Debugging on the ALCF BG/Q and XC40 Systems

Computational Performance Workshop
May 3, 2017

Ray Loy
ALCF
INTERACTIVE RUNS FOR TESTS (BG/Q AND THETA)

- Submit an interactive job to the queue, e.g.
  - `qsub -l -t 30 -n 512`

- When job "runs", the nodes are allocated, and you get a (new) shell prompt.

- This shell behaves like the one in a Cobalt script job
  - BG/Q: Just one difference: do "wait-boot" before proceeding
  - Start your compute node run just like in a Cobalt script job.
    - BG/Q: `runjob --block $COBALT_PARTNAME --np 512 --p 16 : myprogram.exe`
    - Theta: `aprun -N 64 -d 1 -j 1 -cc depth myprogram.exe`

- When you exit the shell, the Cobalt job will end

- Note: When the Cobalt job runs out of time, there is no message.
  - `Runjob or aprun will fail`.
  - Check your job status with "qstat $COBALT_JOBID"
BG/Q LIGHTWEIGHT CORE FILES

- When run fails, look for core files
  - core.0, core.1, etc.

- Lightweight core files
  - One for each rank that failed before job teardown
  - Contain stack backtrace in address form
  - Decode to symbolic (useful!) form

- Environment settings to control core files
+++PARALLEL TOOLS CONSORTIUM LIGHTWEIGHT COREFILE FORMAT version 1.0
+++LCB 1.0
Program : /gpfs/vesta-home/rloy/src/test/idie

[...]
+++ID Rank: 0, TGID: 1, Core: 0, HWTID:0 TID: 1 State: RUN

***FAULT Encountered unhandled signal 0x00000006 (6) (SIGABRT)

[...]
+++STACK
Frame Address   Saved Link Reg
00000001fbfffb700 000000000001001848
00000001fbfffb8c0 00000000010003e8
00000001fbfffb960 0000000000100438

[...]
---STACK
[...]
**BG/Q: DECODING LIGHTWEIGHT CORE FILES**

- bgq_stack [optional_exename] [corefile]

---

+++ID Rank: 0, TGID: 1, Core: 0, HWTID:0 TID: 1 State: RUN

00000000001001848

abort

/bgsys/drivers/V1R2M2/ppc64/toolchain-gnu/glibc-2.12.2/stdlib/abort.c:77

000000000010003e8

barfunc

/gpfs/vesta-home/rloy/src/test/idie.c:6

00000000001000438

foofunc

/gpfs/vesta-home/rloy/src/test/idie.c:12

00000000001000498

main

/gpfs/vesta-home/rloy/src/test/idie.c:19

[...]
BG/Q: COREPROCESSOR

- Useful when you have a large set of core files
  - Shows symbolic backtrace
  - Groups ranks that aborted in the same location together
  - *Can also attach to a running job to take snapshot*

- Location
  - coreprocessor.pl is in your default PATH
    - Attaching to running job does **not** require administrator
    - coreprocessor -nogui -snapshot=<filename> -j=<jobid>
      - Use the back-end (ibm.runjob) jobid from the .error file, not the Cobalt jobid

- Scalability limit
  - **Absolute maximum** 32K ranks. Practical limit lower.

- Instructions:
  - BG/Q Application Developer Redbook
<table>
<thead>
<tr>
<th>Function</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compute Node</td>
<td>1</td>
</tr>
<tr>
<td>Oxfffffffe</td>
<td>1</td>
</tr>
<tr>
<td>__libc_start_main</td>
<td>2</td>
</tr>
<tr>
<td>genxio_start_main</td>
<td>3</td>
</tr>
<tr>
<td>main</td>
<td>4</td>
</tr>
<tr>
<td>Allgather</td>
<td>5</td>
</tr>
<tr>
<td>DMDI_Allgather</td>
<td>6</td>
</tr>
<tr>
<td>DMDI_Allreduce</td>
<td>7</td>
</tr>
<tr>
<td>MPI2_Allreduce</td>
<td>8</td>
</tr>
<tr>
<td>MPI2_Process_wait</td>
<td>9</td>
</tr>
<tr>
<td>DCMF_criticalSection_cycle</td>
<td>10</td>
</tr>
<tr>
<td>MPI2_Process_wait</td>
<td>11</td>
</tr>
<tr>
<td>DCMF_Messenger_advance</td>
<td>12</td>
</tr>
<tr>
<td>DCMF::Queueing::Lockbox::Device::advance()</td>
<td>13</td>
</tr>
<tr>
<td>DCMF_Messenger_advance</td>
<td>14</td>
</tr>
<tr>
<td>DCMF::Queueing::Tree::Device::advance()</td>
<td>15</td>
</tr>
<tr>
<td>DCMF_Messenger_advance</td>
<td>16</td>
</tr>
<tr>
<td>DCMF::DMA::Device::advance()</td>
<td>17</td>
</tr>
<tr>
<td>DCMF::DMA::RecIfGroup::advance()</td>
<td>18</td>
</tr>
<tr>
<td>DMA_RecIfSimplePollNormalFifoById</td>
<td>19</td>
</tr>
<tr>
<td>DCMF::DMA::Device::advance()</td>
<td>20</td>
</tr>
<tr>
<td>MPI2_Allreduce</td>
<td>21</td>
</tr>
<tr>
<td>MPI2_Process_wait</td>
<td>22</td>
</tr>
<tr>
<td>DCMF_Messenger_advance</td>
<td>23</td>
</tr>
<tr>
<td>DCMF::Queueing::Lockbox::Device::advance()</td>
<td>24</td>
</tr>
<tr>
<td>DCMF_Messenger_advance</td>
<td>25</td>
</tr>
<tr>
<td>DCMF::Queueing::Tree::Device::advance()</td>
<td>26</td>
</tr>
<tr>
<td>DCMF::DMA::RecIfGroup::advance()</td>
<td>27</td>
</tr>
<tr>
<td>DMA_RecIfSimplePollNormalFifoById</td>
<td>28</td>
</tr>
</tbody>
</table>
BG/Q: GDB

- A single gdb client can connect to single rank of your job

- **BG/Q Limitations**
  - Each instance of gdb client counts as a “debug tool”
  - Only 4 tools may be connected to a job
    - At most 4 ranks can be examined

- Start a debug session using `qsub –l` (interactive job)
  - `qsub –l –q default –t 30 –n 64`
  - See Redbook for more info on starting gdb with runjob

- gdb can also load a compute-node **binary** corefile
  - *Use extreme caution when generating binary corefiles*

- Generally a parallel debugger (e.g. DDT) will be more useful
THETA

- *Will come back to DDT on BG/Q later*
THETA: ATP

- ATP = Abnormal Termination Processing
  - generates a STAT format merged stack backtrace (file atpMergedBT.dot)
  - view the backtrace file with stat-view

- Link your app with ATP
  - Before linking, make sure the "atp" module is loaded (check using module list)
  - Cray compiler will link in ATP automatically
  - Intel compiler needs a work-around for now:
    - `-Wl,-T/opt/cray/pe/cce/8.5.2/craylibs/x86-64/2.23.1.cce.ld`
THETA: STAT

- While program is running (e.g. deadlocked), you can generate a merged backtrace snapshot showing where your program is.
- module load stat
- On the MOM node, invoke "stat-cl pid" where pid is the aprun pid

Method 1:
- In your job script, run "hostname" to output the MOM node's hostname
- During the run, read the MOM hostname from your output file, ssh to that hostname, use ps to determine the pid of your aprun, then invoke stat-cl on that pid

Method 2:
- Use the example job script in /soft/debuggers/stat/job-stat.sh
  - Modify the aprun command to run what you need
- When the job is running, run the command "touch STAT_NOW". The script will check for this file's existence every 60 seconds. If it sees the file, it will generate a STAT snapshot. You can create multiple snapshots.
LGDB

- Igdb connects a gdb to each rank and provides a text interface
- module load cray-lgdb
- Modify your script job.sh to mark your aprun:
  ```
  #cray_debug_start
  aprun -n 1 -N 1 -d 1 -j 1 a.out
  #cray_debug_end
  ```
- Igdb
  - launch $a(8) --qsub=job.sh a.out
    - Submits job.sh to run 8 ranks, your executable is a.out
- Useful commands
  - backtrace (bt), continue (cont), break, print
  - See "man lgdb"
ALLINEA DDT

- **BG/Q, Theta, Cooley**
  - MAP available on Theta (not supported on BG/Q)

- **Environment**
  - BG/Q: softenv key “+ddt”
  - Theta: module load forge/7.0 (/soft/environment/modules/modulefiles)

- **Compiling your code**
  - Compile –g –O0
  - Note: XL compiler option -qsmp=omp also turns on optimization within OMP constructs. To override, use "noopt", e.g.
    - -qsmp=omp:noauto:noopt

- **More details:**
  - [http://www.alcf.anl.gov/user-guides/allinea-ddt](http://www.alcf.anl.gov/user-guides/allinea-ddt)
ALLINEA DDT STARTUP (BG AND Theta)

- Run using remote client (RECOMMENDED)
  - Optional: use ssh master mode so you only need log in once per session
    • Note: supported on Mac OS/X; not supported in Windows <= XP (? for >XP)
    • ~/.ssh/config
      – ControlMaster auto
      – ControlPath ~/.ssh/master-%r%@h:%p

- Run from login node
  - Need X11 server on your laptop and ssh –X forwarding
  - Run ddt and let it submit job through GUI
DDT REMOTE CLIENT (0)

GUI LOOKS JUST LIKE THE X11 CLIENT
DDT REMOTE CLIENT (1)

SELECT "CONFIGURE" TO ADD A NEW REMOTE HOST
DDT REMOTE CLIENT (2)

NOTE: THIS REMOTE INSTALLATION DIRECTORY IS THE DEFAULT VERSION OF DDT, CORRESPONDING TO +DDT OR MODULE
CLICK "TEST REMOTE LAUNCH" TO VERIFY
DDT REMOTE CLIENT (3)

NOW THAT IT IS DEFINED, SELECT REMOTE MACHINE
CONNECTED (NOTE LICENSE INFO IN LOWER LEFT CORNER)
FROM THIS POINT, REMOTE GUI WORKS SAME AS LOCAL
DDT STARTUP – REVERSE CONNECT (BG, THETA)

- Start remote client and connect to login node (or start X11 client on login node)

- In an ssh session to the login node
  - Run an interactive job (qsub -I)
    - BG/Q: Instead of runjob
      - ddt --connect --mpiargs="--block $COBALT_PARTNAME" --processes=8 -procs-per-node=16 myprog.exe
    - Theta: Instead of aprun ... myprog.exe
      - /soft/debuggers/forge/bin/ddt --connect aprun ... myprog.exe

- Likewise with Allinea MAP
  - Theta: /soft/debuggers/forge/bin/map --connect aprun ... myprog.exe
  - BG/Q: MAP is not supported on BG (but other perf tools available)
DDT (5) – BG/Q DIRECT JOB SUBMIT
CLICK "RUN" TO START A DEBUGGING SESSION
DDT (6) – BG/Q DIRECT JOB SUBMIT

REMEMBER TO SET WORKING DIRECTORY

IMPORTANT! ENABLE THE CHECKBOX "SUBMIT TO QUEUE"
- CLICK "CONFIGURE" AND "PARAMETERS" FOR ADDITIONAL SETTINGS
DDT (6.1) – BG/Q DIRECT JOB SUBMIT

JOB SUBMISSION TAB

USE SUBMISSION TEMPLATE: /SOFT/DEBUGGERS/DDT/TEMPLATES/ALCF-BGQ.QTF
DDT (6.2) – BG/Q DIRECT JOB SUBMIT

REMEMBER TO SET YOUR PROJECT
DDT (7) – BG/Q DIRECT JOB SUBMIT

JOB MUST GO THROUGH QUEUE
DDT (8) – REVERSE CONNECT OR DIRECT SUBMIT
WHEN JOB STARTS RUNNING, CONNECTION STATUS WILL SHOW
DDT (9)
READY TO DEBUG!
QUESTIONS

- See also

  - http://www.alcf.anl.gov/user-guides/mira-cetus-vesta
    - Theta docs coming soon. For now, see Confluence (wiki)

  - support@alcf.anl.gov