Table of Contents

1. HW 2a: Running Batch Queue Jobs (5 points)
2. HW 2b: Passing Data to Apps Running on Batch Nodes (10 points)
3. HW 2c: Running Histogram Batch Queue Jobs (15 points)
4. General Instructions
You will modify `batch.hello-cpuid`

- Run batch jobs for the following test cases:
  - 5 cores one 1 node
  - 5 cores on 2 nodes
  - 11 cores on 1 nodes, specify the node
  - 21 cores on 3 nodes

- Use example code found in `/COMP605/hello` directory.

- Turn in a copy of the batch script and the output for each test: produce output similar to the slides below (see Slide 4).
Showing Node & Core ID
Set Attributes: #PBS -l nodes=1:ppn=8

[mthomas@tuckoo hello]$ qsub batch.hello-cpuid
198.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ qstat -a
tuckoo.sdsu.edu:

<table>
<thead>
<tr>
<th>Job ID</th>
<th>Username</th>
<th>Queue</th>
<th>Jobname</th>
<th>SessID</th>
<th>NDS</th>
<th>TSK</th>
<th>Memory</th>
<th>Time</th>
<th>Elap</th>
</tr>
</thead>
<tbody>
<tr>
<td>220.tuckoo.sdsu.</td>
<td>mthomas</td>
<td>batch</td>
<td>hello-cpuid</td>
<td>--</td>
<td>1</td>
<td>8</td>
<td>--</td>
<td>--</td>
<td>Q</td>
</tr>
</tbody>
</table>

[mthomas@tuckoo hello]$ cat hello-cpuid.o197
bash: BASH_FUNC_module(): line 0: syntax error near unexpected token ')'  
bash: BASH_FUNC_module(): line 0: 'BASH_FUNC_module() () { eval `/usr/bin/modulecmd bash $*`'  
bash: error importing function definition for 'BASH_FUNC_module'

hello-cpuid-test using 16 cores...
[mthomas@tuckoo hello]$ cat hello-cpuid.o220

PBS: job name is hello-cpuid
hello, world from node: node8, core: 7
hello, world from node: node8, core: 3
hello, world from node: node8, core: 4
hello, world from node: node8, core: 5
hello, world from node: node8, core: 2
hello, world from node: node8, core: 1
hello, world from node: node8, core: 5
hello, world from node: node8, core: 6
For this problem you will create a batch queue script that will pass 4 variables, to the hello-arg.c application.

- hello-arg should parse 2 long integer and 2 double precision variables.
- name them LINTARG1, LINTARG2, DBLARG1, DBLARG2

- modify `batch.hello-arg-qsubv` to process these variables
  - You can use the code that can be found in the `/COMP605/hello` directory
  - See the slides below for examples of how to pass arguments to an application running on a batch queue.

- Print out the node and core ID’s, along with the 4 variables passed
Example of a serial code that parses command line arguments

/*File: hello-arg.c by Mary Thomas, 2/12/15 */
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
/* This code is adapted from an example at:
http://brokestream.com/procstat.html */
int get_cpu_id() {
    ...
}

int main (int argc, char* argv[]) {
    long cmdarg;
    char cptr[100];
    int cpuid;

    /* read long integer from the command line */
    if (argc != 2) {
        printf("usage: hello-arg <integer number>\n");
        return 0;
    }
    cmdarg = strtol(argv[1], NULL, 10);

    gethostname(cptr,100);
    cpuid = get_cpu_id();
    printf("hello, world from node: %s, core: %d, cmdarg= %ld\n", cptr, cpuid, cmdarg);
    return 0;
}
Pass Data to Application Within the Batch Script

Batch nodes typically do not have interactive capabilities. How to pass data to your application?

Solution #1: Hard code the arguments within the script. This solution is not very flexible

```
[mthomas@tuckoo hello]$ cat batch.hello-arg
#!/bin/sh
#PBS -V
#PBS -l nodes=1:ppn=4:core4
#PBS -N hello-arg
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
echo "PBS: job name is $PBS_JOBNAME"
NCORES='wc -w < $PBS_NODEFILE'
echo "$PBS_JOBNAME running using $NCORES cores..."
mpirun -np 8 -hostfile $PBS_NODEFILE --nooversubscribe ./hello-arg 12345
[mthomas@tuckoo hello]$ qsub batch.hello-arg
229.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ cat hello-arg.o229
PBS: job name is hello-arg
hello-arg running using 8 cores...
hello, world from node: node1, core: 3, cmdarg= 12345
hello, world from node: node1, core: 2, cmdarg= 12345
hello, world from node: node1, core: 3, cmdarg= 12345
hello, world from node: node1, core: 0, cmdarg= 12345
```
Pass Data to Application as Part of \textit{qsub} Command

How to pass data to your application?

\textbf{Solution \#2: Pass data to the batch script by defining KEY=VALUE pairs to the \textit{qsub} command}

[mthomas@tuckoo hello]$ cat batch.hello-arg-qsubv
#!/bin/sh
# this batch script takes variables passed to the qsub command and passes them to the hello-arg executable as command line arguments. Usage:
# %qsub -v CMDARG=7654  batch.hello-arg-qsubv
#
#PBS -V
#PBS -l nodes=1:ppn=4:core4
#PBS -N hello-arg-qsubvars
#PBS -j oe
#PBS -q batch
cd $PBS_O_WORKDIR
echo "PBS: job name is $PBS_JOBNAME"
NCORES='wc -w < $PBS_NODEFILE'
echo "$PBS_JOBNAME running using $NCORES cores..."
mpirun -np 4 -hostfile $PBS_NODEFILE --nooversubscribe ./hello-arg $CMDARG

[mthomas@tuckoo hello]$ qsub -v CMDARG=9876 batch.hello-arg-qsubv
231.tuckoo.sdsu.edu
[mthomas@tuckoo hello]$ cat hello-arg-qsubvars.o231
PBS: job name is hello-arg-qsubvars
hello-arg-qsubvars running using 8 cores...
hello, world from node: node1, core: 0, cmdarg= 9876
hello, world from node: node1, core: 1, cmdarg= 9876
hello, world from node: node1, core: 2, cmdarg= 9876
hello, world from node: node1, core: 3, cmdarg= 9876
HW 2c: Running the Histogram Program as a Batch Queue Job (15 points)

- Run your tests on a batch node, using a single core.
- Time how long the code takes to run as a function of $data_count$
- Use a timer routine with micro-second resolution (see Slide 14).
- Time the two longest running functions.
- Run the following test case:
  - $bin\_count = 10$
  - $min\_meas = 1$
  - $max\_meas = 100000$
  - Vary $data\_count = 10^n$, where $n = 0, 1, 2, \ldots, n, \ldots, N_{\text{max}}$
  - What is $N_{\text{max}}$? Explain the limit.
- Compute statistics on the timings (see Slide 15).
- Print out the bins, ranges, count, and your timing statistics.
- Report results in a summary table (see Slide 12).
- Plot your timing results (see Slide 10).
Example: Table showing ProbSize

<table>
<thead>
<tr>
<th>n</th>
<th>$10^n$</th>
<th>$T_{wall}$ (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
<td>$1.14 \times 10^{-04}$</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>$2.00 \times 10^{-4}$</td>
</tr>
<tr>
<td>4</td>
<td>10000</td>
<td>$1.05 \times 10^{-3}$</td>
</tr>
<tr>
<td>5</td>
<td>100000</td>
<td>$9.42 \times 10^{-3}$</td>
</tr>
<tr>
<td>6</td>
<td>1000000</td>
<td>$6.94 \times 10^{-2}$</td>
</tr>
<tr>
<td>7</td>
<td>10000000</td>
<td>$6.58 \times 10^{-1}$</td>
</tr>
<tr>
<td>8</td>
<td>100000000</td>
<td>$6.52 \times 10^{0}$</td>
</tr>
<tr>
<td>9</td>
<td>1000000000</td>
<td>$6.52 \times 10^{1}$</td>
</tr>
</tbody>
</table>

A table of your test data and a plot of at least two results. Include labels and units.
HW Example: Plotting Family of Curves

Example of way to plot multiple results. Include labels and units. Use $\log_{10}$ where needed.
HW Example: Table of Statistics

<table>
<thead>
<tr>
<th>Double</th>
<th>Data Size</th>
<th>BW</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Median</th>
<th>Variance</th>
<th>Standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,0E+00</td>
<td>2,038479E-06</td>
<td>3,522754E-04</td>
<td>1,097202E-05</td>
<td>5,137920E-06</td>
<td>1,397227E-09</td>
<td>3,737949E-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,0E+01</td>
<td>3,945827E-06</td>
<td>2,325773E-05</td>
<td>5,178452E-06</td>
<td>5,137920E-06</td>
<td>3,636904E-12</td>
<td>1,907067E-06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,0E+02</td>
<td>2,276897E-06</td>
<td>2,611876E-05</td>
<td>8,463860E-06</td>
<td>8,945307E-12</td>
<td>2,990871E-06</td>
<td>8,312687E-06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,0E+03</td>
<td>4,113913E-05</td>
<td>2,409220E-05</td>
<td>6,90077E-11</td>
<td>-</td>
<td>3,653623E-10</td>
<td>1,911445E-05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,0E+04</td>
<td>1,505656E-04</td>
<td>1,606536E-04</td>
<td>2,327512E-10</td>
<td>1,525618E-05</td>
<td>-</td>
<td>3,653623E-10</td>
<td>1,911445E-05</td>
<td></td>
</tr>
<tr>
<td>1,0E+05</td>
<td>1,433980E-03</td>
<td>1,443000E-03</td>
<td>1,441133E-03</td>
<td>1,441133E-03</td>
<td>3,653623E-10</td>
<td>1,911445E-05</td>
<td>1,911445E-05</td>
<td></td>
</tr>
<tr>
<td>1,0E+06</td>
<td>1,552309E-02</td>
<td>1,421810E-02</td>
<td>1,420463E-02</td>
<td>1,724856E-08</td>
<td>1,313338E-04</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
</tbody>
</table>
Put homework into a directory:

\texttt{HOME/ \textless your\_username \textgreater /hw/hw1}

- include the source code(s), compiled binaries
- see sections above for what to include
- Write a simple report (this can be TEXT, Word, PDF Doc).
- Turn in hard copy (condensed/minimal number of pages) at start of class.

Once the submission timeline has closed
DO NOT CHANGE THE FILE TIMESTAMPS!
Timing code (cont.)

- Suggestions for timing:
  - Use C/Fortran internal timer
  - Example: C `gettimeofday()` function
  - Example timer code located in `/COMP605/tools/code_timer.c`
  - Avoid using the unix `date` function, it is not very sensitive
Statistical Methods

Run times on any computer are not reproducible, hence, it is important to analyze the distribution of a codes’ run times, and not just take one measurement.

- Standard statistical variables used to describe the distribution of the data include:
  - Max/Min (maximum/minimum values between all runs)
  - Mean (average value)
  - Median (central value)
  - Variance (variance)
  - StandardDeviation ($\sigma$) of the timings.

- To test your codes:
  - Run and time critical blocks
  - Vary key parameters (packet/problem size, # of cores, etc.).
  - Calculate the statistics at run-time.

- Refs:
  - http://edl.nova.edu/secure/stats/