



**Collaborative Computational Project for
Biomolecular Simulation**



Practical 1:

Running Biomolecular Simulations on the NGS with the Globus Toolkit



101010001000000100100
101010001000000100100



1010100010000001

Outline

In this practical you will learn how to run biomolecular simulations on the National Grid Service using three different simulation packages (AMBER, GROMACS and NAMD) via the command line using the Globus Toolkit. Using each of these packages you will simulate the Major Urinary Protein (MUP) for 10ps on 16 CPUs on the Leeds node.

Starting structures, parameter and input files for these simulations have been prepared and are installed in your home directories on each of the training workstations. These files are for training purposes only, and CCPB offers no guarantee that these files are fit for research purposes. For details on how to set up protein simulations visit <http://www.ccpb.ac.uk/events/workshops>.

Stage 1: Connecting to the Leeds host and uploading files

Step 1.1: Log into the Leeds node using GSI-SSH

In a web browser that contains your e-science certificate goto;

<http://www.grid-support.ac.uk/content/view/259/195/>

Within this page (assuming you have Java Software Development Kit (JDK or SDK) 1.5 or higher installed) an GSI-SSHTerm applet will launch in the browser window.

To connect to the Leeds node, choose **New Connection** from the **File** menu.

In the **Connect to host...** dialog box press **Advanced** button. Then, fill the **Hostname** field with **ngs.leed.ac.uk**, and **Port** with **2222**, leave **Username** blank, and select **publickey** option in the **Authentication Methods** list box. Press the **Connect** button. When asking for Grid certificate passphrase, enter the passphrase you saved, and press **OK** button. You're logged into the host when you see a welcome page.

Step 1.2: Upload files onto the Leeds node

The starting structures, parameter and input files for MUP for all three simulation packages are located in the ccpb directory in your home directories on each of the training workstations.

Choose **SFTP Session** from the **Tools** menu within the GSI-SSHTerm applet.

A new window will open showing a list of the files in your home directory on the Leeds node.

Choose **Upload Files** from the **File** menu.

In the **Select files to upload** dialog box enter ccpb in the **Selection** field and press the **Upload** button.

In the **Copy Directory** dialog box press **Recurse Copy**.

The files will now be copied onto the Leeds NGS node and a directory called ccpb will be visible in the SFTP session window. Verify that all the files have been copied over

successfully by listing the contents of the ccpb directory;

```
[ngsXXXX@ngs ~]$ ls ccpb/*
ccpb/amber:
M01.md9  M01.top  md.in

ccpb/gromacs:
M01.gro  M01.top  md.mdp

ccpb/namd:
M01  M01.cor  M01.prm  M01.psf  M01.vel  M01.xsc
```

Stage 2: Running an AMBER simulation

The first example that we are going to use to perform a biomolecular simulation is the **pmemd** module of AMBER. The input files for this example are located in

```
/home/ngsXXXX/ccpb/amber
```

where ngsXXXX is your username, and we will run the code in this directory so that the input and output are located in the same place.

The command that we want to run is

```
pmemd.MPI -O -i md.in -o M01.out -p M01.top -c M01.md9
```

(the .MPI indicates that there are separate serial and parallel versions of the code installed on this node. To find out more about the software that you want to run on a particular node visit <http://www.grid-support.ac.uk/content/view/261/132>).

To run this simulation on 16 CPUs submit it use the command;

```
[ngsXXXX@ngs ~]$ globus-job-submit \
  ngs.leeds.ac.uk/jobmanager-pbs -np 16 \
  -d /home/ngsXXXX/ccpb/amber -x '(jobtype=mpi)' \
  /usr/ngs/AMBER pmemd.MPI -O -i md.in -o M01.out \
  -p M01.top -c M01.md9
```

Once the job is submitted you will be given a URI of the form <https://ngs.leeds.ac.uk:64049/28952/1227105032/>. This is identity of the job and can be used to query the status of the job;

```
[ngsXXXX@ngs ~]$ globus-job-status \
  https://ngs.leeds.ac.uk:64049/28952/1227105032/
ACTIVE
```

The job can also be monitored by looking at the output files which are located in /home/ngsXXXX/ccpb/amber;

```
[ngsXXXX@ngs ~]$ cd ccpb/amber/  
[ngsXXXX@ngs amber]$ ls  
logfile M01.md9 M01.out M01.top mdcrd md.in mdinfo  
restrt
```

Five new files have been created, logfile, M01.out, mdcrd, mdinfo and restrt. The status of the simulation can also be monitored by viewing the output file M01.out

```
[ngsXXXX@ngs amber]$ tail -f M01.out
```

This job has 5,000 MD steps and will take approximately 2.5 minutes to run. When the job is complete the globus-job-status command will return DONE.

Stage 3: Running a GROMACS simulation

The next simulation package that we are going to use is GROMACS. The input files for this example are located in

```
/home/ngsXXXX/ccpb/gromacs
```

and we will run the code in this directory so that the input and output are located in the same place.

Running a GROMACS is a two stage process. Firstly a binary input file needs to be built for a 16 CPU simulation;

```
grompp_d -np 16 -c M01.gro -p M01.top -f md.mdp -o md.tpr
```

and then the molecular dynamics is performed using the resulting md.tpr file;

```
mdrun_mpi_d -s md.tpr
```

To achieve this we firstly use globus-job-run to execute the grompp_d command in the directory containing the input files;

```
[ngsXXXX@ngs gromacs]$ globus-job-run ngs.leeds.ac.uk \  
-d /home/ngsXXXX/ccpb/gromacs \  
/usr/local/NGS/GROMACS-3.3.1/bin/grompp_d -np 16 \  
-c M01.gro -p M01.top -f md.mdp -o md.tpr
```

When this command has run, which may take a few seconds to complete, the binary input file md.tpr will have been created;

```
[ngsXXXX@ngs amber]$ cd ~/ccpb/gromacs/  
[ngsXXXX@ngs gromacs]$ ls  
M01.gro M01.top md.tpr md.mdp mdout.mdp
```

The MD simulation is then submitted using the command;

```
[ngsXXXX@ngs gromacs]$ globus-job-submit \  
ngs.leeds.ac.uk/jobmanager-pbs -np 16 \  
-d /home/ngsXXXX/ccpb/gromacs -x '(jobtype=mpi)' \  
/usr/ngs/GROMACS_3_3_1 mdrun_mpi_d -s md.tpr
```

The status of the job may then be monitored with the globus-job-status command, and by monitoring output files;

```
[ngsXXXX@ngs gromacs]$ ls  
M01.gro  md0.log  md11.log  md13.log  md15.log  ....  
M01.top  md10.log  md12.log  md14.log  md1.log   ....
```

The files md*.log contain output information from each CPU. The file md0.log can be monitored to keep track of the progress of the job;

```
[ngsXXXX@ngs gromacs]$ tail -f md0.log
```

This job has 5,000 MD steps and will take approximately 2.5 minutes to run. When the job is complete the globus-job-status command will return DONE.

Stage 4: Running a NAMD simulation

The final simulation package that we are going to use is NAMD. The input files for this example are located in

```
/home/ngsXXXX/ccpb/namd
```

and we will run the code in this directory so that the input and output are located in the same place.

To run this example we will run the command;

```
namd2 M01
```

To run this example on four CPUs we use globus-job-submit;

```
[ngsXXXX@ngs gromacs]$ globus-job-submit \  
ngs.leeds.ac.uk/jobmanager-pbs -np 16 \  
-d /home/ngsXXXX/ccpb/namd -stdout result.out \  
-x '(jobtype=mpi)' /usr/ngs/NAMD_2_6 namd2 M01
```

The status of the job may then be monitored with the globus-job-status command, and by monitoring output files;

```
[ngsXXXX@ngs namd]$ ls  
M01.cor  M01.psf  M01.xsc      result.out  
M01      M01.prm  M01.vel      output.dcd
```

The standard output is written to the file results.out, and can be monitored to keep track of

the progress of the job;

```
[ngsXXXX@ngs gromacs]$ tail -f result.out
```

This job has 5,000 MD steps and will take approximately 4 minutes to run.

Stage 5: Compare performance of each simulation

Each of the software packages we have used in this practical gives some information in the output file about the amount of time the simulation took to execute. AMBER and GROMACS also provide detailed information about how long different parts of the code took to run.

Firstly we will examine the performance of AMBER;

```
[ngsXXXX@ngs gromacs]$ cd ~  
[ngs0207@ngs ~]$ tail -68 ccpb/amber/M01.out
```

```
-----  
5. TIMINGS  
-----  
  
NonSetup CPU Time in Major Routines, Average for All Tasks:  
  
| Routine           Sec           %  
|-----|-----|  
| DataDistrib       25.24      17.90  
| Nonbond           110.66     78.48  
| Bond              0.02       0.02  
| Angle             0.21       0.15  
| Dihedral          0.80       0.57  
| Shake             0.92       0.65  
| RunMD             3.02       2.14  
| Other             0.14       0.10  
|-----|-----|  
| Total            141.01  
  
....  
  
| Master Setup CPU time:           0.51 seconds  
| Master NonSetup CPU time:       140.96 seconds  
| Master Total CPU time:          141.47 seconds      0.04 hours  
  
| Master Setup wall time:          1 seconds  
| Master NonSetup wall time:       142 seconds  
| Master Total wall time:          143 seconds      0.04 hours
```

From the output of this run we can see that pmemd took 143 seconds to perform the simulation. Of that time ~80% was spent calculating non-bonded interactions, and ~18% was spent transmitting data between each CPU. Additional information about where time was spent when calculating the non-bonded interactions is also provided, but not shown here for brevity.

Now looking at the performance of GROMACS;

```
[ngs0207@ngs ~]$ tail -85 ccpb/gromacs/md0.log
```

```
      M E G A - F L O P S   A C C O U N T I N G

      Parallel run - timing based on wallclock.
      RF=Reaction-Field  FE=Free Energy  SCFE=Soft-Core/Free Energy
      T=Tabulated       W3=SPC/TIP3p    W4=TIP4p (single or pairs)
      NF=No Forces

Computing:                M-Number          M-Flops    % of Flops
-----
Coul(T)                   776.270595    32603.364990    1.6
Coul(T) [W3]              3.777794     472.224250     0.0
Coul(T) + LJ             1878.856716   103337.119380    5.2
Coul(T) + LJ [W3]       389.276988    53720.224344    2.7
Coul(T) + LJ [W3-W3]    1933.810546   738715.628572   37.1

....

      NODE (s)   Real (s)      (%)
Time:    155.000   155.000    100.0
          2:35
      (Mnbf/s)   (GFlops)   (ns/day)   (hour/ns)
Performance:  32.142    12.834     5.574      4.306

Detailed load balancing info in percentage of average
Type      NODE:  0  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  Scaling
-----
Coul(T):763 836  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  11%
Coul(T) [W3]: 0  0  0  0  0  0  26 40 75 154 185 207 247 193 213 256 38%
Coul(T) + LJ:993 606 0  0  0  0  0  0  0  0  0  0  0  0  0  0  10%
Coul(T) + LJ [W3]: 0  0  0  0  0  0  25 59 100 117 167 221 236 234 207 228 42%
Coul(T) + LJ [W3-W3]: 0 21 158 158 156 157 151 134 120 106 86 73 68 69 69 68 62%
Outer nonbonded loop:333 300 57 55 57 57 58 65 68 70 74 79 80 81 79 80 29%
1,4 nonbonded interactions:868 731 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 11%
Spread Q Bspline:128 128 127 117 107 100 82 67 61 68 81 91 97 105 111 120 77%

....

Total Scaling: 65% of max performance

Finished mdrun on node 0 Fri Nov 21 10:08:01 2008
```

We can see that the time taken to perform this simulation was 155 seconds, which is slightly longer than the AMBER simulation took. The output from GROMACS is different from that of AMBER in that it provides detailed information about different parts of the code in terms of flops (floating point operations per seconds) rather than real time. This means that we cannot determine how much time has been spent transmitting data between nodes. The output does however contain detailed information about the load balancing so you can see how well different parts of the code are balanced across all of the processors. The output also informs you as to how efficient the parallel performance has been on the whole, in this case the simulation only ran at 65% of its maximum performance.

Finally looking at the output from NAMD;

```
[ngs0207@ngs ~]$ tail -5 ccpb/namd/result.out
CLOSING COORDINATE DCD FILE
WRITING VELOCITIES TO OUTPUT FILE AT STEP 5000
=====
WallClock: 260.310394  CPUTime: 260.310394  Memory: 16215 kB
End of program
```

The minimal timing data presented by NAMD tells us that this simulation took 260 seconds, which is considerably slower than AMBER or GROMACS, but does not provide us with any further information to determine why this is the case.

The results of these timing runs is surprising as it is generally accepted that the pmemd module or AMBER and NAMD use very efficient inter-processor communication algorithms, whereas GROMACS uses a very inefficient algorithm. It should be noted that the test case used here is a relatively small system and we would expect the relative performance of NAMD to improve, and the performance of GROMACS to worsen, as larger systems are considered.

These observations show that if you are choosing to use a particular piece of software solely based upon its performance, then you should run a short series of simulations to determine which software package is the most efficient for the system that you are simulating.