Large Scale Debugging on Titan and Mira with Allinea DDT

David Lecomber
Allinea Software
david@allinea.com
Allinea Software

• Our mission: to make HPC software development fast, simple and successful
  – A modern integrated environment for HPC developers
  – Scalable tools for any scale of system

• Supporting the lifecycle of application development and improvement
  – Allinea DDT : Productively debug code
  – Allinea MAP : Enhance application performance

• Designed for productivity
  – Consistent integrated easy to use tools
  – Enables effective use of HPC resources and expertise
Major Supercomputing Centers
Extreme machines are everywhere

Machine sizes are exploding

Software scale grows as machines grow
Some Software Challenges for the Extreme

Algorithmic: Compilers are not enough!
- Restructure for SIMD threads and vectorization
- Fundamental changes: Do we really need FFTs here?
- Rediscover PRAM and 0-1 Sorting Networks(!)

Programmer Efficiency
- MPI alone is not sufficient: Hybrid required
- Performance trade-offs harder to understand
- Software bugs harder to fix
Bugs in Practice
<table>
<thead>
<tr>
<th>Bug Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bohrbug</td>
<td>Steady, dependable bug</td>
</tr>
<tr>
<td>Heisenbug</td>
<td>Vanishes when you try to debug (observe)</td>
</tr>
<tr>
<td>Mandelbug</td>
<td>Complexity and obscurity of the cause is so great that it appears chaotic</td>
</tr>
<tr>
<td>Schroedinbug</td>
<td>First occurs after someone reads the source file and deduces that it never worked, after which the program ceases to work</td>
</tr>
</tbody>
</table>
Debugging in practice...

- Run
- Compile
- Crash
- Insert print statements
- Hypothesis
Print statement debugging

- The first debugger: print statements
  - Each process prints a message or value at defined locations
  - Diagnose the problem from evidence and intuition
- A long slow process
  - Analogous to bisection root finding
- Broken at modest scale
  - Too much output – too many log files
Titan and Mira

Titan
- 18,688 nodes
- 18,688 NVIDIA Kepler K20 GPUs
- 299,008 CPU cores
- 50,233,344 CUDA cores

Mira
- 49,152 nodes
- 786,432 cores
- 3,145,728 hardware threads

Does the printf workflow “work”? 
Bug fixing as scale increases

Reproduce at a smaller scale?
- Reduced data set - may not trigger the problem?
  - Didn't you already try the code at small scale?
    - Is it a system issue?
    - Is probability stacking up against you?

Debugging at extreme scale is a necessity
Three Challenges for tools

Scalability
• Speed and Simplification

Heterogeneity
• Accelerators and Coprocessors

Adoption
• Ease of Use and Education
What you should expect (demand!) for debugging at scale

Scalability
• A debugger that works to at least as high a scale as you need

Hardware and software support
• Whatever software you use and wherever you use it – the debugger supports it

Assistance
• Debugger is installed, configured, and documented – with site experts and training
ALCF, OLCF and Allinea deliver

2009 - Allinea and Oak Ridge begin collaboration to provide super-Petascale debugging

2010 - Allinea and Argonne collaboration to extend scaling to BlueGene systems

2013 - Mira and Titan full size debugging in place
Beneath the Petascale Allinea DDT

- **Scalable tree network**
  - Sends bulk commands and merge responses
  - Aggregations maintain the essence of the information
  - Optimizations to enable BlueGene architecture

- **Usability matters**
  - The interface is as important as the speed
  - Focus on scalable components
Allinea DDT
Fix software problems, fast

• Powerful graphical debugger designed for:
  – C/C++, Fortran, UPC, …
  – MPI, OpenMP and mixed-mode code
  – Accelerators and coprocessors: CUDA and Intel Xeon Phi

• Unified interface with Allinea MAP:
  – One interface eliminates learning curve
  – Spend more time on your results

• Slash your time to debug
  – Reproduces and triggers your bugs instantly
  – Helps you easily understand where issues come from quickly
  – Helps you to fix them as swiftly as possible
Allinea DDT: Scalable debugging by design

• **Where did it happen?**
  – Allinea DDT leaps to source automatically
  – Merges stacks from processes and threads

• **How did it happen?**
  – Some faults evident instantly from source

• **Why did it happen?**
  – Real-time data comparison and consolidation
  – Unique “Smart Highlighting” – colouring differences and changes
  – Sparklines comparing data across processes

  – **Force crashes to happen?**
  – Memory debugging makes many random bugs appear every time
Interlude: Local Demonstration

- Simple persistent hanging
  - Stepping through a code

- Process count dependent hanging:
  - Attaching to the running job
Example – ORNL’s Jaguar

- HPC code fails on 98,304 cores
- Random processes crashing
- Printf? Which processes and where?
- Too costly to repeat
- Allinea DDT finds cause first time
Getting started on Titan

• How?
  module load ddt
ddt

• Congratulations, you are now ready to debug.
Titan Interlude

• Learn how it feels to debug at scale
Example – ANL Mira

HPC code fails on 16,384 cores

Code abandoned – bug couldn’t be fixed

Machine too busy for interactive debugging

Allinea DDT offline mode runs bug case overnight

Found error in initialization
Getting started on Mira

• How?
  – soft add +ddt
  – ddt

• Congratulations you are now ready to debug.
Offline debugging

- Interactive access difficult
- Used offline mode
- Submit and forget
- Post-mortem analysis
- A scalable print alternative
  - Merged print – with a sparkline graph showing distribution
  - No recompilation required
Mira Interlude

• Use offline debugging to full advantage
Top 5 features at scale

- Parallel stack view
- Automated data comparison: sparklines
- Parallel array searching
- Step, play, and breakpoints
- Offline debugging
Summary

Debugging at scale is not difficult

- 300,000 cores is as easy as 30 cores
- The user interface is vital to success

Debugging at scale is not slow

- High performance debugging – at Mira and Titan scale
- Logarithmic performance

Stable, in production and well supported

- Routinely used over 100,000 cores