PRACTICAL MACHINE SPECIFIC COMMANDS
KRAKEN

Myvizhi Esai Selvan
Department of Chemical and Biomolecular Engineering
University of Tennessee, Knoxville
October, 2008

Machine: a Cray XT4 system with 4512 compute nodes (each with 4 cores)
Relevant webpages: http://www.nics.tennessee.edu/computing-resources/kraken

I. How to connect

a. For Windows operating systems use ssh and on Linux operating systems for interactive window:

ssh userid@kraken.nics.tennessee.edu

Use your pin followed by the secured id to log in.

b. For using the password to log in, connect to

kraken-pwd.nics.tennessee.edu

A sftp session can not be started in this mode but you can monitor your jobs. You can not use the HPSS storage.

kraken.nics.tennessee.edu
kraken-pwd.nics.tennessee.edu

II. How to compile and link

Kraken has two kind of nodes i) login nodes ii) compute nodes. Login nodes should be used for basic tasks such as file editing, code compilation, data backup and job submission. Only compute nodes can be used for production.

command: **make**

The make command uses the macros defined in the file "makefile" to compile and link the code.
The contents of the makefile are shown in Figure 1.
Additional comments on non-intuitive traits of the makefile in Figure 1.

1) Tabs appear after all colons (:). If you use spaces, it won't work.
2) If a space appears after back-slash (\), it won't work.

III. How to execute codes

1. *Move to the lustre scratch directory*
2. *Copy the executable, input and batchscript file to the working subdirectory of scratch directory*
3. *Submit the job*
4. *Monitor job progress*

1. Move to the lustre scratch directory

   Compute nodes can see only lustre scratch directories. The kraken scratch directory is located at:

   cd /lustre/scratch/userid

   where the subdirectory for the jobs can be created. If this is not done, you may see an error like:
   aprun: [NID 94]Exec /lustre/scratch/userid/a.out failed: chdir /nics/b/home/userid No such file or directory

2. Copy the executable, input and batchscript file to the working directory

   Copy your executable file, any input files, and the batch command file to your working subdirectory of the scratch directory. The executable file and input files are obvious. A sample command file, named, "batch.cmd", is shown in Figure 2. The command file determines the files for standard output and standard error. It sets the maximum time and the number of processors required.

   #PBS -A UT-TENN0002
   Specifies the account to be charged to
   #PBS -N mddriver280
   Indicates the job name
   #PBS -j oe
   Mentions job's standard output and error would be combined
   #PBS -l walltime=12:00:00,size=8
   The maximum time per processor is twelve hours for a small job. You can also set the number of nodes processors, so setting size=8, yields 8 cores.
   Kraken has nodes which are quad-core in nature. Size in the above line must state a size which is a multiple of 4. The above line allocates 2 nodes (8 cores) to the job.
   #PBS -q batch
   Directs the job to the specified queue but it is optional because the default queue is "batch" and the other is known as debug queue. In the batch queue it is redirected to other queues depending on the number of cores requested.

Queue	Min Size	Max Size	Max Wall Clock Limit
small	0	512	12:00:00
*longsmall	0	512	60:00:00
medium	513	2048	24:00:00
large	2049	8192	24:00:00
capability	8193	18040	48:00:00

For other PBS options refer to http://www.nics.tennessee.edu/computing-resources/kraken/running-jobs/common-pbs-options.

aprun -n 8 -N 4 ./mddriver

The above line runs the executable mddriver on 8 compute cores (-n) and uses 4 cores per socket(node)(-N).

3. Submit the job

On Kraken the compute nodes only have access to the lustre file system. (/lustre/scratch/[USERNAME]). You will run the job via the command "qsub [batch_script]" You should do this from the directory in lustre in which you want to work. This directory should contain the executable "mddriver".

qsub batch.cmd

If successfully submitted, a job ID will be returned. This ID can be used to track the job.

4. Monitor job progress

qstat -a

Use the above command to check the status of submitted jobs.

```
esaiselv@krakenl:~> qstat -a

nid00004: UT/NICS
                                                                  Req'd  Req'd   Elap
Job ID               Username Queue    Jobname    SessID NDS   Tasks Memory Time  S Time
-------------------- -------- -------- ---------- ------ ----- ----- ------ ----- - -----
100997.nid00004      jpfaendt longsmal F3walk1     12478     1    16     --  24:00 E   --
100998.nid00004      jpfaendt longsmal F3walk2     12596     1    16     --  24:00 E   --
101097.nid00004      jpfaendt large    test4         --      1  2600     --  14:00 Q   --
101098.nid00004      jpfaendt large    test5         --      1  2600     --  14:00 H   --
101390.nid00004      acarr    small    LD_450      30809     1    48     --  12:00 R 05:34
101395.nid00004      andrew   medium   f40.035c    14588     1  1536     --  01:30 R 01:14
```

The S denotes the status of the job R – Running, Q – Queued, C – Recently completed, H – held, E – Exiting after having run.

When the job is done, all your output files, standard output, and standard error are located in the working directory.

If you make a mistake and need to kill a job, use **qdel** with the job id.

Other useful commands to monitor the job can be found in http://www.nics.tennessee.edu/computing-resources/kraken/running-jobs/monitoring-job-status

IV. Conversion of Newton Code to kraken

Replace flush_ with flush in the code.

IV. Debugging

For debugging use interactive jobs submitted in the debug queue. The maximum clock limit is 2 hours.

qsub -I -A XXXYYY -q debug -V -l size=16,walltime=1:00:00

- I – starts interactive
- A XXXYYY – account XXXYYY to be charged

To kill the job use the command **exit**

IV. Scheduling Policy

First In First out policy. The job priority increases with

1. short clock limit
2. large number of processors
3. waiting period

```
.SUFFIXES:  .f

COMP = ftn

OPT = -O3 -tp barcelona-64

MAIN_LIB_DIR =
OTHER_MAIN_LIB_DIR =

SCALAPACK_LIB_DIR = $(OTHER_MAIN_LIB_DIR)
BLACS_LIB_DIR = $(OTHER_MAIN_LIB_DIR)
PLBAS_LIB_DIR = $(OTHER_MAIN_LIB_DIR)

LIB_SCALAPACK =
LIB_BLACS =
LIB_PBLAS =
LIB_BLAS =

LIB = $(LIBSCALAPACK) $(LIB_PBLAS) $(LIB_BLAS) $(LIB_BLACS)

TARGETS = mddriver

OBJS =  md_mix_v70.o transport_d.o md_mpi_extras.o pbc_multi.o linkedcell.o \
      self_d_corr.o transport_dmut.o onsagerL.o onsagerL_corr.o adriver.o \
      md_ewald_real.o md_ewald_reciprocal.o erfc.o xinvmat.o \
      make_molecules.o droulet.o intra_001.o intra_002.o intra_003.o \
      ewald_real_correct.o ewald_recip_correct.o pair_corr_f.o w_cluster.o \
      h3o_stats.o profiles.o restart.o reaction.o trigger.o local_eqb.o

mddriver:       $(OBJS)
        $(COMP) -o mddriver $(OPT) $(OBJS) $(LIB)

clean:
        \rm -f *.o $(TARGETS) *.mod

all:    $(TARGETS)

.f.o:
        $(COMP) $(OPT) $(INCCH)  -c $<
```

Figure 1. Contents of makefile

```
#!/bin/ksh
#PBS -A UT-TENN0002
#PBS -N mddriver280
#PBS -j oe
#PBS -q batch
#PBS -l walltime=12:00:00,size=16
$PBS -W group_list=users
cd $PBS_O_WORKDIR
aprun -n 16 -N 4 ./mddriver
```

Figure 2. Sample command file, “batch.cmd”, for a batch job