Debugging on the ALCF BG/Q and XC40 Systems

Computational Performance Workshop
May 16, 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
BG/Q Lightweight Core File Example

+++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
0000001fbfff700 0000000001001848
0000001fbfffb8c0 0000000001003e8
0000001fbfff960 0000000001000438
[...]
---STACK
[...]
BG/Q: Decoding Lightweight Core Files

- \texttt{bgq\_stack [optional\_exename] [corefile]}

  +++ID Rank: 0, TGID: 1, Core: 0, HWTID:0 TID: 1 State: RUN
  0000000001001848
  abort
  \texttt{/bgsys/drivers/V1R2M2/ppc64/toolchain/gnu/glibc-2.12.2/stdlib/abort.c:77}

  00000000010003e8
  barfunc
  \texttt{/gpfs/vesta-home/rloy/src/test/idie.c:6}

  0000000001000438
  foofunc
  \texttt{/gpfs/vesta-home/rloy/src/test/idie.c:12}

  0000000001000498
  main
  \texttt{/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>Group Mode:</th>
<th>Stack Traceback (condensed)</th>
<th>Session 1 (MMC)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0: Compute Node (128)</td>
<td>1: 0xffffffffc (128)</td>
<td></td>
</tr>
<tr>
<td>2: __libc_start_main (32)</td>
<td>3: generic_start_main (32)</td>
<td></td>
</tr>
<tr>
<td>4: main (16)</td>
<td>5: Allgather (16)</td>
<td></td>
</tr>
<tr>
<td>6: PMPI_Allgather (16)</td>
<td>7: MPID0_Allgather (8)</td>
<td></td>
</tr>
<tr>
<td>8: MPID0_Allreduce (8)</td>
<td>9: MPID0_Progress_wait (1)</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>11:</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>11: BDCF::CriticalSection_cycle (1)</td>
<td></td>
</tr>
<tr>
<td>9:</td>
<td>10:</td>
<td></td>
</tr>
<tr>
<td>9:</td>
<td>10: BDCF::Progress_wait (7)</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>11:</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>11: BDCF::Message_advance (1)</td>
<td></td>
</tr>
<tr>
<td>11:</td>
<td>12: BDCF::Queueing::LockBox::Device::advance() (1)</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>11:</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>11: BDCF::Queueing::Tree::Device::advance() (1)</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>11:</td>
<td></td>
</tr>
<tr>
<td>10:</td>
<td>11: BDCF::Message_advance (5)</td>
<td></td>
</tr>
<tr>
<td>11:</td>
<td>12: BDCF::DMA::Device::advance() (2)</td>
<td></td>
</tr>
<tr>
<td>12:</td>
<td>13: BDCF::DMA::RecFifoGroup::advance() (2)</td>
<td></td>
</tr>
<tr>
<td>13:</td>
<td>14:</td>
<td></td>
</tr>
<tr>
<td>13:</td>
<td>14: BDCF::RecFifoSimplePollNormalFifoByI (2)</td>
<td></td>
</tr>
<tr>
<td>14:</td>
<td>15:</td>
<td></td>
</tr>
<tr>
<td>14:</td>
<td>15: BDCF::DMA::Device::advance() (3)</td>
<td></td>
</tr>
<tr>
<td>15:</td>
<td>16:</td>
<td></td>
</tr>
<tr>
<td>15:</td>
<td>16: BDCF::DMA::RecFifoSimplePollNormalFifoByI (3)</td>
<td></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 –I` (interactive job)

– qsub –I –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 and Intel compilers will link in ATP automatically

Set environment before running your app
– export ATP_ENABLED=1
– aprun ...
THETA: STAT

While program is running (e.g. deadlocked), you can generate a merged backtrace snapshot showing where your program is.

On the MOM node, invoke "stat-cl pid" where pid is the aprun pid

In job script (or interactive job shell)
- hostname # identify the MOM node you are on
- module unload xalt # xalt wraps aprun resulting in 2 processes named "aprun"
- aprun ...

During the run, ssh to the same MOM node
- ps -u username # Determine pid of aprun
- module load stat
- DISPLAY="" stat-cl pid

Optional
- aprun ... &
- echo "aprun pid is $!"
- wait
lgdb 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
lgdb
   – 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, Cooley (not supported on BG/Q)
Environment
– BG/Q: softenv key “+ddt”
– Theta: module load forge/18.0.2 (/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
**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
DDT (4)

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)
When job starts running, connection status will show
DDT

Ready to debug!
Questions

See also

–http://www.alcf.anl.gov/user-guides

–support@alcf.anl.gov