NAMD

- NAMD Application Information for Beagle2
  - Building and using NAMD 2.8/2.9
    - Using non-smp NAMD
      - Example of non-smp script
    - Using smp NAMD
      - Example of smp script
    - Using replica-exchange MPI-based NAMD
  - Miscellaneous information, that is either outdated or under construction
    - Charm++
    - Building NAMD On Beagle2 with PGI compiler
    - Running NAMD 2.8b1 on Beagle2
    - Running NAMD on Beagle2

NAMD Application Information for Beagle2

NAMD is a molecular dynamics (MD) program designed for parallel computation. NAMD official web site

Many thanks to Yanhua Sun and Jim Phillips for porting charm++ and NAMD to the Cray XE6, particularly Beagle2.

Please note: NAMD is built for MPI and the compute nodes therefore it won't run on login nodes. In order to run it you must use aprun otherwise it won't work.

Building and using NAMD 2.8/2.9

NAMD requires a number of libraries to function properly. A number of this libraries need to be tuned to the Gemini interconnect or Beagle2's general architecture to perform well. We will discuss in detail which implementations are available on Beagle and why. Alternatives might be available and we will attempt to provide users with sufficient information to be able to find/install those variants for themselves. In general the best information on such alternatives can be found directly form the developers themselves.

SMP or non SMP To the best of our knowledge this affects mostly Charm++'s behavior. In non-smp, we fire a process on each physical core. Even they are on same physical node, the memory address space for different processes is not visible for each other. So to send a message from one process to another process on same node, either the messages can be passed using network stack or using posix shared memory. Thus, to be simple, you can think that posix shared memory is used to share memory for different process on same node. In smp case, we fire one process on one node and then spawn multiple threads. One thread is dedicated to one physical core. In os, all the memory within one process is accessed for all threads. Thus, when we send messages from one thread to another, we can just directly send the memory point.

Usually on small number of cores, non-smp gives better performance. However, when scales above 1000 cores, smp is usually faster. When you run smp, cpuaffinity is required. On Beagle the following flags are recommended when calling smp-based namd2 add this namd2 +ppn 11 +setcpuaffinity +pemap 1-11,13-23 +commap 0,12 to your aprun command and they are expected to give the best performance.

Simulations indicate that on Beagle2 it is best to run it on two independent memory processes (-N 2 ) each with 12 threads ( -d 16 ) for each node used. Let me know if you have more question about running NAMD or performance issues. (Thanks to Yanhua Sun from the University of Illinois for this explanation.)

Overall in the case of NAMD itself, it appears that smp uses less memory, while non-smp runs a little faster (Jim Phillips)

For acknowledgement, remember citing Cham++ and NAMD in addition to Beagle.

Using non-smp NAMD

module load namd loads the non-smp version ( 2.8-nonsmp is default) it can also be done as

module load namd/2.8-nonsmp

module load namd/2.9-nonsmp
this automatically sets:

```bash
setenv MPICH_ENV_DISPLAY 1
setenv MPICH_MAX_SHORT_MSG_SIZE 50000
setenv MPICH_PTL_UNEX_EVENTS 200000
```

Which is considered optimal on Beagle2, feel free to change those environmental settings should you have reason to believe otherwise.

- The loading sets namd2 in your path, and you can simply call it as `namd2` (computations should be done only on the compute nodes)

### Example of non-smp script

**IMPORTANT NOTES:**

- The fields surrounded by <> need to be filled
- To run this script is necessary to load namd first using `module load namd`
- This script should be submitted using `qsub`
- number of cores set in `mppwidth` and in `aprun -n` should generally be the same

```bash
#!/bin/bash
#PBS -j oe
#PBS -N <job name, useful to recognize a job when running batch>
#PBS -l walltime= <time_requested, the shorter the more likely to run soon>
#PBS -l mppwidth= <number_of_cores, generally a multiple of 32 -- nr of cores per node on Beagle>
# If testing for short and small jobs (less than 30 minutes less than 32 cores or so) skip the the following line
#PBS -q batch

cd $PBS_O_WORKDIR
#mynamd=namd/2.8-nonsmp
mynamd=namd/2.9-nonsmp

# Load modules
. /opt/modules/3.2.6.7/init/bash
module swap PrgEnv-cray PrgEnv-gnu 2>&1
module load $mynamd 2>&1
module list 2>&1

# Load modules

echo "#!/bin/bash
namd2 <namd configuration file> >& <namd logfile>
">my_namad.sh
chmod +x my_namad.sh

# -n followed by the same number used for mppwidth above
aprun -n <number_of_cores> -cc cpu my_namad.sh
```

### Using smp NAMD

module load namd loads the non-smp version (default) it can also be done as

```bash
module load namd/2.8-smp
module load namd/2.9-smp
```

this automatically sets (maybe the same as non-smp)

```bash
setenv MPICH_ENV_DISPLAY 1
setenv MPICH_MAX_SHORT_MSG_SIZE 50000
setenv MPICH_PTL_UNEX_EVENTS 200000
```
Which is considered optimal on Beagle2, feel free to change those environmental settings should you have reason to believe otherwise.

- The loading sets namd2 in your path, and you can simply call it as namd2 (computations should be done only on the compute nodes). Note that in this case it will be the smp version even if the call will be the same.

**Example of smp script**

**IMPORTANT NOTES:**

- The fields surrounded by <> need to be filled
- To run this script is necessary to load namd first using `module load namd/2.9-smp`
- This script should be submitted using `qsub` from the directory where one wants to use namd (or change the script)
- number of cores set in `mppwidth` and in `aprun` -n should NOT be the same for the smp submission

```bash
#!/bin/bash
#PBS -j oe
#PBS -N <job name, useful to recognize a job when running batch>
#PBS -l walltime= <time_requested, the shorter the more likely to run soon>
#PBS -l mppwidth= <number_of_cores, number of mpi processes times 32>
# If testing for short and small jobs (less than 30 minutes less than 32 cores or so) skip the following line
#PBS -q batch

cd $PBS_O_WORKDIR

# Load modules
. /opt/modules/3.2.6.7/init/bash
module swap PrgEnv-cray PrgEnv-gnu 2>&1
module load namd/2.9-smp 2>&1
module list 2>&1

echo >my_namd.sh <<END
#!/bin/bash
namd2 +ppn 11 +setcpuaffinity +pemap 1-11,13-23 +commap 0,12 <namd configuration file> >& <namd logfile>
END
chmod +x my_namd.sh

aprun -n <number_of_mpi> -N 2 -d 16 my_namd.sh
```

**Using replica-exchange MPI-based NAMD**

module load namd loads the non-smp version (default). This can also be done as:

namd/2.9-ReplicaExchangeMPI

this automatically sets:

```bash
setenv MPICH_ENV_DISPLAY 1
setenv MPICH_MAX_SHORT_MSG_SIZE 50000
setenv MPICH_PTL_UNEX_EVENTS 200000
```

Which is considered optimal on Beagle2, feel free to change those environmental settings should you have reason to believe otherwise.

- The loading sets namd2 in your path, and you can simply call it as namd2 (computations should be done only on the compute nodes)
- It is recommended to run it using the setting `-cc cpu` for aprun
Miscellaneous information, that is either outdated or under construction

Charm++

http://charm.cs.uiuc.edu/software

Current default version is 6.2.1 check out from the University of Illinois on on Thu Feb 9 10:52:47 CST 2012:

git clone git://charm.cs.illinois.edu/charm
to build

module swap PrgEnv-cray PrgEnv-gnu
module load rca
module list 2>&1 | less
... 
  (3) PrgEnv-gnu/3.1.61
  14) atp/1.3.0
  ... 
  21) gcc/4.6.1
  ... 
  25) rca/1.0.0-2.0301.23101.14.211.qem

Building NAMD On Beagle2 with PGI compiler

Source: Email from Lorenzo, 2011.0513

This is the recipe Chris and Lorenzo used. Chris tested it to 2.8. Greg Cross is installing 2.8 as the official version (PGI only for now)

I built it and tested it for PGI only, the other ones break (I can try to fix them, but I see no reason, since gcc showed worse scaling than PGI on the other MD programs).

Location of my copy of the build directory on Beagle2: /home/lpesce/Voth/CRAY_NAMD_2.7

The file to run is ./build-xe , which has been modified as follows:

This needs to be replaced:

```
# MOD by LP 4/15/2011 for Beagle XE , orig: setenv XTPE_LINK_TYPE shared
setenv XTPE_LINK_TYPE dynamic
```

The makefile needs to be changed by removing all traces of libfpmpi.a on line 367 (unless one wants to have that working, in which case we need to install it. I saw no reason for it).

Running NAMD 2.8b1 on Beagle2

Example PBS script (and PATH) for running NAMD 2.8b1 (courtesy of Chris Rowley):
export MPICH_PTL_SEND_CREDITS=-1
export MPICH_MAX_SHORT_MSG_SIZE=8000
export MPICH_PTL_UNEX_EVENTS=80000
export MPICH_UNEX_BUFFER_SIZE=100M

. /opt/modules/3.2.6.7/init/bash
module swap PrgEnv-cray PrgEnv-gnu
module load fftw/2.1.5.2

export LD_LIBRARY_PATH=/soft/torque/2.5.4/lib:/opt/cray/mpt/5.2.0/xt/gemini/mpich2-pgi/lib:/opt/cray/mpt/5.2.0/xt/gemini/sma/lib64:/opt/pgi/11.1.0/linux86-64/11.1.1/lib64:/opt/pgi/11.1.0/linux86-64/11.1.1/lib64:/opt/pgi/11.1.0/linux86-64/11.1.1/lib64:/opt/pgi/11.1.0/linux86-64/11.1.1/lib64:/opt/cray/xpmem/0.1-2.0301.24575.5.2.gem/lib64:/opt/cray/dmapp/2.2-1.0301.2791.5.1.gem/lib64:/opt/cray/ugni/2.1-1.0301.2798.5.2.gem/lib64:/opt/cray/udreg/2.1-1.0301.2797.5.2.gem/lib64:/opt/cray/MySQL/5.0.64-1.0301.2899.20.1.gem/lib64:/opt/cray/MySQL/5.0.64-1.0301.2899.20.1.gem/lib64:/opt/cray/MySQL/5.0.64-1.0301.2899.20.1.gem/lib64:/opt/cray/MySQL/5.0.64-1.0301.2899.20.1.gem/lib64:

aprun -n $NCPUS /home/cnrowley/programs/NAMD_2.8bi_Source/CRAY-XE/namd2 $JOBNAME.namd  >& $JOBNAME.out

---

Running NAMD on Beagle2

(From Lorenzo's email notes 2011.0513)

The PBS script needs to include the following (shown for BASH, other shells need other steps to make the module command work)

```
# Load modules and set for dynamic environment
# Some are to give us more postmortem info when it does not work for
# someone

. /opt/modules/3.2.6.7/init/bash
module load fftw/2.1.5.2
module list 2>&1

# Set the dynamic library runtime env.

export CRAY_ROOTFS=DSL
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/fftw/2.1.5.2/lib
```