Introduction to the usage of the HP computing cluster of Fak II at Institut für Mathematik

Sebastian Kraus

Team IT am Insitut für Chemie

May 4, 2017

HP computing cluster at Institut für Mathematik

Cluster infrastructure

- compute hosts grouped into **18 clusters: within each cluster interconnection via infiniband**, between clusters Gigabit Ethernet
- **about 600 machines in total:** different types of machine architectures (shared memory/NUMA) with distinct resources (amount of main memory, number of cpu cores, HDD swap space...)
- several clusters include hosts with GPU accelerator cards, but **cluster18 has 4 compute hosts each with modern Nvidia Geforce GTX cards from Pascal architecture**

Cluster infrastructure

- resources within cluster managed by batch job/queueing system (in our case a descendent from Sun's Gridengine); each host attached to different batch queues
- batch job system manages usage of the compute resources offered by each compute host; each host's cpu core attached to a compute slot
- jobs automatically **scheduled to specific batch queue according to job resource demands** and **distribution from queues to hosts according to host loads**

Overview of cluster18 - GPU host units

- **nodes in cluster18 for "big" GPU jobs:** 4 machines (node561 to node565) each equipped with **four Nvidia Geforce GTX 1080 GPU accelerator cards**
- **each Geforce GTX 1080** with **20 MIMD processors at 1.6 GHz** containing **2560 GPU SIMD cores**; total **main/global memory of 7.9 GiB GDDR-5**
- How to use host compute power? - **Serveral jobs in parallel** or **one exclusive job demanding all CPU cores**? - Answer: Depends on specific job!

First steps at the HP compute cluster

- **Login via:**

  `ssh -2 -e none -C -l` *`username`*`@cluster-[i|v|x].math.tu-berlin.de`

  (for graphical display: -[X|Y] and if necesary [-p 22])

- **Copy files/directories to cluster nodesr:**

  `scp -2 [-P 22] [-r] /path/to/local/[file|directory] \`
  `username``@cluster-i.math.tu-berlin.de:/path/to/remote/[file|directory]`

- program environments loadable via predefined modules:

  `module load name_of_module (e.g. module load namd/2.12-cuda)`

Submitting jobs

Using module system

About batch job scheduler/Queueing system

- batch job system allows **fair use/sharing** and **quoting of compute resources** wrt. to jobs and users
- each **job must demand for resources** to be runnable, e.g. **number of used cpu cores**, **resident main memory**, ...
- priority of jobs in queues: **long** or **short** jobs (running minutes/hours/several days) AND **resource demanding or non-demanding** jobs

Job submission procedure

- **type of job submission:**
  - b) **qsub** for batch jobs
  - b) **qrsh** for interactive test jobs (if enough free resources)

- **submitting batch jobs:**

```
a) command line: qsub -N [job_name] -q [queue] \
                                    -l h_rt=[runtime] ... job_command
b) via script: qsub < jobscript  (script on stdin to qsub)
c) array jobs (advanced): qsub -n[start_index]:[end_index]]:[incr]
  ( $SGE_TASK_ID to access index within your script )
```

Writing batch job scripts

- **Syntax batch job scripts:**

```
#!/bin/bash --login
# mandatory settings
#$ -N [jobname]
#$ -l mem_free=[xxxM]|[yyyG]  # resident memory per process/thread
#$ -l h_rt=[zzz]  # wall clock time in seconds
#$ -l [name_of_resource]=[amount_of_resource]  # other resources
#$ -pe [name_of_parallel_env] [no_of_cpu_cores]  # parallel environment
#$ -cwd  # cd to current working directory of job
# optional settings
#$ -q [name_of_job_queue]  # queue to be scheduled
# settings for job notifications
#$ -o stdout.out
#$ -e stderr.out
#$ -j [no|yes]
#$ -m [n|b|e|a|b,e,...]
#$ -M [name_of_your_mailbox]

(shell_)commands
```

About module system

- nearly each **program package needs set of path/environment variables** in order to execute correctly

$\Longrightarrow$ **module system: easily load predefined environment** needed by your program/library/compiler (i. e. reliable user environment)

- **(un)loading/switching/listing needed modules:**

```
module load|unload|switch|list|avail|show name_of_module
```

- **advantage:** loading via moudule files can be done automatically within your ∼/.bashrc

Running batch jobs - An example with namd

Preparing NAMD job runs

- **use special parallel environment (pe)** for parallel jobs

```
#$ -pe ompi[no_of_cluster|*]_[procs_per_node] [(range_)no_procs]
```

```
e.g.: -pe ompi* 5 (five cpu cores in pe ompi
                   of any cluster)
-pe ompi18_5 5-10 (between five to ten cpu cores
     in pe ompi of cluster18; five on each host)
```

Preparing NAMD job runs

- load CUDA and NAMD module; launch NAMD with cuda wrapper (remark: **gpurun wrapper not needed for batch job runs**):

1. `module load cuda/8.0 namd/2.12-cuda`
2. `gpurun -g [no_gpu] namd2 +p [no_proc] input > output`

Batch job script for NAMD

- **Example batch job script for jobs with NAMD:**

```
#!/bin/bash --login
#$ -N namd_gpu
#$ -l mem_free=500M  # 500 MiB resident memory
#$ -l h_rt=3600  # one hour wall clock time
#$ -l gpus=1  # no of GPU accelerators
#$ -l cluster18  # runnable on hosts in cluster18
#$ -pe ompi18* 5  # parallel environment with 5 cpu
#$ -cwd
#$ -o stdout.out
#$ -e stderr.out
#$ -j no
# #$ -m a,e
# #$ -M name@tu-berlin.de

module load cuda/8.0 namd/2.12-cuda
namd2 +p 5 input.namd > output.out
```

Best practice NAMD jobs

- benchmarking NAMD on nodes of cluster18 showed: **use only 1 Geforce GTX 1080 GPU accelerator card for each NAMD process**
- **run four NAMD jobs in parallel on one host/machine; each using one accelerator card and five cpu cores**
- collect **information about expected runtime, resource usage** and syntatic correctness of your input **via short test runs**

Best practice NAMD jobs

- narrow user quota limitations imposed on your nfs HOME:
  $\Longrightarrow$ **put permanent data into your nfs WORK; huge temporary job files can be put on node local scratch directory (see your Wiki for how to do that)**
- **please have a look at your file and space usage on WORK and remove from time to time undeeded files** (to be included within backup of your work group as necessary)
- results of benchmarking tests will be - if there is interest - presented on behalf of following talk

Further sources of information

Finding further information

- **have a look at the Wiki of Team IT**: open user's manual with example scripts and the description of installed program packages offered (more documentation will be added in near future)

- http://it.chem.tu-berlin.de/wiki/doku.php?id=numericserveruser:start and also http://www.math.tu-berlin.de/iuk/computeserver/einfuehrung_in_die_benutzung/