Debugging Essentials via Allinea DDT

Ian Lumb
Senior Systems Engineer, Allinea Software Inc.
ilumb@allinea.com

OLCF Spring Training 2012
Agenda

- Bugs and Debugging
- Debugging Essentials via Allinea DDT
- Live Demo
"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

Brian Kernighan
Software Challenges in HPC

- The relentless march towards larger machines ...
  - Does your application scale?
    - Can it perform efficiently/effectively – or does it crash?
- Hardware flux – multicore, GPU, Intel MIC, ARM, ...
  - Developing and maintaining code for many targets
  - New execution models
    - Are software changes introducing bugs?
      - If so, how can they be fixed?
- Debugging is a vital piece of HPC development
Bugs in Practice
Some Types of Bugs

● Bohr bug
  • Steady, dependable bug
● Heisenbug
  • Vanishes when you try to debug (observe)
● Mandelbug
  • Complexity and obscurity of the cause is so great that it appears chaotic
● Schroedinbug
  • First occurs after someone reads the source file and deduces that it never worked, after which the program ceases to work
A `New' Vernacular for Bugs

- **Defect**
  - An incorrect program code
    - A bug in the code

- **Infection**
  - An incorrect program state
    - A bug in the state

- **Failure**
  - An observable incorrect program behaviour
    - A bug in the behaviour

Debugging

Transforming a broken program into a working one

How?

Track the problem

Reproduce

Automate - (and simplify) the test case

Find origins – where could the “infection” be from?

Focus – examine the origins

Isolate – narrow down the origins

Correct – fix and verify the testcase is successful

How to Focus and Isolate

- A scientific process?
  - Hypothesis, trial and observation, ...
- Requires the ability to understand what a program is doing
  - Printf
  - Command-line debuggers
  - Graphical debuggers
- Other options
  - Static analysis
  - Race detection
  - Valgrind
  - Manual source code review
What are Debuggers?

- Tools to inspect the insides of an application whilst it is running
  - Ability to inspect process state
    - Inspect process registers, and memory
    - Inspect variables and stacktraces (nesting of function calls)
    - Step line by line, function by function through an execution
    - Stop at a line or function (breakpoint)
    - Stop if a memory location changes
  - Ideal to watch how a program is executed
    - Less intrusive on the code than printf
    - See exact line of crash – unlike printf
    - Test more hypotheses at a time
How Debuggers Work

- Multiple methods of operation/implementation
  - Interpreted interactive environments – Ruby, Perl, etc.
    - Everything is under control of the implementation – easy access to the state of the system
    - Relatively easy extension to any interpreter
  - Virtual/managed environments – eg. Java
    - Public protocols hook into the virtual machine (ie. JDWP API)
      - Insert breakpoint, inspect classes and data
  - Native executables
    - A harder challenge – binaries run wild under operating system control
      - Examples: Eclipse, DDT, GDB, Allinea DDT
Debugging Parallel Applications

- The same need: observation, control, ...
  - A complex environment – with complex problems
    - More processes, more data
    - More Heisenbugs – MPI communication library introduces potential non-determinism
  - Few options ...
    - Cannot use printf or command line debuggers
  - Some bugs only occur at scale
    - Need to handle thousands of threads/processes
    - Needs to be fast to use and easy to understand
Debugging Parallel GPU Applications

- The same need: observation, control, ...
- A complex environment – with complex problems
  - Explicit data transfer between host and GPU
  - Hierarchy of memory levels
  - Grid/block layout and thread scheduling
  - Synchronization
  - Massively fine-grained parallel model
- Debugging options ...

Processing flow on CUDA
About Allinea

- HPC development tools company
  - Flagship product Allinea DDT
    - Now the leading debugger in parallel computing
    - The scalable debugger
      - Record holder for debugging software on largest machines
      - Production use at extreme scale – and desktop
    - Wide customer base
      - Blue-chip engineering, government and academic research
      - Strong collaborative relationships with customers and partners
Allinea DDT in a nutshell

- Graphical source level debugger for
  - Parallel, multi-threaded, scalar or hybrid code
  - C, C++, F90, Co-Array Fortran, UPC
- Strong feature set
  - Memory debugging
  - Data analysis
  - Managing concurrency
    - Emphasizing differences
    - Collective control
- “Make as simple as possible, no more”

www.allinea.com
Demo

- Crashes
- Memory errors and leaks
- Deadlocks
- Incorrect results
- GPU support

http://www.olcf.ornl.gov/kb_articles/software-jaguar-ddt/
http://www.allinea.com/downloads/ddt_training.tar.gz
Debugging Parallel CUDA Applications

- **Current status**
  - Software complexity reflects hardware complexity
  - cuda-gdb
    - Direct use challenging
    - Indirect use via a debugger
Allinea DDT and CUDA

- Supports
  - CUDA toolkits 3.1 -- 3.2 -- 4.0 -- 4.1

- Makes use of
  - NVIDIA C/C++ compiler - nvcc
  - NVIDIA debugger - cuda-gdb

- Execution model is unusual
  - GUI work required to support 32-thread units (warps) in blocks and grids

- Mixed GPU/CPU in one interface
  - Interaction with CPUs
  - Easy to switch between contexts (stacks, threads, data...)
  - Support multiple nodes
Allinea DDT and CUDA
Core Debugging Capabilities

• The first graphical debugger for NVIDIA CUDA
  – Simple and easy to use
  – As easy as debugging ordinary (i.e., non-GPU) code

• Core debugging capability
  – Breakpoints
  – Stepping warps
  – Viewing data and thread stacks within the GPU

• Supports advanced features
  – CUDA memcheck – memory debugging for CUDA
Allinea DDT and CUDA Seamless Integration within the GUI

- View all existing threads in parallel stack view
  - At one glance, see all GPU and CPU threads together
  - Links with thread selection
  - Pick a tree node to select one of the CUDA threads at that location

- Full MPI support
  - See GPU and CPU threads from multiple nodes
Allinea DDT and CUDA
Kernel Progress

- Has my thread calculated the output yet? Is it to be scheduled?
  - Contrast with scalar programming

- Keep an eye on your kernel progress across processes

![Kernel Progress View](image)
Array Visualization Support

- Browse arrays
  - 1, 2, 3, … dimensions
- Table view
- Filtering
  - Look for an outlier
- Export
  - Save to a spreadsheet
Summary

- Bugs and Debugging
- Debugging Essentials via Allinea DDT
- Live Demo