Improvements for Multicore Performance and Domain Choice within DL_POLY_4,

Ian Bush
Valène Pellissier
(NAG Ltd.)

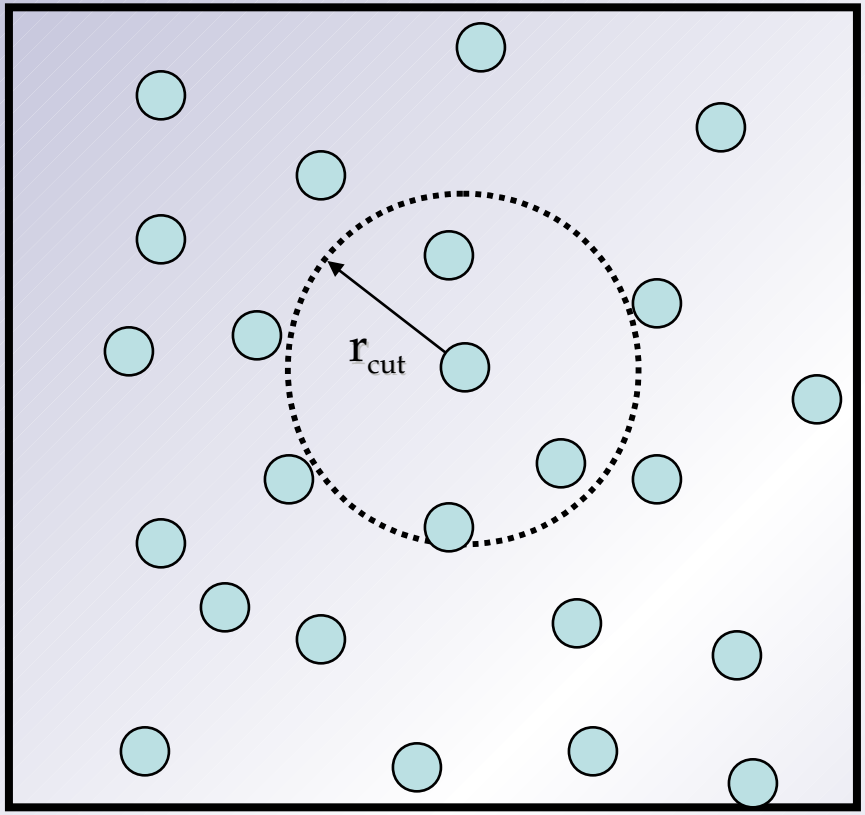
Molecular Dynamics: What is it?

- Theoretical tool for modelling the detailed microscopic behaviour of many different types of systems, including gases, liquids, solids, surfaces and clusters.
- In an MD simulation, the **classical equations of motion** governing the **microscopic time evolution** of a many body system are solved numerically, subject to the boundary conditions appropriate for the geometry or symmetry of the system.

Molecular Dynamics

- Can be used to monitor the microscopic mechanisms of energy and mass transfer in chemical processes, and dynamical properties such as absorption spectra, rate constants and transport properties can be calculated.
- Can be employed as a means of sampling from a statistical mechanical ensemble and determining equilibrium properties. These properties include average thermodynamic quantities (pressure, volume, temperature, etc.), structure, and free energies along reaction paths.

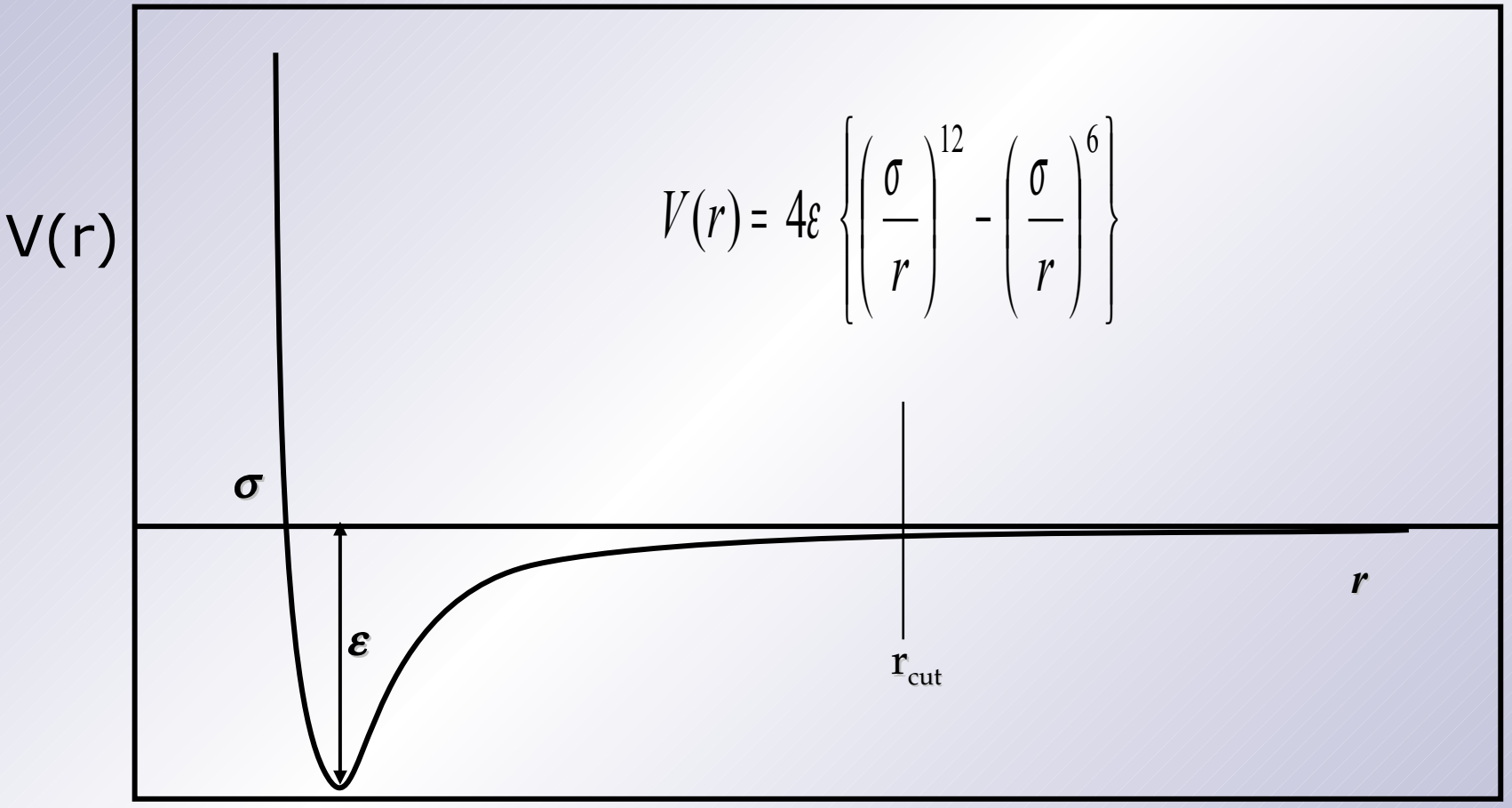
Example: Argon



Pair potential:

$$V(r) = 4\epsilon\left(\left(\frac{\sigma}{r}\right)^{12} - \left(\frac{\sigma}{r}\right)^{6}\right)$$

Lennard-Jones Potential



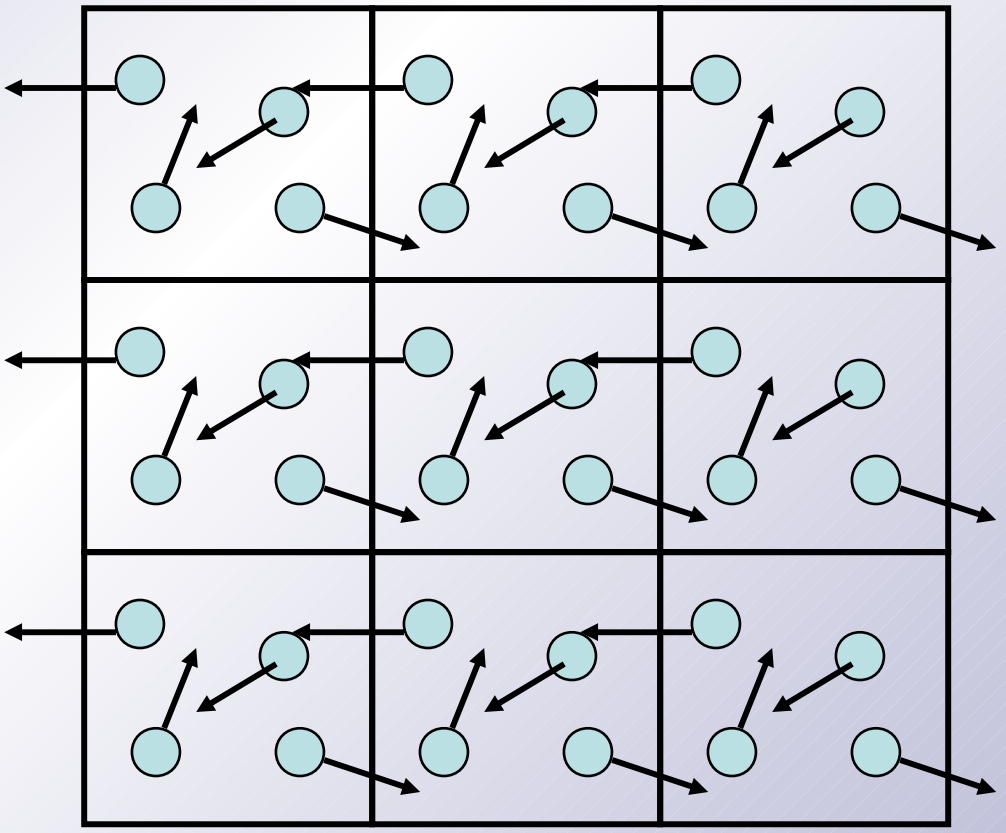
Equations of Motion

- Force on atom i due to atom j: $f_{ij} = -\nabla V(r_{ij})$
- Total force on atom i: $F_i = \sum_{j=1}^{N} f_{ij}$
- Newton 2: $a_i = \frac{F_i}{m_i}$
- So can update position by, e.g., Verlet's algorithm

Boundary Conditions

- Open – biopolymer simulations
- Stochastic boundaries – biopolymers
- Hard wall boundaries – pores, capillaries
- **Periodic boundaries** – most MD simulations

2D Cubic Periodic:



So What Can We Calculate?

- Kinetic Energy: $$\langle KE \rangle = \langle \frac{1}{2} \sum_{i}^{N} m_i v_i^2 \rangle$$
- Temperature: $$T = \frac{2}{3\mathrm{Nk}_B} \langle KE \rangle$$
- Configuration Energy: $$U = \sum_{i} \sum_{j}^{j>i} V(r_{ij})$$
- Pressure: $$PV = Nk_B T - \frac{1}{3} \langle \sum_{i}^{N} \boldsymbol{r}_i \cdot \boldsymbol{f}_i \rangle$$
- Specific heat: $$\langle \delta U \rangle = \frac{3}{2} Nk_B^2 T^2 \left(1 - \frac{3\mathrm{Nk}_B}{2\mathrm{C}_v} \right)$$

What Else?

- Pair correlation (Radial Distribution Function):

$$g(r)=\frac{V}{N^2}\left\langle \sum_{i}^{N} \sum_{j,\, j\neq i}^{N} \delta\left(r-r_{ij}\right)\right\rangle$$

- Structure factor:

$$S(k)=1+4\pi\rho\int_{0}^{\infty}\frac{\sin(kr)}{kr}\left(g(r)-1\right)r^2\,dr$$

- Note: S(k) available from diffraction experiments
  - e.g. Diamond, ISIS

So How Does It Scale?

- Provided we can cut off the potential …
  - Each atom interacts with a finite volume
  - Within that volume on average there is a constant number of atoms
    - Related to the density of the system
  - So the evaluation of the force on each atom on average takes a time that is independent of system size
  - So classical MD is O(N) as we have to evaluate the forces on N atoms for each time step
- **BUT – can we always cut off the potential?**

No!!!!

- Consider a radius σ atom interacting with a uniform "sea" by a $1/r^n$ potential

- The energy of interaction is

$$E = \int_{\sigma}^{\infty} d\boldsymbol{r}\, \frac{1}{r^n} = 4\pi \int_{\sigma}^{\infty} dr\, r^2 \frac{1}{r^n}$$

- This diverges for $n \leq 3$; such potentials are termed **long ranged**
  - The problem is that the volume element grows faster than the potential decays
  - Get conditionally convergent sums
- Is this important?

Yes!!!!

- The fundamental interaction of chemistry is the Coulomb potential

$$V(r_{ij}) = \frac{q_i q_j}{r_{ij}}$$

- So as we can't cut off the potential we have to sum **every** pair
  - Which is quite a large number in a system with periodic boundary conditions (have to include all images!)
  - And therefore might take quite a long time!

Ewald Summation

Get round this by use of the *Ewald Sum*

- Takes advantage of the periodic nature of the system to evaluate the long range terms
- Effectively splits the potential into two parts
  - Short ranged
    - Can cut off, won't talk about more here
  - Long ranged but not singular at r=0
    - Use Fourier techniques to evaluate
    - This is what we are interested in

$$V(r) = \frac{erfc(\beta r)}{r}$$

$$V(r) = \frac{erf(\beta r)}{r}$$

Long Range Terms

$$U_{lr} = \frac{1}{2\pi V} \sum_{\boldsymbol{m} \neq 0} S(\boldsymbol{m}) S^{\dagger}(\boldsymbol{m}) \frac{e^{-\frac{\pi^2 |\boldsymbol{m}|^2}{\beta^2}}}{|\boldsymbol{m}|^2}$$

$$\boldsymbol{m} = m_1 \boldsymbol{b_1} + m_2 \boldsymbol{b_2} + m_3 \boldsymbol{b_3}$$

$$S(\boldsymbol{m}) = \sum_{j=1}^{N} q_j e^{2\pi i \boldsymbol{m} . \boldsymbol{r}_j}$$

Structure Factor

$$S(\boldsymbol{m}) = \sum_{j=1}^{N} q_j e^{2\pi i \boldsymbol{m}.\boldsymbol{r}_j}$$

$$S(\boldsymbol{m}) = \sum_{j=1}^{N} q_j e^{2\pi i \frac{m_1 u_1}{\mathrm{K}_1}} e^{2\pi i \frac{m_2 u_2}{\mathrm{K}_2}} e^{2\pi i \frac{m_3 u_3}{\mathrm{K}_3}}$$

This looks almost like a $K_1 * K_2 * K_3$ 3D FT, but not quite as the atoms are almost certainly not on a nice regular grid

The Particle Mesh Approximation

- So can we calculate the structure factor by only sampling on a regular grid?
  - If so can calculate it using a FFT
- Yes!
- Number of ways, most commonly used in MD is the *Smooth Particle Mesh Ewald* method due to Darden *et al.*:

$$e^{2\pi i \frac{m_\mu u_\mu}{\mathrm{K}_\mu}} \approx b_\mu(m_\mu) \sum_{k_\mu=-\infty}^{\infty} M_n(u_\mu - k_\mu) e^{2\pi i \frac{m_\mu k_\mu}{\mathrm{K}_\mu}}$$

SPME

$$e^{2\pi i \frac{m_\mu u_\mu}{K_\mu}} \approx b_\mu(m_\mu) \sum_{k_\mu=-\infty}^{\infty} M_n(u_\mu - k_\mu) e^{2\pi i \frac{m_\mu k_\mu}{K_\mu}}$$

M_n are the cardinal B-splines of order n

They have some nice properties including:

- Easily evaluated by a simple recurrence
- Differentiable n–2 times
  - Forces
- $M_n(u)$ is non-zero only in the range $0<u<n$
  - Only need evaluate at grid points near each atom
  - So effectively a short range interaction!

SPME Steps

1 Evaluate $Q(\boldsymbol{k}) = \sum_{i=1}^{N} \sum_{n_1, n_2, n_3} q_i \prod_{\mu=1,2,3} M_n(u_{\mu i} - k_\mu - n_\mu \mathrm{K}_\mu)$

2 Fourier transform $Q(\boldsymbol{k}) \rightarrow Q(\boldsymbol{m})$

3 $V(\boldsymbol{m}) = \left[\prod_{\mu} \left|(b_\mu)^2\right|\right] Q(\boldsymbol{m}) \frac{e^{-\frac{\pi^2 |\boldsymbol{m}|^2}{\beta^2}}}{|\boldsymbol{m}|^2}$

4 Inverse Fourier transform $V(\boldsymbol{m}) \rightarrow V(\boldsymbol{k})$

5 From $Q(\boldsymbol{k})$, $V(\boldsymbol{k})$ easy to calculate energy and forces

So…

Therefore:

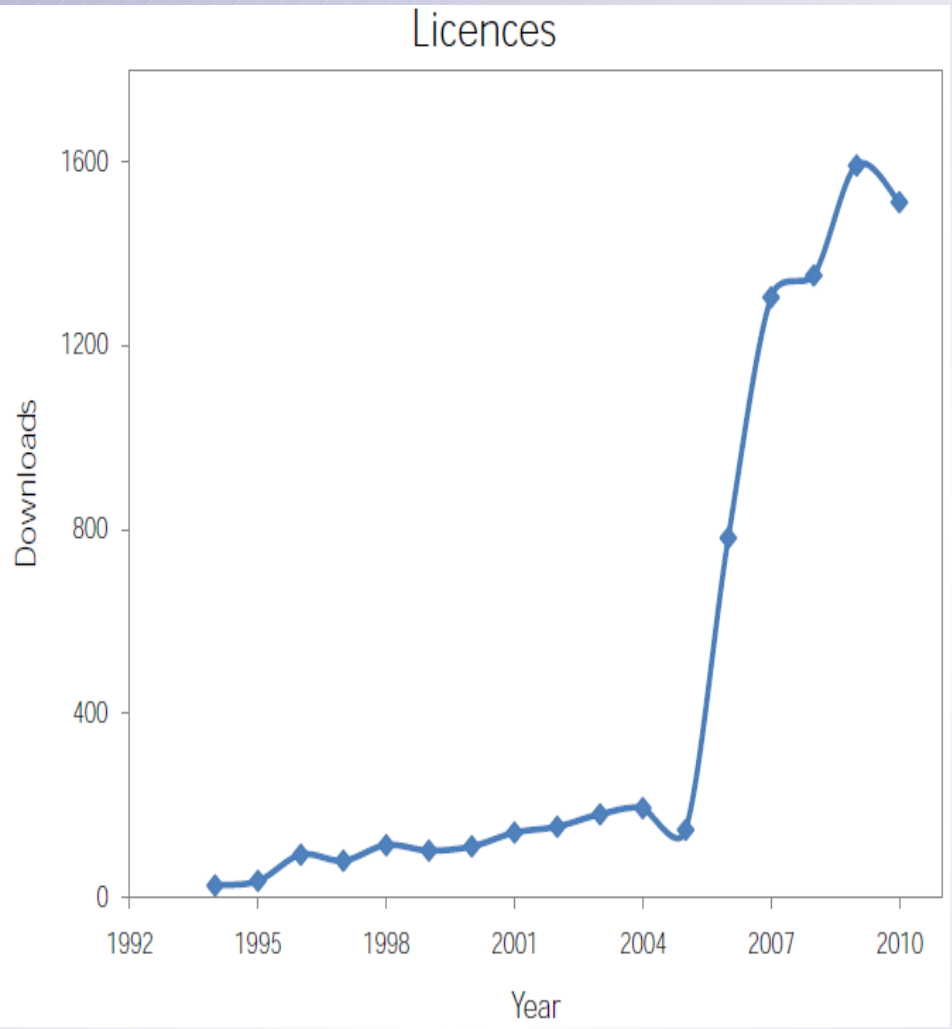
- To evaluate short ranged terms we only need to know the positions of the atoms near to the atom of interest
- To evaluate long range terms we
  - Need to evaluate terms on the grid points near the reference atom
  - But also need a FFT

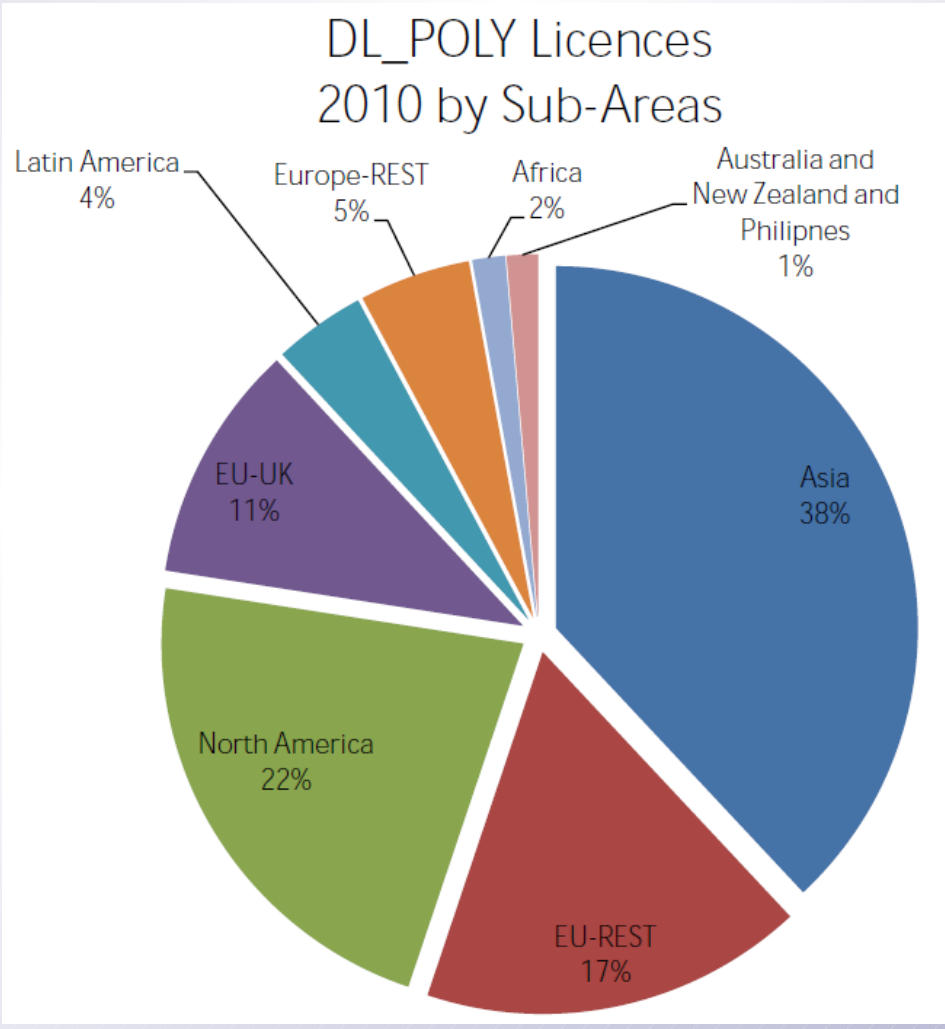
Thus when planning how to do this in parallel we should use a decomposition that reflects the spatial locality

DL_POLY – What is it?

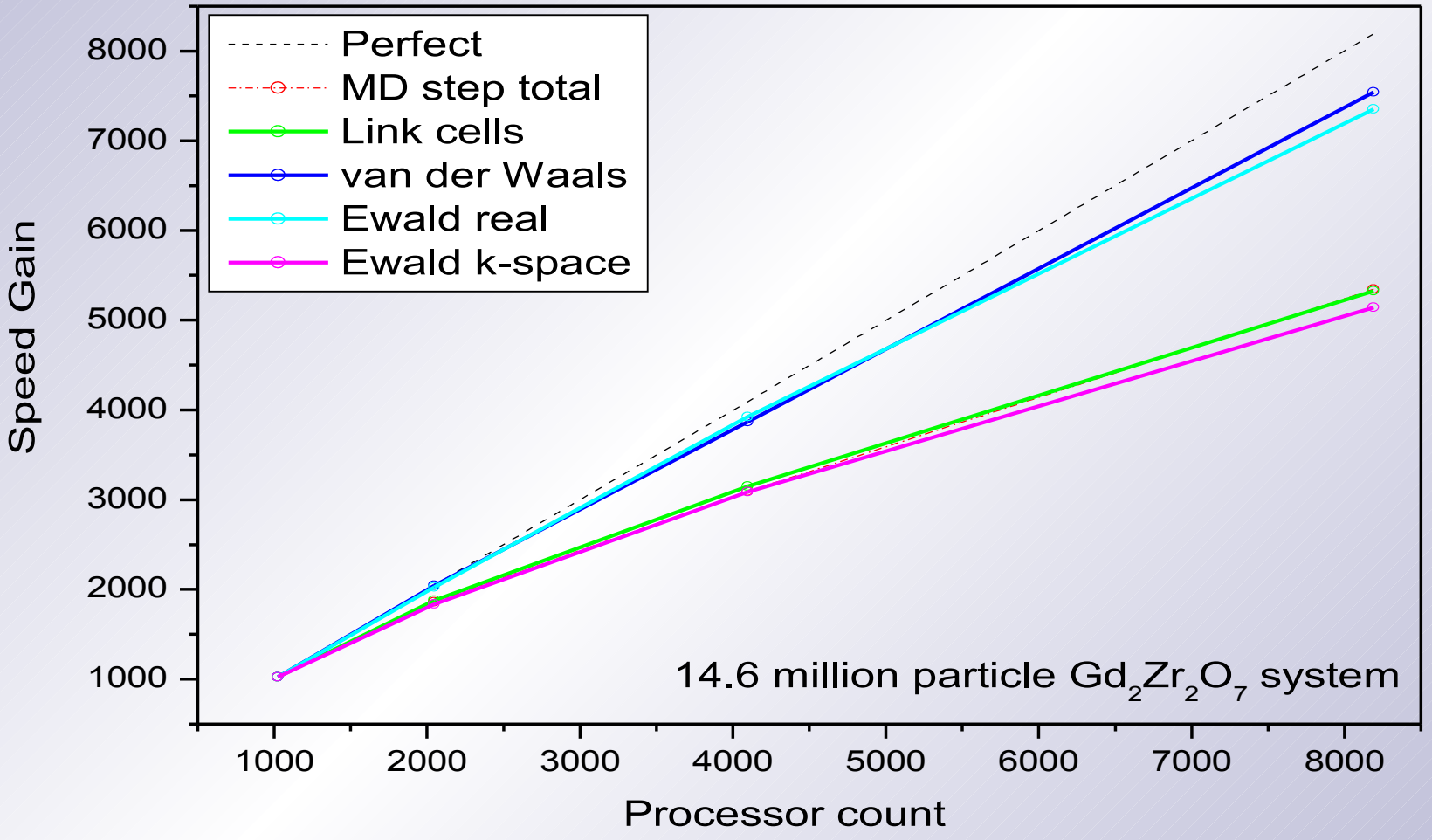
- General purpose parallel (classical) MD simulation software
- Originally funded by CCP5
- Written in modularised Fortran95 with MPI1+MPI-I/O
- 1994 – 2011: DL_POLY_2 (RD) by W. Smith & T.R. Forester
- 2003 – 2011: DL_POLY_3 (DD) by I.T. Todorov & W. Smith
- Available free of charge (under licence) to University researchers (provided as code) and at cost to industry

Widely Used


Licences
Downloads
1600
1200
800
400
0
1992
1995
1998
2001
2004
2007
2010
Year


DL_POLY Licences
2010 by Sub-Areas
Latin America
4%
Europe-REST
5%
Africa
2%
Australia and
New Zealand and
Philipnes
1%
EU-UK
11%
Asia
38%
North America
22%
EU-REST
17%

How well does it work?

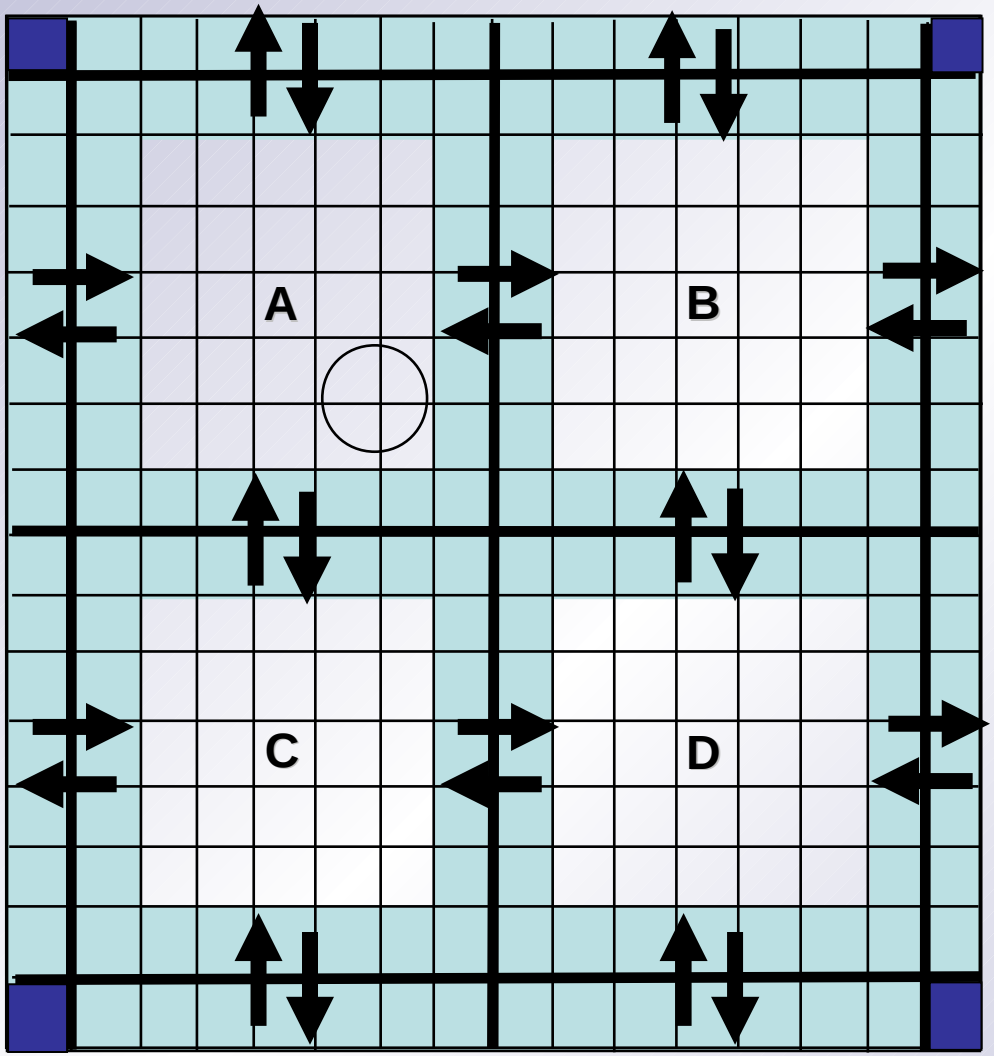


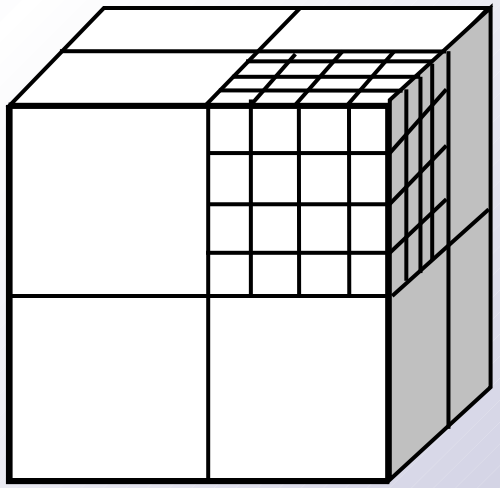
Two Main Versions

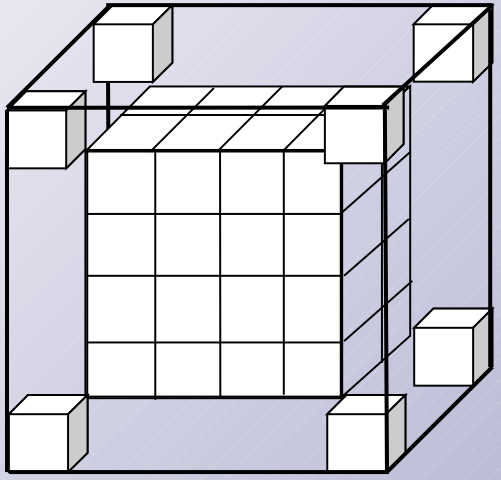
- **DL_POLY_4** (version 1.2)
  - Parallelisation based on domain decomposition
  - limits up to ≈2.1×10^{9} atoms with inherent parallelisation
    - i.e. 2^{31}
  - Full force field and molecular description with rigid body description

- DL_POLY Classic (version 1.6)
  - Replicated Data parallelisation, limits up to ≈30,000 atoms with good parallelisation up to ~64 cores
  - Full force field and molecular description
  - Hyper-dynamics, Temperature Accelerated Dynamics, Solvation Dynamics, Path Integral MD

DL_POLY_4 Domain Decomposition



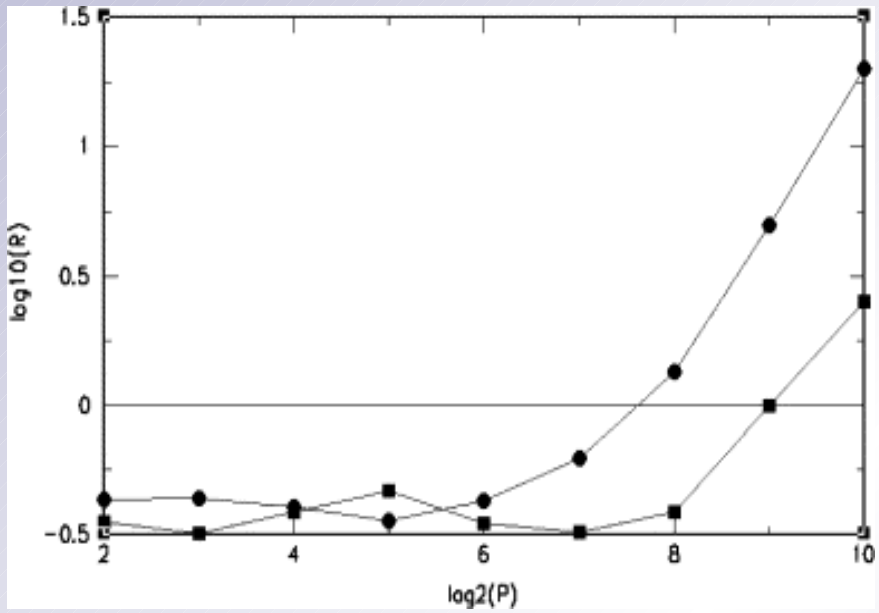




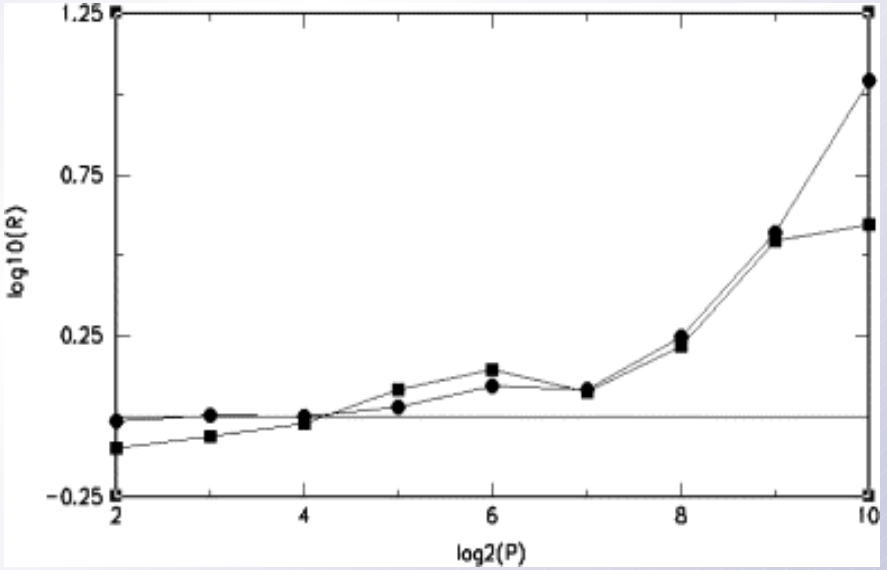
Domain Decomposition

- So domain decomposition fits the short range terms very nicely
- But not “standard” library parallel 3D FFTs
  - FFTW, IBM's PESSL, Cray's SciLib all use a decomposition by planes (“slabs”)
- So two choices:
  1) Use a library routine, but this will require an expensive data redistribution
  2) Write an FFT that uses the domain decomposed form directly – DaFT
     - But now we need to parallelise the 1D FFTs

So how DaFT is it?



Without time for redistribution



Including time for redistribution

Moral of this story

- Have to use the data decomposition that fits the problem
  - Sometimes “force fitting” a standard library may not be the best solution
    - Libraries for distributed data problems are **hard**:
      - ~Infinitely different ways to distribute the data
      - Have to use best distribution for whole application, not just the library routine
        - Redistribution will (eventually) kill you
      - Somebody from NAG shouldn't be saying this!

So How to Parallelise a 1D FFT?

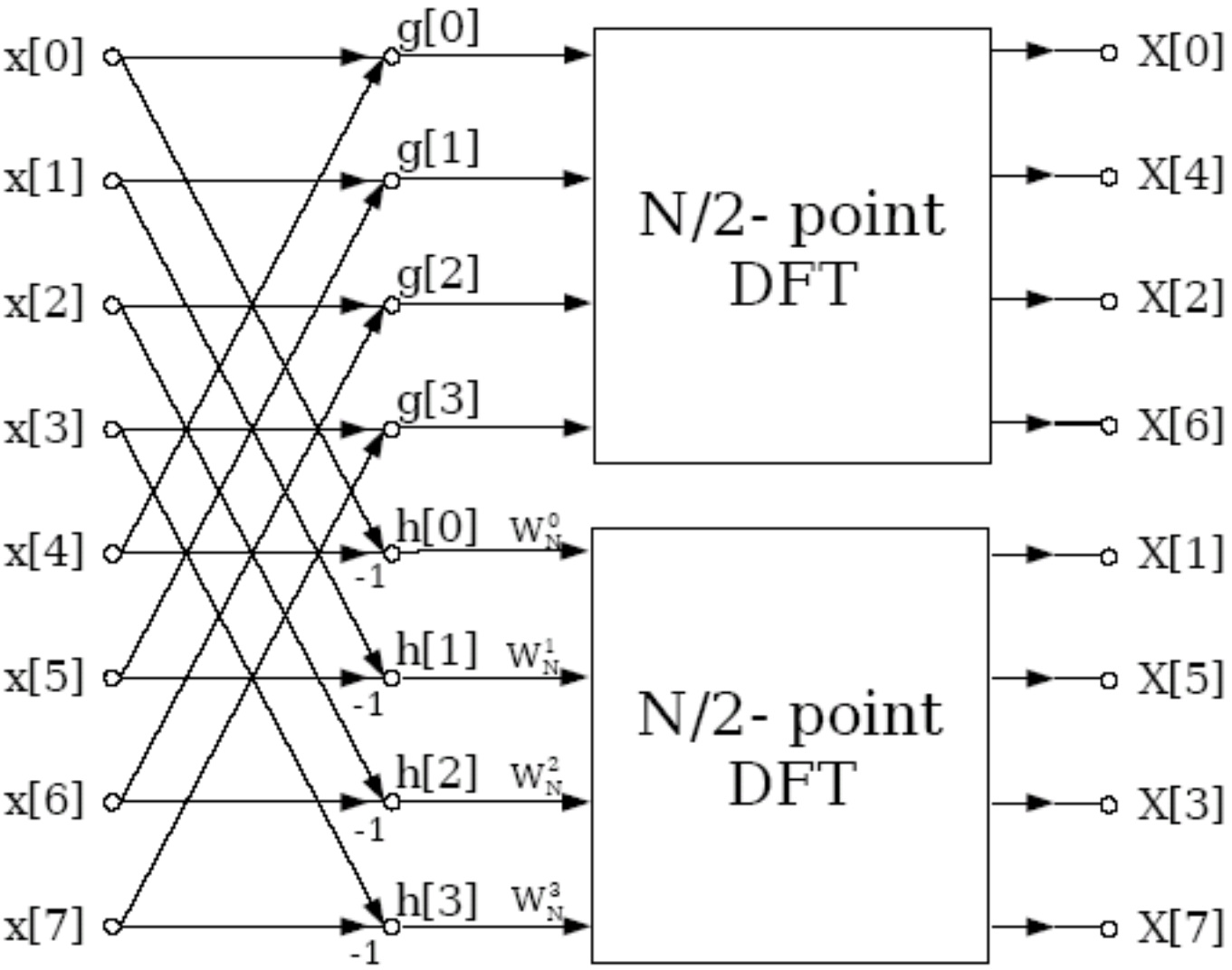
$$X(k) = \sum_{j=0}^{N-1} x(j) e^{\frac{2\pi i}{N} jk}$$

$$X(2r) = \sum_{j=0}^{N/2-1} x(j) e^{\frac{2\pi i}{(N/2)}(2r)j} + \sum_{j=0}^{N/2-1} x(j+N/2) e^{\frac{2\pi i}{(N/2)}(2r)j}$$

$$X(2r+1) = \sum_{j=0}^{N/2-1} x(j) e^{\frac{2\pi i}{(N/2)}(2r+1)j} - \sum_{j=0}^{N/2-1} \left[x(j+N/2) e^{\frac{2\pi i}{N} j} \right] e^{\frac{2\pi i}{(N/2)}(2r+1)j}$$

- "Decimation in frequency"
  - Sande-Tukey Algorithm

Graphically



So for a 3D FFT

- The method is obvious!
  - Do the x transforms in parallel
  - Then do the y transforms
  - And finally the z

- No nassty transposes
  - So no mpi_alltoallvs
    - Much longer messages
  - Communication purely along the principle directions of the process grid
  - If 512=8*8*8 cores only need to get the 1D transforms to scale to 8 cores
  - BUT more data needs to be sent

So DaFT for the 1D transforms...

- $Log_2(P)$ communication steps
  - Obviously communicate all at once
    - Sending V/P amount of data
  - Big messages
    - Doing many FFTs at once
- Followed by a standard serial FFT library call

- Inverse FFT can be done by "Decimation in Time" – essentially the reverse of the above
- First a standard serial FFT library call
- Then $Log_2(P)$ communication steps
  - Also undoes the reordering, but won't go into this

But

- This only works on powers of 2 numbers of processes
- And the problem size might not naturally fit onto such a number of processors
  - Restrictions due to cutoff
- Scientist
  - Might use more procs than he/she needs
    - Poor scaling?
  - Might scale up the problem size to fit
- Both a waste of computer budget
- Thus dCSE funded to reduce this restriction on core counts

Work Funded

- Funded: to allow $P=2^{n}x3^{0-2}x5^{0-1}$

- What actually done – allow P to be *any* number
  - But performance will be poor if includes a large prime factor

Mixed Radix

$$X(k)=\sum_{j=0}^{N-1} x(j)e^{\frac{2\pi i}{N}jk}$$

$$X(N_2k_1+k_2)=\sum_{j_1=0}^{N_1-1}\left[e^{\frac{2\pi i}{N}j_1k_2}\right]\left(\sum_{j_2=0}^{N_2-1} x(N_1j_2+j_1)e^{\frac{2\pi i}{N_2}j_2k_2}\right)e^{\frac{2\pi i}{N_1}j_1k_1}$$

So if have multiple factors

- So I split the prime factors into two sets
  - Short
    - Small primes
    - Usually raised to high powers
    - Done by full FFT algorithm
    - Code designed to make this set easily added to
  - Long
    - Larger primes
    - Usually raised to low powers
    - Done by simple DFT algorithm
    - Note if the power raised to is 1 DFT=FFT
- Note though any number of cores the length of the FFT must still be a multiple of P
  - Not in practice a problem

An Example

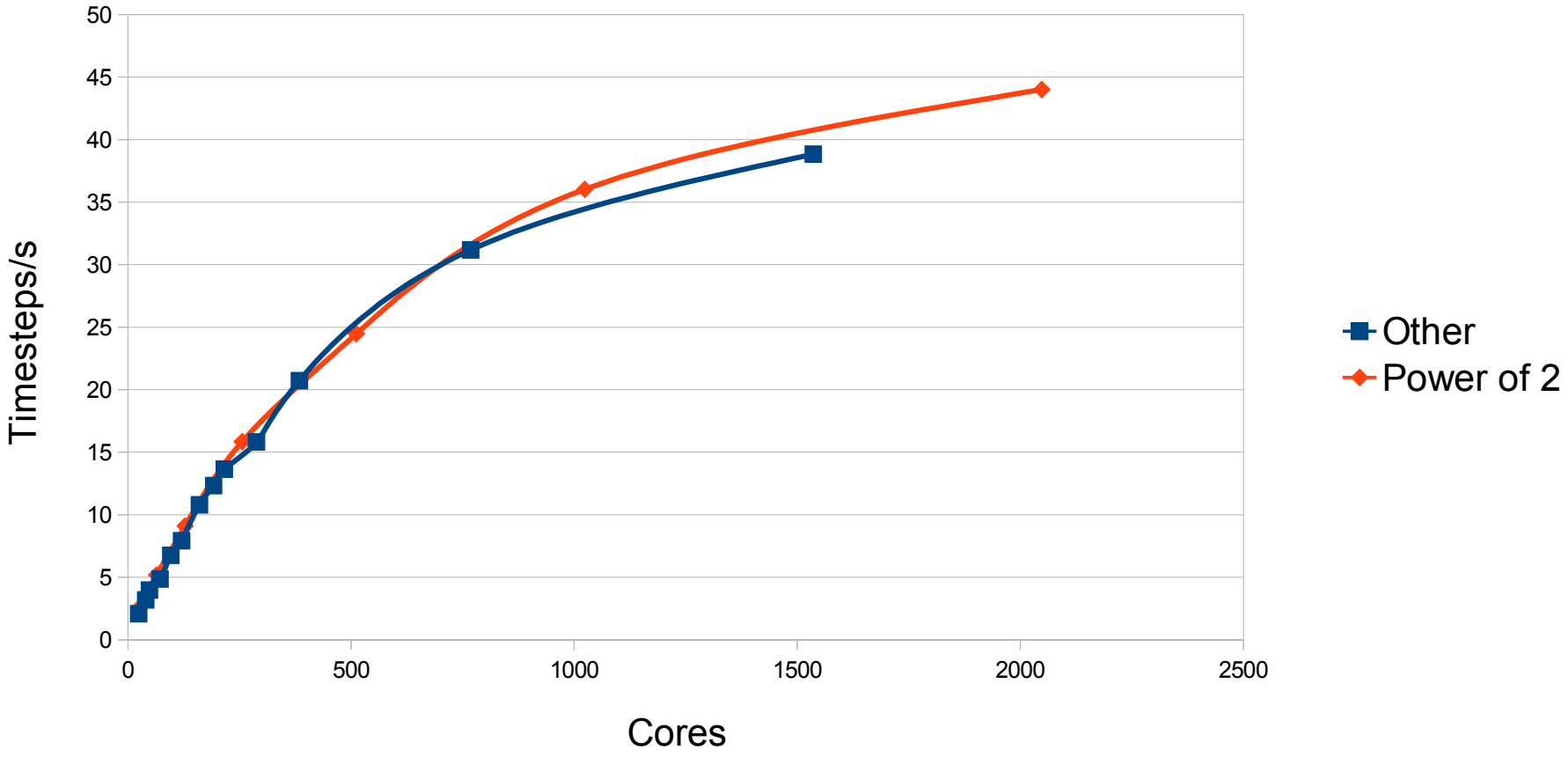
- If $P_x = 2^n * L$
  - First do 2^n DFTs of length $N_x/2^n$ each split over L cores
    - Simultaneously
    - Need to circulate data – double buffered systolic loop found best
  - Next apply the phase ("twiddle") factors
  - Finally do L FFTs of length N_x/L each split over 2^n cores
    - Simultaneously
- And then similarly for y and z

Other Issues

- The domain decomposition algorithm in DL_POLY had to be generalised to any number of cores
- Recast as an optimisation problem
  - Want to minimise amount of data sent
  - Related to the the sum of the surface areas of the domains
    - c.f. Halo exchange
  - Nice side effect of 3D, tends to factorise the long factors – $1728 = (2^2*3)*(2^2*3)*(2^2*3)$

So How Well Does It Work

- If not a power of 2 how hard hit is the performance?



Briefly

- Other work by Valène Pellissier focused on profiling and optimising DL_POLY
  - Loss in performance as the number of cores in the node has increased
- Valène is now based in France so I'll give a quick overview of some of her work
  - Full report going up on the web page

Profiling

- DL_POLY_4 was profiled extensively
  - Main tests on a large biomolecule
- Problem areas identified include
  - Frozen atoms in general
    - Conditionals in inner loops but most simulations have no frozen atoms
  - Link cell pairs
    - Conditionals in inner loops avoided by slight change of algorithm
  - Ewald routines
    - Improved vectorisation – see later
  - Constraints
    - Build list of constrained atoms avoids conditionals in inner loops

Ewald Summation

- Problem: Inner loop short and of variable length (typically 1-12)
- Solution:
  - Length of loop can be determined outside main loop nest
  - Therefore use a select construct to choose outside the loop an unrolled version
  - Allows vectorising over both inner and second innermost loop
  - Early compiler versions dramatic improvements: 189s → 109s!
  - Compilers getting better but still ~20% improvement to do it manually.

Overall Improvement

- Rather case dependent but each improvement gives around 10-35% improvement
  - Depends on number of cores as well
- So whole run times around 10-25% better due to Valène's work
  - XE6 now faster than the XT4!

Summary

- Through dCSE funded work DL_POLY_4 has
  - Been made more flexible by removing restrictions due to the FFT
    - Can run on any number of cores now
  - Run faster due to identifying hotspots and optimising