

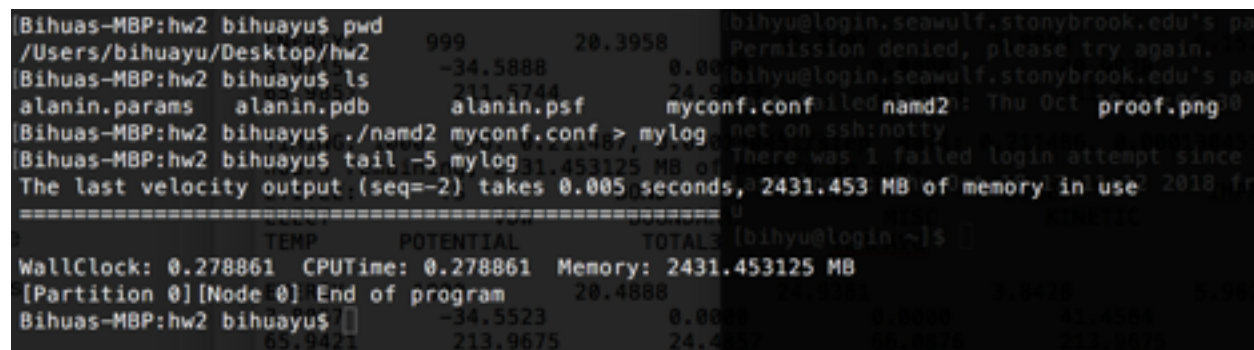
Project Description:

NAMD is a parallel molecular dynamics code designed for highperformance simulation of large bio-molecular systems.

Our project requires you to do the following:

- (1) Download the latest source and compile it for your system.
- (2) Run the program on the supercomputer you can access. You test it with 8 processing cores for 100 time steps (usually the variable is named as numsteps). You may use the default parameters or change them to a reasonable set that you can explain. Please tabulate the wall clock times on all processors for their local calculations (accumulated over the 100 time steps);
- (3) Repeat the above experiment described in (2) with exactly the same parameters except that you increase the core count to 16.
- (4) Repeat the above experiment described in (2) with exactly the same parameters except that you increase the core count to 32.

- (1) This is the screen shot showing that the program is successfully installed and run. The system is MacOS X86-64 multicore.



```
[Bihuas-MBP:hw2 bihuayu$ pwd
/Users/bihuayu/Desktop/hw2
[Bihuas-MBP:hw2 bihuayu$ ls
alanin.params  alanin.pdb  alanin.psf  myconf.conf
[Bihuas-MBP:hw2 bihuayu$ ./namd2 myconf.conf > mylog
[Bihuas-MBP:hw2 bihuayu$ tail -5 mylog
The last velocity output (seq=-2) takes 0.005 seconds, 2431.453 MB of memory in use
=====
      TEMP      POTENTIAL      TOTALS
WallClock: 0.278861 CPUTime: 0.278861 Memory: 2431.453125 MB
[Partition 0][Node 0] End of program 20.4888
[Bihuas-MBP:hw2 bihuayu$
65.9421 213.9675 24.4888
```

- (2) The program is run on seawulf, with following modules loaded:
 - module load shared
 - module load namd/2.9
 - module load gcc-stack

Since the program is pretty small, in order to get results soon, I ran an interactive job. Thus, I also did

module load torque/6.0.2

Then, make following command to run the program on 8 processing cores.

```
mpirun -n 8 namd2 myconf.conf
```

The 12th line in following stdout shows the program runs successfully on 8 process:

```
Info: NAMD 2.9 for Linux-x86_64-MPI
Info: Then, make following command to run the program on 8 proce
Info: Please visit http://www.ks.uiuc.edu/Research/namd/
Info: for updates, documentation, and support information.
Info:
Info: Please cite Phillips et al., J. Comp. Chem. 26:1781-1802 (2005)
Info: in all publications reporting results obtained with NAMD.
Info:
Info: Based on Charm++/Converse 60500 for mpi-linux-x86_64
Info: Built Thu Mar 8 13:27:33 EST 2018 by decarlson on login
Info: 1 NAMD 2.9 Linux-x86_64-MPI 8 login bihyu
Info: Running on 8 processors, 8 nodes, 1 physical nodes.
Info: CPU topology information available.
Info: Charm++/Converse parallel runtime startup completed at 0.0632775 s
Info: 121.062 MB of memory in use based on /proc/self/stat
Info: Configuration file is myconf.conf
Info: Working in the current directory /gpfs/projects/AMS530/hw2
TCL: Suspending until startup complete.
Info: SIMULATION PARAMETERS:
Info: TIMESTEP 1
Info: NUMBER OF STEPS 100
Info: STEPS PER CYCLE 20
Info: LOAD BALANCER Centralized
Info: LOAD BALANCING STRATEGY New Load Balancers -- DEFAULT
```

The last 4 lines show that the timestep is set as 100. And the load balance is centralized, which is dynamic. According to some tutorials of NAMD, we know

NAMD only print out “wallclock time used by the first processor (not the aggregate of all processors in the simulation).”(https://www.ks.uiuc.edu/Training/Tutorials/namd/namd-tutorial-unix-html/node27.html);

“If specified and nonzero, CPU and wallclock times will be output to stdout. These data are from node 0 only; CPU times for other nodes may vary.”(https://udel.edu/~arthij/namd-ug.pdf).

Thus, I will only report the wall clock time of the first processor for all questions.

#processor in total	processor id	wall clock time
8	0	0.214950

(3)

#processor in total	processor id	wall clock time
16	0	0.251867

(4)

#processor in total	processor id	wall clock time
32	0	0.337982