Distributed Computing

Christopher Woods, Robert Gledhill, Sarah Williams, Jonathan Essex
Comb*e*chem Project, University of Southampton

What do we want to Compute?

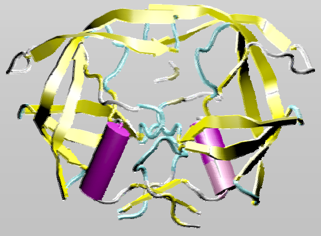
Comb*e*chem is compiling a large database of molecules. The database contains the properties of these molecules, e.g. their crystal structure or solvent accessible surface area (SASA). Some of these properties are measured from experiment while others are calculated from simulations run on the GRID.

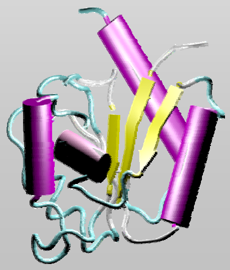
Molecule ID	pK_a	SASA	Melting Point
1CD34	2.3	Unknown	58
1CD35	1.3	Unknown	112
1CD36	Unknown	36543	96
1CD37	5.3	58435	78
1CD38	Unknown	Unknown	110

Comb*e*robots continually scan the database for empty fields. They can automatically submit simulations to calculate any unknown properties. These simulations run on the GRID by stealing the spare cycles of a heterogeneous network of computers.

Distributed Algorithms

The database of molecules can also be screened against pharmaceutical protein targets. To do this accurately requires knowledge of how the protein changes shape upon ligand binding. We can use the GRID to investigate protein conformational change via *Replica Exchange* simulations. Multiple simulations of the protein are run in parallel, each running under a different condition, e.g. temperature. Periodically the simulations running at neighbouring temperatures are tested and swapped. This enables simulations at high temperatures, where there is rapid conformational change, to rain down to biologically relevant temperatures where conformational change occurs more slowly.


HIV Protease

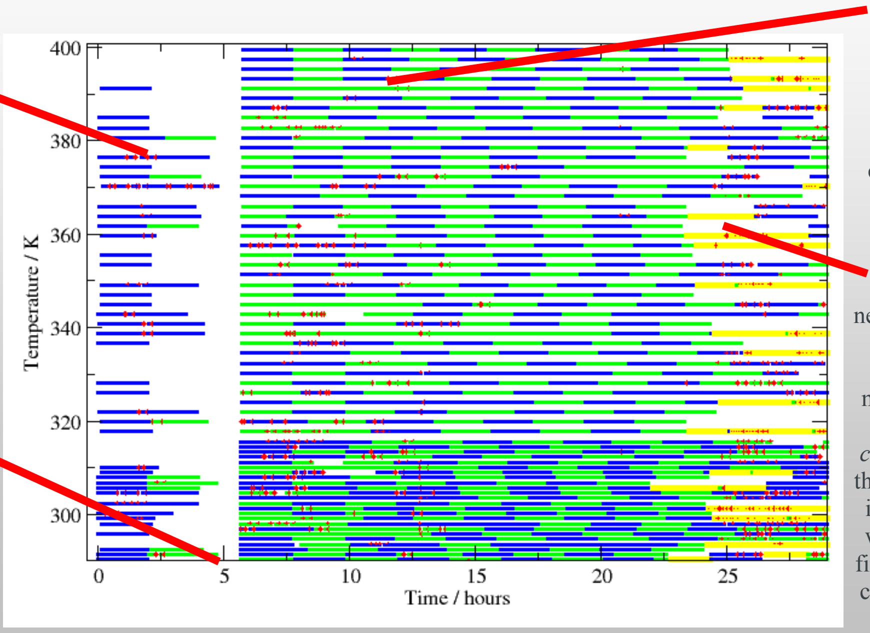

Nitrogen Regulatory Protein C

The reality of working on the GRID

We run our simulations by stealing the spare cycles of the University's machines using Condor. These form a large distributed GRID. This GRID contains a wide variety of computers, from underused Windows desktops, to highly used Linux servers.

Iterations are run at each temperature. Even iterations are shown in green, odd iterations in blue.

Our algorithms have been designed to cope with extreme events, e.g. the total failure of the distributed cluster after only five hours of simulation!



The cluster contains some slow nodes, so some iterations take longer than others.

The owners of the computers may also wish to use them! (shown by red dots)

This all means that neighbouring temperatures will be ready to test at different times. To minimise the waiting, we implemented a *catchup cluster* (shown in yellow) that catches up an iteration if another temperature is waiting too long for it to finish. Jobs on the catchup cluster are run in parallel, so are completed more quickly.