Debugging Large Scale and Hybrid Parallel Programs

David Lecomber
david@allinea.com
CTO
Interesting Times ...

• A recent history of parallel computing
  – Increasing core counts
  – Increasing cores per node
  – Established programming models
    • Near-100% of HPC software using MPI and/or OpenMP
    • A close match of software to hardware - good portability
  – The challenge of **application scalability** remains!

• Times change ...
  – GPUs entering HPC – power, performance, ...
  – Massive multi-core clusters – with many GPUs
  – The challenge of **hybrid software** is here!
“Software has become the #1 roadblock ... Many applications will need a major redesign” - IDC HPC Update, June 2010
- Most ISV codes do not scale
- High programming costs are delaying GPU usage

Development tools are a vital part of the solution
• UK based HPC tools company since 2001
  - Allinea DDT – the scalable parallel debugger
  - Allinea OPT – the optimization tool for MPI and non-MPI
  - Allinea DDTLite – the parallel debugging plugin for Microsoft Visual Studio

• Large European and US customer base
  - Ease of use – means tools get used
  - Users debugging regularly at all scales – at 1 or 100,000 cores
  - World's only Petascale debugger!
Some Clients and Partners

- Academic
  - Over 200 universities
- Major research centres
  - ANL, EPCC, IDRIS, Juelich, NERSC, ORNL,
- Aviation and Defence
  - Airbus, AWE, Dassault, DLR, EADS, ...
- Energy
  - CEA, CGG Veritas, IFP, Total, ...
- EDA
  - Cadence, Intel, Synopsys, ...
- Climate and Weather
  - UK Met Office, Meteo France, NOAA ...
• Debugging: A necessary part of development
  – Reproducing and fixing software problems
    • Complexity of scaling and GPU architecture will introduce bugs
  – Debuggers interactively examine processes and data
    • Fastest way to debug – with less chance of introducing more bugs

  – Bugs at **scale** need a debugger at **scale**
    • ... until recently debuggers limited to ~4,000-8,000 cores
  – Bugs **on GPUs** need a debugger **for GPUs**
    • ... until recently GPU software couldn't be debugged

... Allinea DDT is the first graphical debugger to do both
DDT in a nutshell

- **Scalar features**
  - Advanced C++ and STL
  - Fortran 90, 95 and 2003: modules, allocatable data, pointers, derived types
  - Memory debugging

- **Multithreading & OpenMP features**
  - Step, breakpoint etc. one or all threads

- **MPI features**
  - Easy to manage groups
  - Control processes by groups
  - Compare data
  - Visualize message queues
Memory Debugging

• Find memory leaks

• Or stop on read/write beyond end of array
Is scale relevant?

- **A major problem**
  - Was exclusive to big labs
  - Everyone is joining the fun
  - If you can't debug problems at scale, you can't fix them

- **Historic debugger limits**
  - Linear or worse performance
  - Subjective view of limit: pain threshold varies: 1 second, 1 minute, 1 hour?
DDT: Petascale Debugging

- **DDT is delivering petascale debugging today**
  - Collaborations with ORNL on Jaguar Cray XT and CEA
  - Tree architecture – logarithmic performance
  - Many operations now faster at 220,000 than previously at 1,000 cores
  - \(~1/10^{th}\) of a second to step and gather all stacks at 220,000 cores
Scalable Process Control

- **Parallel Stack View**
  - Finds rogue processes faster
  - Identifies classes of process behaviour
  - Allows rapid grouping of processes

- **Control Processes by Groups**
  - Set breakpoints, step, play, stop etc. using user-defined groups
  - Mutates to scalable groups view
  - Compact group representations
• Gather from every node
  - Potentially costly – if all data different
  - Easy if data mostly same
  - New ideas
    • Aggregated statistics
    • Probabilistic algorithms optimize performance – even in pathological case

• Watch this space!
  - With a fast and scalable architecture, new things become possible
• Hybrids are today's hottest topic
  - Technology is moving quickly – compilers, SDKs, hardware
  - NVIDIA CUDA leads in tool support

• Many lines of code need rewriting for GPUs
  - Memory hierarchy
  - Explicit data transfer between host and accelerator
  - Unusual execution model -
    • Kernels, thread blocks, warps, synchronization points
  - Massively fine-grained parallel model

• Inevitable that we need to debug!
Debugging Options

• Old world “printf”
  – NVIDIA SDK 3.0 allows this (new) – but has limitations

• Fake it – run the kernel on the host x86_64 processor
  – Languages often support targeting host CPU instead of GPU
  – Different numeric precision – different answer?
  – Different scheduling – different answer?
  – A reasonable option for some bugs

• Or run on the GPU with Allinea DDT...
Introducing DDT for CUDA

• The first graphical debugger for NVIDIA CUDA
  – Simple and easy to use
  – As easy as debugging ordinary code

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

• Supports advanced features
  – CUDA memcheck – memory debugging for CUDA

• More to come!
• Run the code
  - Browse source
  - Set breakpoints
  - Stop at a line of CUDA code
  - Stops once for each scheduled collection of blocks

• Select a CUDA thread
  - Examine variables and shared memory
  - Step a warp
Easy to understand scale

- View all extant 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
Some Common Problems

• Incorrect logic (if-statements, calculations)
  – Loop iteration to GPU thread analogy - threads identified by grid and block indexes
  – **Solution:** Select a thread and step with DDT; look at the local state and shared data
    • Cherry-pick important threads: start, end, a few interior points

• Kernel bounds – getting the right grids and blocks
  – Incorrect kernel thread boundaries can lead to incomplete results or crashing of the kernel
  – **Solution:** Bugs will often trigger “CUDA memcheck” errors - run with DDT and CUDA memory debugging enabled
  – **Solution:** Use DDT's advanced multi-dimensional array viewer to look at data and find the missing indexes
Current Limitations

• SDK 3.0 is a big leap forward
• SDK and driver limitations
  – Only one GPU can be debugged per O/S (per physical node)
  – Cannot currently read launch failure codes (without breaking your code)
  – Only one warp can be stepped per GPU at any time
  – Cannot debug GPU part of (attach to) an already running job

• Strong partnership with NVIDIA, CAPS and others is helping to extend capabilities
Questions?