allinea

Leaders in parallel software development tools

Debugging HPC Applications

David Lecomber

CTO, Allinea Software

david@allinea.com

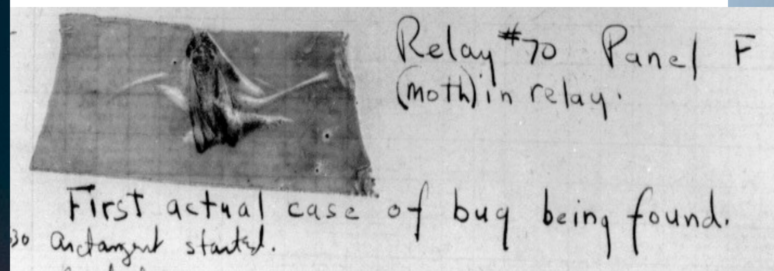
Agenda

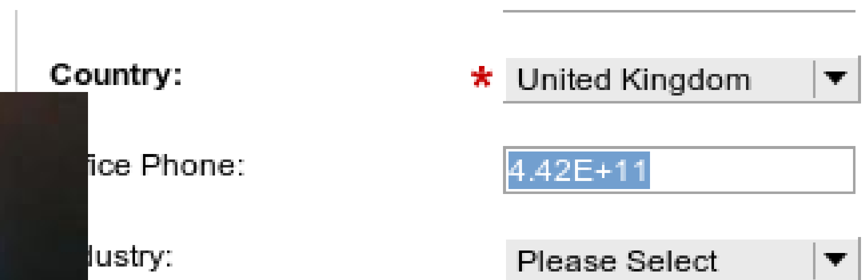
- Bugs and Debugging
- Debugging parallel applications
- Debugging OpenACC and other hybrid codes
- Debugging for Petascale (and beyond)

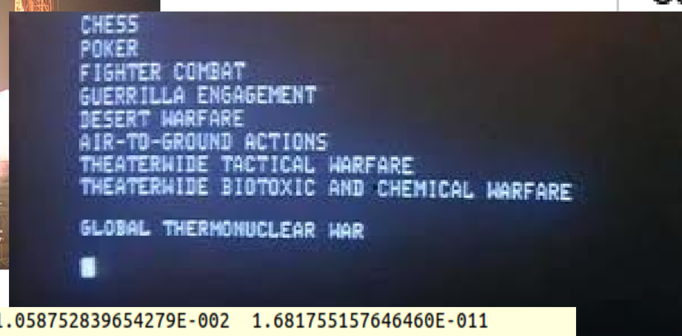
About Allinea

- 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

Bugs in Practice








```
   124395.444928040      1.058752839654279E-002  1.681755157646460E-011
   124395.444323148      1.119242021240481E-002  1.681205346551746E-011
   124395.443701451      1.181411574161518E-002  1.680444969505865E-011
   124395.443062951      1.245261508079283E-002  1.679731384893576E-011
   124395.442407647      1.310791832922166E-002  1.679052894606482E-011
   124395.441735539      1.378002558885051E-002  1.678304215668999E-011
^\forrtl: error (79): process quit (SIGQUIT)
Image              PC                Routine            Line        Source

omp-break          0000000000405400  Unknown               Unknown  Unknown
omp-break          0000000000404B23  Unknown               Unknown  Unknown
libiomp5.so        00007F6E3A7C6B93  Unknown               Unknown  Unknown
Aborted (core dumped)
```

Some types of bug

- Some Terminology
  - 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

Debugging

- Transforming a broken program to a working one
- **How?**
  - **T**rack the problem
  - **R**eproduce
  - **A**utomate - (and simplify) the test case
  - **F**ind origins – where could the "infection" be from?
  - ***F****ocus – examine the origins*
  - ***I****solate – narrow down the origins*
  - ***C****orrect – fix and verify the testcase is successful*
- TRAFFIC
- Suggested Reading:
  - Zeller A., "Why Programs Fail", 2nd Edition, 2009

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

Debugging Parallel Applications

www.allinea.com

Debugging Parallel Applications

- Scalar bugs can be challenging: parallel even more so!
- The same need: observation, control
  - Complex environment – with complex problems
    - More processes, more data
    - More Heisenbugs – MPI communication library introduces potential non-determinism
  - Fewer options
    - Printf or command line debuggers are not quick enough

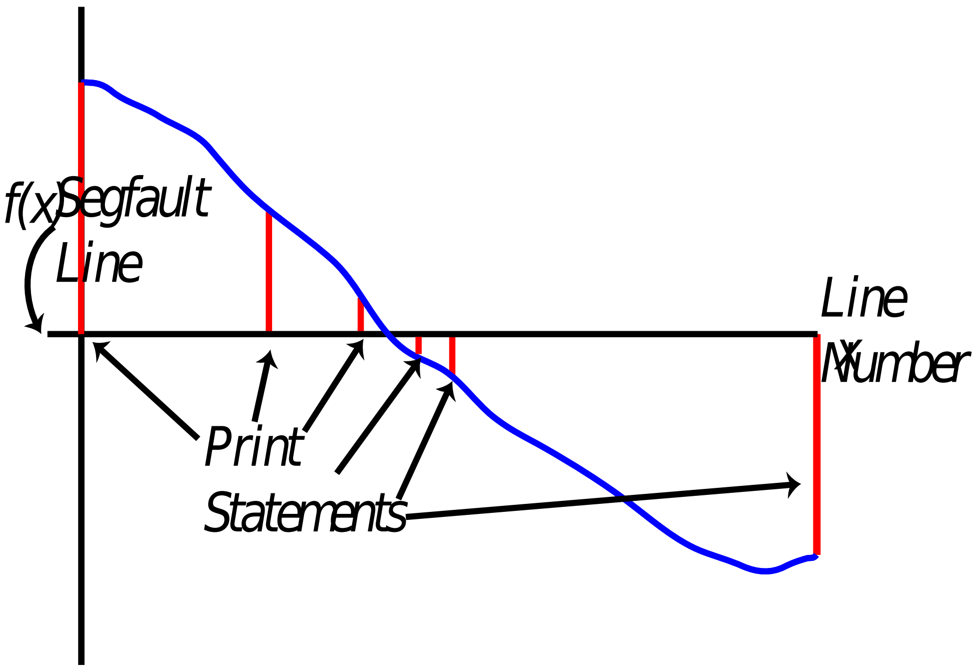
First example

- Typical problem scenario: application ends abruptly
  - Example potential causes
    - Segmentation fault
    - Early termination due to invalid parameters
- Where do we start?

Print statement debugging

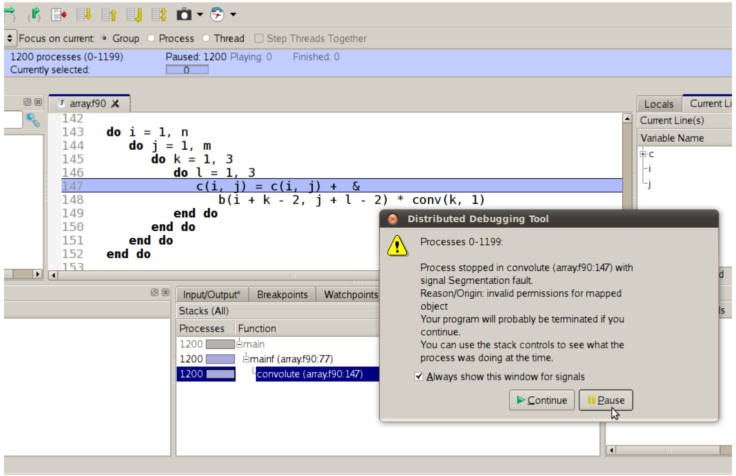
- The first debugger: print statements
  - Each process prints a message or value at defined locations
  - Diagnose the problem from evidence and intuition
- A long slow process
  - Analogous to bisection root finding

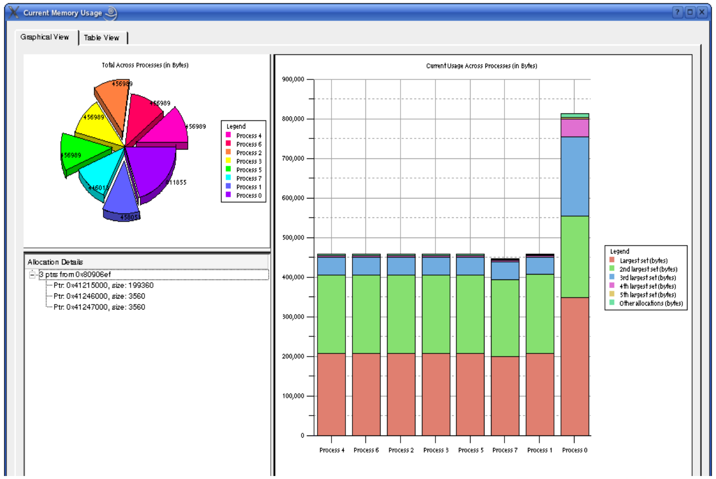
    Thanks to Rebecca Hartman-Baker of ORNL for the analogy and animation
- Broken at modest scale
  - Too much output – too many log files



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



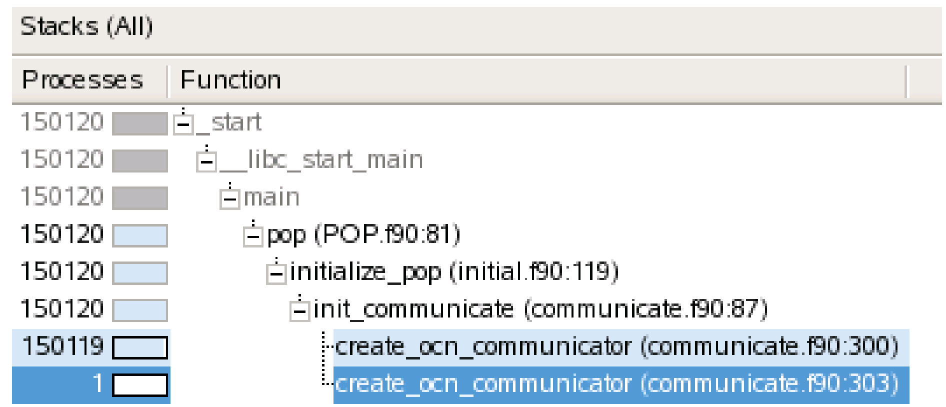


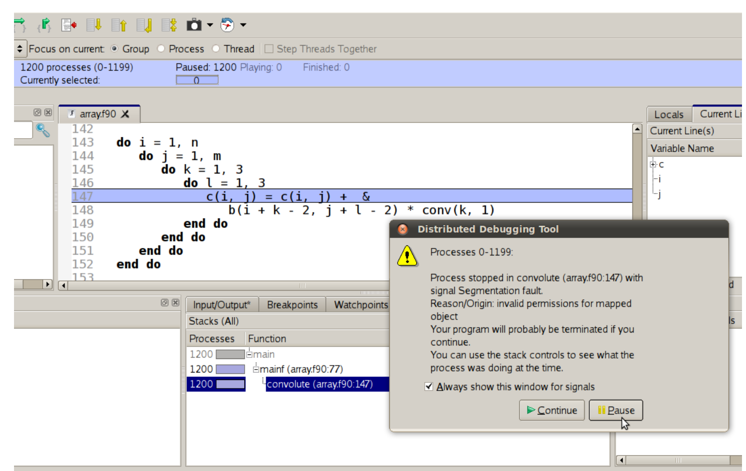
www.allinea.com

Fixing the everyday crash

- The typical application crash or early exit:
  - Run your program in the debugger

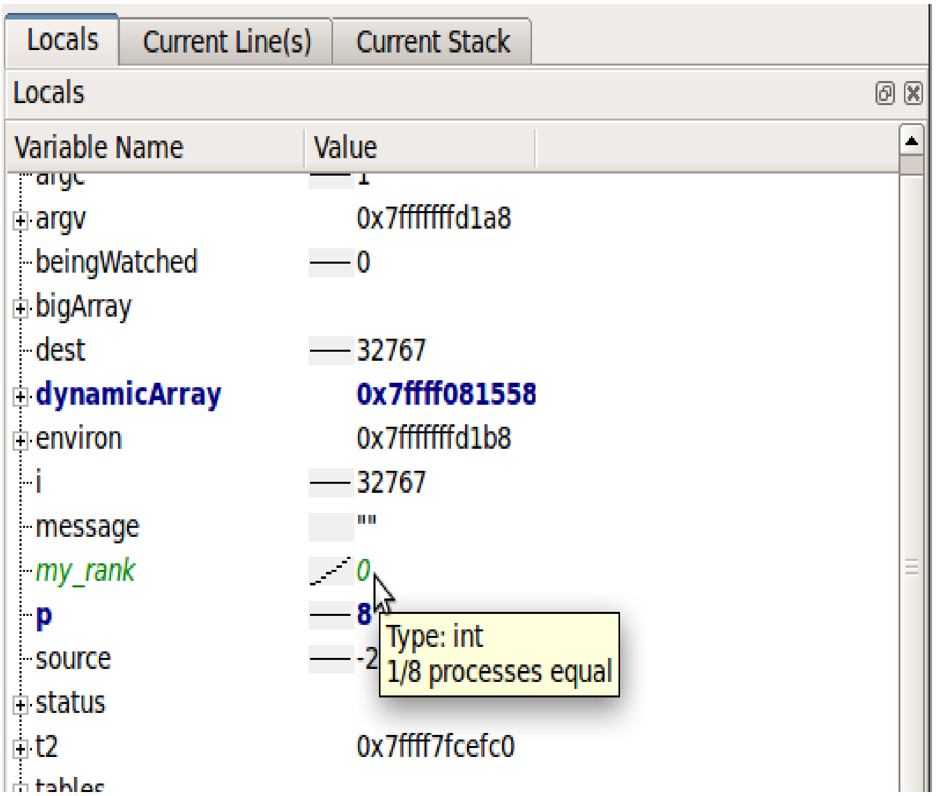
    ddt {application} {parameters}
  - Application crashes or starts to exit
- **Where** did it happen?
  - Allinea DDT merges stacks from processes and threads into a tree
  - Leaps to source automatically
- **Why** did it happen?
  - Some faults evident instantly from location(s)
  - But for others we need to look further – at variables





Simplifying data divergence

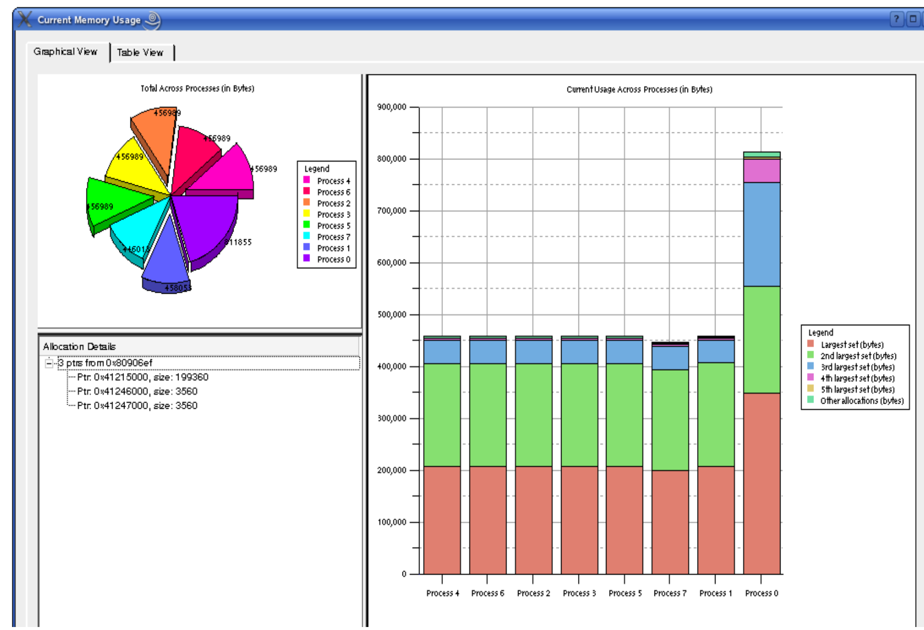
- Need to understand the data
  - Too many variables to trawl manually
  - Allinea DDT compares data automatically
- Smart highlighting
  - Subtle hints for differences and changes
  - With sparklines!
- More detailed analysis
  - Full cross process comparison
  - Historical values via tracepoints



allinea
www.allinea.com

Memory debugging

- Random errors are the worst kind
  - You can't fix a bug that doesn't repeat - memory debugging can force the bug
  - Better to crash every time, than only during product demos
- Allinea DDT helps eliminate random memory bugs
  - Enable memory debugging by ticking an option
  - Monitors usage: detects memory leaks
  - Automatically protects ends of arrays
  - Trigger instant stop on touching invalid memory
  - Also with CUDA support



allinea
www.allinea.com

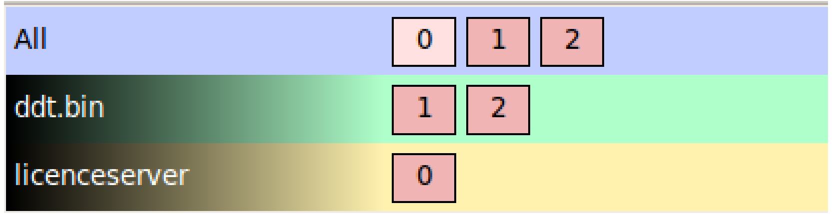
Interlude: Fixing a simple MPI bug

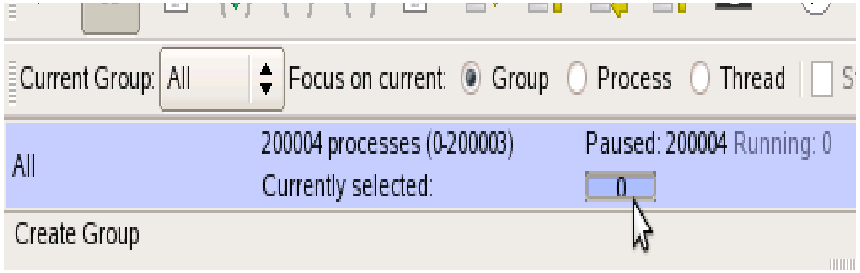
Second example

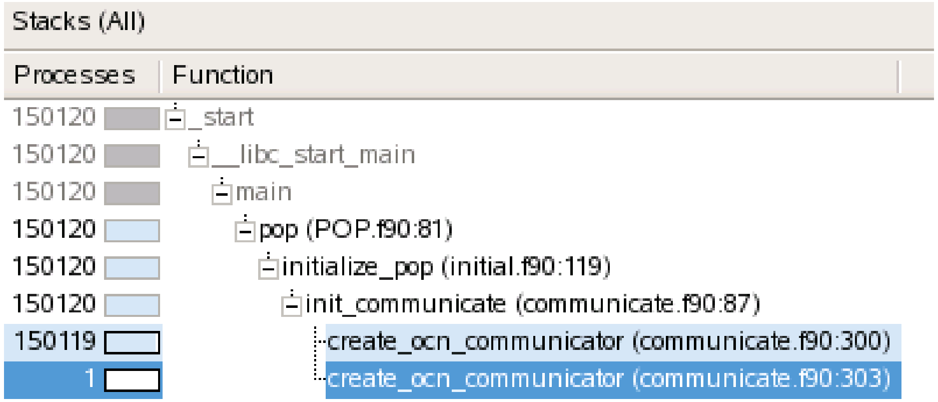
- More subtle issue
  - Not an immediately obvious crash
    - Crash occurs in self-evidently correct code
    - Something went wrong – somewhere else!
- How can a debugger help here?
  - Observation: we can watch things go bad

Controlling execution

- Observe application behaviour by controlling execution
  - Step, play or run to line based on groups
  - Change interleaving order by stepping/playing selectively
  - Set breakpoints
  - Set data watchpoints
- Examine data and progress at each point
- Group creation is easy
  - Integrated throughout Allinea DDT - eg. stack and data views

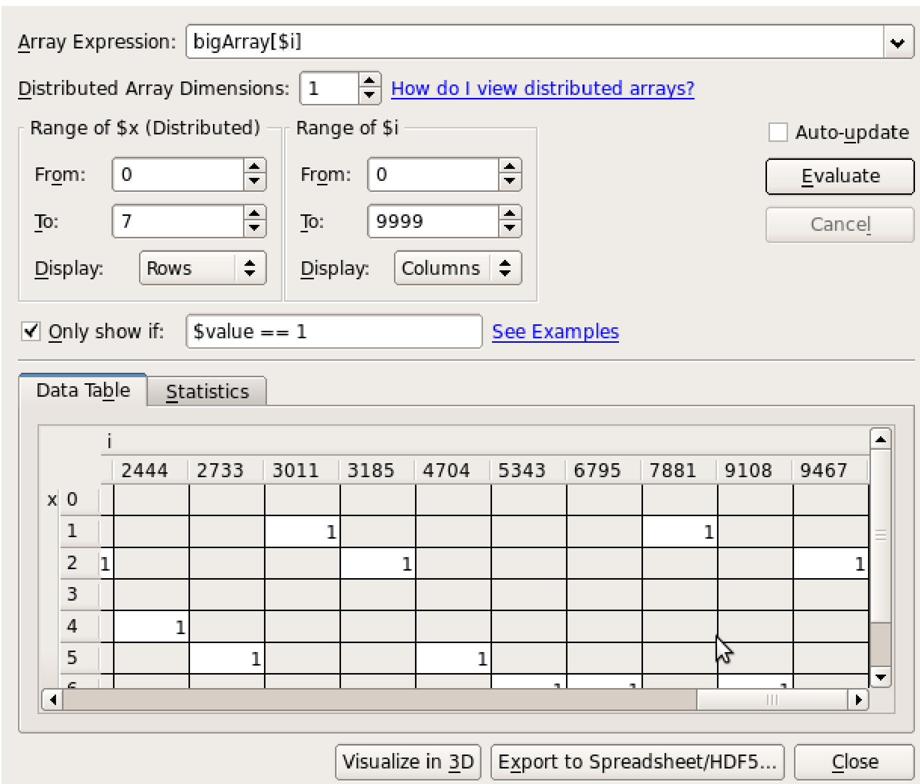






www.allinea.com

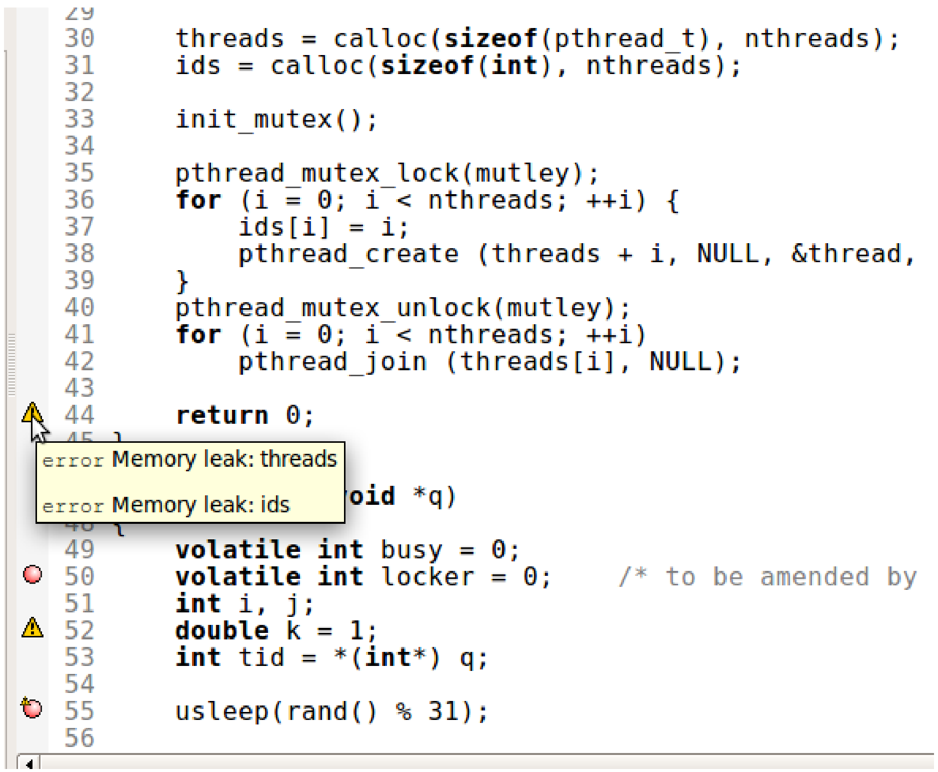
Exploring large arrays



- Browse arrays
  - 1, 2, 3, … dimensions
  - Table view
- Filtering
  - Look for an outlier
- Export
  - Save to a spreadsheet
- View and search arrays from multiple processes
  - Search terabytes for rogue data – in parallel

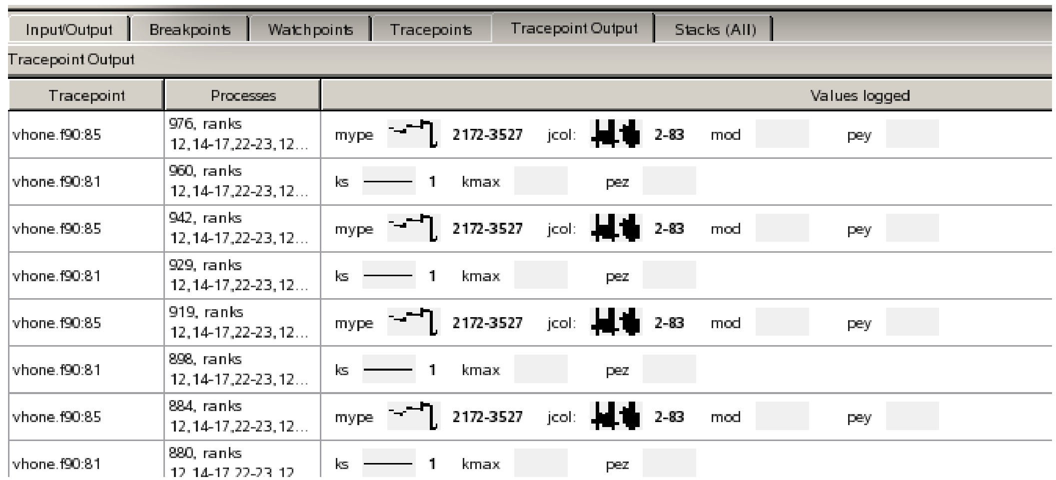
www.allinea.com

Static analysis



- Analyzes source code
  - Detects some common errors (eg.)
    - Memory leaks
    - Buffer overflow
    - Unused variables
  - Not exhaustive – but a useful hint for debugging

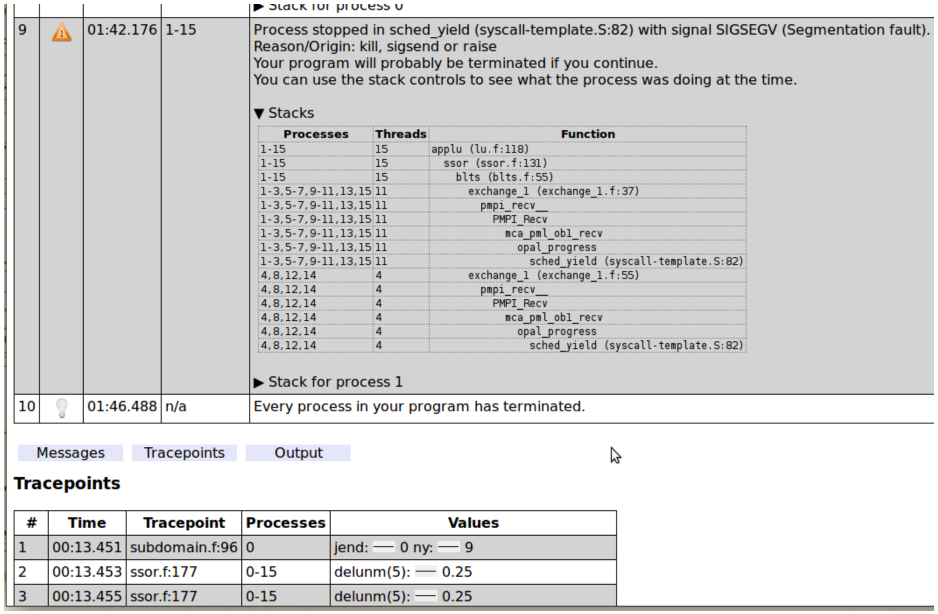
Tracepoints



- A scalable print alternative
  - Merged print – with a sparkline graph showing distribution
  - Change at runtime – no recompilation required

Offline debugging

- Machine access can be a problem
  - New offline mode
    - Set breakpoints, tracepoints from command line
    - Memory debugging
    - Record variables, stacks – crashes and breakpoints
  - Submit and forget
    - Post-mortem analysis
    - HTML/plain text
  - Debug while you sleep



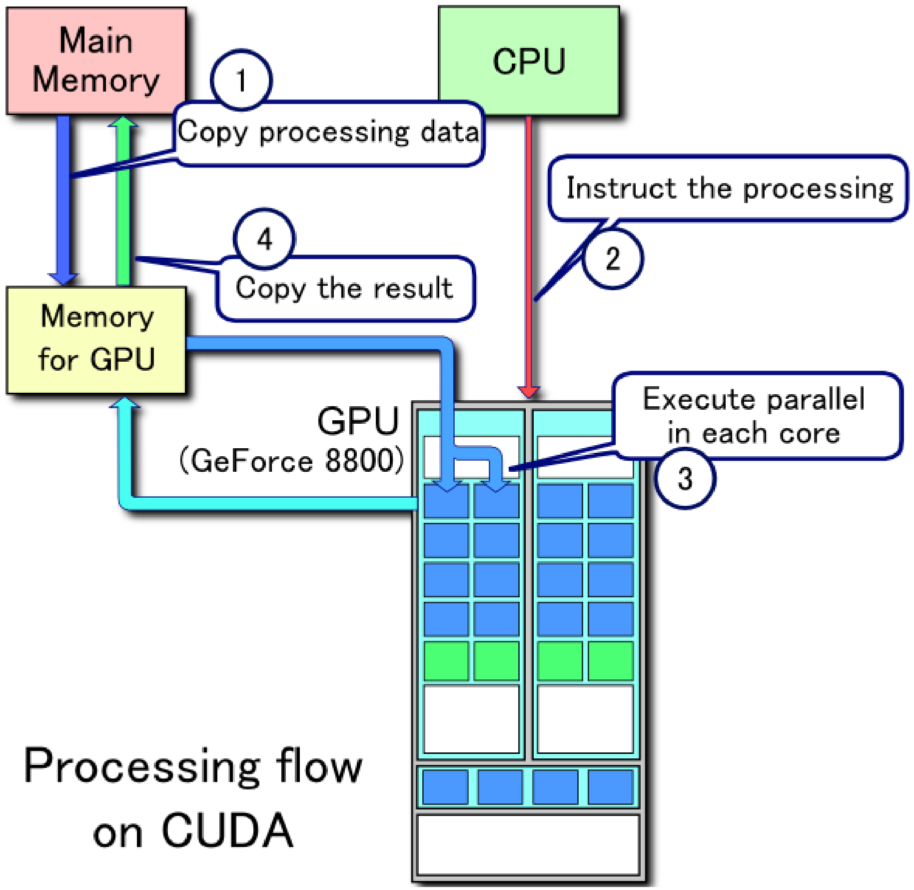
Interlude: Fixing a more unusual bug

Hands on: A first exercise with Allinea DDT

Debugging OpenACC and other hybrid codes

HPC's current challenge

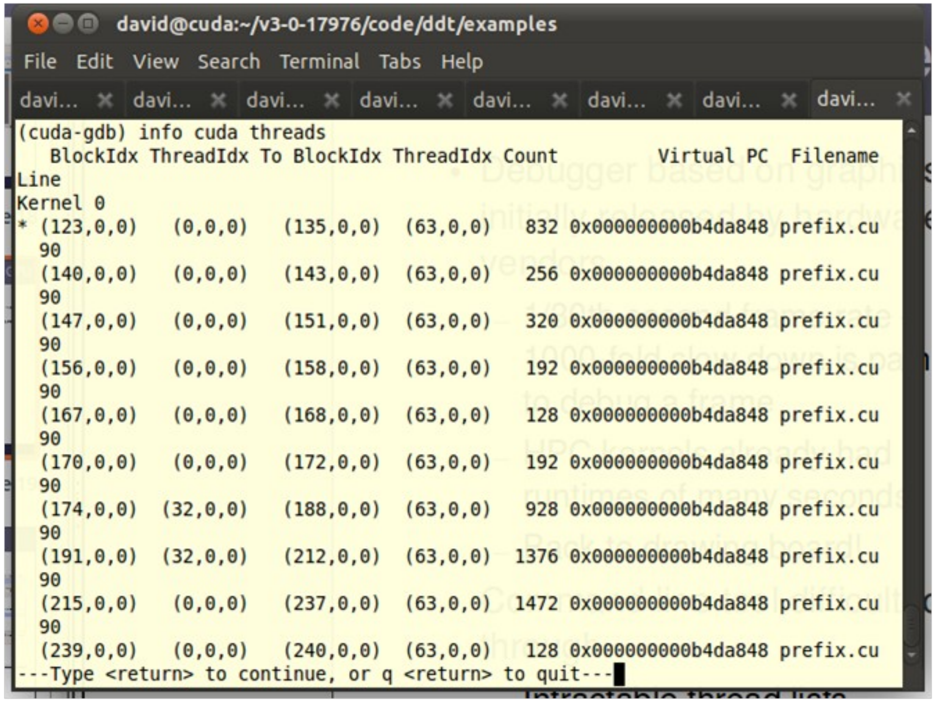
- GPUs – a rival to traditional processors
  - AMD and NVIDIA
  - OpenCL, CUDA
- New languages, compilers, standards
- Great bang-for-bucks ratios
- A big challenge for HPC developers
  - Data transfer
  - Several memory levels
  - Grid/block layout and thread scheduling
  - Synchronization
- Bugs are **inevitable**



allinea
www.allinea.com

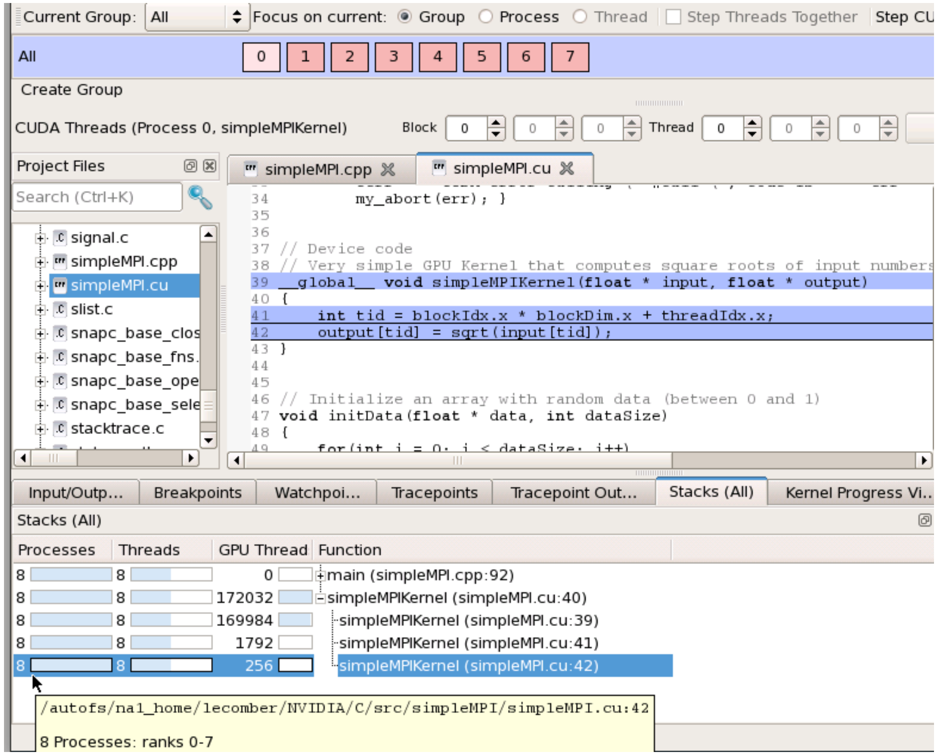
How do we fix GPU bugs?

- Print statements
  - Too intrusive
- Command line debugger?
  - A good start:
    - Variables, source code
    - Large thread counts overwhelming
  - Too complex
- A graphical debugger...



GPU Debugging

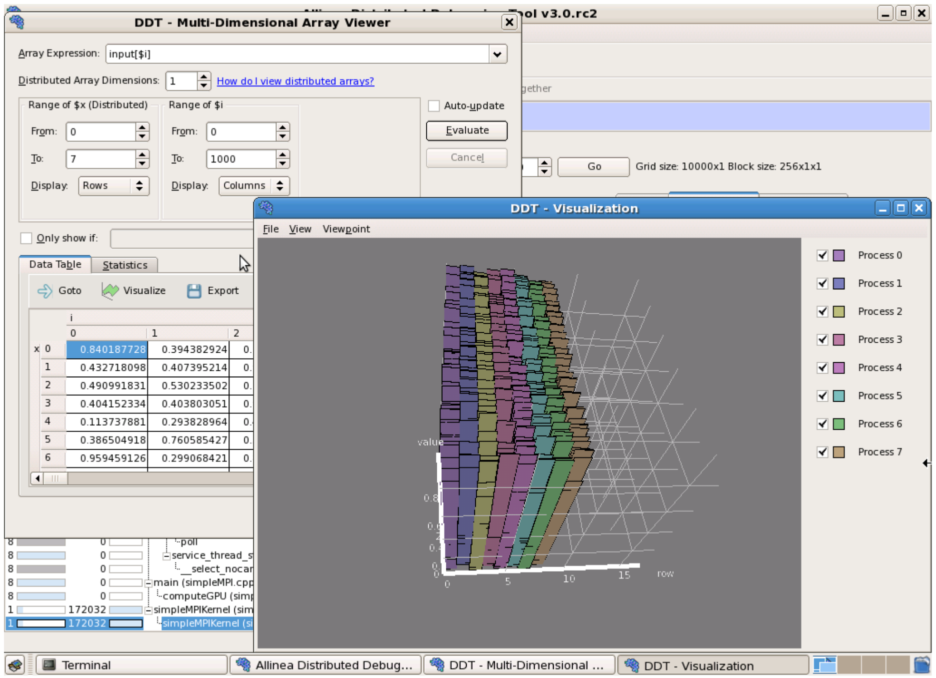
- Almost like debugging a CPU – we can still:
  - Run through to a crash
  - Step through and observe
- CPU-like debugging features
  - Double click to set breakpoints
  - Hover the mouse for more information
  - Step a warp, block or kernel
  - Follow threads through the kernel
- Simultaneously debugs CPU code
- CUDA Memcheck feature detects read/write errors



allinea
www.allinea.com

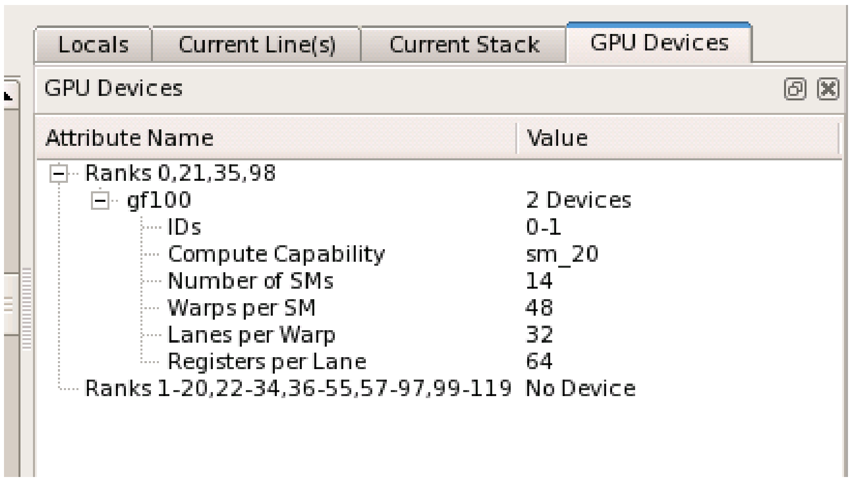
Examining GPU data

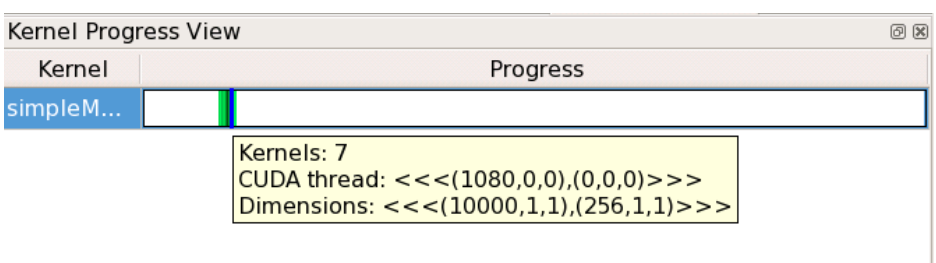
- Debugger reads host and device memory
  - Shows all memory classes: shared, constant, local, global, register..
  - Able to examine variables
  - … or plot larger arrays directly from device memory



allinea
www.allinea.com

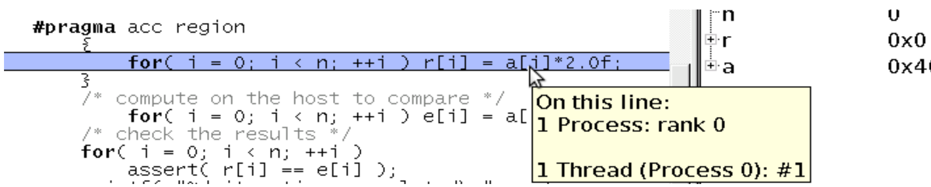
New overviews of GPUs



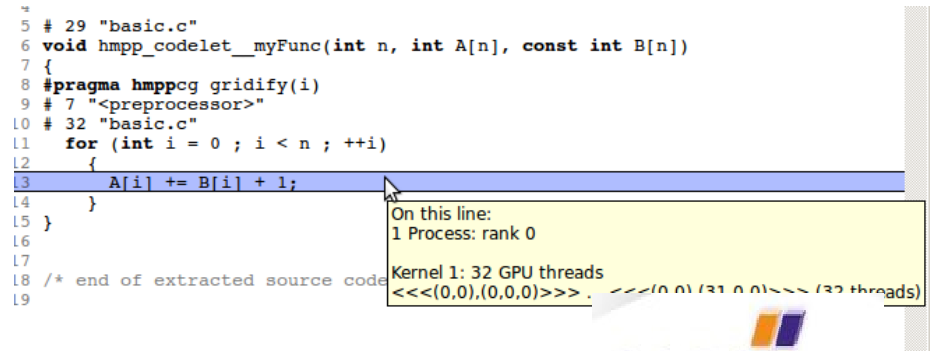


- Device overview shows system properties
  - Helps optimize grid sizes
  - Handy for bug fixing – and detecting hardware failure!
- Kernel progress view
  - Shows progress through kernels
  - Click to select a thread

Debugging for Directives



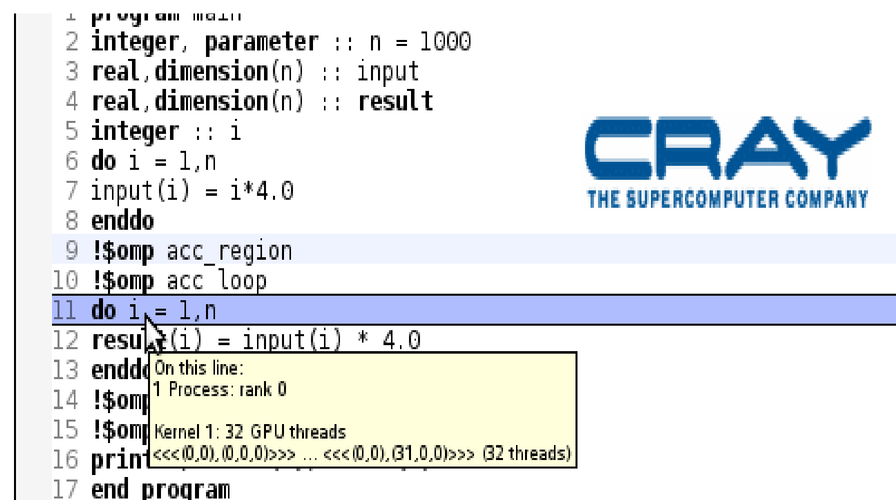
PGI



CAPS

Innovative software for manycore paradigms

- Supporting the environments that you use for hybrid development



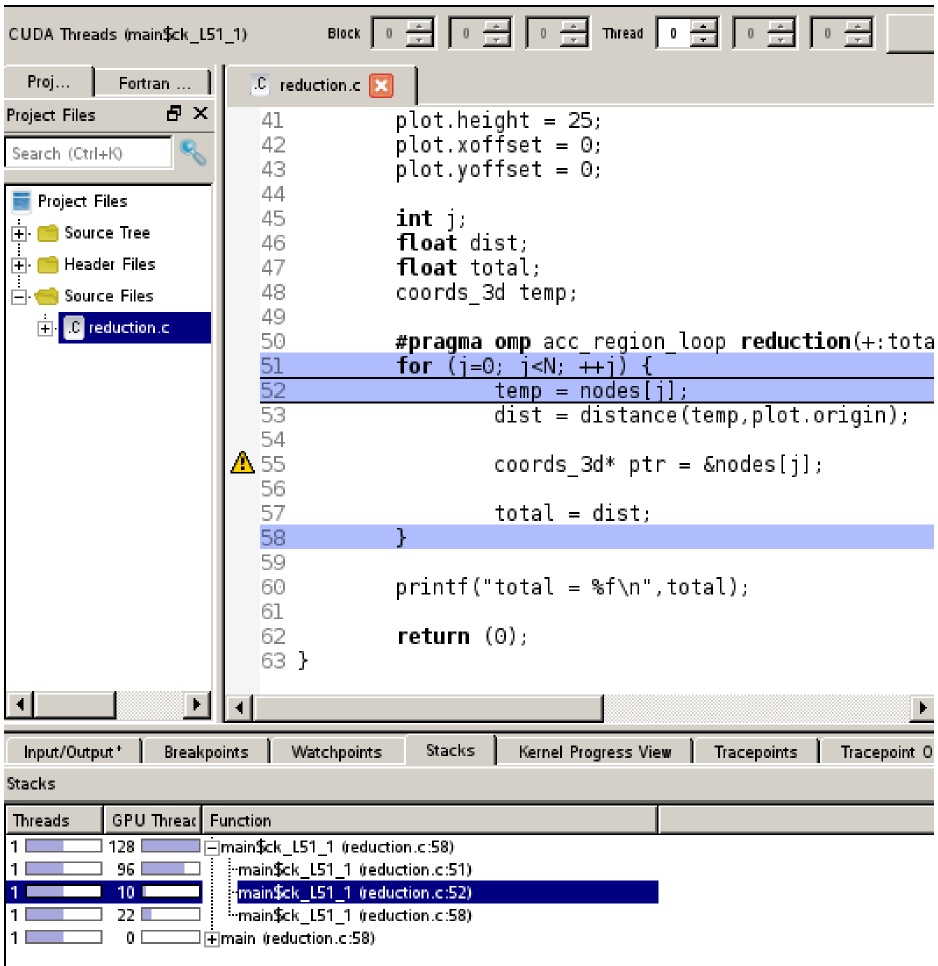
CRAY

THE SUPERCOMPUTER COMPANY

allinea

www.allinea.com

OpenACC debugging



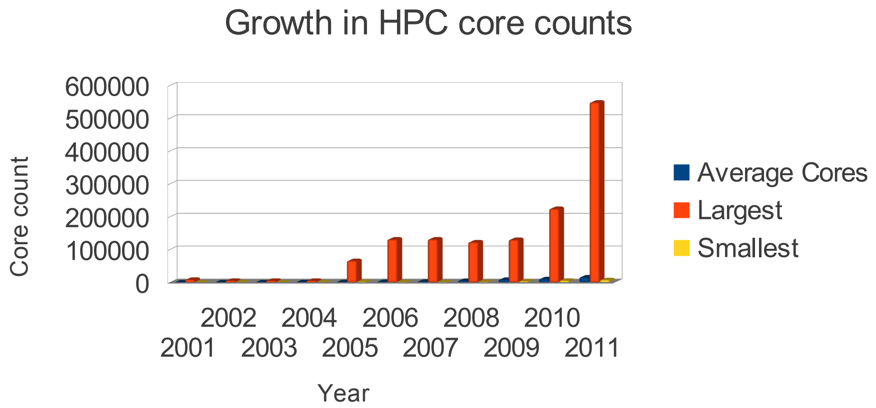
- OpenACC for C and F90
  - Straightforward CUDA compute power
  - Getting code onto the GPU quickly
    - Optimization may still be required
- On device debugging with Allinea DDT
  - Variables – arrays, pointers, full F90 and C support
  - Set breakpoints and step warps and blocks
- Requires Cray compiler for on device debugging
  - Other compilers to follow

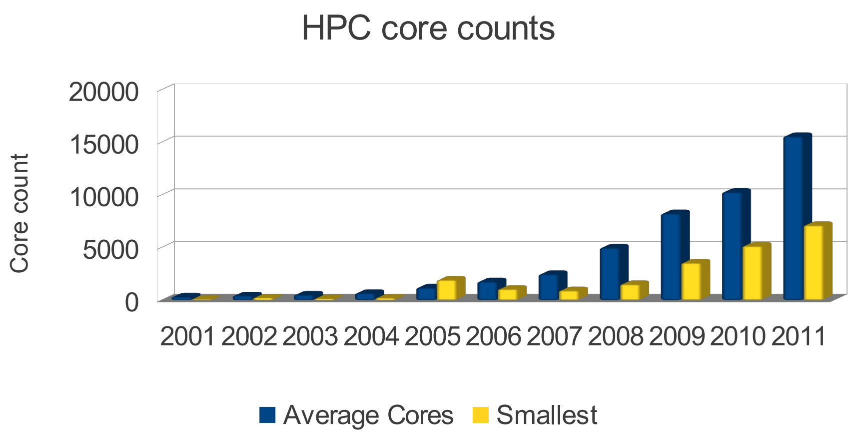
Interlude: CUDA Debugging

Debugging for Petascale

www.allinea.com

Extreme machine sizes





- Progress requires ever more CPU hours
  - Machine sizes are exploding
  - Skewed by largest machines
  - … but a common trend everywhere else
  - Software is changing to exploit the machines

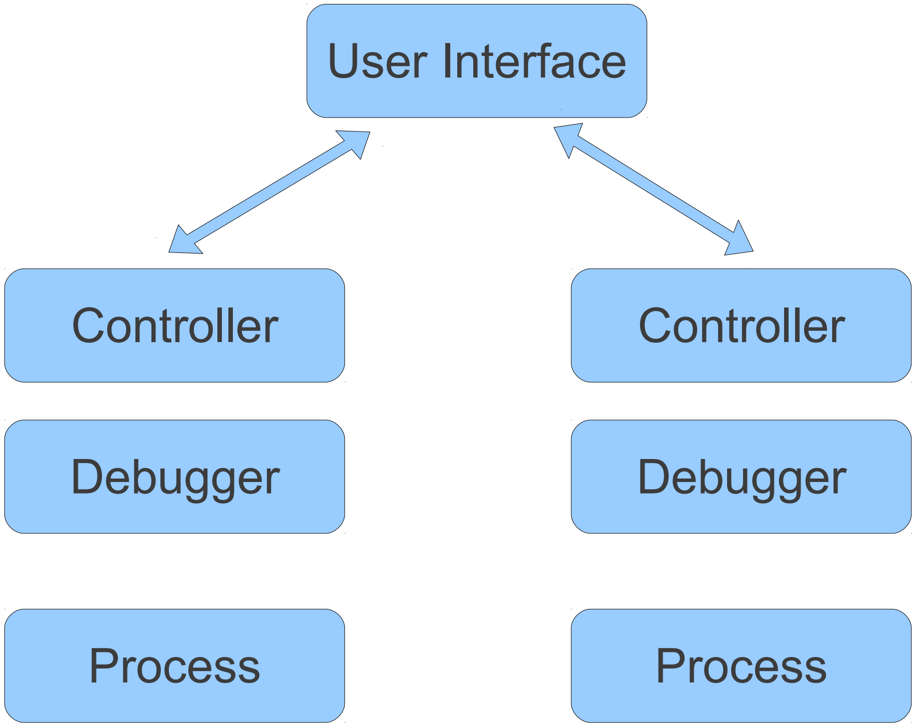
allinea
www.allinea.com

Bug fixing as scale increases

- Can we reproduce at a smaller scale?
  - Attempt to make problem happen on fewer nodes
    - Often requires reduced data set – the large one may not fit
      - Smaller data set may not trigger the problem
    - Does the bug even exist on smaller problems?
      - Didn't you already try the code at small scale?
    - Is it a system issue – eg. an MPI problem?
  - Is probability stacking up against you?
    - Unlikely to spot on smaller runs – without many many runs
    - But near guaranteed to see it on a many-thousand core run
- Debugging at extreme scale is a necessity

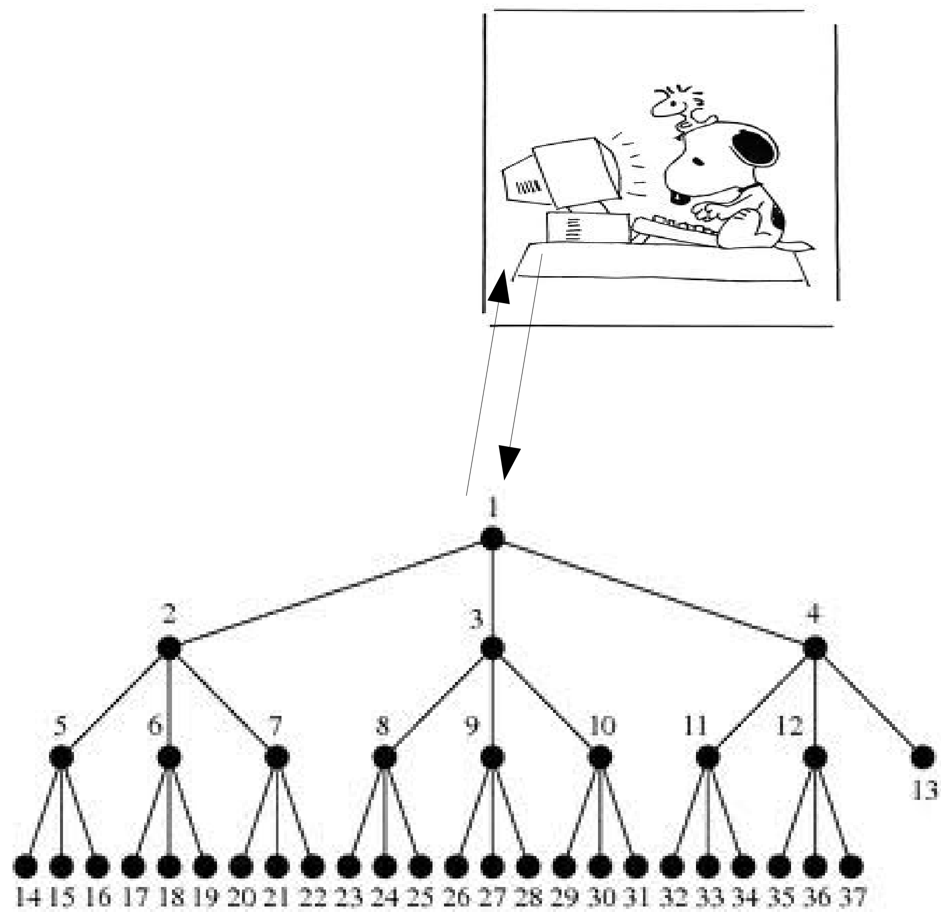
A simple parallel debugger

- A basic parallel debugger
  - Aggregate scalar debuggers and control asynchronously
  - Implement support for many platforms and MPI implementations
  - Develop user interface: simplify control and state display
- Initial architecture
  - Scalar debuggers connect to user interface
  - Eventual scalability bottlenecks
    - Operating system limitations: file handles, threads, processes
    - I/O limitations, memory and computation limitations
  - Machines still getting bigger...

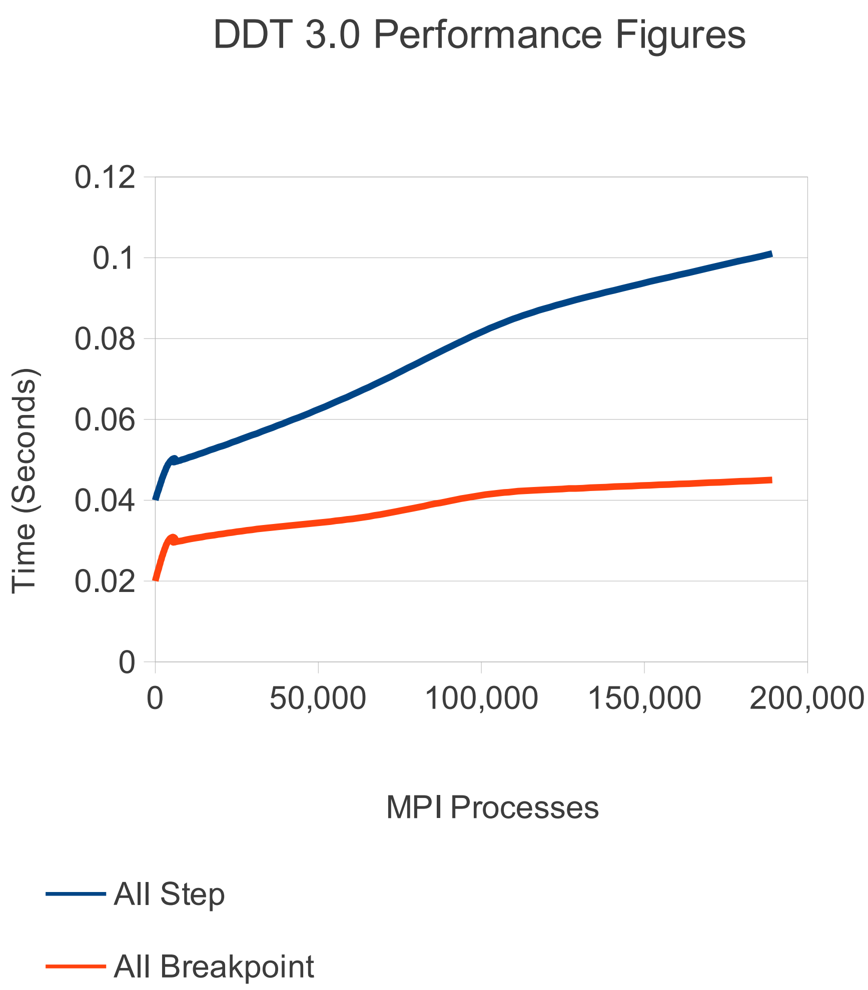


How to make a Petascale debugger

- A control tree is the solution
  - Ability to send bulk commands and merge responses
    - 100,000 processes in a depth 3 tree
  - Compact data type to represent sets of processes
    - eg. For message envelopes
    - An ordered tree of intervals?
    - Or a bitmap?
  - Develop aggregations
    - Merge operations are key
    - Not everything can merge losslessly
    - Maintain the essence of the information
      - eg. min, max, distribution



For Petascale and beyond

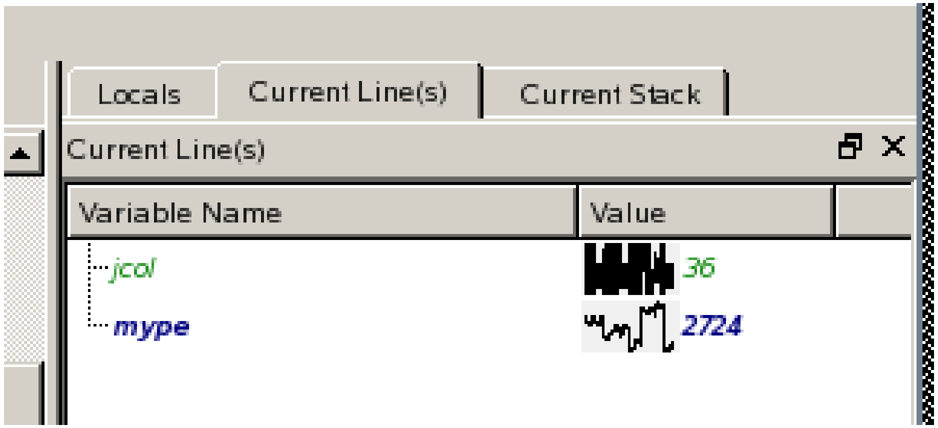


- Scale doesn't have to be hard
  - 100,000 cores should be as easy as 100 cores
  - The user interface is vital to success
- Scale doesn't have to be slow
  - High performance debugging - even at 200,000 cores
  - Step all and display stacks: 0.1 seconds
  - Logarithmic performance
- Stable and in production use
  - Routinely used by end users at over 100,000 cores

www.allinea.com

Key features at scale

- Top 5 features at scale
  - Parallel stack view
    - Ideal for divergence or deadlock
  - Automated data comparison: sparklines
    - Rogue data is easily seen
  - Parallel array searching
    - Data is too large to examine manually
  - Process control with step, play, and breakpoints
    - Still essential
  - Offline debugging
    - Access to machine may be hard – try offline debugging instead



www.allinea.com

Interlude: Petascale demonstration