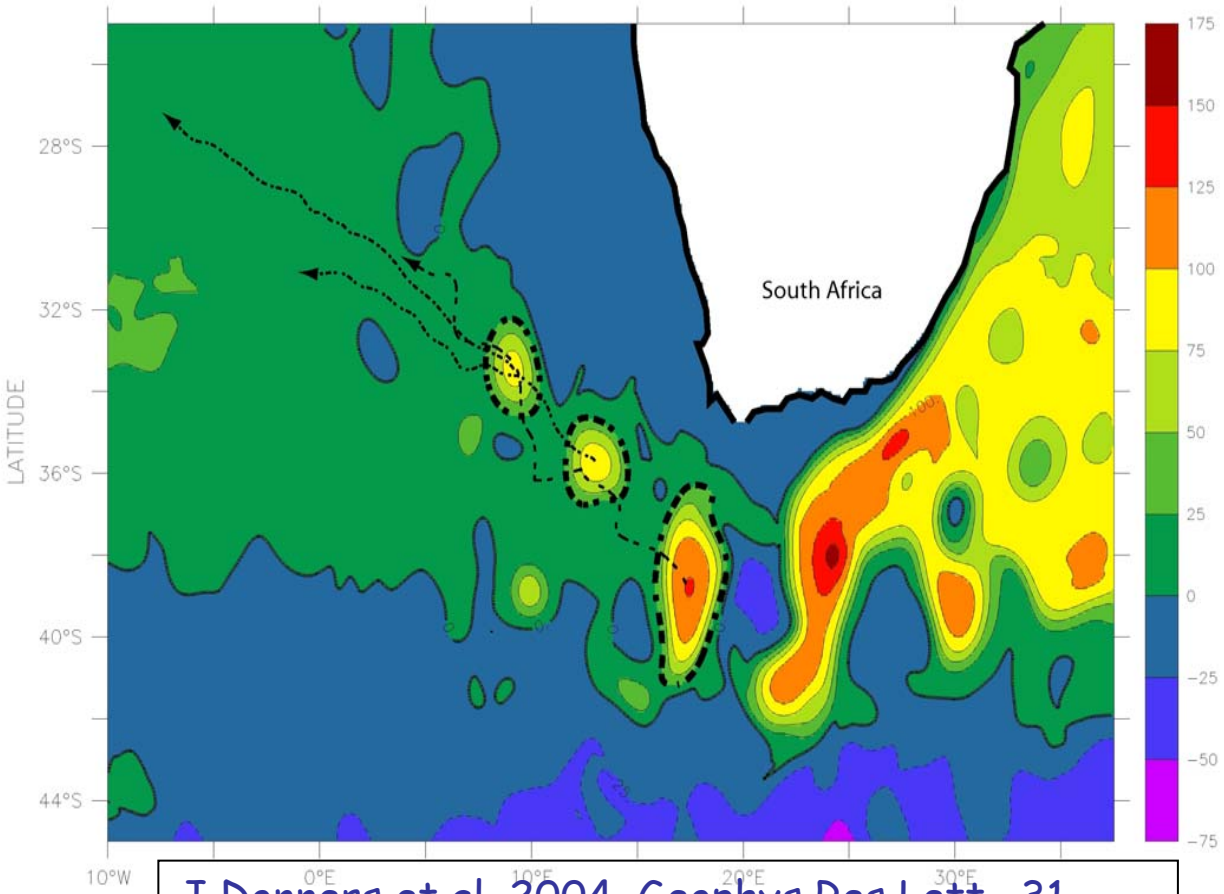
- We also run a number of small clusters (16-64 processors)
- These are a very cost effective way of performing computational science
  - **Just commodity parts so cheap**
  - **Interconnects can be as cheap as ethernet**
    - *Higher performance networks are available but cost more*
- However if it's two processors or 1000 the same principles apply !
  - **It's just the downside hits you harder !**
- Possible to get a very powerful resource for your group/department at a very reasonable cost, c.f. vector machines

So What Can We Do With Parallel Computing ?

- A quick run through one or two examples of what we are currently involved with

Long-Term Ocean Circulation

- Impact of cooling on the water mass exchange of Agulhas rings in a high resolution ocean model



J Donners et al, 2004, Geophys Res Lett., 31, L16312

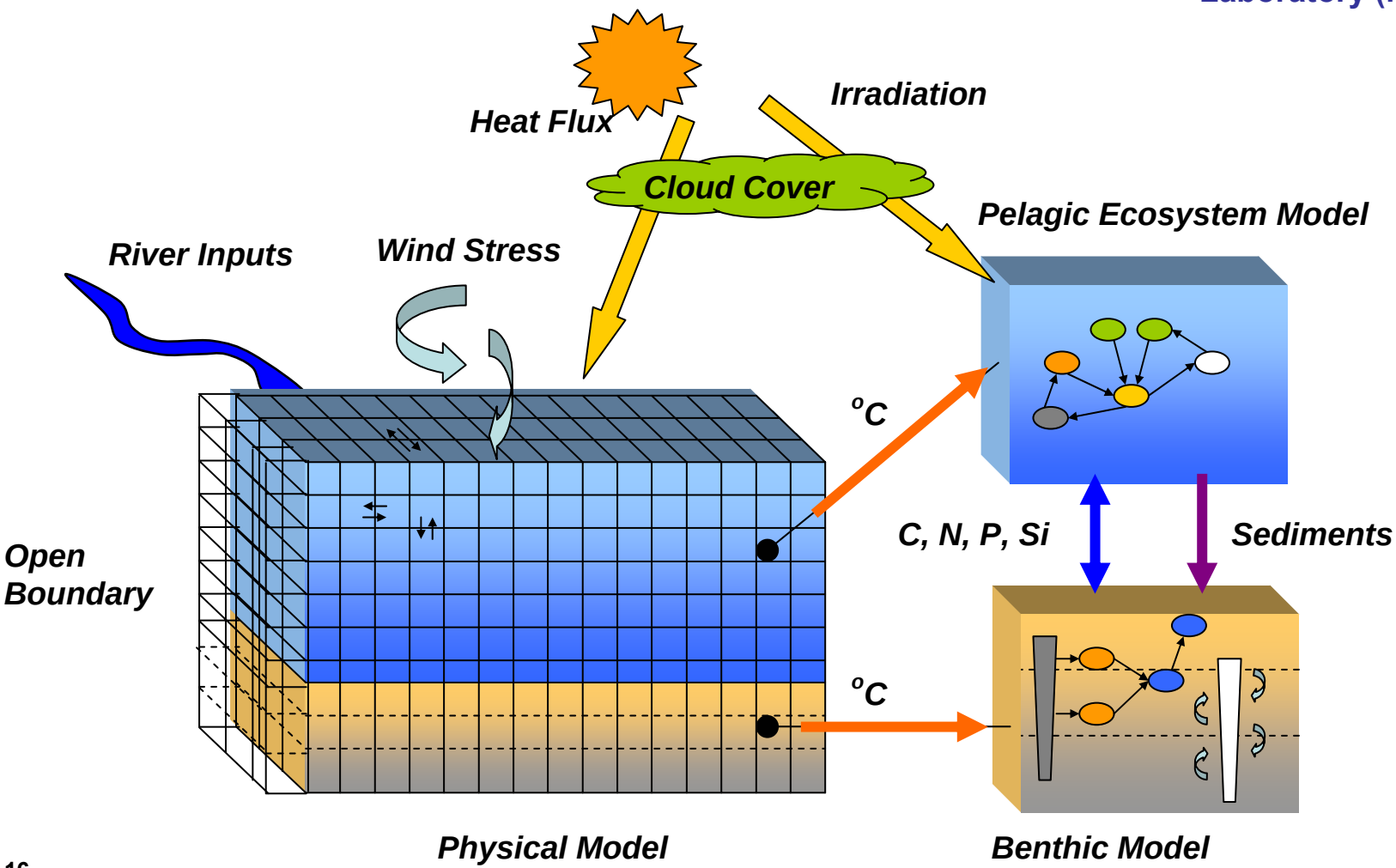
Vortex shedding has an important role in the global thermohaline circulation

The UK's global ocean model (OCCAM) is being run at an eddy-resolving resolution of 1/12°

The level of realism that can now be achieved is providing an invaluable compliment to observational programmes

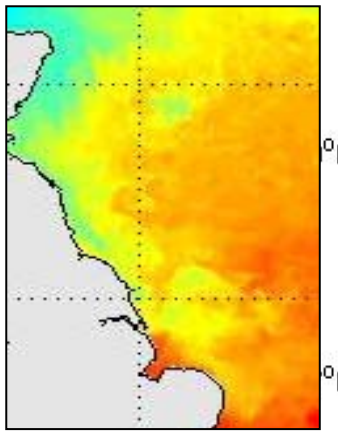
Coupled Marine Ecosystem Model (POLCOMS)

Proudman Oceanographic Laboratory (NERC)



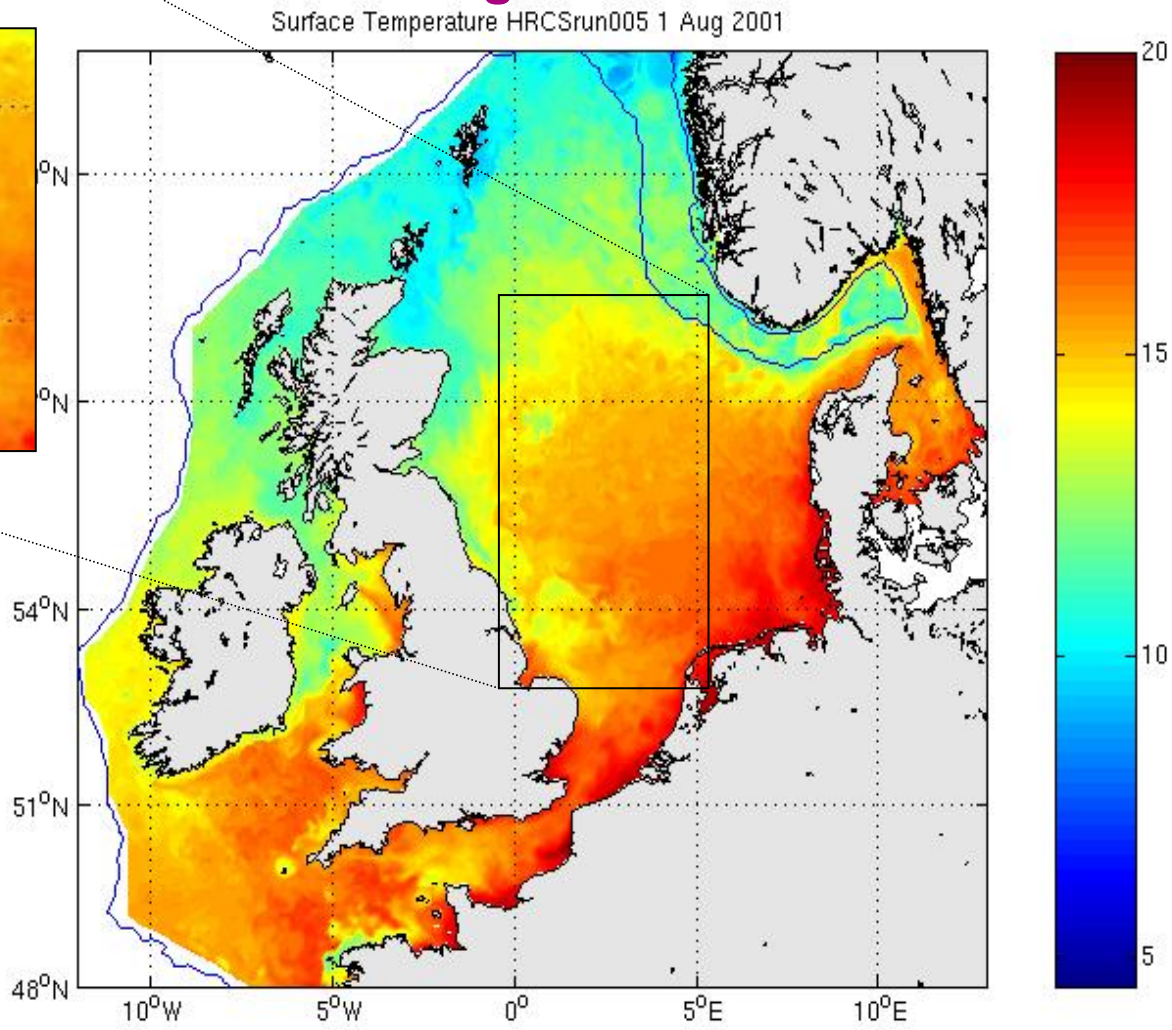
High-resolution coastal ocean modelling

POLCOMS code

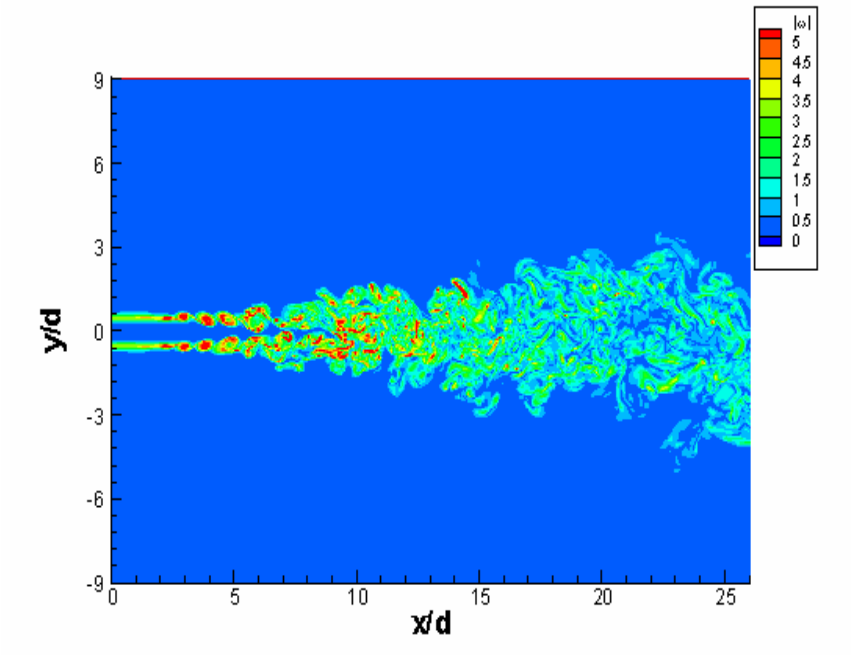


A 1.8km model of the NW european shelf seas is being run

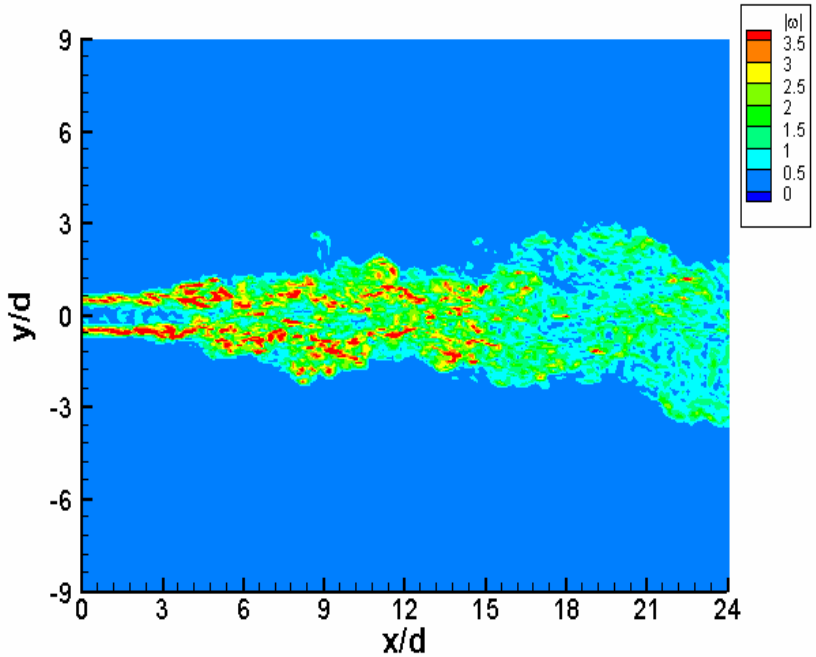
Early results show unprecedented detail in the resolution of fronts and eddies.



Inter-comparison of Direct Numerical Simulation and Large Eddy Simulation



DNS
721x351x256



LES
181x181x64

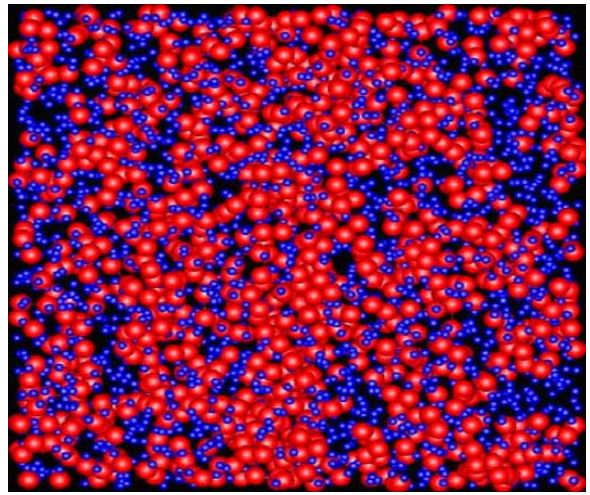
Yorke, C.P et al, European J.Mech. B – Fluids, **23**, 319, 2004.

Methanol-water solutions

Neutron diffraction experiments and MD simulations (DLPOLY) show that mixtures of methanol and water exhibit extended structures in solution despite the components being fully miscible in all proportions.

- Of particular interest is a concentration region where both methanol and water appear to form separate, percolating networks, a concentration range where many transport properties reach extremal values.
- The observed concentration dependence of several of these material properties of the solution may thus have a structural origin.

L. Dougan et al, J Chem Phys., 2004 (in press)



Front face of the MD simulation box for the methanol-rich solution (water oxygens- red; methyl carbons- blue).

The entire box comprises ~20000 atoms. A large water "super-cluster", containing over 400 molecules, can be seen running top to bottom of the middle of the box

From CO_2 to Methanol by QM/MM Embedding

Sam French, Alexey Sokol, Richard Catlow and Paul Sherwood

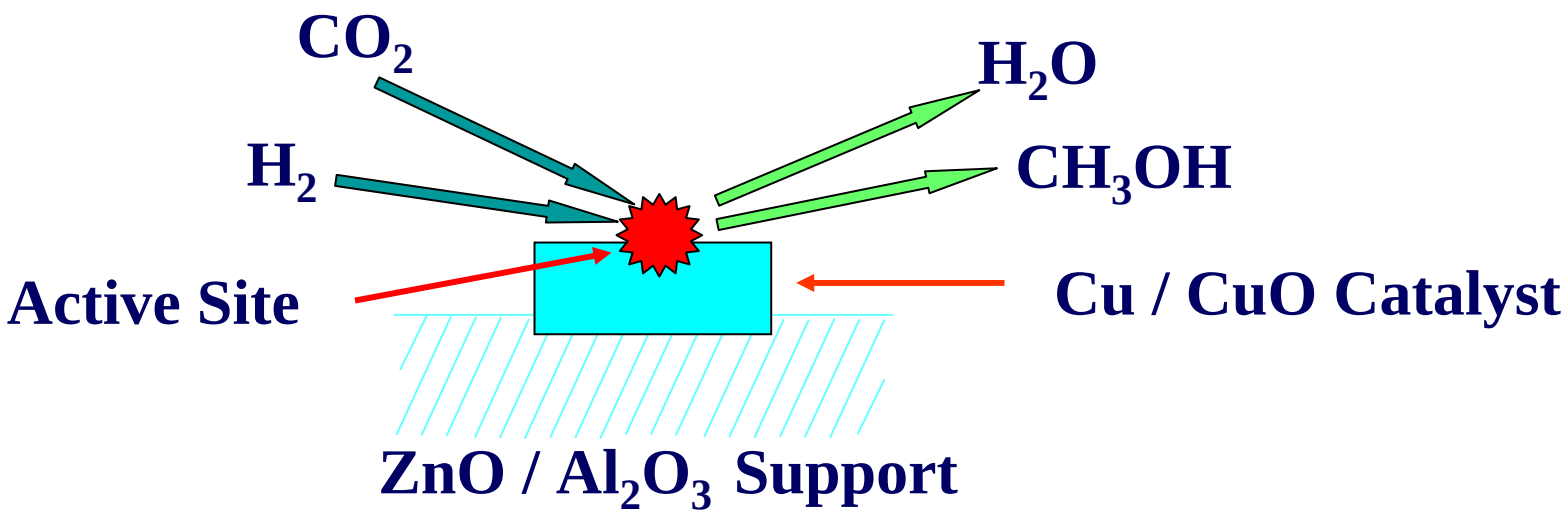
- **BASF 1923** **high pressure catalyst** **300 bar, < 300 °C**
  Zinc Oxide / Chromia
- **ICI 1965** **low pressure catalyst** **40-110 bar, 200–300 °C**
  Copper Oxide / Zinc Oxide / Alumina

Syngas →  → **Methanol**

From CO_2 to Methanol by QM/MM Embedding

Sam French, Alexey Sokol, Richard Catlow and Paul Sherwood

- **BASF 1923 high pressure catalyst 300 bar, < 300 °C**
  Zinc Oxide / Chromia
- **ICI 1965 low pressure catalyst 40-110 bar, 200–300 °C**
  Copper Oxide / Zinc Oxide / Alumina



$H_2(g) \rightarrow 2H(a)$

$CO_2(g) \rightarrow CO_2(a)$

$CO_2(a) + H(a) \rightarrow HCO_2(a)$

$HCO_2(a) + 2H(a) \rightarrow CH_3O(a) + O(a)$

$CH_3O(a) + H(a) \rightarrow CH_3OH(g)$

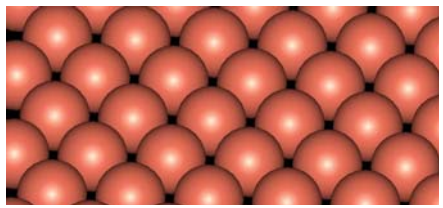
$CO(g) + O(a) \rightarrow CO_2(g)$

$H_2(g) + O(a) \rightarrow H_2O(g)$

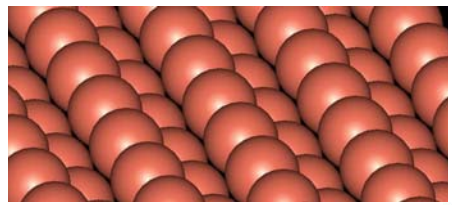
Chinchen *et al.*, *J. Chem. Soc. Faraday 1*, 83 (1987) 2193

Problems / Goals

- Morphology of active site: Cu surface and Cu/ZnO interface
  - The nature of the interaction of Cu with the catalytically important (0001)-Zn and (000-1)-O surfaces of ZnO ?
  - The oxidation state of the active copper sites (0, +1, +2) ?
  - The surface morphology of the catalytically active copper clusters involved - 111, 110 or 110-like?
- From reactants to intermediates to products
  - structure, spectra (vibrational) & energetics



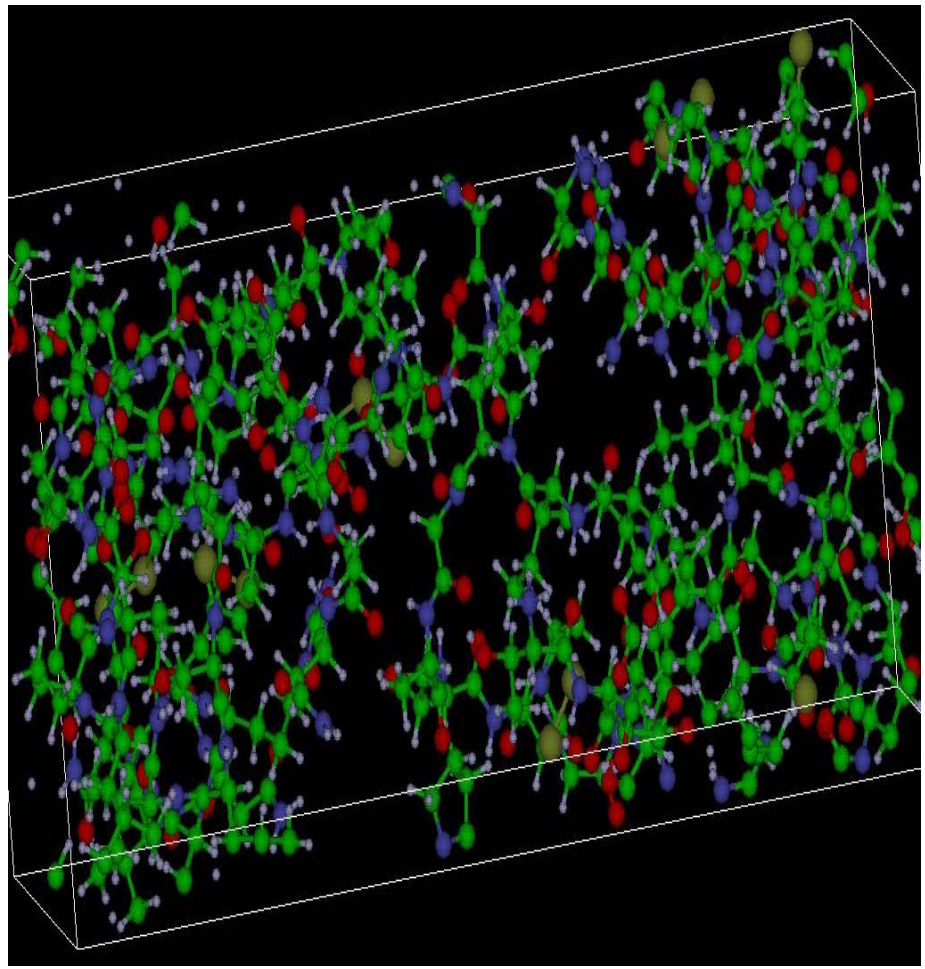
Cu (111) surface



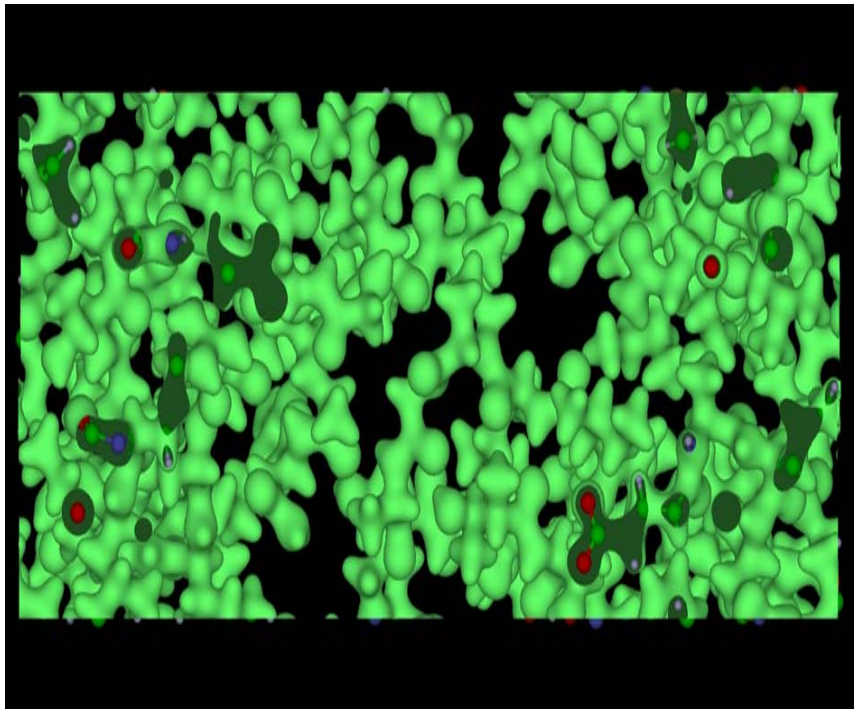
Cu (110) surface

Crambin - The Crystal

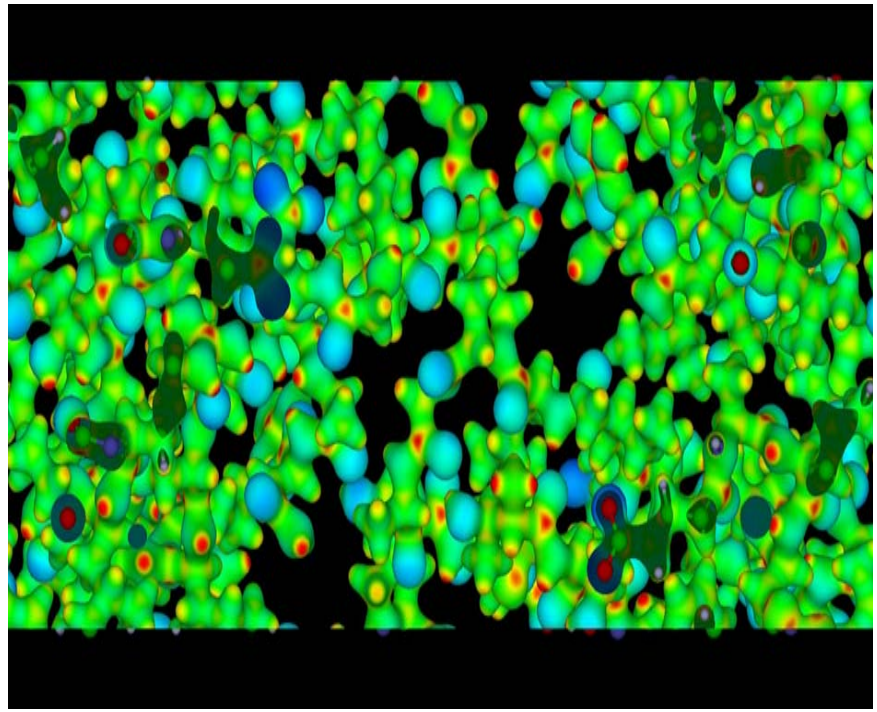
- 1284 Atoms
- Initial studies using STO3G (3948 basis functions)
- Improved to 6-31G** (12354 functions)
- All calculations Hartree-Fock
- Calculations run on HPCx
- **As far as we know the largest HF calculation ever converged**



Results - charge density and electrostatic potential



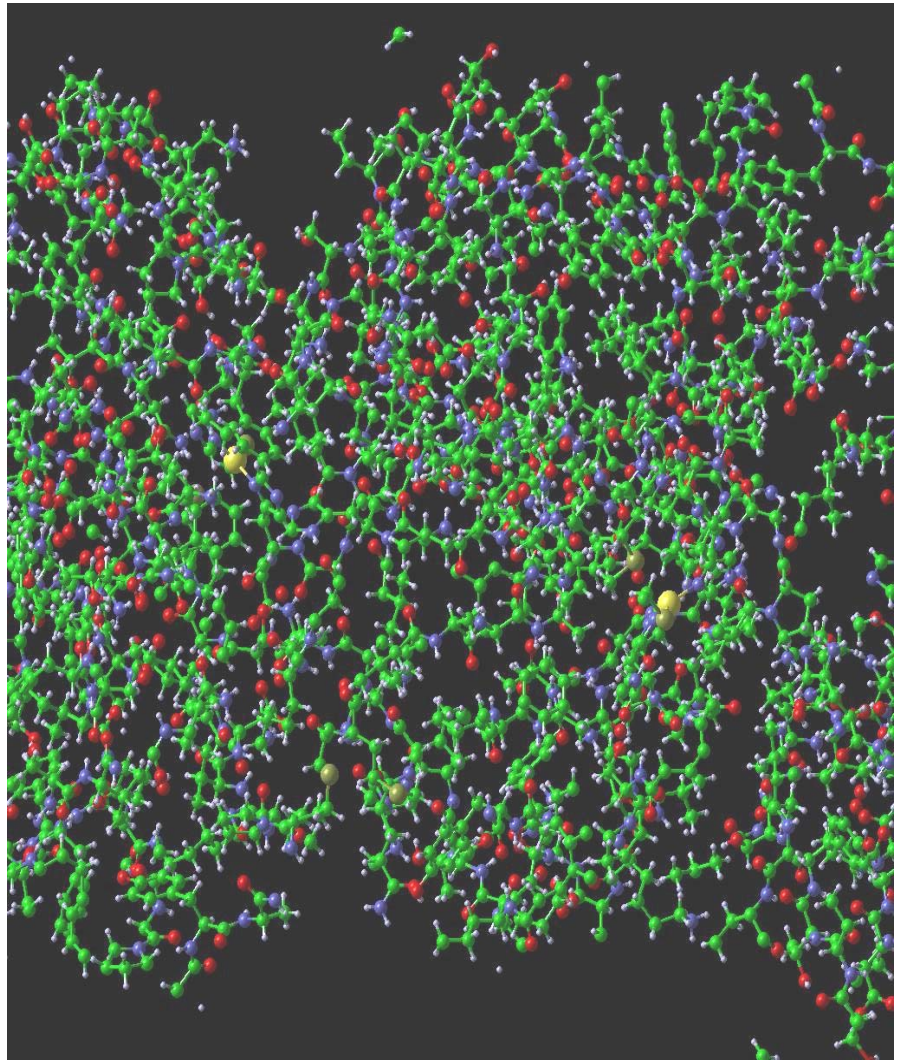
- Isosurface of the ***charge density*** at 0.1Å resolution: You name the resolution, in principle we can get it !
- Can be compared with SR results



- Charge density isosurface coloured according to ***electrostatic potential***
- Useful to determine possible chemically active groups

Rusticyanin

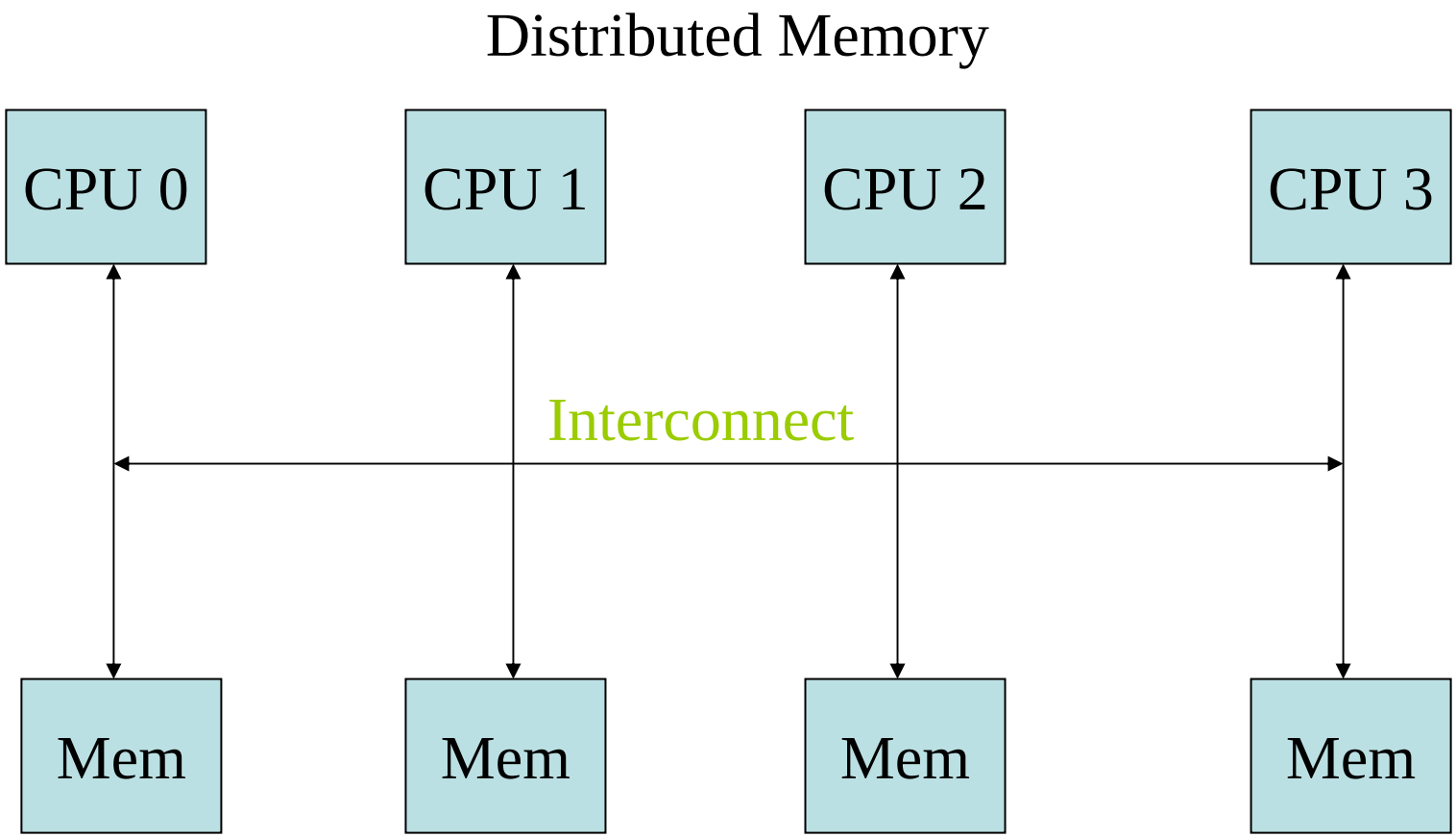
- Rusticyanin, a blue copper protein, has ~6300 atoms and is involved in redox processes
- We have started calculations using over 30000 basis functions
- In collaboration with S.Hasnain (DL) we want to calculate redox potentials for rusticyanin and associated mutants. Rusti has a large potential, 680mV

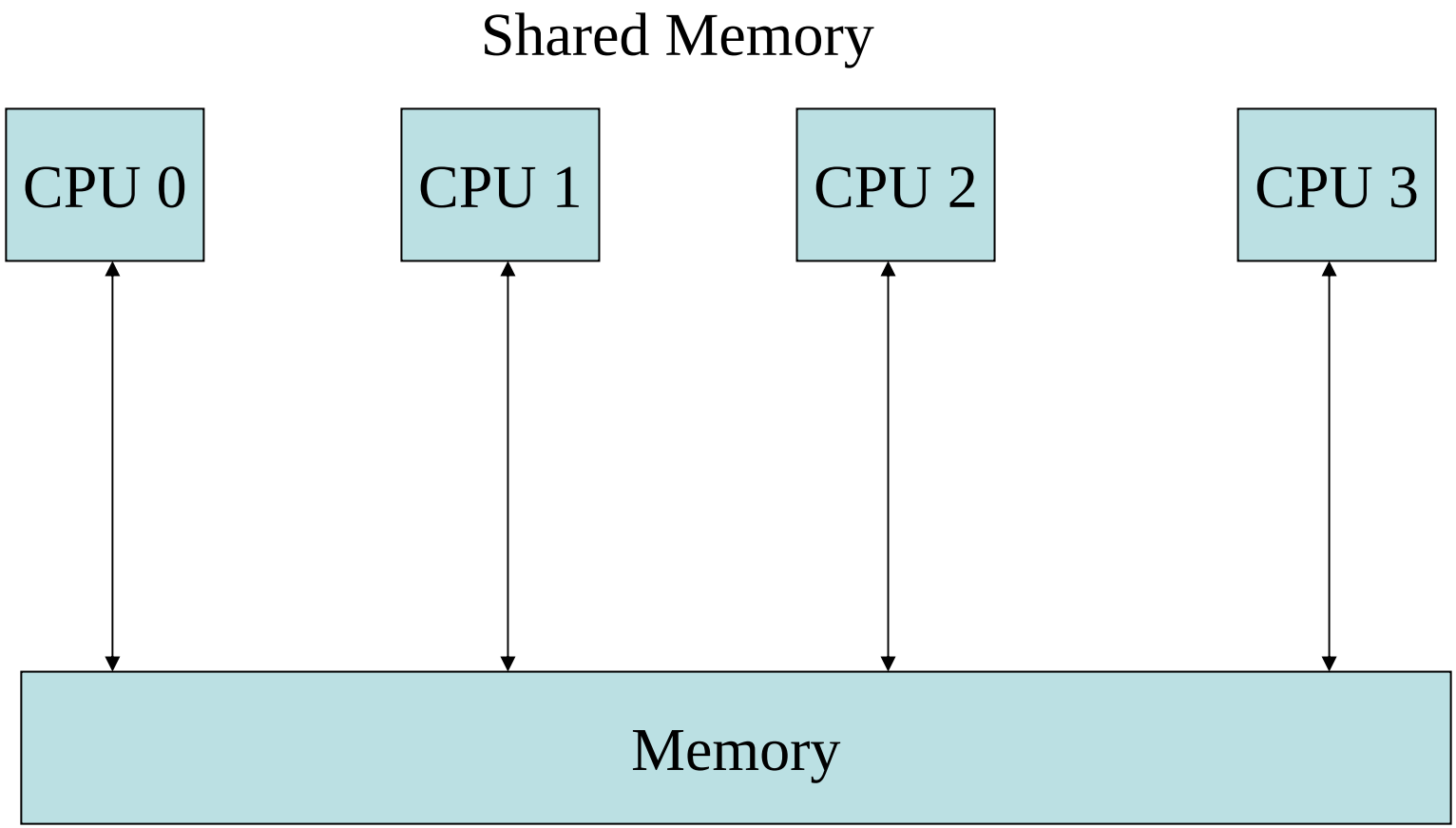


But Why Should You Know About Parallel Computing ?

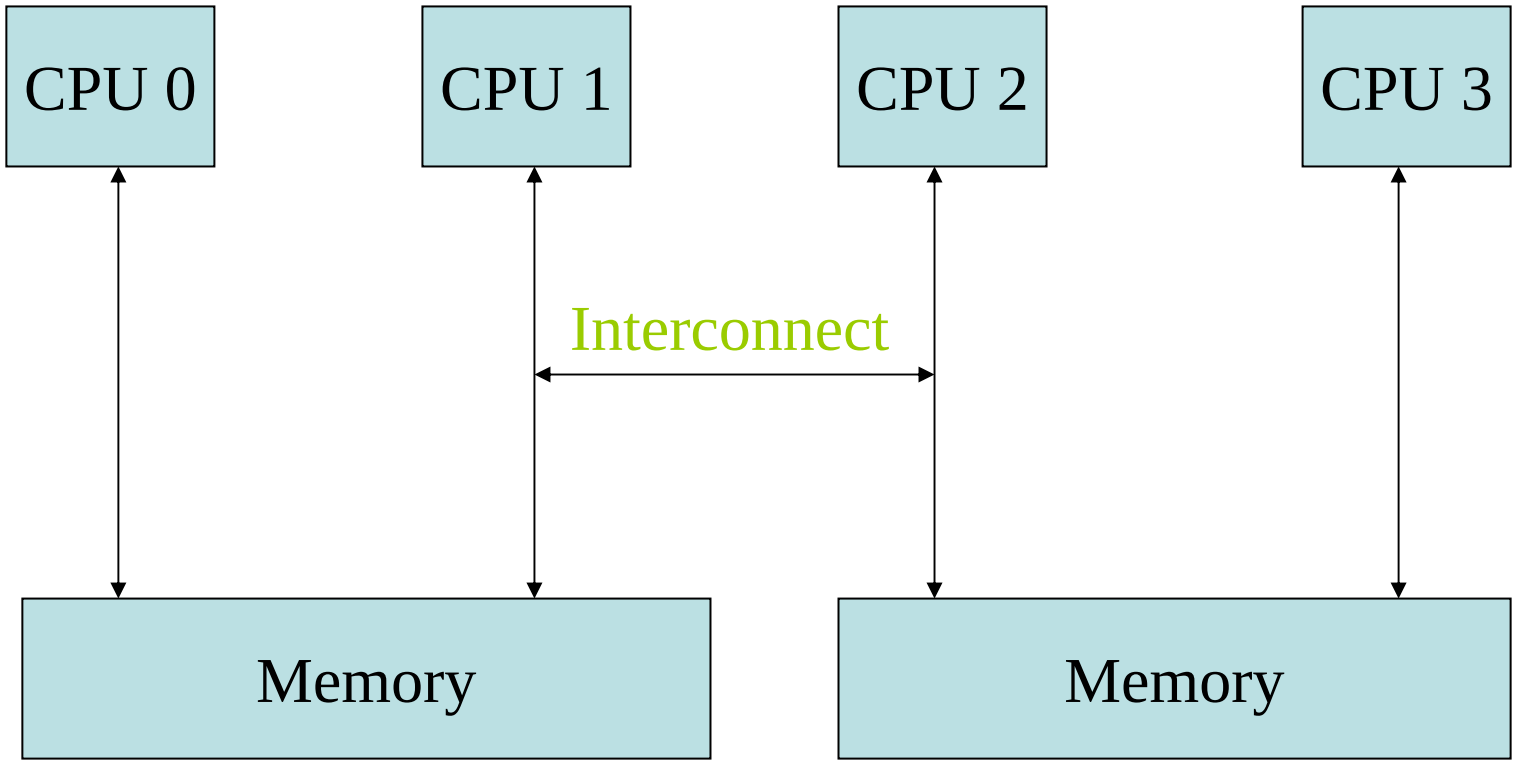
- Do you run your science packages without knowing the theory ?
- To exploit those packages better
  - **If you know the pitfalls you'll know how to get the best out of them**
- To write your own software !
  - **Commercial or open source software does not always address what you want to do**
  - **Commercial packages often do not work very well in parallel even if the company has claimed to parallelized the code**
    - *They don't make a large amount of money from it so why invest a lot of effort ?*
  - **It makes you think and is good for your soul**

So What Is a Parallel Computer ?




Shared Memory
CPU 0
CPU 1
CPU 2
CPU 3
Memory

Mixed



Why Do We Want To Do Parallel Computing ?

- Faster results
  - **One might hope that given P CPUs you can solve your problem P times quicker**
- Larger Problems
  - **Typically parallel machines have lots of memory so can do bigger problems**
    - e.g. 16 CPU system with 2 Gigabyte per processor has a lot more memory than your desktop
    - Side issue – are you restricted to 32 bit pointers ? Much less of a problem on a distributed memory machine
- Larger Problems Faster ???????

These are actually all related …….

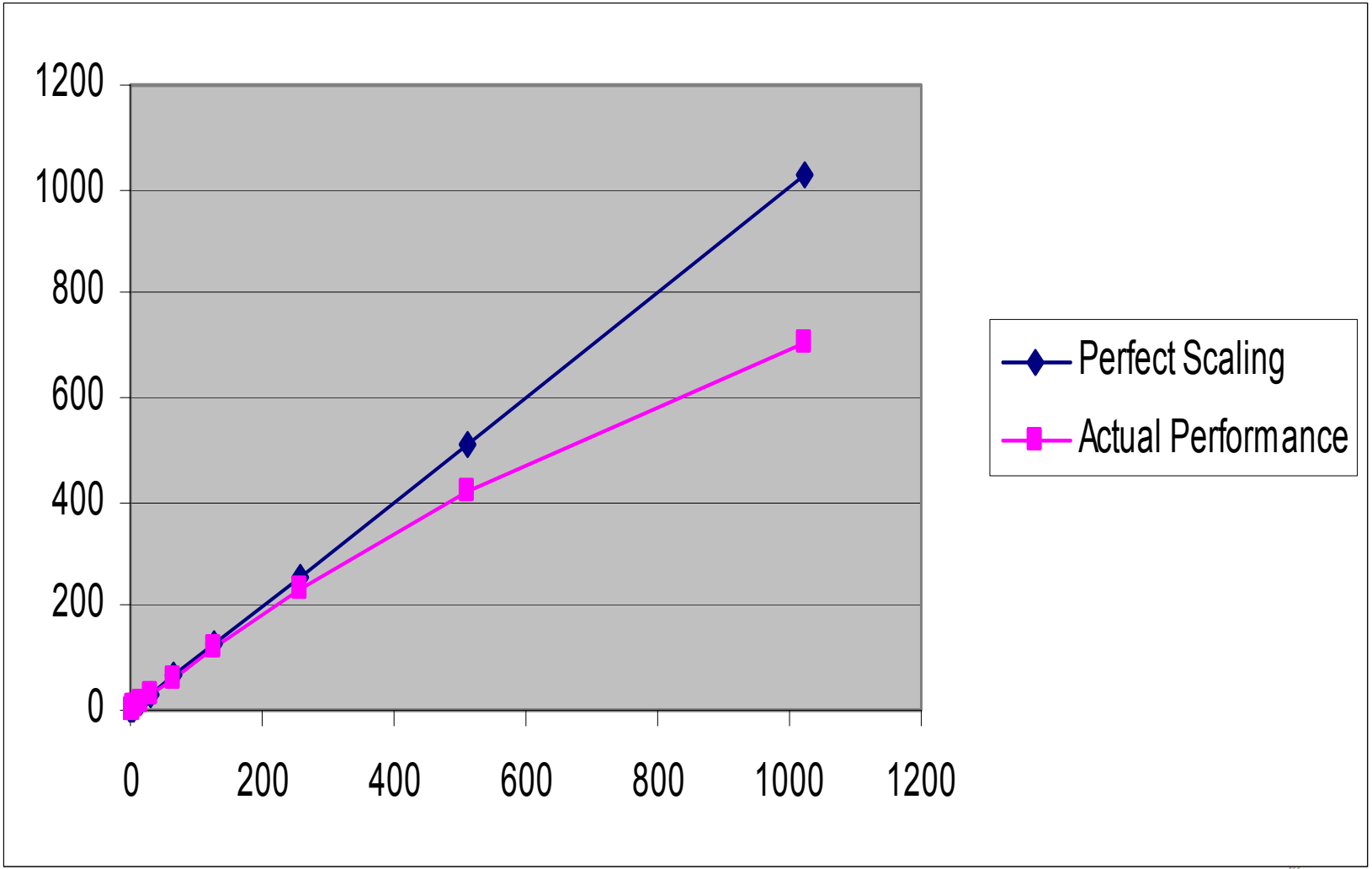
Speed Up

Speed up is how much faster your program runs on P processors relative to one

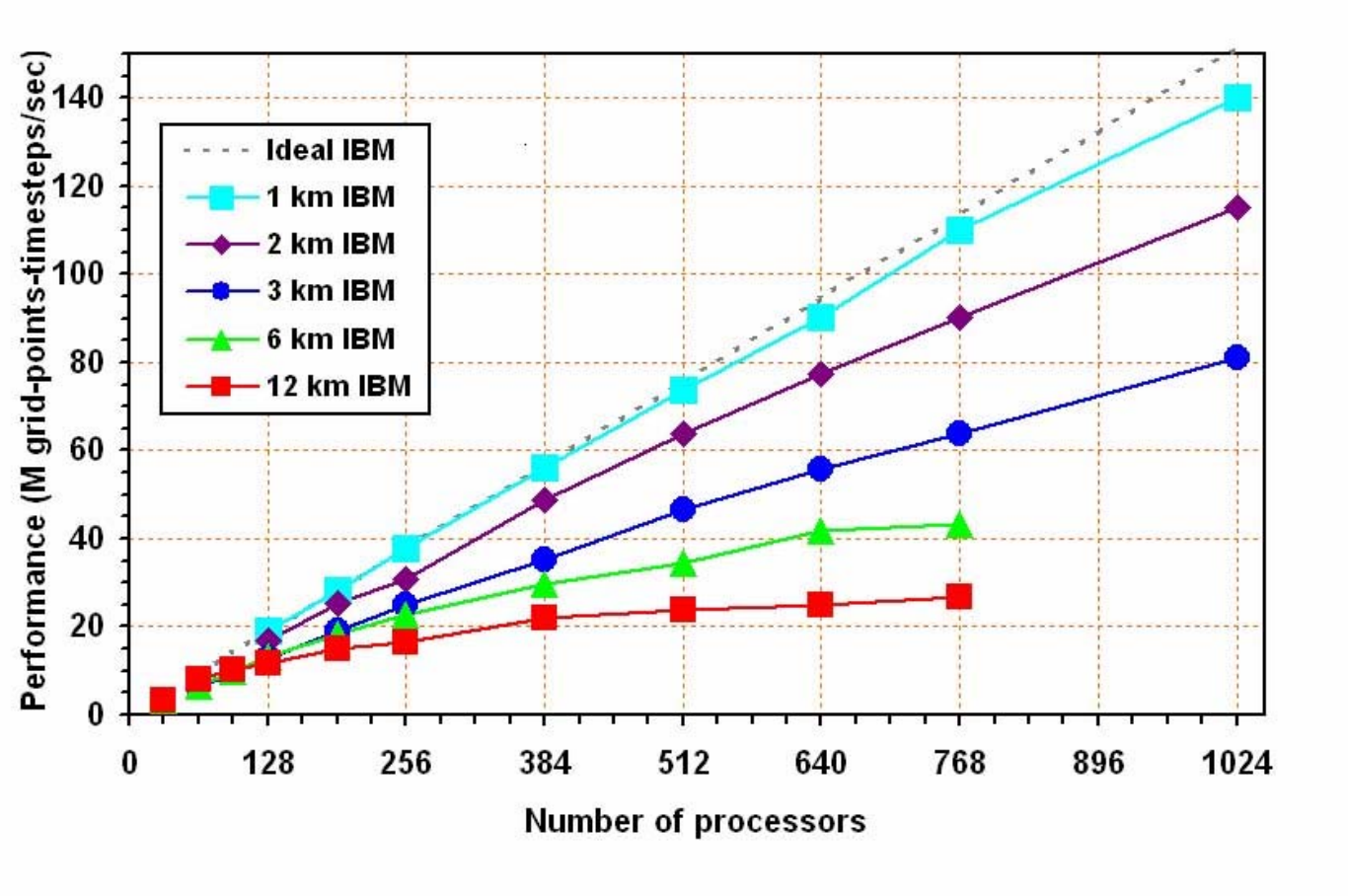
- Relative speed up is how much faster your program runs on P processors relative to one

- Absolute speed up is how much faster your program runs on P processors relative to THE BEST SERIAL IMPLEMENTATION
  - **Sometimes these are the same. Often there is little difference. Sometimes the difference can be quite marked**
    - *Sometimes the best serial algorithm is not the best for parallel machines*

A Speed Up Curve for CRYSTAL



Speed Up Curves for POLCOMMS



So Why Isn't it Perfect ….

- Remaining Serial code
- Load Imbalance
- Message Passing
- Memory and cache issues
- Other shared Resources
- It just doesn't like you !

Actually these two examples are very good, something to aim at.

  - **Sometimes your program gets SLOWER with more processors**

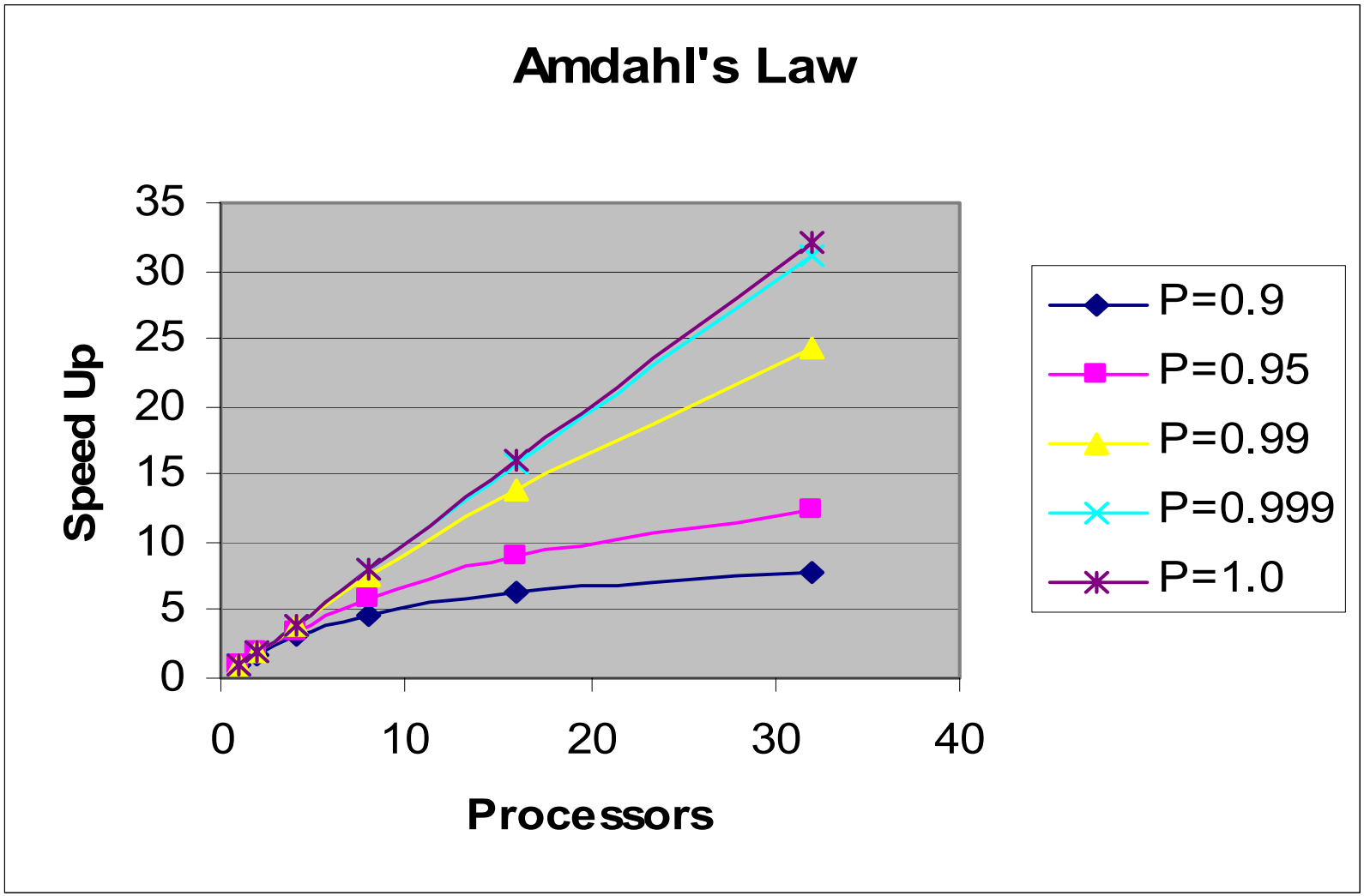
Remaining Serial Code

Say your program on one processor runs for P% of the time in perfectly parallel code, so it takes (1-P)% in serial. So on N processors it runs for P/N% of the time in parallel routines, but still (1-P)% in the serial ones. The speed up is therefore

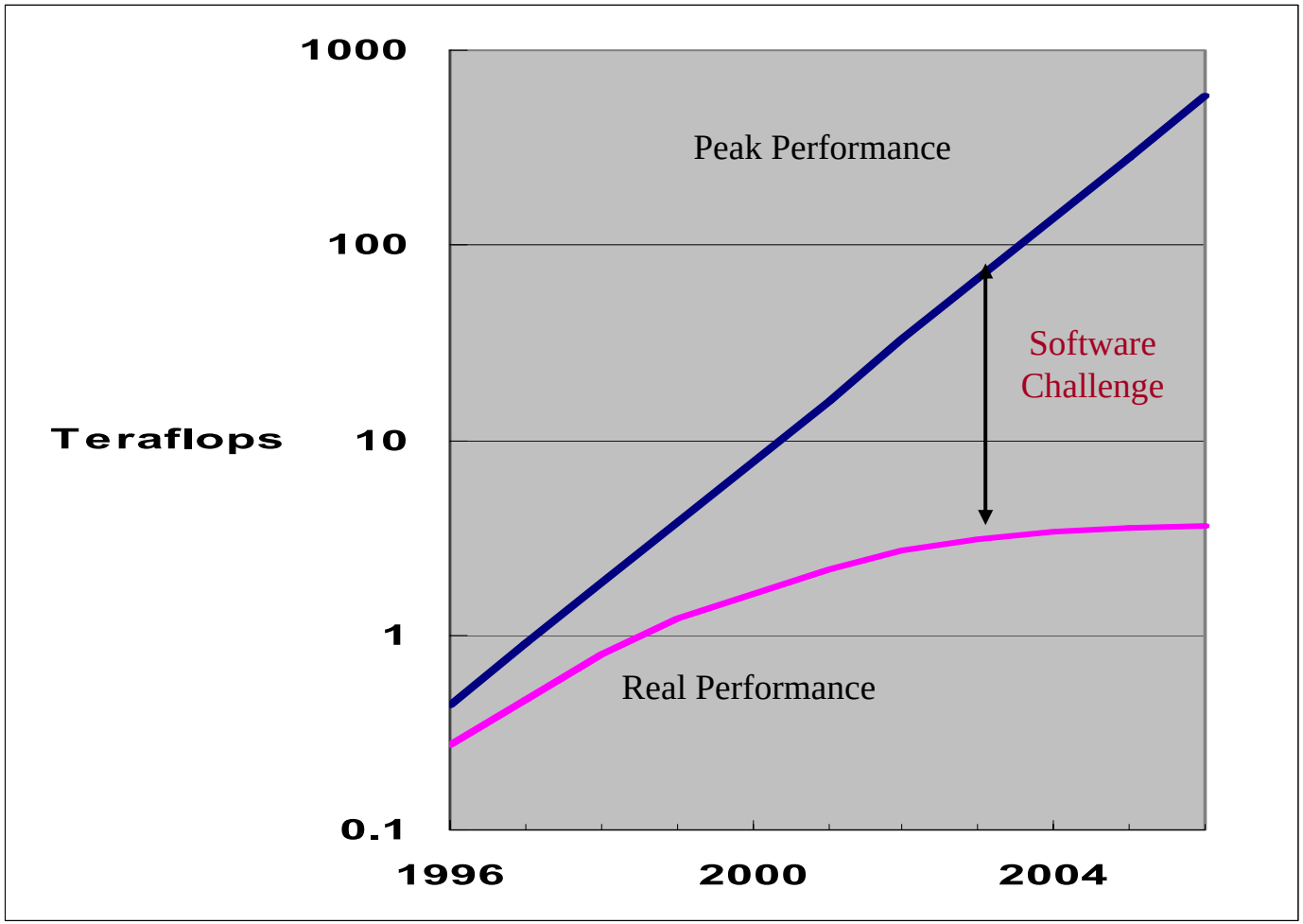
Speed up = 1/((1-P)+P/N)

This is known as Amdahl's law. It is somewhat scary !

- **On a infinite number of processors the speed up is 1/(1-P), so even if your program is running 90% in parallel the best speed up you can EVER get is 10 !**
- **The CRYSTAL results presented above when fitted to Amdahl's law give P=99.95%**


Amdahl's Law
Speed Up
35
30
25
20
15
10
5
0
0
10
20
30
40
Processors
P=0.9
P=0.95
P=0.99
P=0.999
P=1.0

The Software Challenge



Is It Really That Bad ?

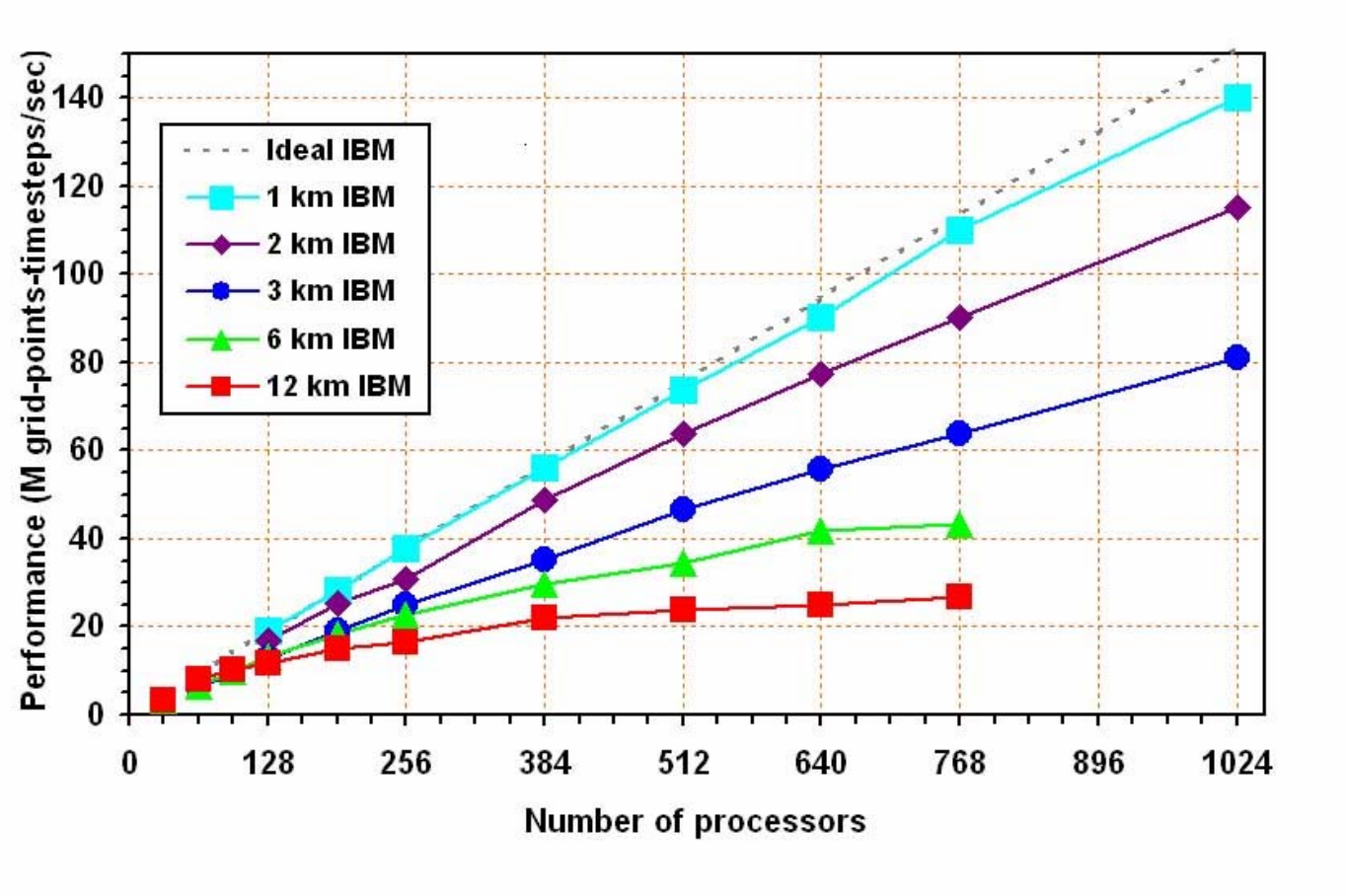
- P is usually a function of the problem being addressed
- In many problems the portion of the code that has been parallelised depends strongly on the system size in some way (e.g. N^3) while the serial portion scales much less strongly
  - **Gustavson's law**

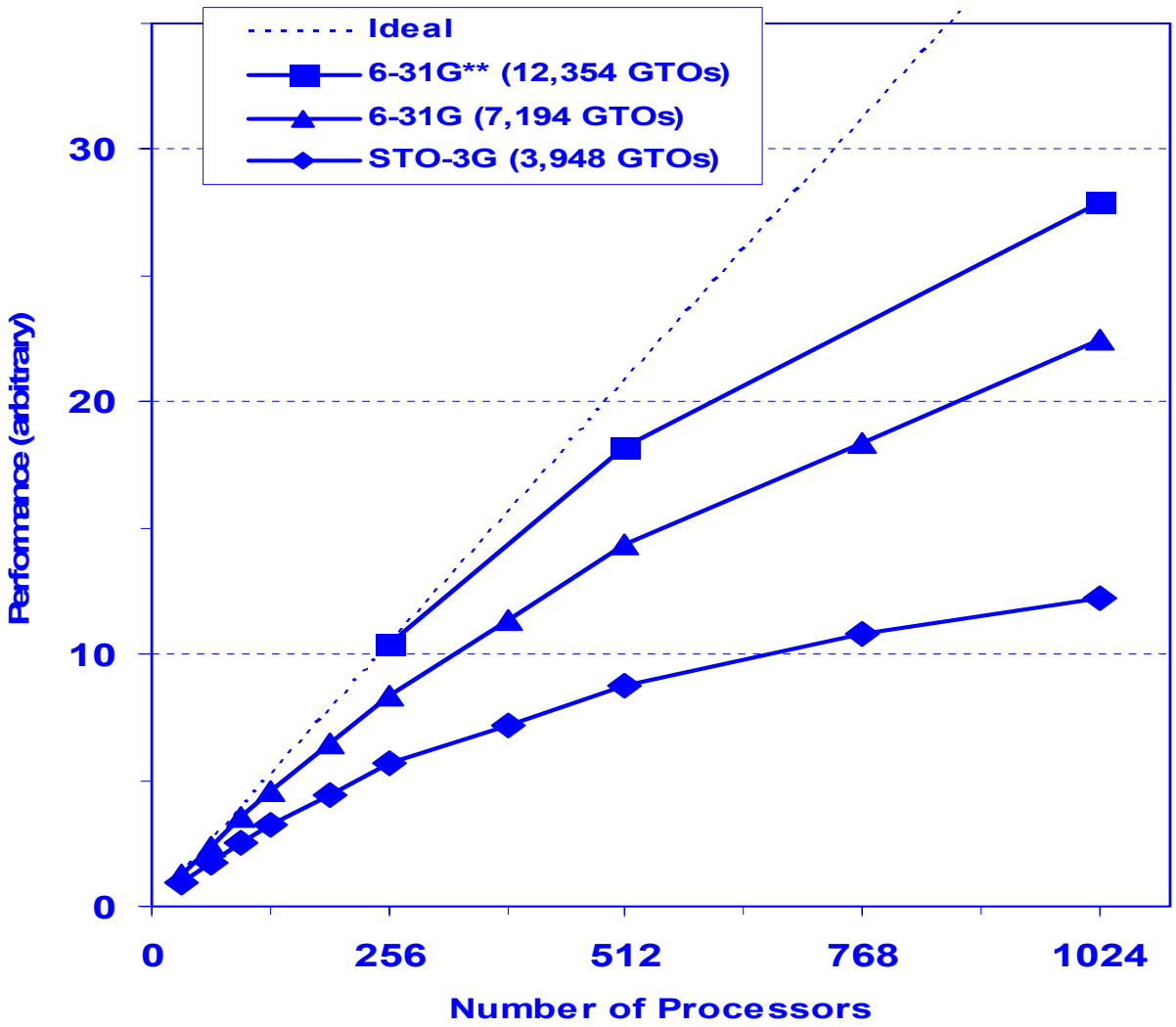
Parallel Computing Is Best For Large Systems

The Prime Directive

Parallel Computing Is Best For Large Systems

Speed Up Curves for POLCOMMS




Ideal
6-31G** (12,354 GTOs)
6-31G (7,194 GTOs)
STO-3G (3,948 GTOs)
Performance (arbitrary)
0
10
20
30
0
256
512
768
1024
Number of Processors

Load Imbalance

- Say you have 10 processors running a parallel job
  - **If all processors are doing the same thing one might hope that they take the same time**
    - *Can hope for optimal speed up*
  - **HOWEVER if they are doing different things, or the whole job is not evenly distributed, you can not hope for the best performance**
    - *Say one processor takes 6s to do it's work and the other take 1s the best you can hope for is a speed up of (6+9x1)/6=5/2*

This is known as **LOAD IMBALANCE**. It can affect all parallel programs, but is probably best known for task farms, grid based codes and, in some cases, classical MD codes.

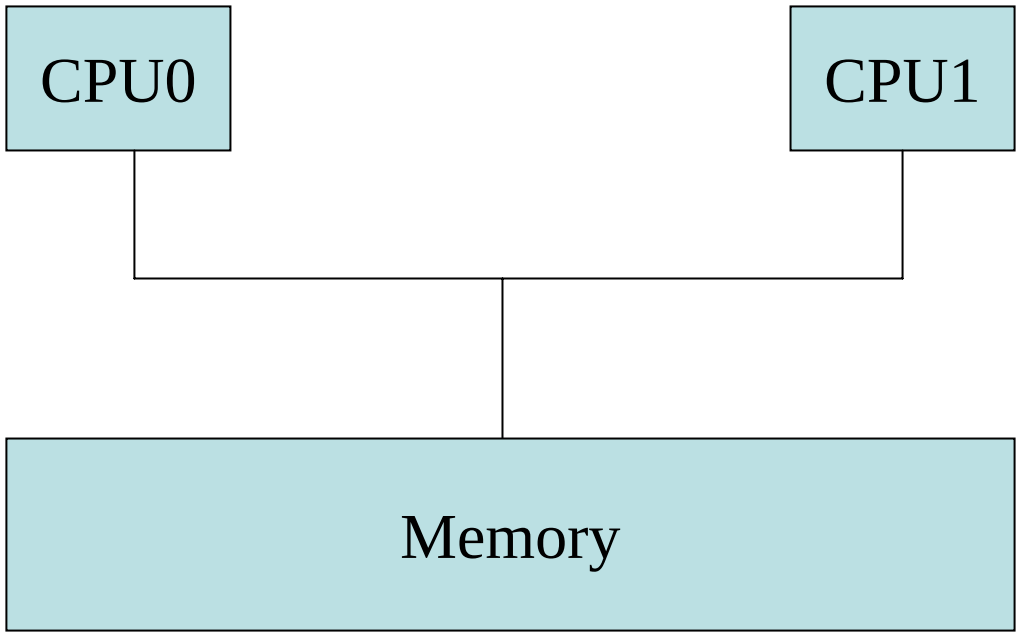
Message Passing

- In parallel computing CPUs have to `talk' to each other
  - **The bit of data you need for this calculation may not be on this processor**
  - **You may have to sum something across all processors (e.g. contributions to the total energy)**
- THIS TAKES TIME
  - **$T = \alpha+\beta N$**
  - **α is the latency – time to send zero bytes – 1-200μs**
  - **$1/\beta$ is the bandwidth – 10Mbytes/s – GBytes/s+**
- Obviously your serial code does not need to do this, so this inhibits get the best performance you might expect
- Avoid where possible – but unfortunately in all practical codes you can't avoid in entirely

Message Passing – is it really that bad ?

- $T = \alpha + \beta N$
- In practice the latency is the real problem
  - **Say N is large, so $T \sim \beta N$**
  - **For a distributed data code if you double the number of CPUs N goes down by a factor of 2, and so does T – scales well**
  - **BUT YOU CAN'T get rid of the latency !!**
- So … SEND LONG MESSAGES

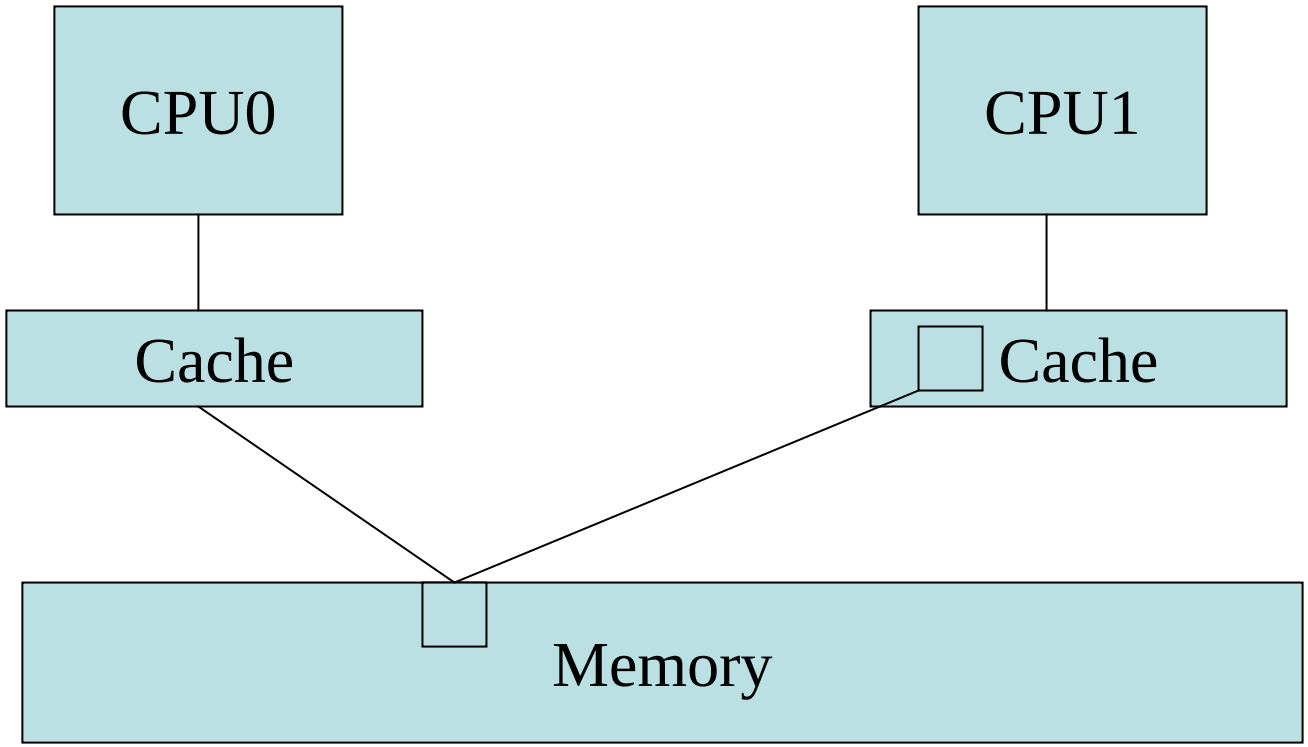
Memory Bus Issues



- This is how one vendor's dual processor system really works
- What if both processors want to use lots of memory real, right NOW ?
- Can see this even just running two identical serial jobs

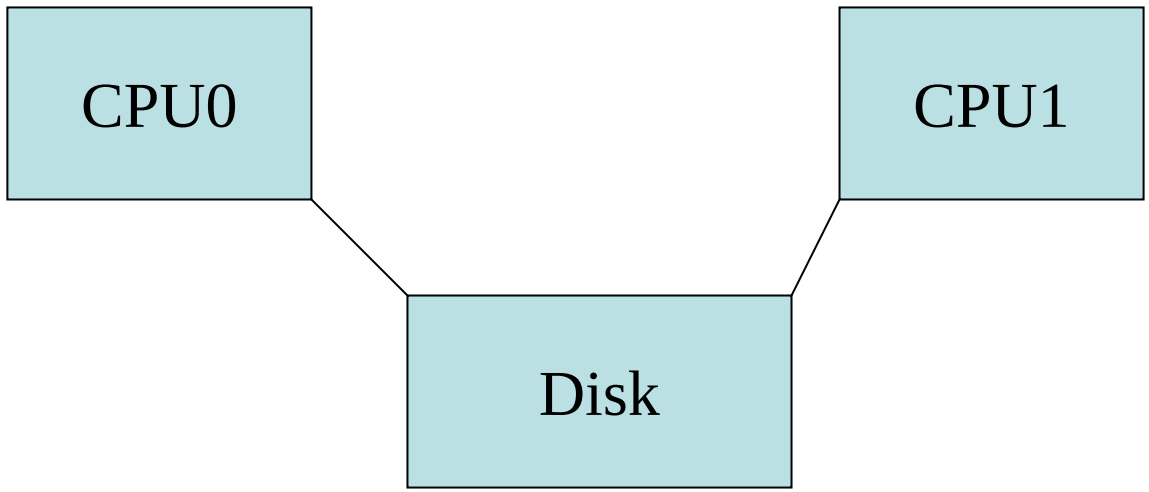
Memory And Cache Issues

CPU0 can't get it's memory until CPU1's cache is flushed



Even worse – **False Sharing**

Shared Resources



- If two CPUs have to share ANY resources there may be contention
- Classic case is disk
  - **May have to share with other users – non-repeatable job times**
  - **DON'T DO PARALLEL I/O OVER NFS !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!**
  - **In fact don't do I/O in parallel codes AT ALL if possible !!!!!!**

It hates you

- Well … Computers are inanimate so this can't occur, right ?
- In practice there are a number of issues that are very difficult to resolve without a very in depth knowledge of the code, and sometimes you simply get unlucky
  - **E.g. because the memory layout depends on how many processors you run on the efficiency of the cache usage may also vary. You may get unlucky. On the other hand … occasionally you might get lucky**
- Also debugging gets trickier, v. brief discussion latter

Other Issues

Well we're almost ready to start thinking about what we really need to do to write parallel codes. What else ?

- Portability
- Portability
- **Portability**

- debugging

Portability

- Portability has for a long time been a major issue with parallel computing
  - **Each vendor supplied it's own way to do it**
  - **Could require a lot of effort to move from one machine to another**

- However it has got a lot better over the last few years, and I VERY strongly suggest you stick to the standard methods now in use
  - **Fortran90 / C / C++ as a base language (and stick to the ISO standards !)**
  - **MPI for message passing**
  - **OpenMP for shared memory programming (or maybe Pthreads)**
  - **Open Source libraries where appropriate e.g.**
    - *Scalapack*
    - *Plapack*
    - *FFTW*
    - *PetSc*

Portability

- The situation is now such that if carefully written a code can run very well on a variety of machines with no change.
- If you suddenly get time on a supercomputer do you really want to blow all your budget porting the code from your cluster ?

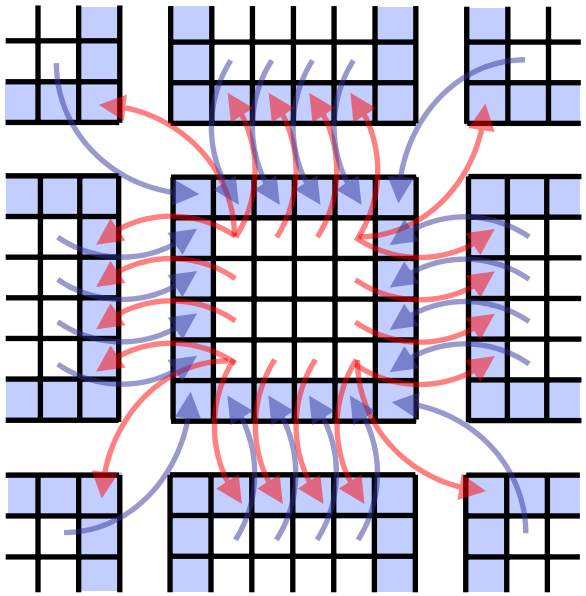
Debugging

- I don't want to say too much about this as it is really something you learn by experience. However parallel computing introduces a number of extra places to go wrong which can be very hard to diagnose
  - **Deadlock**
  - **Irreproducible errors**
    - *Run one time it works, run the next it doesn't*
    - *Typically occurs due to incorrect expectations about what all the processors are doing – e.g. one processor gets the wrong data from another because the other is in another part of the code from that you expect*
- Parallel debuggers do exist (e.g. Totalview, DDT) but they tend to be commercial and expensive, though if you can afford them they can be very useful

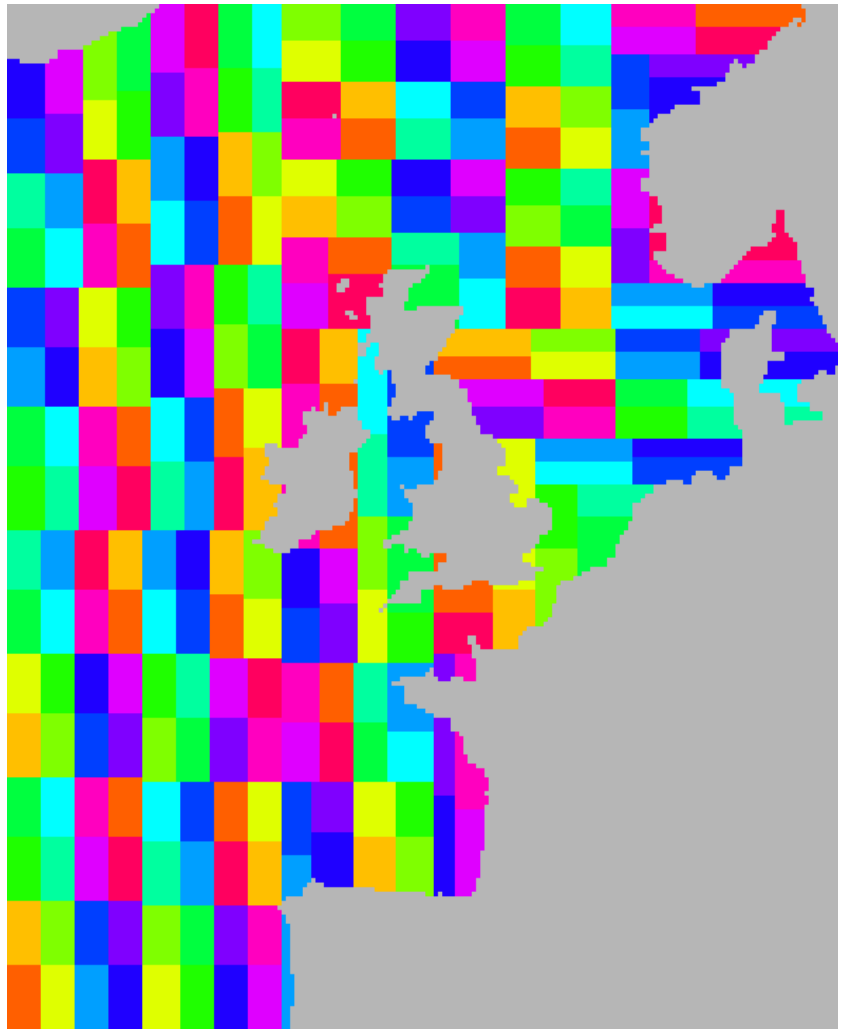
Parallel Programming Paradigms

- There are 3 major ways of programming the beasts
  - **Message passing by MPI**
    - *View memory as totally distributed – i.e. each CPU has it's own local memory which can not be directly accessed by any other, and conversely it can not access the memory of any other CPU*
    - *Most general method*
  - **Shared Memory by OpenMP**
    - *Each CPU can see all the memory.*
  - **Mixed mode using both MPI and OpenMP**
    - *Within the shared memory node use OpenMP*
    - *Between use MPI*
    - *In theory some performance benefits but in practice …. See the `It Hates You' section.*

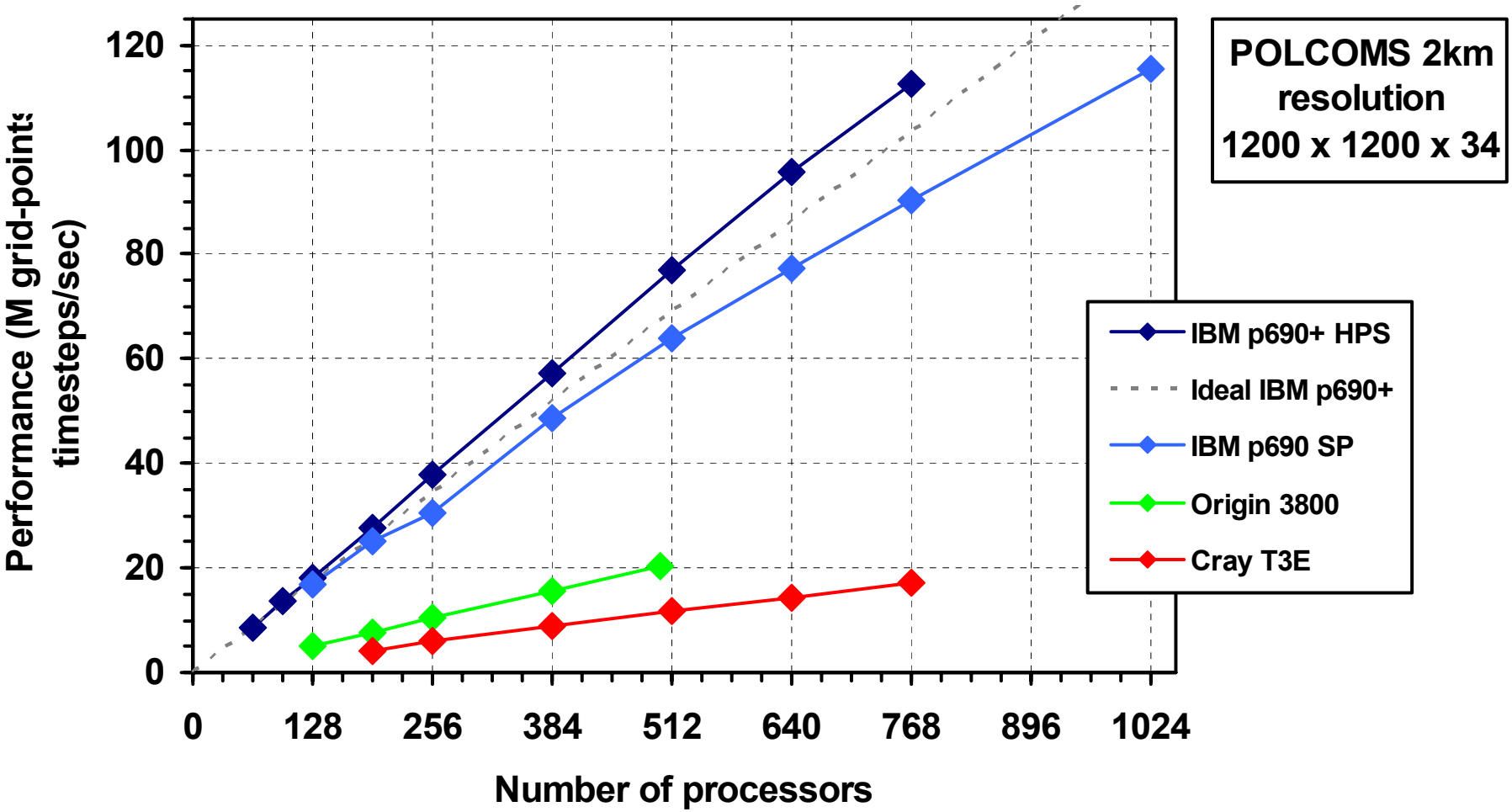
Recursive bisection grid partitioning in POLCOMS



2D halo boundary
data exchange



POLCOMS 2 km benchmark : All systems



Rusticyanin

- Rusticyanin, a blue copper protein, has ~6300 atoms and is involved in redox processes
- We have started calculations using over 30000 basis functions
- In collaboration with S.Hasnain (DL) we want to calculate redox potentials for rusticyanin and associated mutants. Rusti has a large potential, 680mV