Petascale Debugging with Allinea DDT

HPC ADVISORY COUNCIL WORKSHOP 2010
• Processor counts growing rapidly

• GPUs entering HPC

• Large hybrid systems imminent

• But what happens when software doesn't work?
• **Debuggability**
  - A subjective measure of the ability to be debugged
• **Linear tool architectures**
  - Linear (or worse) bottlenecks
  - Pain threshold varies: 1 second, 1 minute, 1 hour?
• **A major problem**
  - Previously exclusive to big labs
  - Now everyone is joining in the fun
Approaches to Scale

• Ignore the problem
  – Pretend bugs at scale do not happen

• Best programming practices
  – Consistency checking and self-diagnosis within code
  – Still frustrated by some types of bug

• Lightweight debugging
  – STAT (LLNL) identifies equivalent processes using stacks
  – STAT calls Allinea DDT (or TTV) to debug representatives
  – Other work is promising

• But what about full-strength debuggers?
• Many benefits to graphical parallel debuggers
  – Large feature sets for common bugs
  – Richness of user interface and real control of processes

• Historically **all** parallel debuggers hit scale problems
  – Bottleneck at the frontend: Direct GUI → nodes architectures
    • Linear performance in number of processes
  – Human factors limit – mouse fatigue and brain overload

• Are tools ready for the task?
  – Allinea DDT has changed the game
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
• Find memory leaks

• Or stop on read/write beyond end of array
GPU Debugging

• 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
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
DDT: Petascale Debugging

- Allinea DDT is delivering petascale debugging today
  - Collaboration with ORNL on Jaguar Cray XT
  - Tree architecture – logarithmic performance
  - Many operations now faster at 220,000 than previously at 1,000 cores
  - ~1/10th of a second to step and gather all stacks at 220,000 cores
Presenting Data, Usefully

• 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
Data Gathering Results

• Benchmarked on five codes on Jaguar XT
  – Stacks gathering mileage can vary: default install at ORNL has full debug info deep into MPI
  – Cross Process Comparison
    • Of equal variable
    • Of MPI rank (a bad case!)
• Most features now scale
  – Attach, run, process control and breakpoints
  – Process stacks
  – Data comparison
  – Memory debugging – out-of-bound array access, leaks, etc.
  – Import/export – stacks (XML/CSV), arrays, compared data
  – Tested at 220k cores on XT; 8k on Blue Gene P (SMP mode) – more timings soon; Ranger (Linux IB cluster)
  – New distributed array features
  – New grow/shrink attached-set - in addition to existing subset capabilities
• **Lessons learnt**
  
  – The scalable tree has really delivered!
    • More optimizations still possible
  
  – Even if you're quick, it's still all about the GUI
    • Present sensibly to the user – parallel stacks, data comparison
    • ... but some machines don't encourage full power of debugging due to their architecture
  
  – MPI spec probably never meant debuggers to scale!
    • Still linear things in there.. eg. MPIR_proctable
  
  – It's hard to debug a debugger without a debugger
Limits of the approach

• Logarithmic performance should last for many years
  – Any linear factors will eventually dominate
    • Must eradicate them all over time
    • Any memory usage on per-process basis
  – More intelligence can be pushed down the tree as need arises
  – Predict core operations on 1M or 10M cores will be under the pain threshold
    – SIMD/almost-SIMD GPUs fit within current approach (as threads, not individual processes)
• ... but bugs can still be hard to find
Collaboration opportunity

- No single organization has the resources to do everything
  - Plenty of opportunity for everyone in debugging
  - We use tools independently – but using together is more compelling
- Examples:
  - MPI correctness checking – Marmot, Intel MPI Checker
  - Library specific sanity checkers for data
  - Comparative debugging
- Ideal scenario: easy to prototype new bug finding ideas
  - Not tied to a particular product – but tied to an open API/scripting language
  - Single process or built from the top (drive a full debugger, or eg. combination of Wisconsin tools)
Come and see us at ISC10
Booth #745

Thank you