Debugging and profiling of MPI programs

The code examples:
http://syam.sharcnet.ca/MPI_debugging.tgz

Sergey Mashchenko
(SHARCNET / Compute Ontario / Compute Canada)

Outline

- Introduction
- MPI debugging
- MPI profiling
- Concluding remarks

Introduction

Parallel vs. serial

- Parallel programming is more difficult than serial programming each step of the way:
  - Designing stage
  - Coding
  - Debugging
  - Profiling
  - Maintenance

Parallel bugs

- In addition to usual, “serial” bugs, parallel programs can have “parallel-only” bugs, such as
  - Race conditions
    - When results depend on specific ordering of commands, which is not enforced
  - Deadlocks
    - When task(s) wait perpetually for a message/signal which never come

Profiling issues

- Similarly to debugging, profiling of parallel codes deals both with issues common between serial and parallel codes (bad patterns for accessing memory, not cache friendly etc.), but also adds new, parallel-only issues, e.g.
  - Workload balancing
  - Costs of communications

Tools

- Debugging of codes (including parallel ones) could be as primitive as inserting multiple printf statements.
- Similarly, for profiling one could resort to using wallclock timers in the code.
- But given the extra difficulty of dealing with parallel code issues, debugging and profiling of parallel (including MPI) codes better be done using proper tools.

Allinea software

- For the rest of this webinar, I will focus on advanced parallel coder tools developed by Allinea and installed on multiple SHARCNET clusters (orca, monk, kraken etc.) Its two main components are
  - DDT: serial and parallel (MPI, multi-threaded, CUDA) debugger
  - MAP: serial and MPI code profiler.

Useful links

- For detailed information on how to use the Allinea tools on our clusters, check these wiki pages:
  - https://www.sharcnet.ca/help/index.php/DDT
  - https://www.sharcnet.ca/help/index.php/MAP

MPI debugging

DDT live demo

- Compile your code with low or zero optimization, and use “-g” switch to add symbolic information.
- Module ddt is loaded by default, so no need to load it manually.
- DDT/MAP can be used interactively on orca development nodes (orc-dev1 ... orc-dev4).
- Simply prepend “ddt” in front of your code + command line arguments; don't use mpirun (it is invoked internally by DDT), e.g.

```
$ ddt ./my_mpi_code  arg1  arg2
```

MPI debugging examples

- Deadlocks:
  - deadlock_simple.c: two MPI ranks using blocking send/receive in the wrong order
  - deadlock_ring.c: more interesting case of multiple ranks in ring topology
  - deadlock_collective.c: deadlocks in collective communications

MPI profiling

MAP live demo

- Compile your code with “-g”, but unlike DDT you can (and should) also use optimization flags, like “-O2”.
- MAP is a part of DDT module, and is loaded by default. MAP is only available on orca.
- Can be used interactively on orca dev nodes (for small jobs: N_ranks<=24), or submitted to the orca scheduler in the batch mode:

```
$ sqsub -q mpi --nompirun -o out -n 24 -r 12h  map -n 24 -profile  ./code
```

Dynamic workload balancing

- Dynamic workload balancing (DWB) is frequently used by MPI programs.
- We use it when the length of time spent on computing different parts of a large workload by different MPI ranks is hard or impossible to predict ahead of time.
- Well written DWB code should have a way to adjust the size of the workload quantum. (In other words – number of chunks.)

DWB example

- Example code:
  - dynamic_workload_balancing.c: using “sleep” function to emulate different processing time for different elements of a large input array
  - On 10 cpus, I got the following wall clock times:

N_chunks / N_CPUs	Wall clock time (s)
1	10.3
10	5.8
100	5.4
1000	5.5
10,000	6.9

Severe workload imbalance

Optimal performance

Latency becomes important

Concluding remarks

- Though it is not possible to cover such complex topics as MPI debugging and profiling at any depth in one hour, I hope the webinar
  - Provided enough of information so you know where to start
  - Demonstrated that with the right tools the parallel debugging/profiling is not as formidable as one might think

Questions?

- You can always contact me directly (syam@sharcnet.ca) or send an email to help@sharcnet.ca .