THE UNIVERSITY *of York*

High Performance Computing - Example Applications

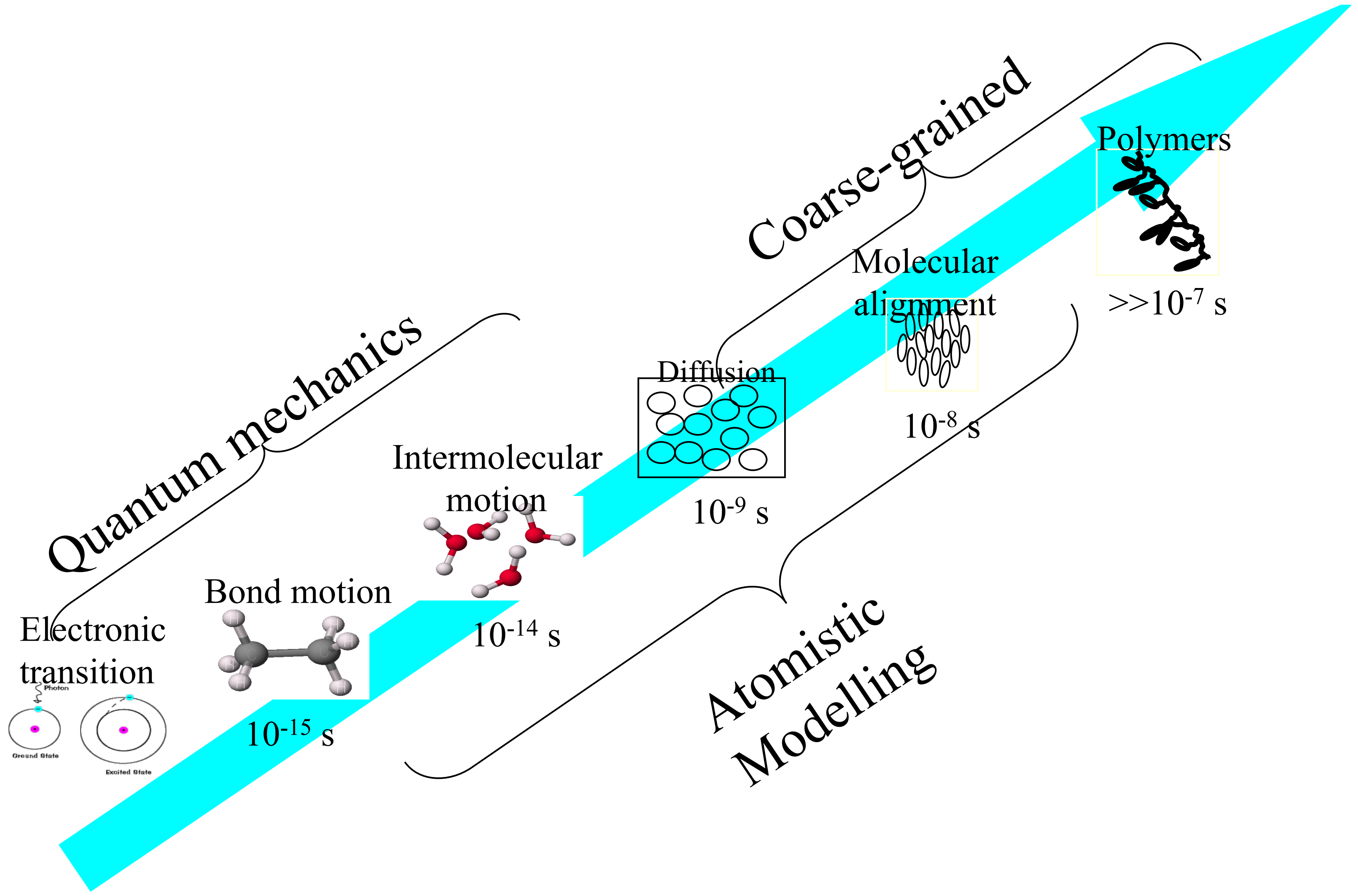
Prof Matt Probert

http://www-users.york.ac.uk/~mijp1

Overview

- Parallel Molecular Dynamics
- Parallel Quantum Mechanics

Length and Time scales

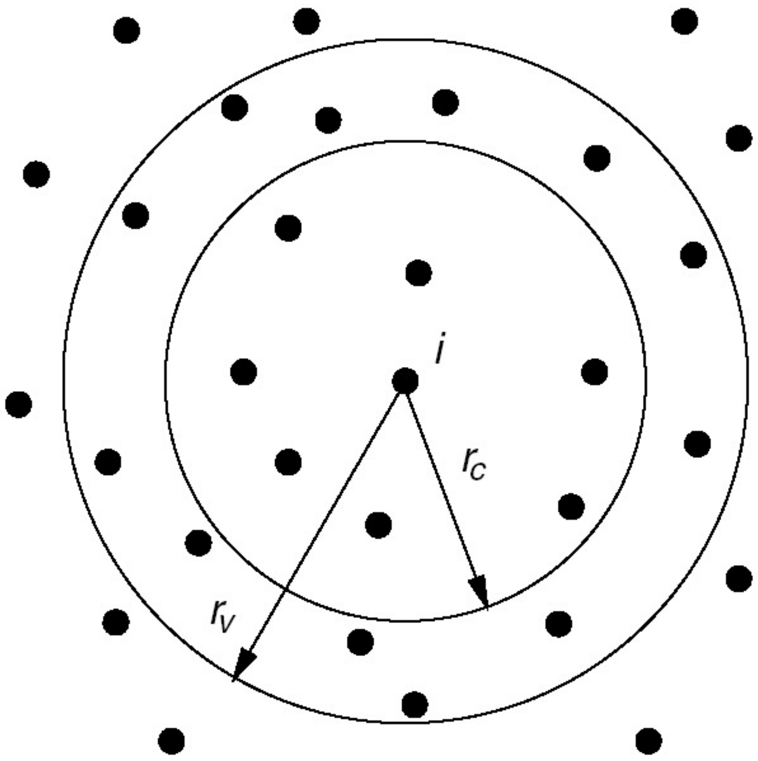


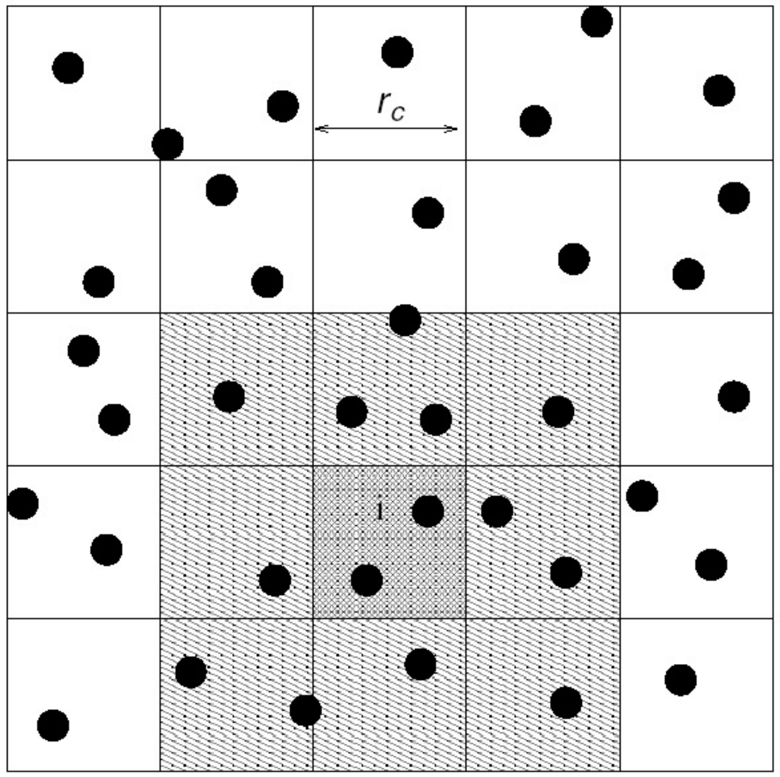
MD Basics

- Integrate the Newtonian equations of motion for atomic positions and momentum to generate a trajectory in phase space
  - Different regimes correspond to different models for calculating the forces on the atoms
  - Often it is this force calculation that is the most time consuming part of the calculation
- Hence need to minimise the number of force calculations and/or make them as fast as possible – need ~10^6 MD steps for a reasonable calculation!

Speeding up Serial MD (I)

- Pair-wise forces:
  - N(N-1)/2 calculations so scaling ~$O(N^2)$
  - But if interaction is short-range then can exploit spatial locality to make this ~O(N) using Verlet neighbour-list and/or cell-list schemes:





Speeding up Serial MD (II)

- Long range interactions (e.g. Coulomb) and periodic boundary conditions look bad
  - suggests that particle will interact with all particles in cell AND all image cells to infinity!
  - Solution is to use Ewald technique to produce finite sum ~O($N^{3/2}$)
  - more advanced techniques (e.g. Smoothed Particle-Mesh Ewald) make this scale as ~O(N*log*N) or even better but approximate the exact answer …

Parallel MD

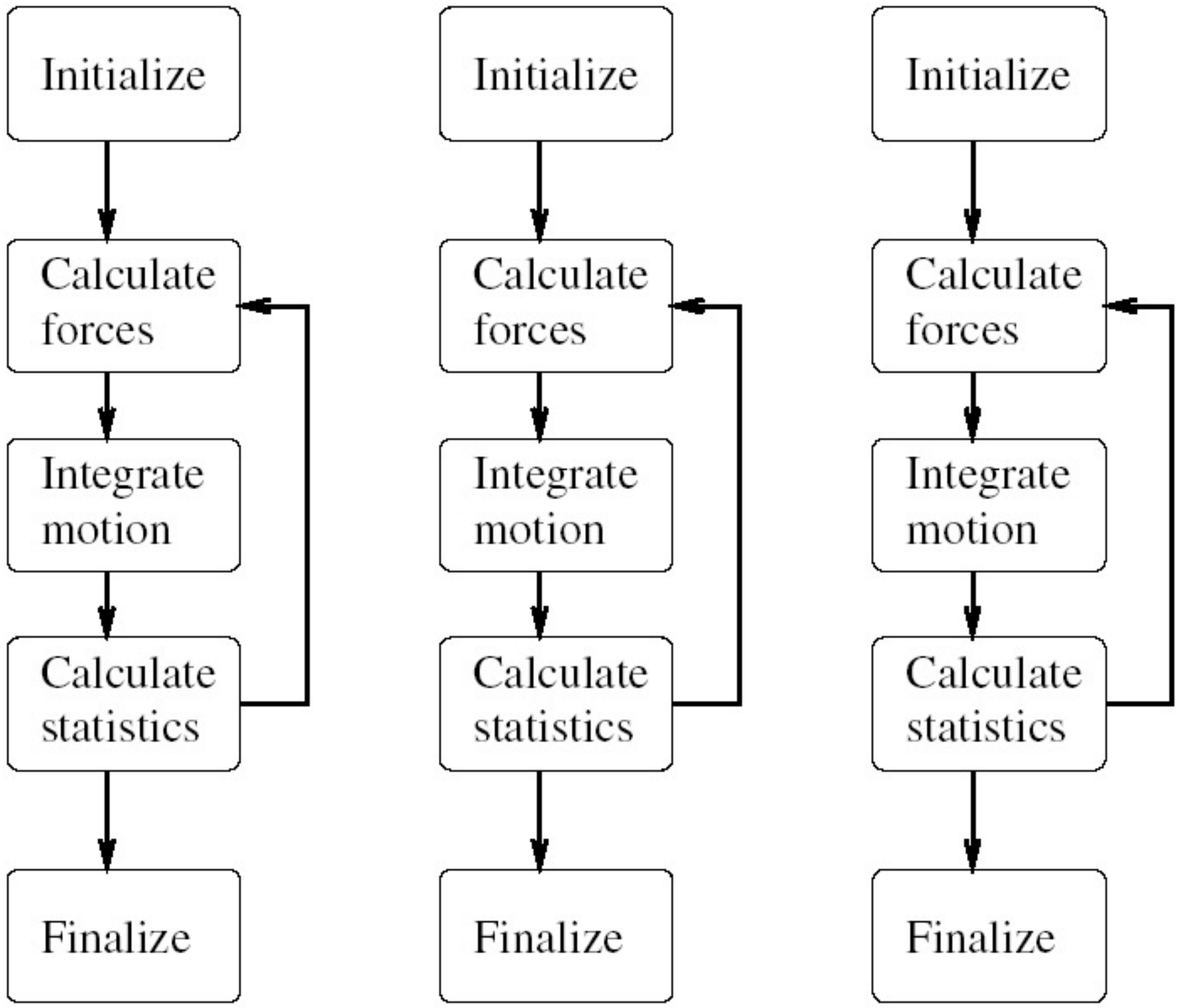
- Cloning
- Master-Slave
- Replicated Data
- Systolic
- Domain Decomposition

Cloning MD

Good way of generating ensemble averages

Run lots of different independent simulations.

Each processor must be capable of running entire simulation so limits size of problem that can be tackled.

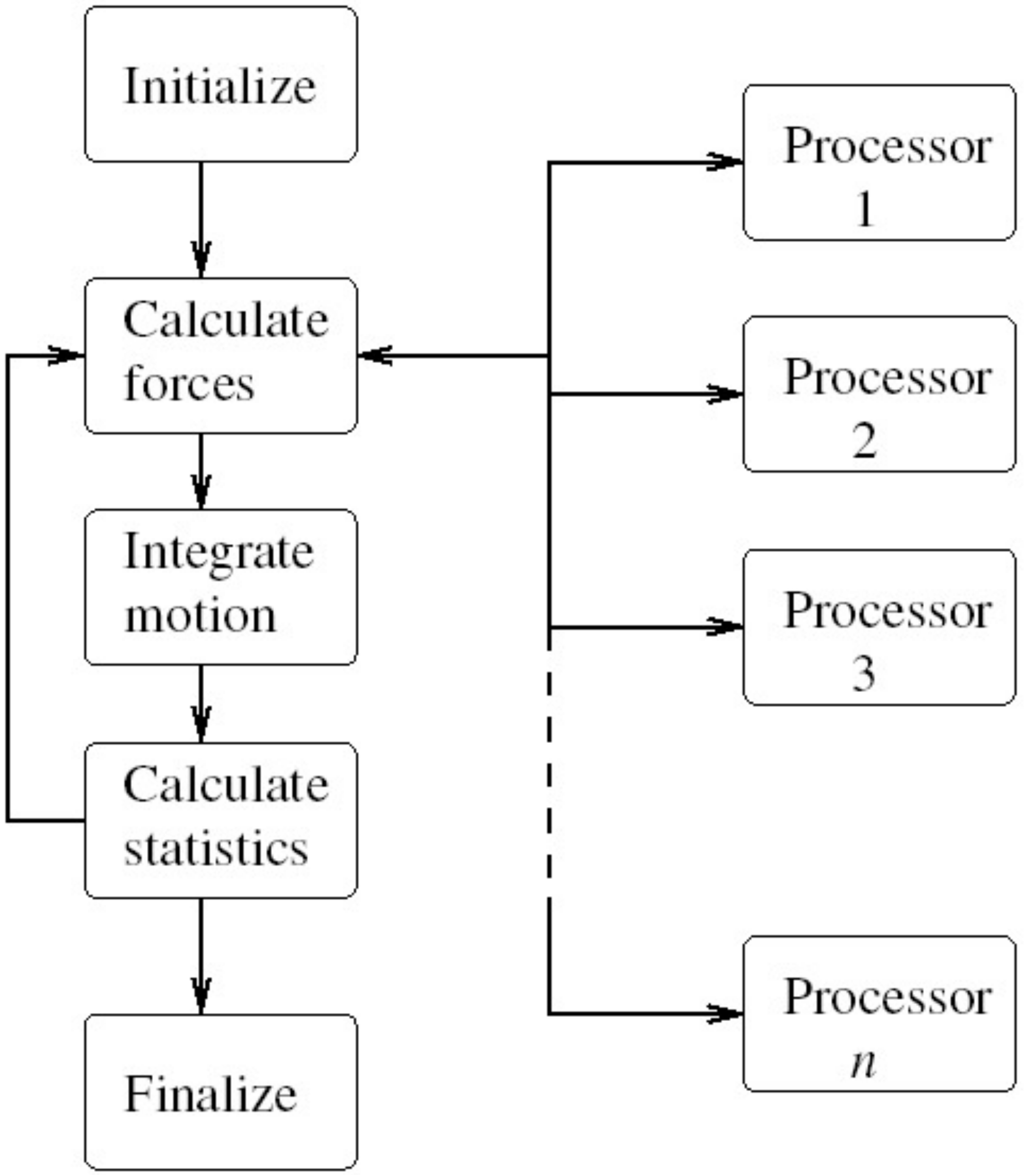


Master-Slave MD

Parallelise the bottleneck - the force calculation

Can be useful with complex force models, e.g. fully QM, especially if distribute data so can tackle larger problems. As used in CASTEP!

May require complex pattern of communications to get good load balancing and global synchronisation to integrate equations of motion.

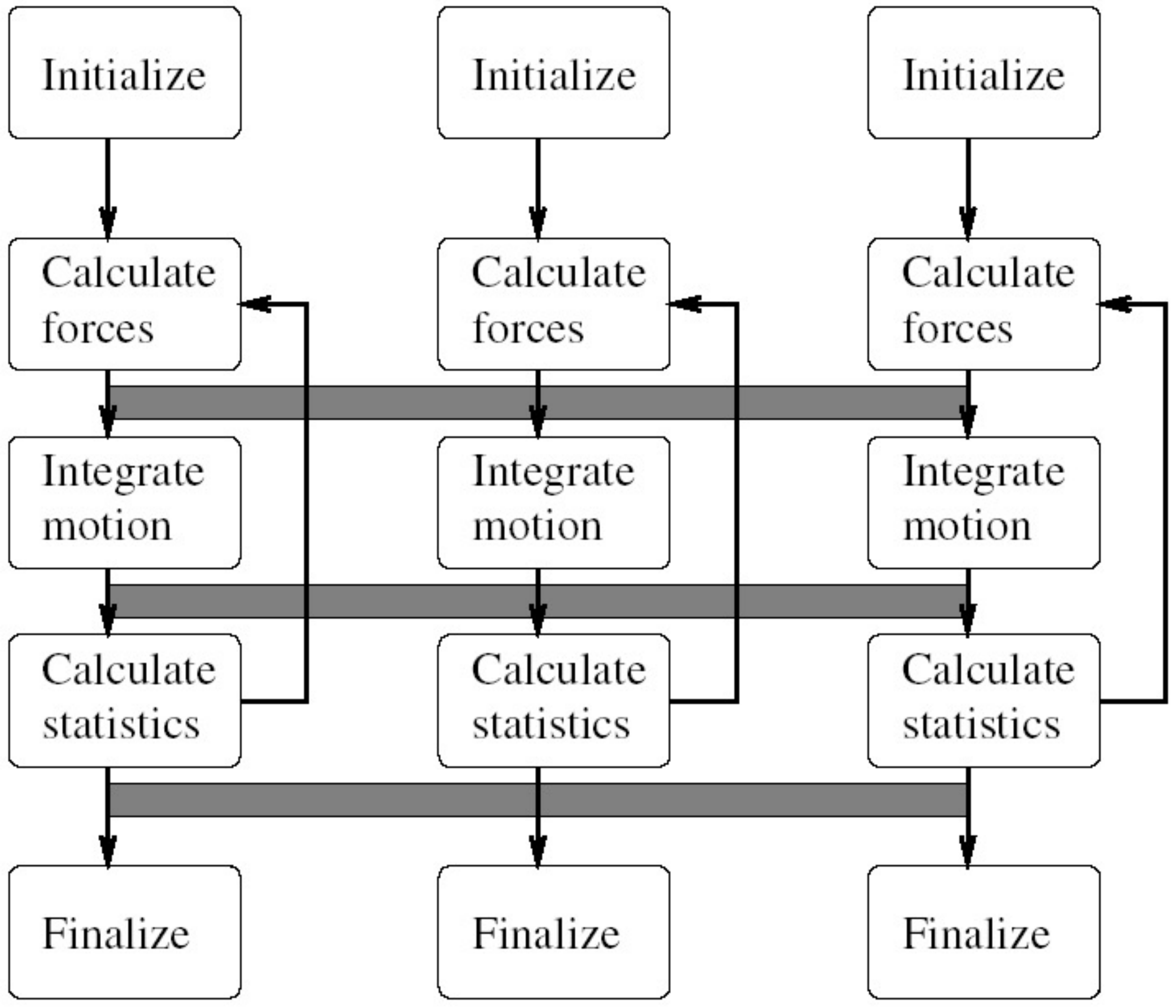


Replicated Data MD

Developed from cloning: each processor has complete set of data but only operates on subset.

E.g. in a pair-potential calculation, calculates N^2/n forces and then communicates to n-1 others. Then global sync as integrate motion.

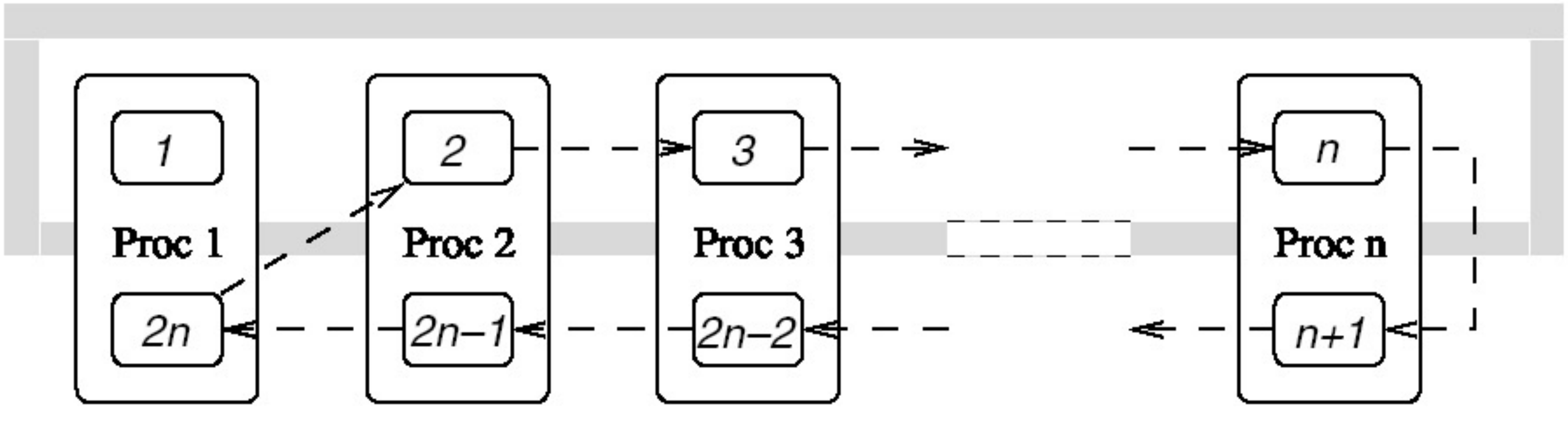
Limited problem size and poor comms strategy.



Systolic MD (I)

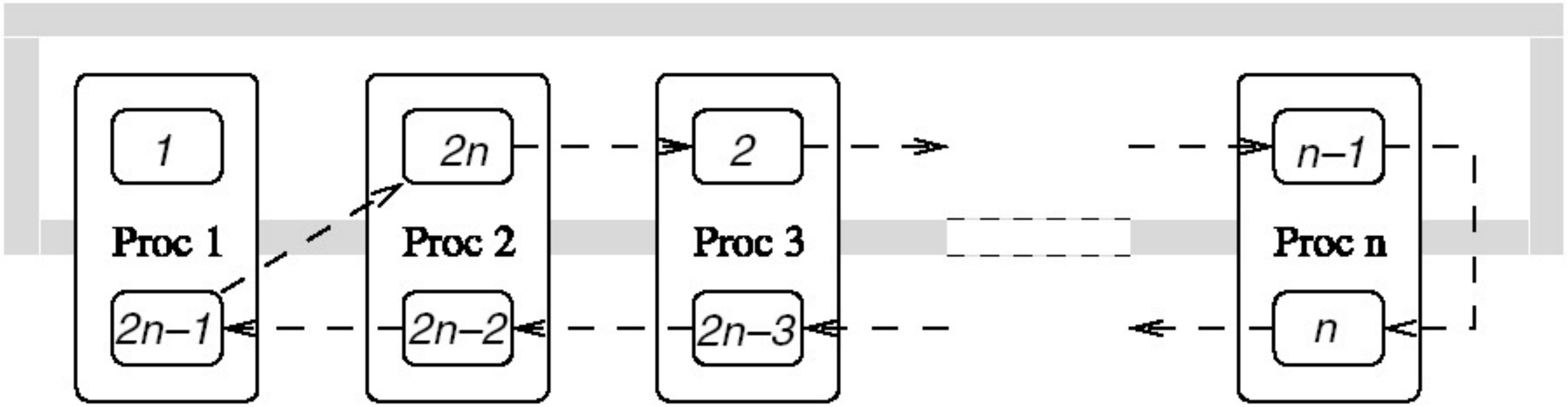
•Uses a (virtual) ring topology and “pulses” data like heart. Initially each processor has 2 packets of data:

•1st pulse: each processor calculates forces within its packets (intra-group), and between the two packets (inter-group). Then every processor (except one) then passes one set of data to its left neighbour and receives in its place the corresponding packet from its right neighbour. It also sends its other packet to its right neighbour and receives one from its left neighbour, as shown:



Systolic MD (II)

•Subsequent pulses: Above process repeats until every processor has received in turn a copy of every packet.



•At the end of this sequence, each processor has calculated the full set of forces and can therefore integrate the equations of motion. A single time step therefore consists of 2n – 1 systolic pulses, after which all the packets are back where they started.

•Good scaling and load balancing but complex to code.

Domain Decomposition MD

- Based upon cell-list approach:
  - Physical system is divided into a number of contiguous domains, which are then mapped onto processors.
  - Each domain must be sufficiently large that the pair forces can be calculated independently for each domain hence need size of system >> interaction range
  - Use cell-list techniques to map the interactions between domains and processors. Equations of motion for each domain can then be integrated independently. Any particles leaving a domain are mapped onto the appropriate domain and processor at the end of the step.
- Good load balancing and simple comms but non-ideal scaling and difficult to apply to complex interactions.

Massively Parallel MD

- Domain decomposition
  - DL_POLY3 used domain decomposition with replicated data which limited scaling and had I/O bottleneck.
  - DL_POLY4 uses MPI-2 (includes parallel I/O) and fully distributed data so now scales much better[1] …
  - Largest(?) MD calculation (using LAMMPS) has 19 billion atoms[2] and also uses this technique.

1. M. F. Guest, A. M. Elena and A. B. G. Chalk "DL_POLY - A performance overview analysing, understanding and exploiting available HPC technology", Mol. Sim. **47**, 194-227 (2021)
2. K. Kadau, T.C. Germann and P.S. Lomdahl, "Large-scale MD simulation of 19 billion particles", Int. J. Mod. Phys. C **15**, 193-201 (2004)

Parallel Quantum Mechanics

- Many different levels of approximation
  - "*ab initio*" means "from the beginning", i.e. calculation using QM without any empirical input, only atomic number of atoms and approximate arrangement
- Density Functional Theory is a very popular *ab initio* scheme – allows efficient first principles calculations of systems containing 100s of atoms on PC or 1000s on parallel computer
- CASTEP is a DFT code developed within UK and one of the flagship codes on successive UK supercomputers
  - Designed to be parallel from the beginning
  - And also run on serial machines – had best single CPU %peak of any of the HPCx flagship codes (lots of BLAS)

Density Functional Theory

- An exact theory for finding the ground state charge density, $\rho(\mathbf{r})$, of any system of atoms
  - Has one basic approximation to make it tractable – the exchange-correlation functional
  - Kohn-Sham equations for $\rho(\mathbf{r})$ need to be solved "self-consistently" as Hamiltonian itself depends upon $\rho(\mathbf{r})$
  - Scheme is made computationally tractable by expanding $\rho(\mathbf{r})$ in terms of "single-electron wavefunctions" $\psi(\mathbf{r})$
  - The $\psi(\mathbf{r})$ are then expanded in terms of coefficients and a known set of basis functions
  - Hence Schrödinger equation as a matrix equation!
  - Hence "diagonalising the Hamiltonian" etc.

Periodic systems - Bloch's Theorem

- Recall that Bloch's theorem lets us write:

$$\psi_k(\mathbf{r}) \;=\; e^{i\mathbf{k}.\mathbf{r}} u_k(\mathbf{r})$$

- Where $u_k(\mathbf{r}+\mathbf{L}) = u_k(\mathbf{r})$ is periodic and $e^{i\mathbf{k}.\mathbf{r}}$ is an arbitrary phase factor. We express $u_{bk}(\mathbf{r})$ as a Fourier series:

$$u_{bk}(\mathbf{r}) \;=\; \sum_{G} c_{Gbk} e^{i\mathbf{G}.\mathbf{r}}$$

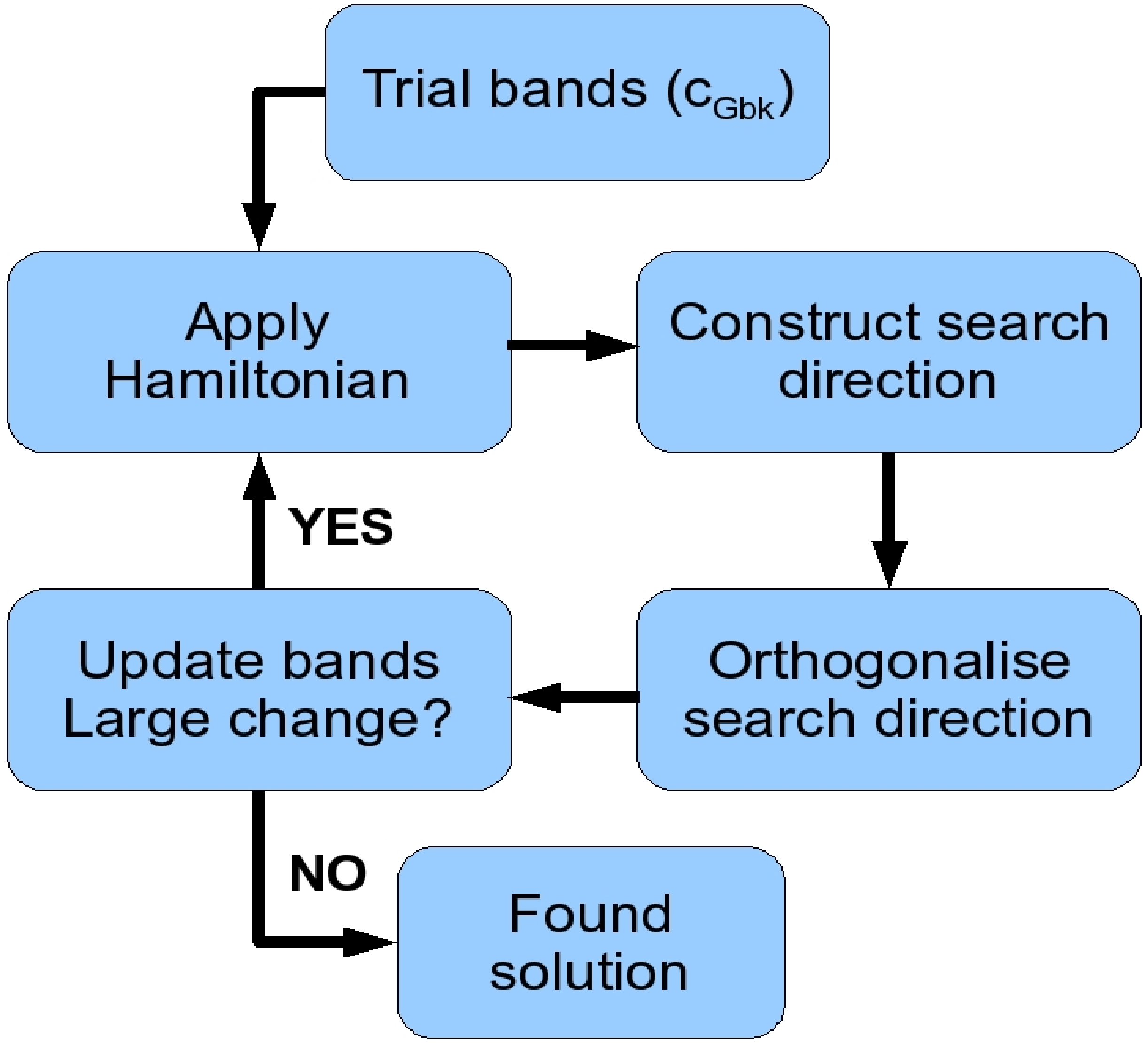
- Where c_{Gbk} are complex Fourier coefficients.

The wavefunction

$$\psi_{b\mathbf{k}} = \sum_{\mathbf{G}} c_{Gb\mathbf{k}} e^{i(\mathbf{G}+\mathbf{k}).\mathbf{r}}$$

- The complex coefficients c_{Gbk} are what CASTEP computes by solving $\hat{H}[\rho]\psi_b = E_b\psi_b$
- Storing these coefficients needs a lot of the computer's memory (RAM).
- **G**: a reciprocal lattice vector ("G-vector")
- b: a band index
- ***k***: a Brillouin zone sampling point ("***k***-point")

Where does CASTEP spend its time?


Trial bands (c_{Gbk})
Apply Hamiltonian
Construct search direction
Orthogonalise search direction
Update bands Large change?
YES
NO
Found solution

CASTEP bottlenecks

- Applying *H*
  - Kinetic energy applied in reciprocal-space
  - Local potential applied in real-space
  - need to (fast) Fourier transform between the two spaces.
- Orthogonalising wavefunctions
  - Need to ensure trial bands are orthogonal
  - Compute the band-overlap matrix, and transform to create an orthonormal set.

Parallelising DFT (I)

Three possible strategies:

1. Distribute real- and reciprocal-space vectors
   - Distributed data => good for memory
   - Good scaling as number of real- and reciprocal-space vectors increases with increasing problem size
   - Operations with G-vectors in reciprocal-space are independent of operations on $\rho(\mathbf{r})$ in real-space

     $$\rho(\mathbf{G}) \leftrightarrow \rho(\mathbf{r})$$
   - Requires 3D parallel FFT to transform
   - Hence complex comms strategy

Parallelising DFT (II)

2. Distribute k-points
   - Most operations at different points in the Brillouin zone are independent of other k-points
   - Distributed data => good for memory but limited scaling as number of k-points reduces with increasing problem size
3. Distribute bands
   - In a non-spin-polarised system, each $\psi_j(\mathbf{r})$ represents a different band
   - Number of bands will scale with system size but memory limited as each node has to store entire $\rho(\mathbf{r})$ - unless combine with g-vector parallelism

- CASTEP can use all 3 in any combination as appropriate
  - hence can scale to 1000s of CPUs on large machines

CASTEP Data Distribution (I)

- Reciprocal space data represented on grid
  - Grid size given by smallest **G** (=2π/|**a**| etc)
  - Largest **G** given by E_{cut}
  - Different **k** have independent **G** grids
- "g-vector group"
  - CASTEP-speak for those processors over which the data (for a given k-point) are distributed.
  - Multiple g-vector groups are possible if the calculation is parallelised over k-points.
  - Each g-vector group has the same number of nodes and has a "master node" to handle inter-group comms
  - Each g-vector group is a separate MPI communicator

CASTEP Data Distribution (II)

- “k-point group”
  - CASTEP-speak for those processors over which the data is k-distributed.
  - The same real and reciprocal space data will be stored on each node in the group but will correspond to a different k-point.
  - Multiple k-point groups are possible if the calculation is parallelised over g-vectors.
  - Each k-point group has a “master node” to handle inter-group comms
  - Each k-point group is a separate MPI communicator
- “Root node”
  - CASTEP-speak for the primary I/O node. There is only one root node and it handles all comms to k-point masters and g-vector masters.

CASTEP Data Distribution (III)

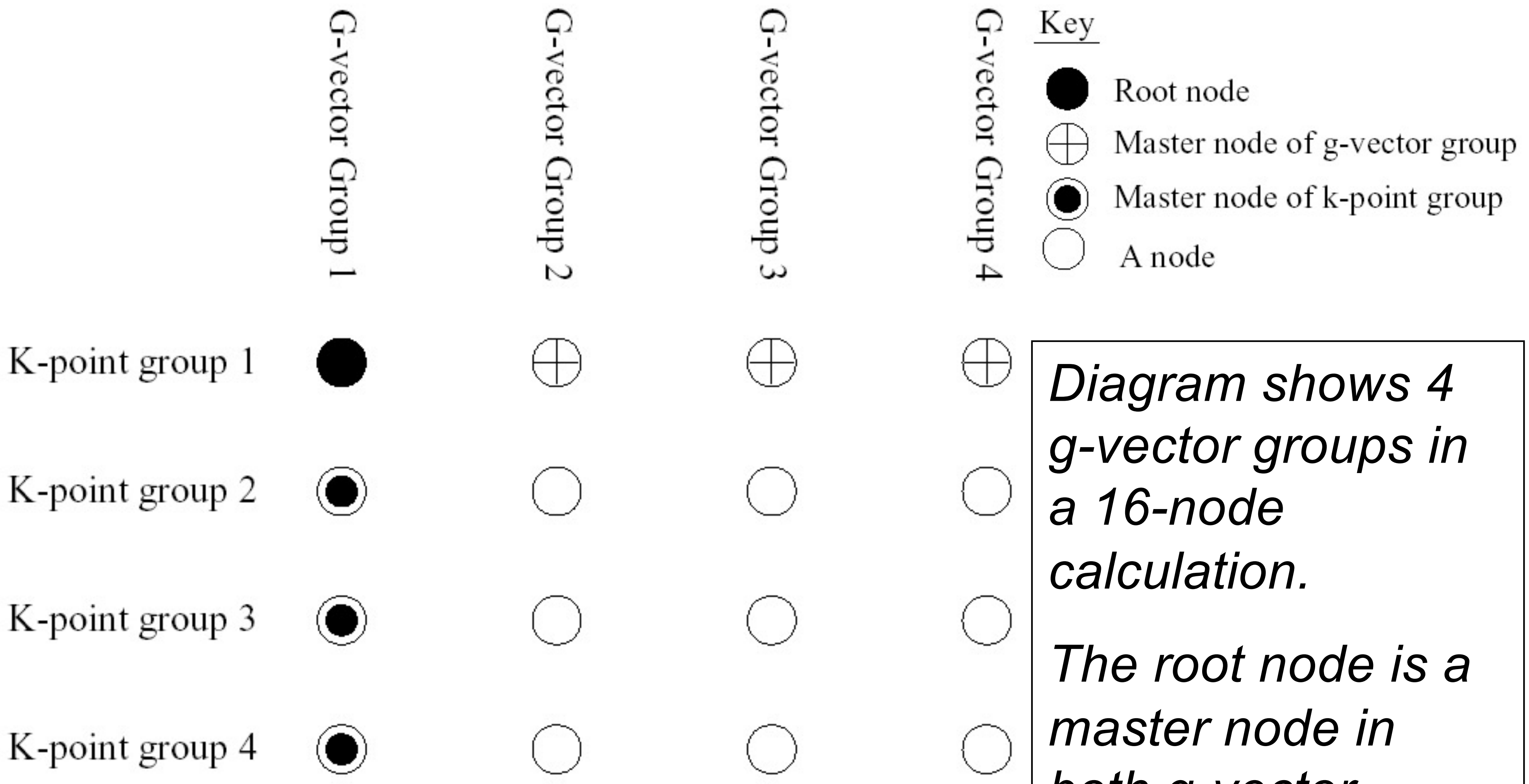


Diagram shows 4 g-vector groups in a 16-node calculation.

The root node is a master node in both g-vector group1 and k-point group1.

CASTEP Design (I)

- Code a total rewrite of an earlier (serial) F77 code – necessitated by inability to further develop and/or maintain code.
- Lot of effort went into code design before coding began
  - Wrote a specification (over 400 pages long) detailing functionality and requirements of each module, and all public data-types and subprogram interfaces
  - Modular F90 with strict hierarchy (now over 650,000 lines long) developed at 6 different academic sites in UK – hence use of mercurial and need for strict design rules
  - Each module "owned" by a single academic – can choose own internal algorithms and responsible for all coding, testing and debugging for that module.

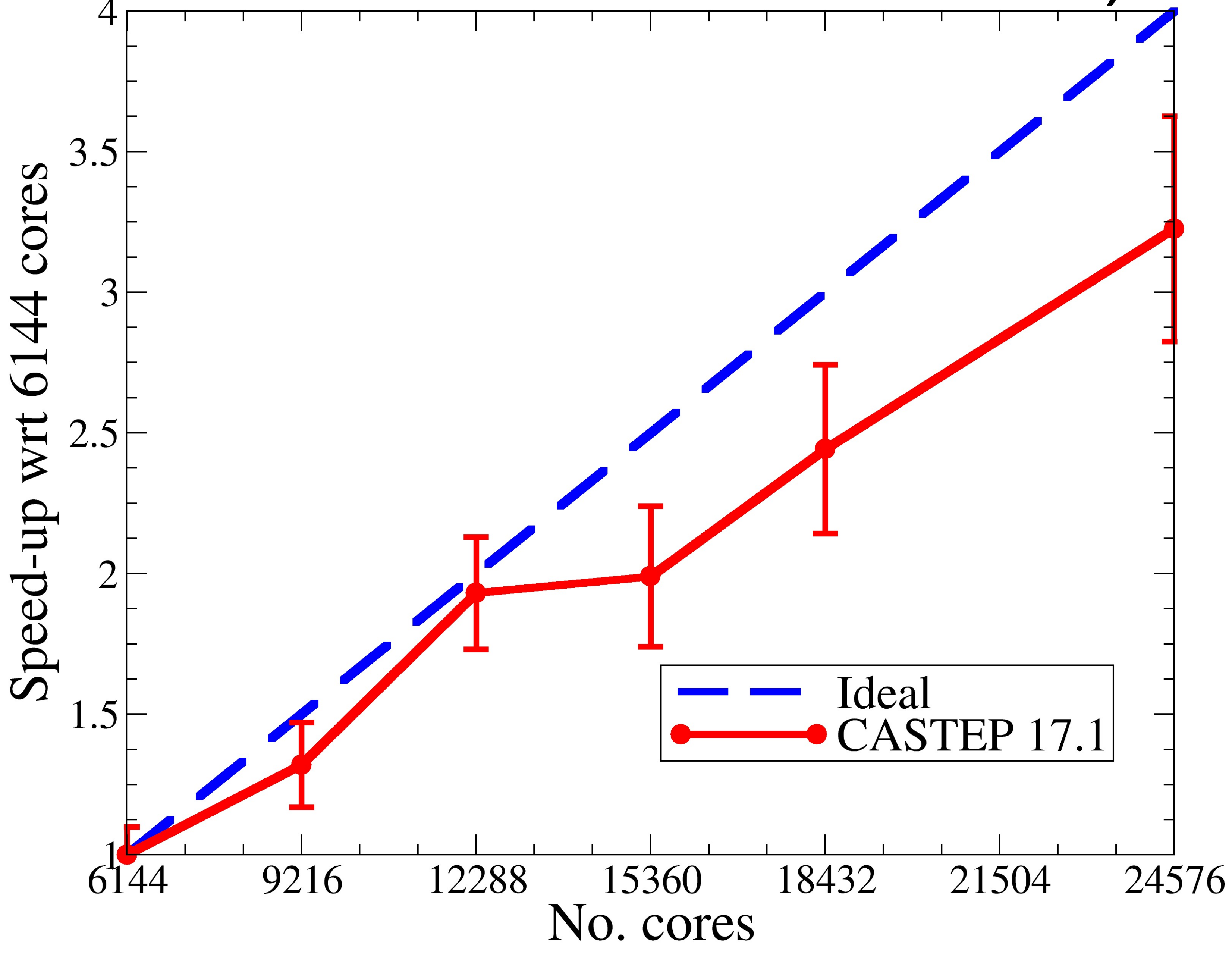
CASTEP Design (II)

- OOP-like within modern Fortran restrictions
  - Utility modules define low-level operations and things which may be machine specific, e.g. `io` module for handling all file I/O, or `FFT` module which may use different vendor-specific libraries.
  - Fundamental modules define basic types and operations, e.g. `wave` module defines a `wvfn` type and all the operations that can be performed upon that type, e.g. `wvfn_add, wvfn_rotate`, etc
  - Functional modules put the physics into action by using the lower level modules/types, e.g. `Hartree` uses `density` to calculate the electron-electron interaction energy/force, etc.

CASTEP – Serial or Parallel?

- So how can CASTEP be serial *and* parallel?
  - Not two separate source trees – avoid a nightmare!
  - Rather, one source tree and two versions of the `comms` module – `comms.mpi.F90` and `comms.serial.F90` – selected at compile time so have serial or parallel binary
  - Hence `comms` module provides "wrappers" to most useful MPI routines, etc. So can substitute serial versions or any other parallel library as long as provide same interface to rest of the code. Localises any required changes!
  - Hence rest of code does *not* do any communications directly, only by calling to `comms` routines.
  - And now has some OpenMP in certain modules to reduce memory usage per node and comms bottlenecks

Putting it all together (1500 atom DNA with b,G and threads)



Further Reading

- "*Molecular dynamics studies of liquids using a Beowulf computer*"

  M.I.J. Probert, *Contemporary Physics* **44** 435-450 (2003)

- "*First-principles simulation: ideas, illustrations and the CASTEP code*"

  M.D. Segall, P.J.D. Lindan, M.J. Probert, C.J. Pickard, P.J. Hasnip, S.J. Clark and M.C. Payne,

  Journal of Physics: Condensed-Matter **14** 2717-2744 (2002)

- "*First-principles methods using CASTEP*"

  S. J. Clark, M. D. Segall, C. J. Pickard, P. J. Hasnip, M.I. J. Probert, K. Refson, M. C. Payne,

  Zeitschrift für Kristallographie **220**(5-6) 567-570 (2005)