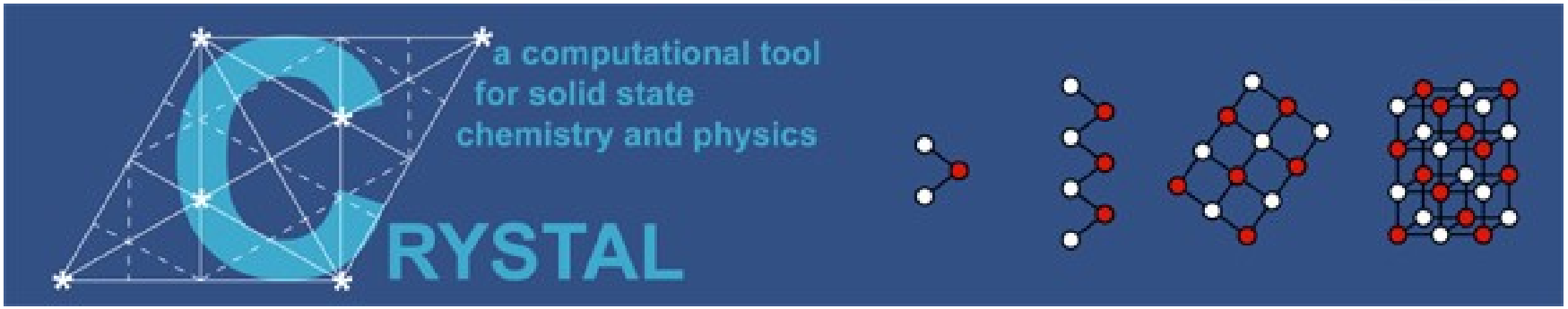
Implementation of a Divide and Conquer Linear Scaling Algorithm in the CRYSTAL Code

Daniel R. Jones
Numerical Algorithms Group

Acknowledgements:
STFC:
Barbara Montanari, Leonardo Bernasconi, Nic Harrison
NAG:
Ian Bush
Computing
HECToR, SCARF

Outline

- An introduction to CRYSTAL
- Motivation
- Divide and conquer scheme –
  What does the algorithm look like?
- Task level parallelism in CRYSTAL
- A very simple test case – first results.
- Continuations
- Conclusions



Basis Set

- LCAO – Gaussians type orbitals
- All electron or pseudopotential

Hamiltonian

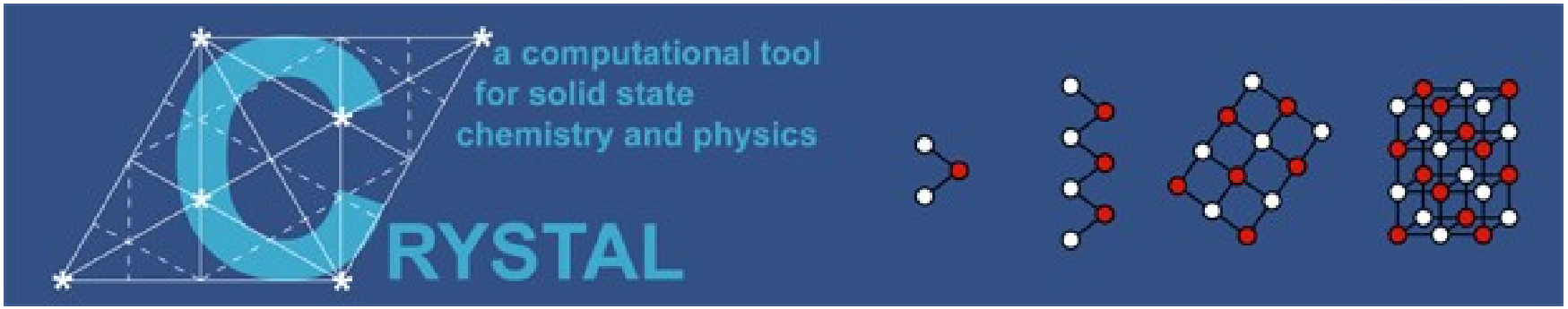
- Hartree-Fock (UHF, RHF)
- DFT (LSDA, GGA)
- Hybrid exchange functionals
- Periodic local MP2 via CRYSCOR

Techniques

- Replicated and massively parallel

Visualisation

- DLVizualise



Energy
Geometry optimisation
Vibrations (phonons)
Elastic tensor
Transition state searching
Ferroelectric polarisation
Piezoelectric constants
X-ray structure factors
Density of States / Bands
Charge / Spin Densities
Magnetic Coupling
Electrostatics (V, E, EFG)
Fermi contact (NMR)
EMD (Compton, e-2e)

TD-DFT (+ hybrids)
- Optical spectra
- Exciton binding energies

Coupled Perturbed HF/KS
- Static polarizabilty
- Dielectric tensor

Empirical dispersion terms

Substitutional & magnetic order

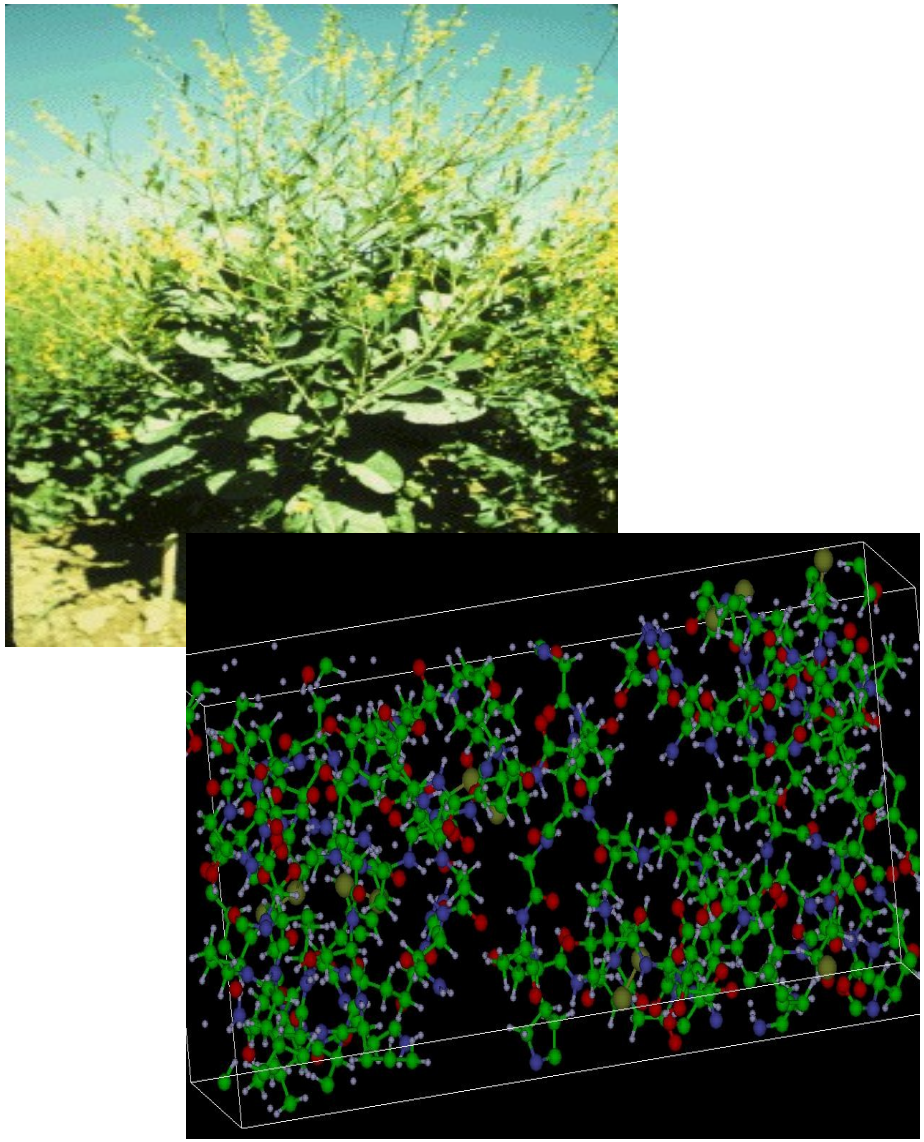
Quantum transport
- Landauer via WaNT interface

Helical symmetry for polymers

Auto-generation of nanotubes

CRYSTAL Example 1 – Crambin

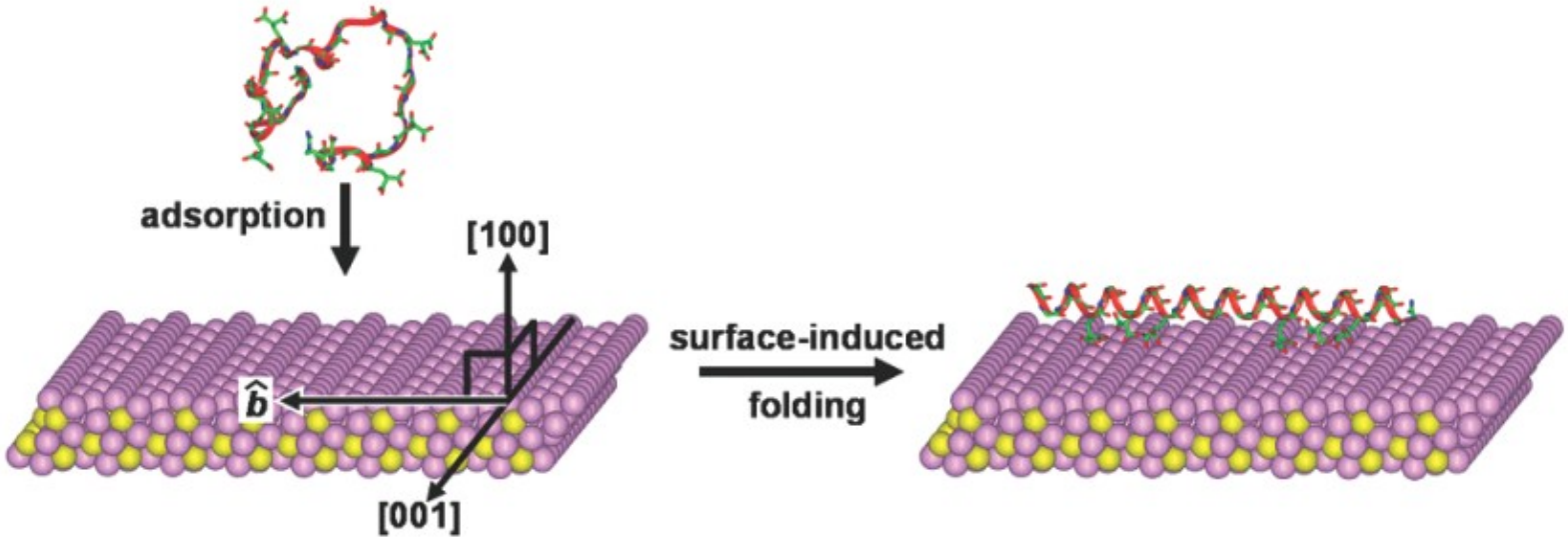
- 2 Chains in unit Cell
- 1284 Atoms
- 6-31G** basis set (12354 functions)
- All calculations B3LYP



Work by Ian J. Bush.

CRYSTAL Example 2 -

Helix stabilization by adsorption on hydroxyapatite



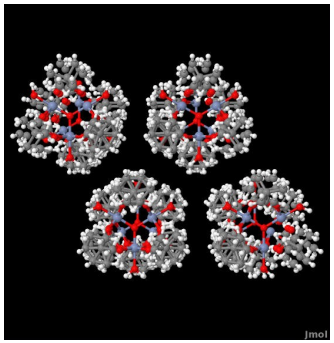
- **B3LYP /6-31G(d,p) periodic level with CRYSTAL06 code**
- **Only electronic energies (no T effects!)**
- **No dynamics (daunting task)**
- **Forced to use fixed peptide conformers (either random coil/alpha-helix)**
- **Protein and adsorption from gas-phase(!)**
- **Water studied as a microsolvation process**

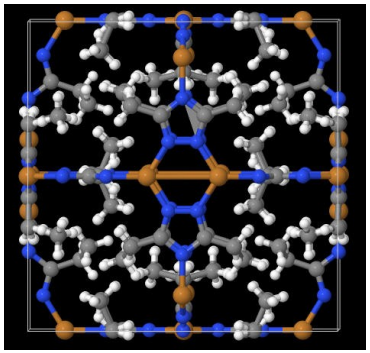
When do we need linear scaling?

- Traditional DFT algorithms scale as $O(N^3)$ – where N is some measure of the system size.
- Prohibitive at large system sizes, e.g:
  - Obvious cases, where a crystal has a large unit cell
  - Surfaces, can require a deep slab so the centre converges to bulk like.
  - Defects, can require a large super-cell to reduce the impact of the defect interacting with its periodic image.

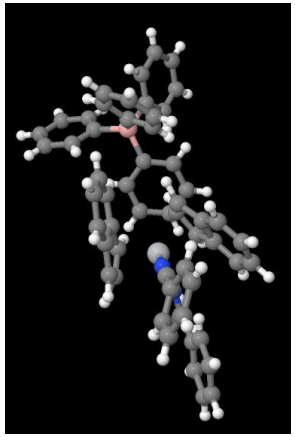
Symmetry

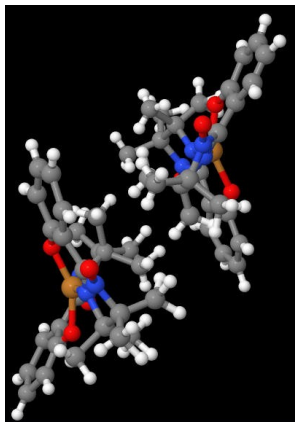
- CRYSTAL is very good at utilising symmetry to minimise computational expense.

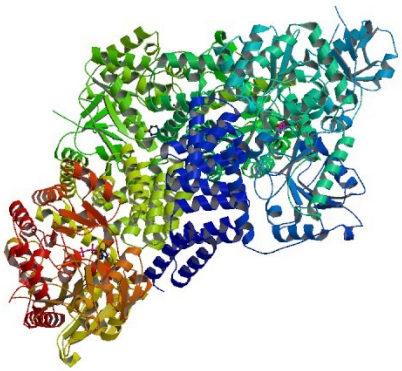




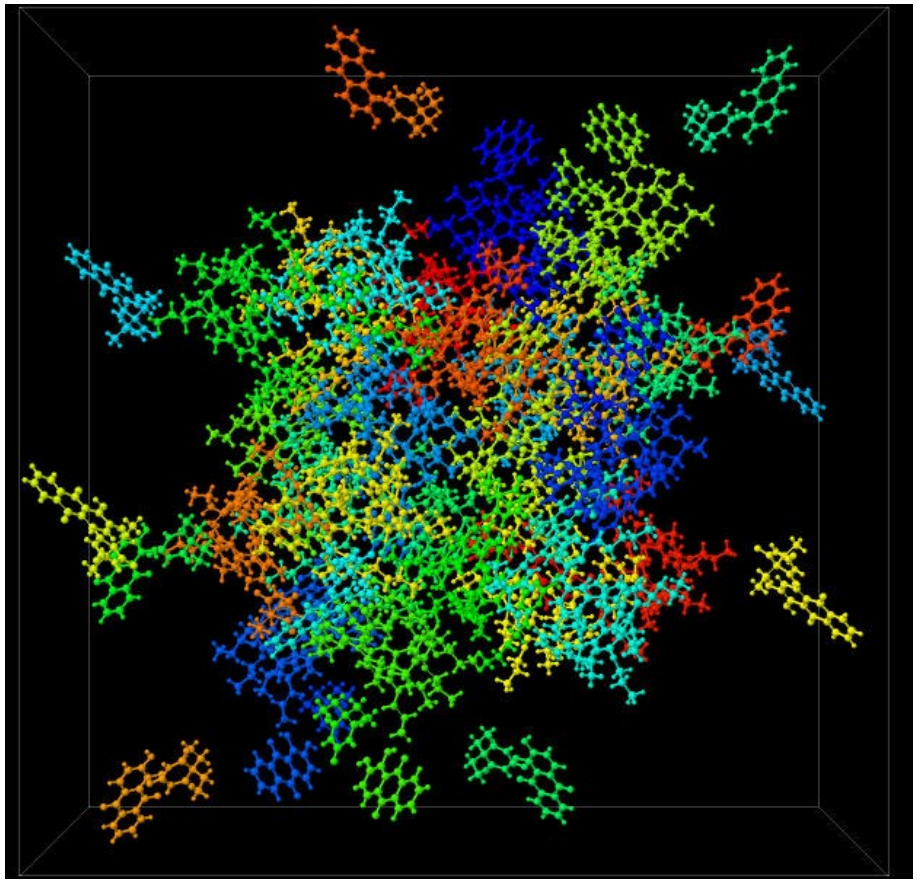
- However:
  - Many systems have very low symmetry (e.g. most proteins)
  - Symmetry is fragile!

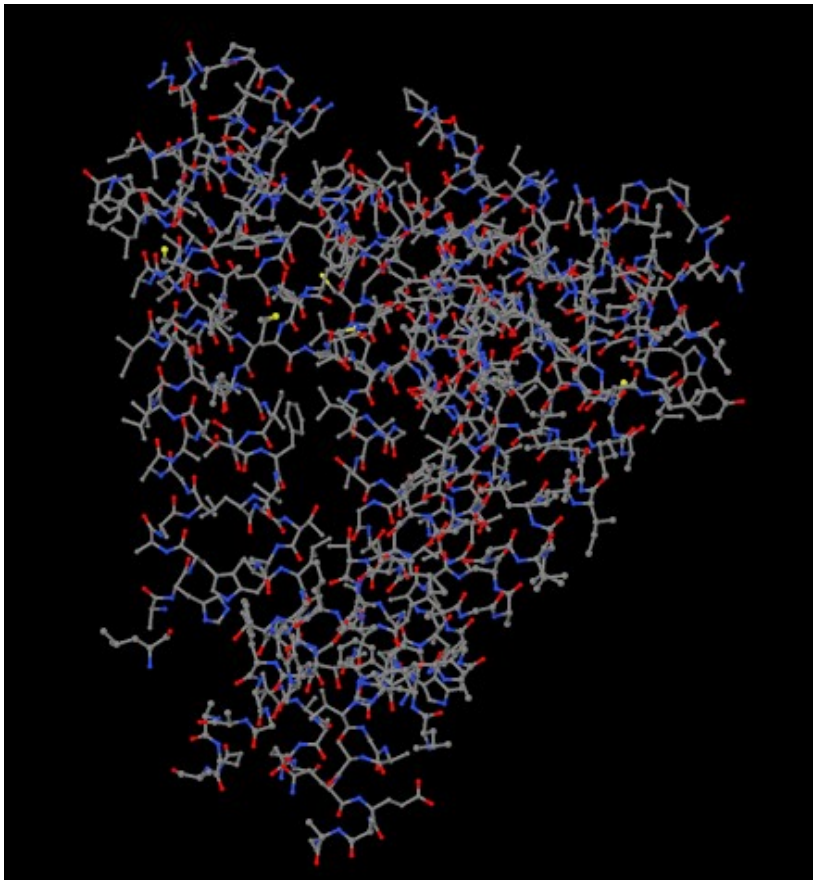






Target Systems





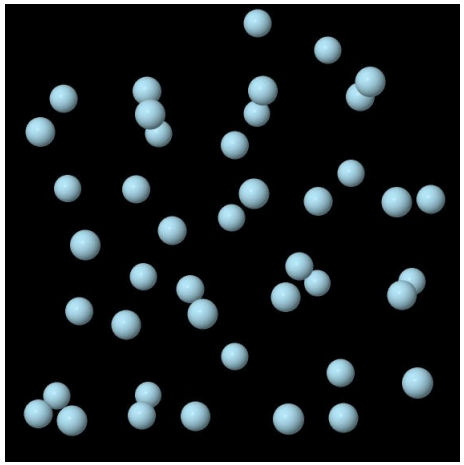
To make best use of the algorithm, the target stystems are large unit cell molecular crystals and (to a lesser extent) biomolecules.

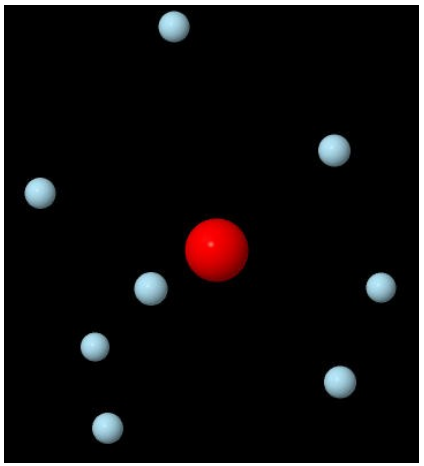
Divide and Conquer Algorithm

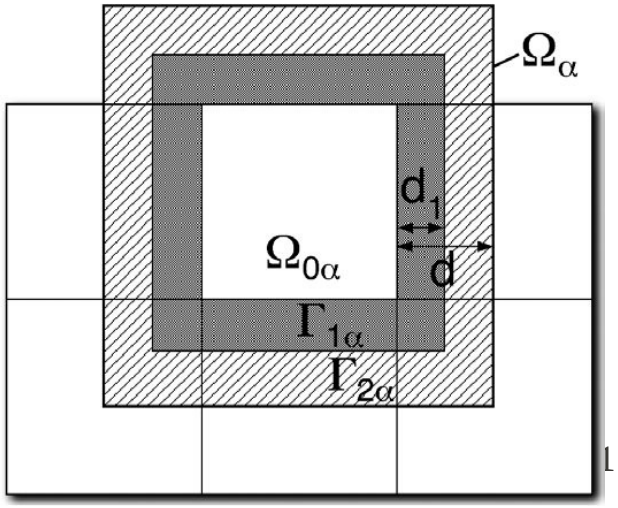
- How can we achieve linear scaling?
- Why is “conventional” DFT $O(N^3)$?
  - In CRYSTAL the Fock matrix in reciprocal space is diagonalised – an $O(N^3)$ operation.
  - Plane wave codes have a number of states which are generally delocalised across the whole system that must be mutually orthogonal.
- “Short sighted” localised electrons
  - Result of destructive wave interference in the presence of many particles.
  - Usually (but not universally) true
  - Already assumed in a lot of chemistry and materials science.

Dividing the system into subsystems

- Use the short-sightedness of electrons. Electrons only interact with other electrons at (relatively) small separations.
- System is separated into subsystems.
- Simplest way (used here and in SIESTA) is to have 1 core atom per subsystem.
- More sophisticated (and sensible) divisions are possible and should be implemented later.







Scaling of Order N In Crystal

- Implemented algorithm named SONIC, to find the electronic structure by divide and conquer.
- The system is divided into a set of subsystems, {α}, and a partition function P_{ij}^{α} is defined.

$$\sum_{\alpha} P_{ij}^{\alpha} = 1$$

$$P_{ij} = 1, i \in \alpha \wedge j \in \alpha$$
$$P_{ij} = 0.5, (i \in \alpha \wedge j \notin \alpha) \vee (j \in \alpha \wedge i \notin \alpha)$$
$$P_{ij} = 0, i \notin \alpha \wedge j \notin \alpha$$

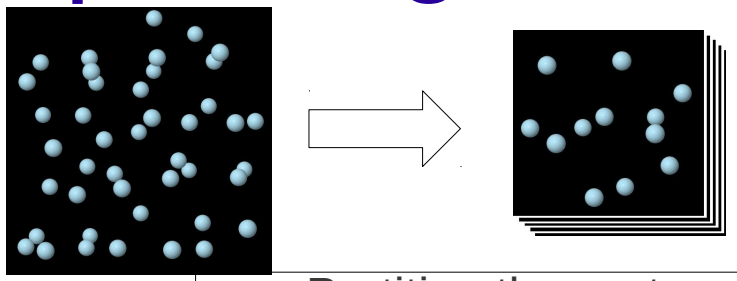
- Once the SCF has converged for all subsystems the global Fermi energy is determined by iteratively improving Ef until there are the right number of electrons in the system.

$$N = \sum_{ij} \rho_{ij} S_{ij} = \sum_{ij} (2 \sum_{\alpha} P_{ij}^{\alpha} \sum_{m} f_{\beta}(\varepsilon_F - \varepsilon_m^{\alpha}) C_{ij}^{\alpha} C_{jm}^{\alpha}) S_{ij}$$

- And the global density matrix is constructed, from which the energy can be evaluated.

$$\rho_{ij} = \sum_{\alpha} 2 P_{ij}^{\alpha} \sum_{m} f_{\beta}(\varepsilon_F - \varepsilon_m^{\alpha}) C_{\Im}^{\alpha} C_{jm}^{\alpha}$$

Divide and Conquer Algorithm



Read Geometry

Begin SCF Cycle

First SCF?

yes

Partition the system into subsystems. Determine partition function.

no

Compute the SCF for each subsystem.

Find the global Fermi Energy. Compute the density matrix.

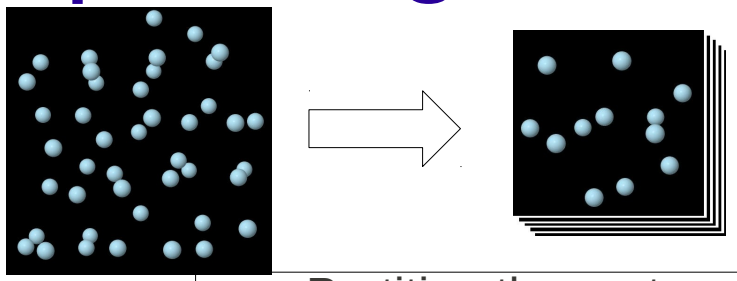
Converged?

no

yes

Compute the Energy

Divide and Conquer Algorithm



Read Geometry

Begin SCF Cycle

First SCF?

yes

Partition the system into subsystems. Determine partition function.

no

Compute the SCF for each subsystem.

Find the global Fermi Energy. Compute the density matrix.

Converged?

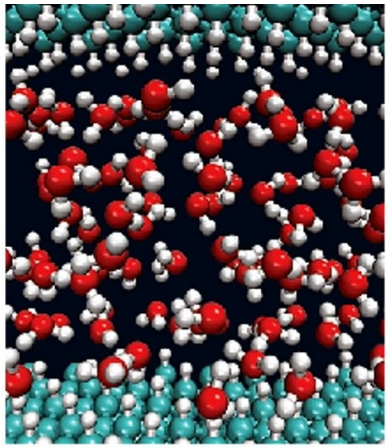
no

yes

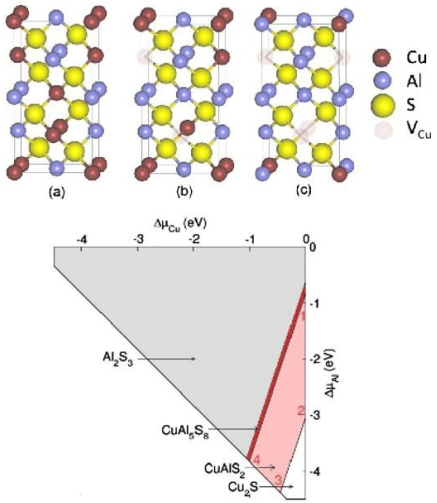
Compute the Energy

Task level parallelism in CRYSTAL

1.



2.



- Aside from the divide and conquer algorithm there are many cases where multiple SCF computations are required.
- e.g. (1). Finding the electronic structure at snapshots of a classical MD run. (2). Finding several points on a phase diagram. (3). (not shown) Finding higher order differentials of the potential energy surface, to compute frequencies etc.
- This is now easy to perform in CRYSTAL and I have developed helper routines so that this task level parallelism can be used within CRYSTAL.

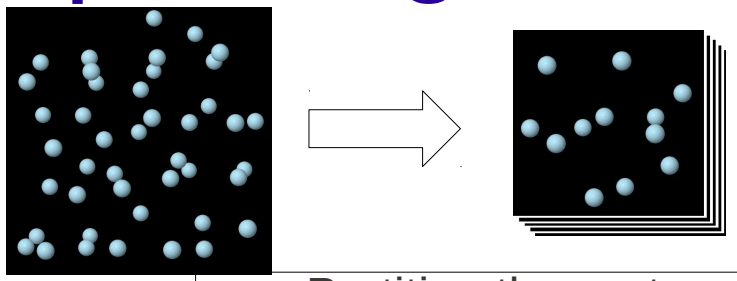
MULTIRUN in CRYSTAL

- I have implemented this task farming within CRYSTAL using the MULTIRUN keyword.
- It is simple to use, an INPUT file containing nothing but
    1. A title line.
    2. MULTIRUN
    3. 2 Integers, NRUNS and TASK_SIZE
        - The total number of tasks, and the number of processors to use for each task.
- A separate normal INPUT file name INPUT.tsk_ is required for tasks 1-NRUNS. Running this job will be identical to running CRYSTAL for each of the INPUT files.
- Should be available soon (after further testing) hopefully in the next release of CRYSTAL.

Integrating MULTIRUN for use with other CRYSTAL routines

- It should be possible to integrate this functionality with routines within CRYSTAL where the computation of many SCFs happens sequentially.
- This is likely to improve the scaling using large numbers of processors.
- Should be ideally suited to finite difference methods – so long as SCFs are independent of each other.

Divide and Conquer Algorithm



Read Geometry

Begin SCF Cycle

First SCF?

yes

Partition the system into subsystems. Determine partition function.

no

Compute the SCF for each subsystem.

Find the global Fermi Energy. Compute the density matrix.

Converged?

no

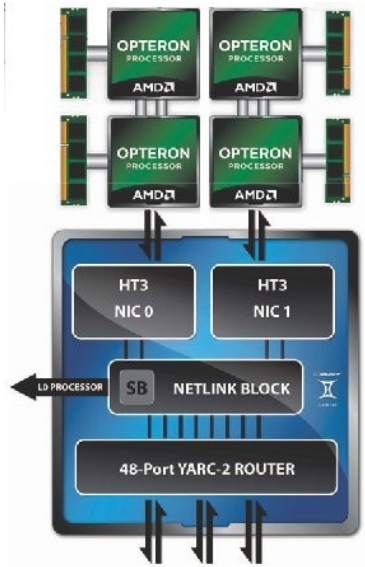
yes

Compute the Energy

Modern HPC Architecture

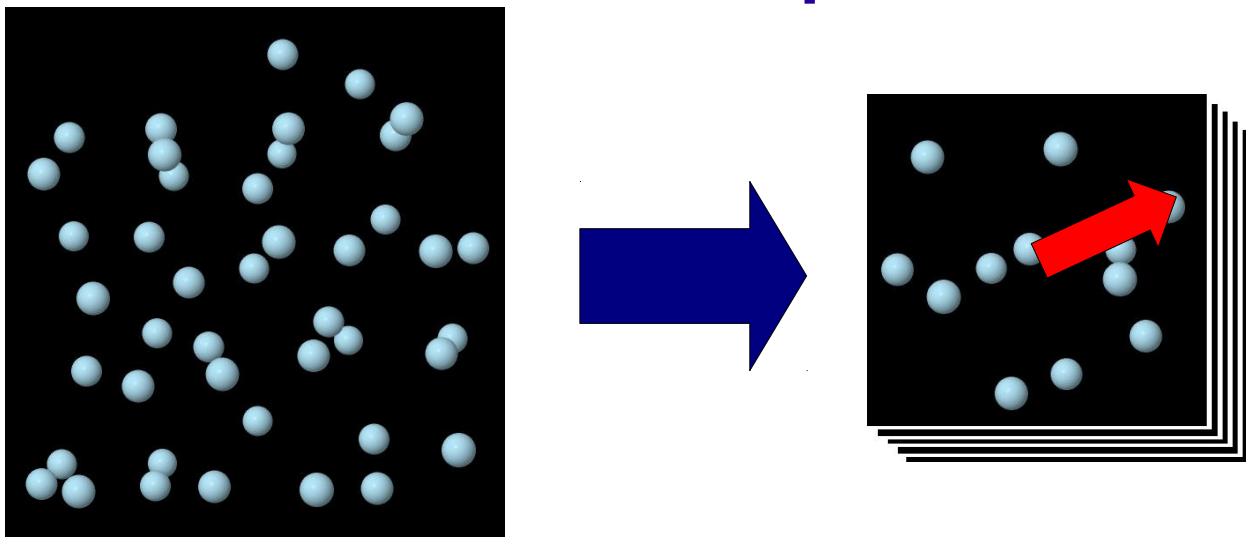
- Multi-core chips are becoming increasingly important
- Task level parallelism means it is possible to keep communication mostly on-chip.
- The D&C algorithm in SONIC should scale very well with number of processors, as well as linearly with system size. Lots of computation with very little communication.

HECToR



XE6 Node

Simple Case



- 10 Å^3 Neon
- Each subsystem contains 1 core atom
- Atoms within a defined radius are included in a halo.

Radius of Halo / Å	E(*O(N)*) – E(Trad) / millihartree cell^{-1}
0.1	17.3
3.0	5.1
4.5	2.6

What's next?

- 2 main problems to work on:
  - Termination of covalent bonds at the surface of subsystems.
  - Long range treatment of electrostatics – CRYSTAL has the machinery for this.
- Implement a more intelligent way to partition a system into subsystems.
- Test the algorithm's performance for more challenging and interesting systems.

Summary

- Implemented an *O(N)* divide and conquer algorithm for the CRYSTAL code.
- As part of this, task level parallelism has been implemented in CRYSTAL.
- Showed the method applies to a simple example case.

Continuations

- Long range Coulombic interactions using a multipole expansion.
- Sensible subsystem termination to deal with covalent systems.
- Splitting the system into a variety of subsystems depending on what is most appropriate.