Parallel Debugging & Profiling

https://github.com/ResearchComputing/USGS_2016/

February 10, 2016

Timothy Brown

Research Computing
UNIVERSITY OF COLORADO **BOULDER**

Overview

Debugging Methods

Common Causes Of Bugs

DDT on Yeti

Profiling

MAP on Yeti

Debugging a parallel program is difficult. Parallel programs have all the usual bugs and new bugs due to

- timing
- synchronization
- shared data (OpenMP)

What makes it worse is these bugs often disappear when you run your program serially or when adding code to isolate and identify the bug.

Overview

Debugging Methods

Common Causes Of Bugs

DDT on Yeti

Profiling

MAP on Yeti

Debugging Methods

- Graphical: Totalview, DDT
- Command line: gdb, iidb
- Write statements

Overview

Debugging Methods

Common Causes Of Bugs

DDT on Yeti

Profiling

MAP on Yeti

Common Causes Of Bugs

- Improper use of language features
- Space decomposition
- Synchronization
- Scalability

Improper Use Of Language Features

Examples

- inconsistent parameter types for send/receive
- inappropriate choice of functions

Indicators

- compile time errors
- incorrect running under certain conditions

Check unfamiliar language routines and features carefully. The MPI routines have man pages.

Domain decomposition

Incorrect mapping between the problem space and the program memory space.

- often segmentation fault
- incorrect output

Check memory allocations and run the program with a memory debugger/checker (valgrind). MPI does provide decomposition routines, such as `MPI_Type_create_subarray` and `MPI_Cart_create`.

Synchronization

Improper coordination between processes.

- deadlocks
- race conditions

Often the program will hang or give incorrect output.
Make sure all communication is correctly coordinated.

Scalability

- Ordinary serial constructs may have unexpected side-effects when they used concurrently.
- Don’t just focus on parallel code, make sure the serial code is working on one processor first.
- Make sure all processors are working that there are no load imbalances.

Overview

Debugging Methods

Common Causes Of Bugs

DDT on Yeti

Profiling

MAP on Yeti

DDT on Yeti

1. Log in to Yeti.

```
laptop ~$ ssh yeti.cr.usgs.gov
```

2. Load the Intel parallel studio module

```
yeti-login01 ~$ module load intel/psxe-2015
```

3. Load the Allinea module

```
yeti-login01 ~$ module load allinea/6.0
```

4. Run DDT (the debugger)

```
yeti-login01 ~$ ddt
```

Overview

Debugging Methods

Common Causes Of Bugs

DDT on Yeti

Profiling

MAP on Yeti

Tools for Profiling

- **Gprof**
  http://www.thegeekstuff.com/2012/08/gprof-tutorial
- **PAPI** http://icl.cs.utk.edu/papi
- **Perfsuite** http://perfsuite.ncsa.illinois.edu
- **TAU** http://tau.uoregon.edu
- **MAP** http://allinea.com/

Profiling Terms

- Event
  - Function entry
  - Message send
  - Cache miss
- Metric
  - Total cycles
  - MFLOPS
  - Cache miss ratio
- Resolution
  - File
  - Subroutine
  - Loop

One metric does matter the most

- Overall wall time.
  Most sites charge utilization for this metric.
- Unix `time`.
  So easy it's practically free.
- System `time`
  Standard C `gettimeofday` or Fortran `cpu_time`.

Measures

Direct

- Observations
  - Code instrumentation
  - Counters
- Execution trace
  - Strace
  - Debuggers (gdb, totalview)

greater overhead, perturbation and resolution

Indirect

- Observations
  - Sampling
  - Triggers
- Execution trace
  - Strace -c
  - gprof, prof, psrun

less overhead, perturbation and resolution

Performance Measures

- Variety of tools available
- Use more than one approach
- Perturbation vs. resolution and accuracy

PerfSuite

- Functionality:
  - Counting overall hardware performance event counts for all or a portion of your application.
  - Profiling statistical sampling using either time or event-based triggers, generalization of the approach used by gprof
- Command line tools + libraries:
  - Commands `psrun`, `psprocess`, `psinv`, `psconfig`
  - Libraries 3 C libraries and 2 JVMTI agents
- PerfSuite itself does not require kernel patches

Four performance counter-related utilities

- `psconfig` configure / select performance events
- `psinv` query events and machine information
- `psrun` generate raw counter or statistical profiling data from an unmodified binary
- `psprocess` pre and post-process data

Three C libraries (shared and static, serial and threaded)

- libperfsuite – the “core” library
- libpshwpc – HardWare Performance Counter library. If counter support unavailable, will only perform time-based profiling through `profil()` or interval timers
- libpshwpc_mpi – a convenience library based on MPI PMPI interface

Processor Inventory

- Lists information about the characteristics of the computer
- This same information is also stored in PerfSuite XML output and is useful for later generating derived metrics (or for remembering where you ran your program!)
- x86/x86-64 version also shows processor features and descriptions
- Lists available hardware performance events

```
node0218 ~$ psinv -v

System Information -
Node Name:              node0218
OS Name:                Linux
OS Release:             2.6.32-279.5.2.el6.x86_64
OS Build/Version:       #1 SMP Tue Aug 14 11:36:39 EDT
OS Machine:             x86_64
Processors:             12
Total Memory (MB):      24150.31
System Page Size (KB): 4.00

Processor Information -
Vendor:                 Intel
Brand:                  Intel(R) Xeon(R) CPU
CPUID info:             family: 6, model: 44, stepping
Revision:               2
Clock Speed:            2800.06 MHz

Cache and TLB Information -
Cache levels:           3

Cache Details -
Level 1:
        Type:           Instruction
        Size:           32 KB
        Line size:      64 bytes
        Associativity:  4-way set associative
```

PAPI Event Summary

```
node0218 ~$ psinv -p

PAPI Standard Event Information -
Standard events:        58
Non-derived events:     44
Derived events:         14

PAPI Standard Event Details -
Non-derived:
        PAPI_BR_CN:     Conditional branch instructions
        PAPI_BR_INS:    Branch instructions
        PAPI_BR_MSP:    Conditional branch instructions mispredicted
        PAPI_BR_TKN:    Conditional branch instructions taken
        PAPI_BR_UCN:    Unconditional branch instructions
        PAPI_FP_INS:    Floating point instructions
        PAPI_L1_DCM:    Level 1 data cache misses
        PAPI_L1_ICA:    Level 1 instruction cache accesses
        PAPI_L1_ICH:    Level 1 instruction cache hits
        PAPI_L1_ICM:    Level 1 instruction cache misses
        PAPI_L1_ICR:    Level 1 instruction cache reads
        PAPI_L1_LDM:    Level 1 load misses
        PAPI_L1_STM:    Level 1 store misses
        PAPI_L2_DCA:    Level 2 data cache accesses
        PAPI_L2_DCR:    Level 2 data cache reads
        PAPI_L2_DCW:    Level 2 data cache writes
        PAPI_L2_ICA:    Level 2 instruction cache accesses
        PAPI_L2_ICH:    Level 2 instruction cache hits
```

psrun Example

```
# Get a debug session (1 node, 12 tasks)
login4 ~$ salloc --qos=janus-debug --time=02:00 \
          -N1 --ntasks-per-node=12

# Load the module
node0214 ~$ module load perfsuite
# Run your program with psrun (default counting)
node0214 ~$ echo 100000 | psrun ./triad.exe
# Process the output
node0214 ~$ psprocess triad.exe.0.5930.node0214.xml \
            | less
# Timing profile
node0214 ~$ echo 100000 | psrun -C \
            -c papi_profile_cycles.xml ./triad.exe
```

psprocess Output

```
Event Count Information
================================================================================
Index Description                                                  Counter Value
--------------------------------------------------------------------------------
    1 Conditional branch instructions.....................          51,274,199
    2 Branch instructions.................................          51,279,642
    3 Conditional branch instructions mispredicted........               6,409
    4 Conditional branch instructions taken...............          51,265,123
    5 Floating point instructions.........................         410,833,868
    6 Floating point operations...........................         410,811,630
    7 Level 1 data cache misses...........................         205,962,349
    8 Level 1 instruction cache accesses..................             436,270
    9 Level 1 instruction cache misses....................               6,655
   10 Level 2 instruction cache accesses..................               5,546
   11 Level 2 instruction cache misses....................               5,441
   12 Level 2 total cache accesses........................         279,059,387
   13 Level 2 cache misses................................         176,321,806
   14 Level 3 instruction cache accesses..................               3,860
   15 Level 3 total cache accesses........................         176,272,500
   16 Level 3 cache misses................................                   0
   17 Load instructions...................................         615,029,501
   18 Cycles stalled on any resource......................         937,233,297
   19 Store instructions.................................          204,980,198
   20 Data translation lookaside buffer misses............           3,265,086
   21 Instruction translation lookaside buffer misses.....                 245
   22 Total cycles........................................       1,590,331,406
   23 Instructions issued.................................             940,536
   24 Instructions completed..............................         973,799,552
```

Overview

Debugging Methods

Common Causes Of Bugs

DDT on Yeti

Profiling

MAP on Yeti

MAP on Yeti

1. Log in to Yeti.

```
laptop ~$ ssh yeti.cr.usgs.gov
```

2. Load the Intel parallel studio module

```
yeti-login01 ~$ module load intel/psxe-2015
```

3. Load the Allinea module

```
yeti-login01 ~$ module load allinea/6.0
```

4. Run MAP (the profiler)
   - 4.1 Interactively

     ```
     yeti-login01 ~$ map
     ```

   - 4.2 Batch mode

     ```
     compute80 ~$ map -n 4 -profile a.out
     ```

Questions?

Online Survey

<Timothy.Brown-1@colorado.edu>

License