Molecular Dynamics

Hybrid parallel execution:
process & thread placement

Aims

The aim of this exercise is investigate how best to run the molecular dynamics application NAMD on ARCHER for a particular simulation problem. NAMD is a hybrid MPI + OpenMP code so can run in parallel simultaneously using both processes and OpenMP threads.

The point of this exercise is to go through a methodical process of finding out how best to run a application in parallel on an HPC machine. This should help you help you when you need to do this for your own particular research problem and an application you would like to use.

Introduction

In this exercise you will run the molecular dynamics application NAMD and investigate process and thread placement. You will:

1. Measure the runtime for different numbers of MPI processes and OpenMP threads using all cores on a single node
2. Determine the best process / thread option
3. Find out if hyperthreads speed up the simulation
4. See how the problem scales to multiple nodes

Instructions

The ApoA1 benchmark consists of a protein (27,850 atoms), solvated in explicit water for a total of 92,224 atoms. The force-field has a 12 Å cut-off for pair interactions, and uses PME (Particle-Mesh Ewald) for long-range electrostatics. The PME calculation is done every 4 timesteps, and a total of 500 1fs timesteps will be carried out in an NVE (microcanonical) ensemble. A tar-ball containing the files can be obtained from:

http://www.ks.uiuc.edu/Research/namd/utilities/apoa1.tar.gz

Download and extract this in a directory on /work on ARCHER. Write or adapt a job script to run this benchmark on a single node based on the guidance given on the ARCHER website:

http://www.archer.ac.uk/documentation/software/namd/

Hint: start by using the following aprun command to run NAMD on a single node:

```
aprun -n 1 -d 24 -cc none namd2 +ppn 23 +pemap 1-23 +commap 0 apoa1.namd > output.log
```

This runs NAMD with 1 MPI process and 24 threads per process.

When NAMD runs it will generate an output file containing information about the setup of the calculation, including how many MPI processes are being used, details relating to the simulation parameters and force-field, and then a summary of the computed energies, temperature, pressure etc. at each timestep. You will also see intermittent messages to indicate that NAMD is performing dynamic load balancing. Periodically, NAMD prints timing information, for example:

```
TIMING: 480  CPU: 22.6909, 0.0448748/step  Wall: 22.6909, 0.0448748/step,
```

```
0.000249304 hours remaining, 672.472656 MB of memory in use
```

You should observe that after a few rounds of dynamic load balancing, the time per step will decrease and converge to some value.

Conveniently, NAMD prints out a Benchmark time, which is designed to provide an accurate estimate of the performance for long runs, ignoring any data before the initial load balancing steps. In the apoa1 benchmark this is printed after 400 and 500 steps:

```
Info: Benchmark time: 23 CPUs 0.0355051 s/step 0.410938 days/ns 2987.35 MB memory
```

Finally NAMD prints the total time it has used:

```
WallClock: 21.276503 CPUTime: 21.276503 Memory: 2987.347656 MB
[Partition 0][Node 0] End of program
```

Warning: the total Wallclock time includes the time taken for initialisation (reading the input files, and also performancing some initial calculations to find the most efficient FFT strategy for the PME calculation). The first time this is done, NAMD will write a file FFTW_NAMD_2.12_CRAY-XC-ugni-smp_FFTW3.txt which it can re-read in later runs to save time. Thus the wallclock time is not a good indicator of performance, the time per step is a better metric.

Once you have verified that this runs, use the guidance on the ARCHER website to construct aprun commands for the following choices of MPI processes and OpenMP threads per process:

1 MPI x 24 OpenMP
2 MPI x 12 OpenMP
4 MPI x 6 OpenMP
6 MPI x 4 OpenMP

Now run the benchmark for each of these options. What is the best choice (i.e. has the fastest time per step)?

Next, enable hyperthreads by including the following aprun option:

```
-j 2
```

This will allow you to also explore the following options:

1 MPI x 48 OpenMP
2 MPI x 24 OpenMP
4 MPI x 12 OpenMP
6 MPI x 8 OpenMP
12 MPI x 4 OpenMP

You will need to further adapt the aprun statement following the examples on the ARCHER website. Does enabling hyperthreading improve the single-node runtime?

Using this optimal choice, now run your benchmark on multiple nodes for a range of different processor counts, and record the Benchmark time. For each processor count, calculate the speedup and the parallel efficiency using the expressions given in the lecture slides. You may also calculate other performance metrics, like the number of ns of simulation time per day of wallclock time.

Plot your data against the number of processors (you may need to use a logarithmic scale). How many processors can you use before the performance starts to drop off? If you were running extended MD calculations with this system, how many MPI processors would you choose to use?

Finally, some additional suggestions for tuning performance are provided at the NAMD website:
http://www.ks.uiuc.edu/Research/namd/wiki/?NamdPerformanceTuning
You can experiment with some of these and see how they affect the performance of benchmark.