allinea

Leaders in parallel software development tools

Debugging Petascale HPC Applications

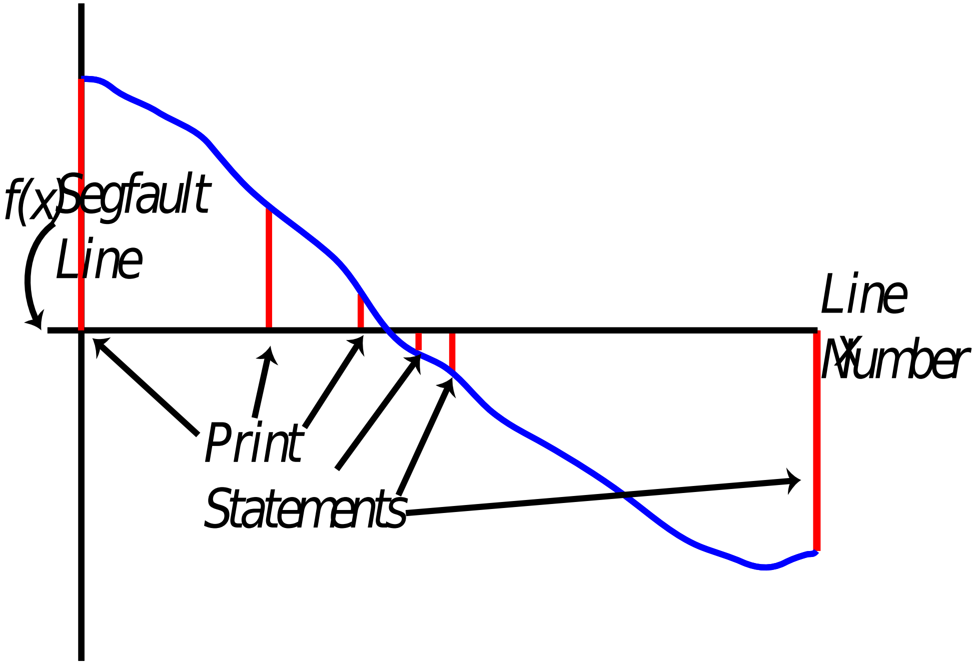
Blue Waters User Workshop 2013

Objectives

- Raise awareness
  - Debugging in general
  - Allinea DDT in particular
- Enhance dexterity
  - Allinea DDT in particular

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



Challenges for Developers

Heterogeneity

Scale

Legacy

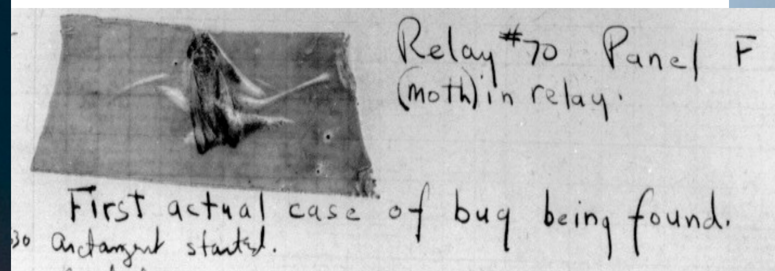
Diversity

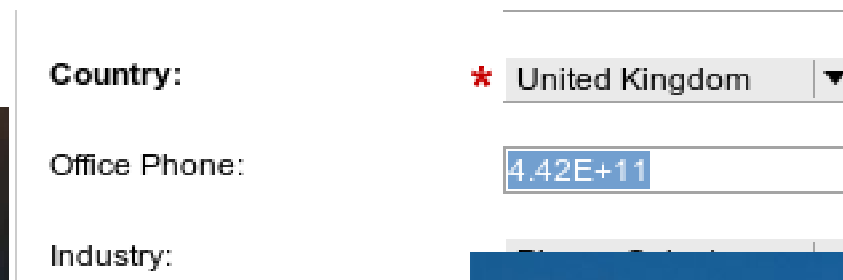
allinea
www.allinea.com

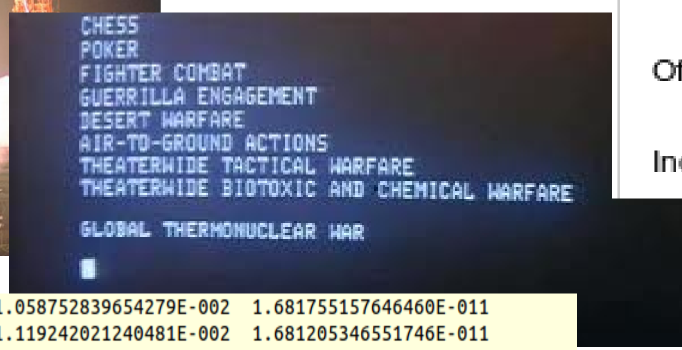
"Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it."

Brian Kernighan

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)
```

www.allinea.com

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

Zeller A., “Why Programs Fail”, 2nd Edition, 2009

TRAFFIC

- Debugging
  - Transforming a broken program into 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*

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

'I' is for Isolate

- Can the issue be isolated?
  - Reduce the process count, data size or some other factor (eg. Time)
  - Simplify the problem?
- Simplifying is not always an option
  - 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 – or is it too unlikely to occur?
- Are there quick ways to just “debug”?

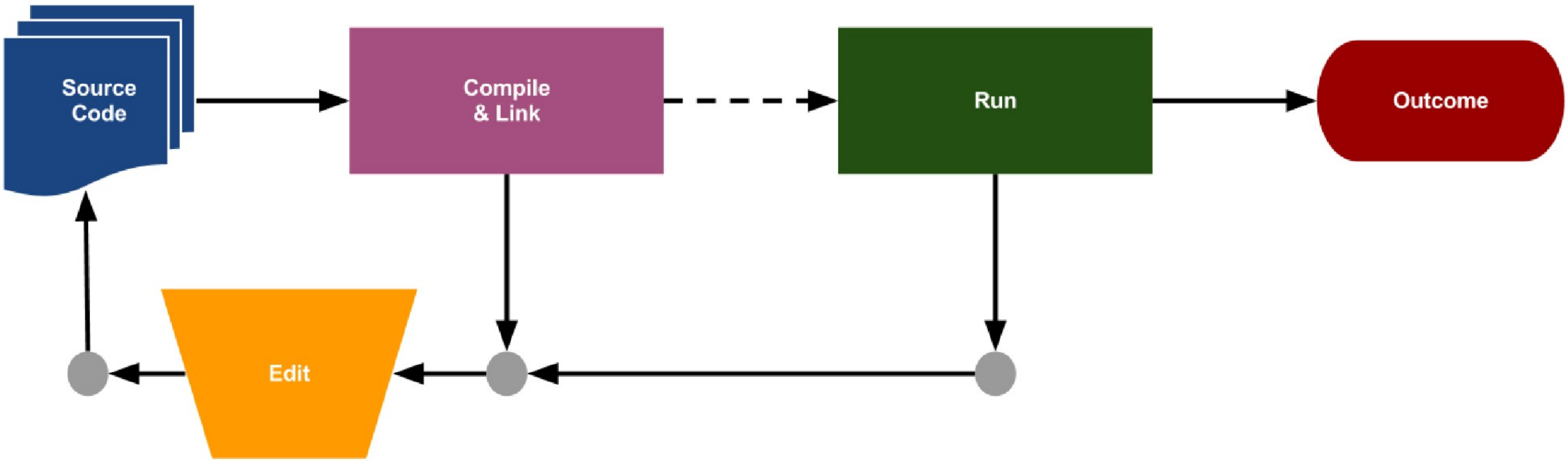
What are Debuggers?

- 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

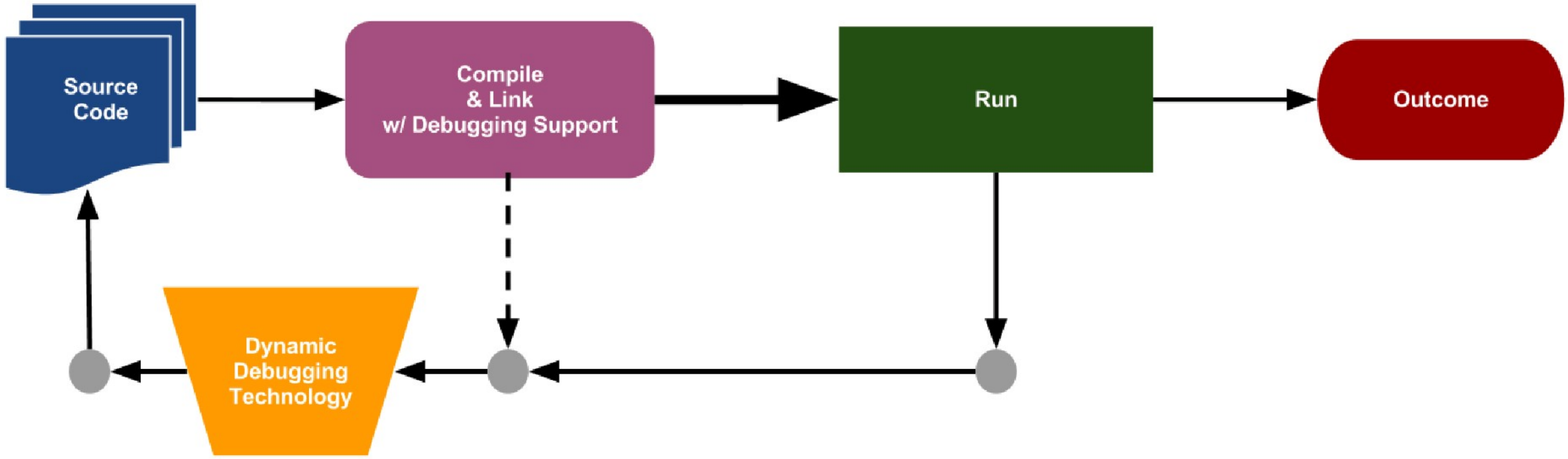
- Multiple methods of operation/implementation
  - Interpreted interactive environments – Ruby, Perl, etc.
    - Everything is under control of the implementation – easy access to the state of the system
    - Relatively easy extension to any interpreter
  - Virtual/managed environments – eg. Java
    - 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

Development Process ... Simplified



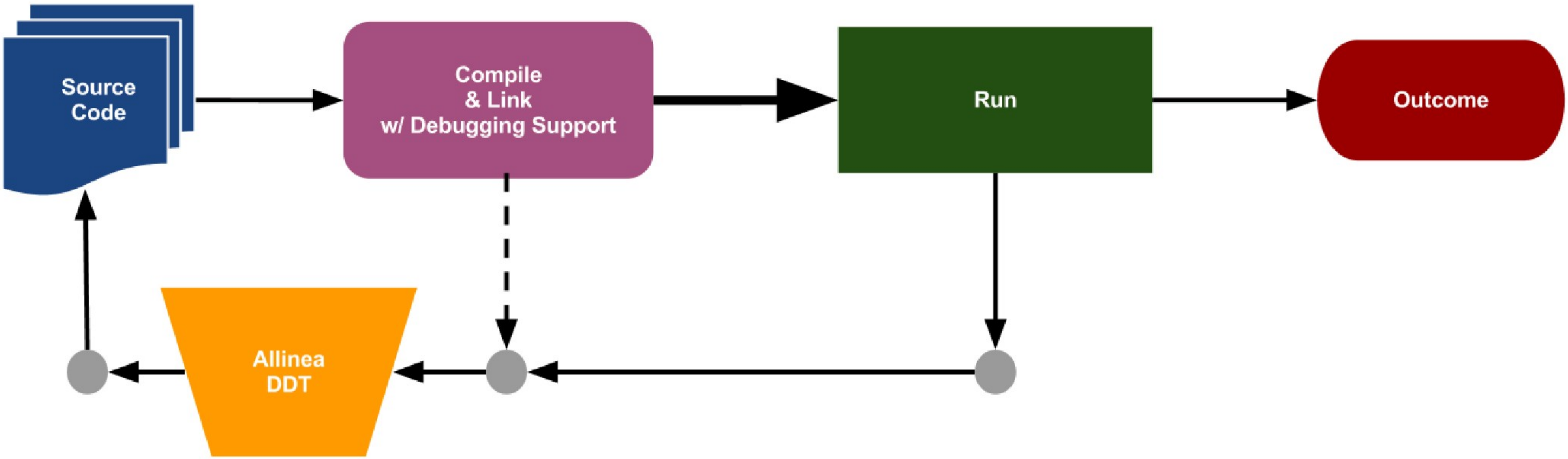
www.allinea.com

Development Process + Debugging



allinea
www.allinea.com

Dev Process + Allinea DDT



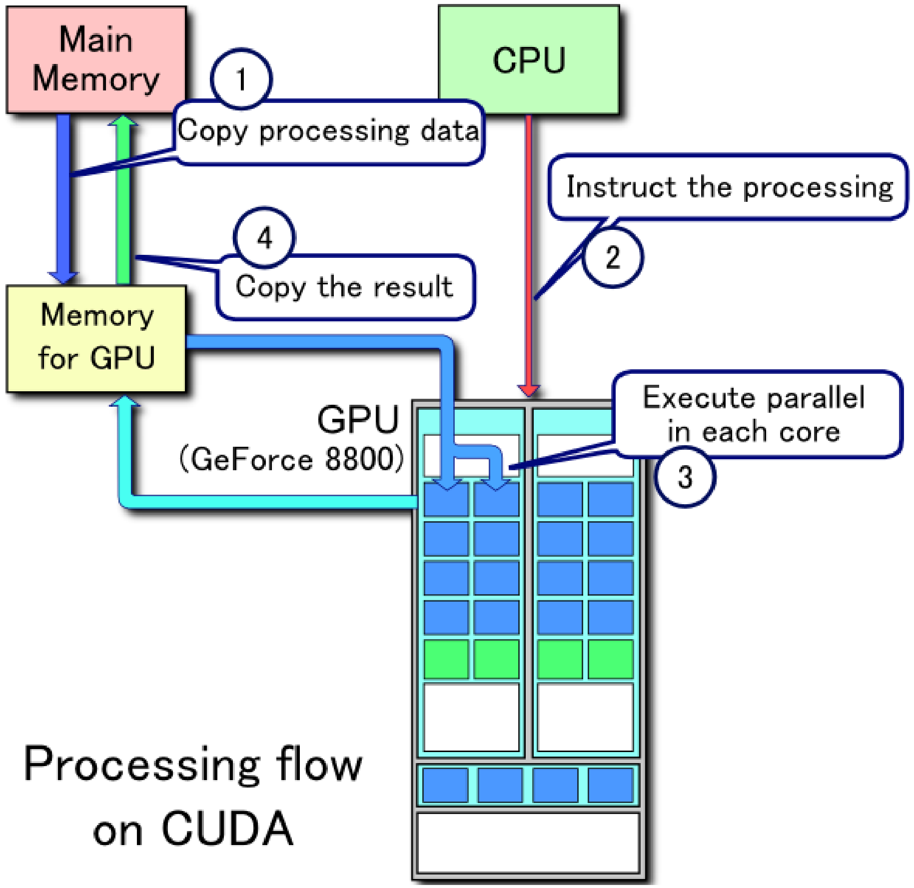
allinea
www.allinea.com

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 ...



The Allinea Environment

allinea MAP

allinea DDT

allinea
www.allinea.com

The Allinea Environment: Benefits

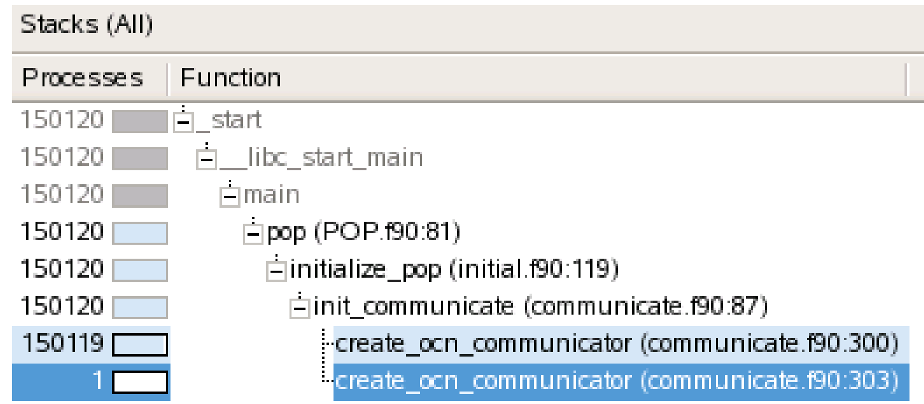
- **At last:** a modern **integrated** environment for the HPC developer
- Supporting the lifecycle of application development and improvement
  - Productively debug code
  - Enhance application performance
- Designed for productivity
  - Consistent integrated easy to use tools
  - Enables effective HPC development
- Improve system usage
  - Fewer failed jobs
  - Higher application performance

allinea
www.allinea.com

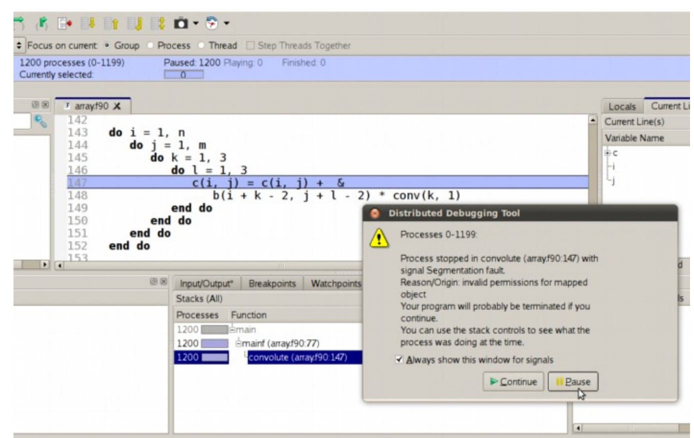
Fixing the everyday crash

- 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

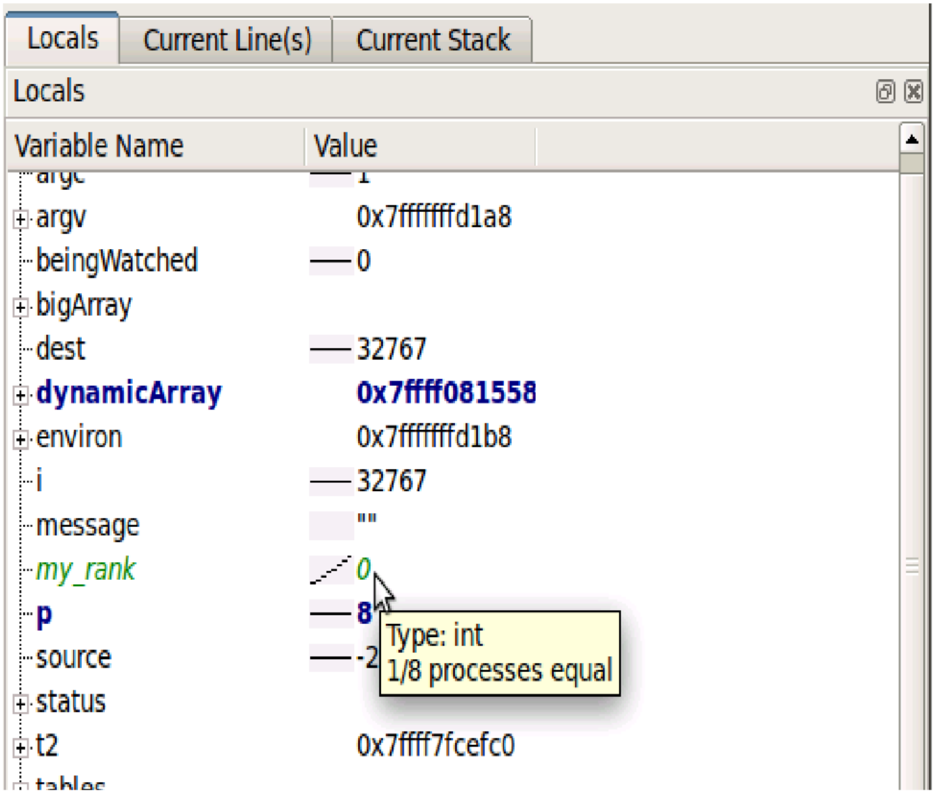


- **Why** did it happen?
  - Some faults evident instantly
  - For others look deeper – at variables

allinea
www.allinea.com

Simplifying the data deluge

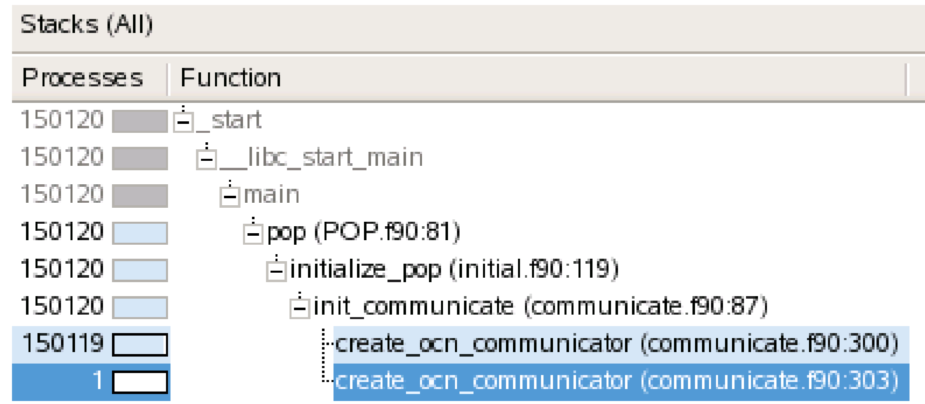
- Allinea DDT compares data automatically
  - Too many variables to trawl manually!
- Smart highlighting
  - Subtle hints for differences and changes
  - With sparklines!
- More detailed analysis
  - Full cross process comparison
  - Historical values via tracepoints

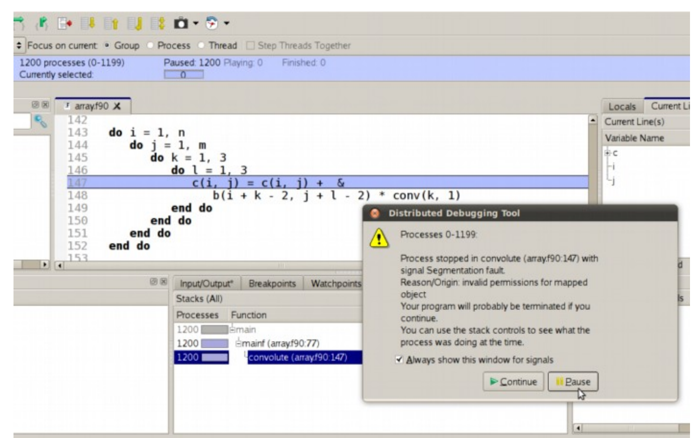


allinea
www.allinea.com

Allinea DDT: Proved to the extreme

- Scalability by design
  - User interface that scales
  - High performance tree architecture
- Proven performance at Petascale
  - Measured in milliseconds
  - **Routine use** at 100,000+ cores
- 300,000+ cores
  - Easy to use
  - Scalable GUI

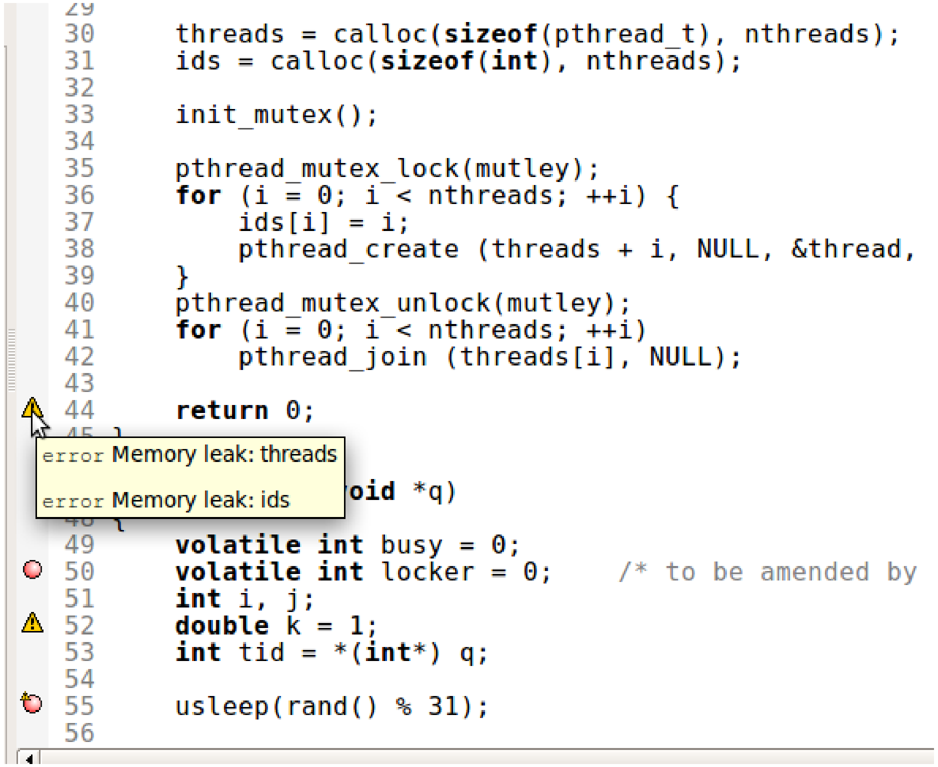




allinea
www.allinea.com

Allinea DDT: More than debugger

- Integrated automated detection of bugs
  - Static analysis
  - Memory leaks and errors
- Open plugin architecture
  - MPI checking tools
- Offline mode - debug in batch mode



allinea
www.allinea.com

Demos

- Crashes
- Memory errors and leaks
- Deadlocks
  - Threads
  - MPI
- Breakpoints and watchpoints
- Offline debugging
- Incorrect results
- GPU support

http://www.allinea.com/downloads/ddt_training.tar.gz

OpenMP Debugging Considerations

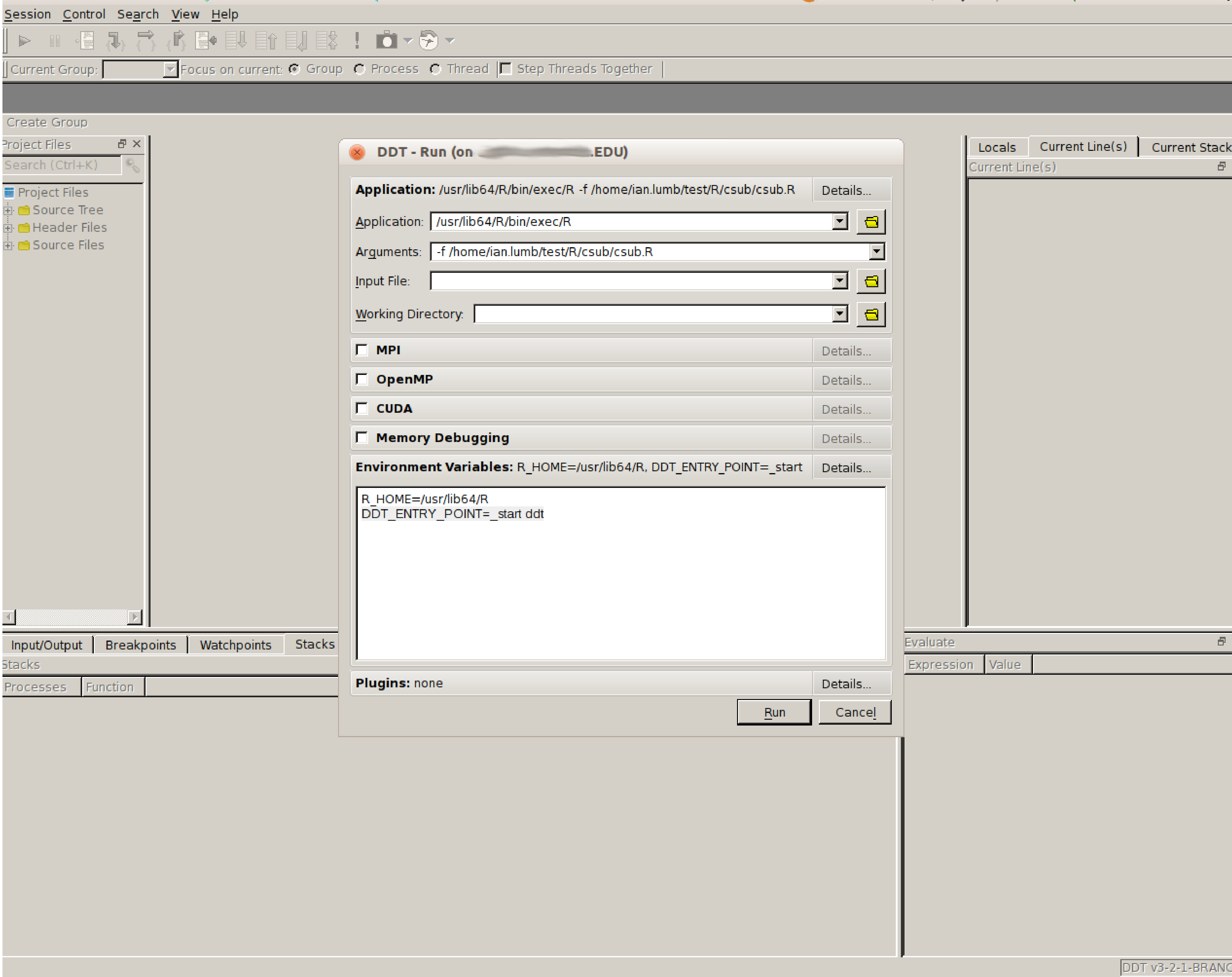
- Threads only created when parallel region reached
  - Applies to some OpenMP libraries
- Can't step into a parallel region
  - Synchronize threads in parallel region then
    - Step threads together
    - Run to a specific line
- Can't step out of a parallel region
    - Step threads together inside parallel regions
    - Run to specific line to exit parallel region

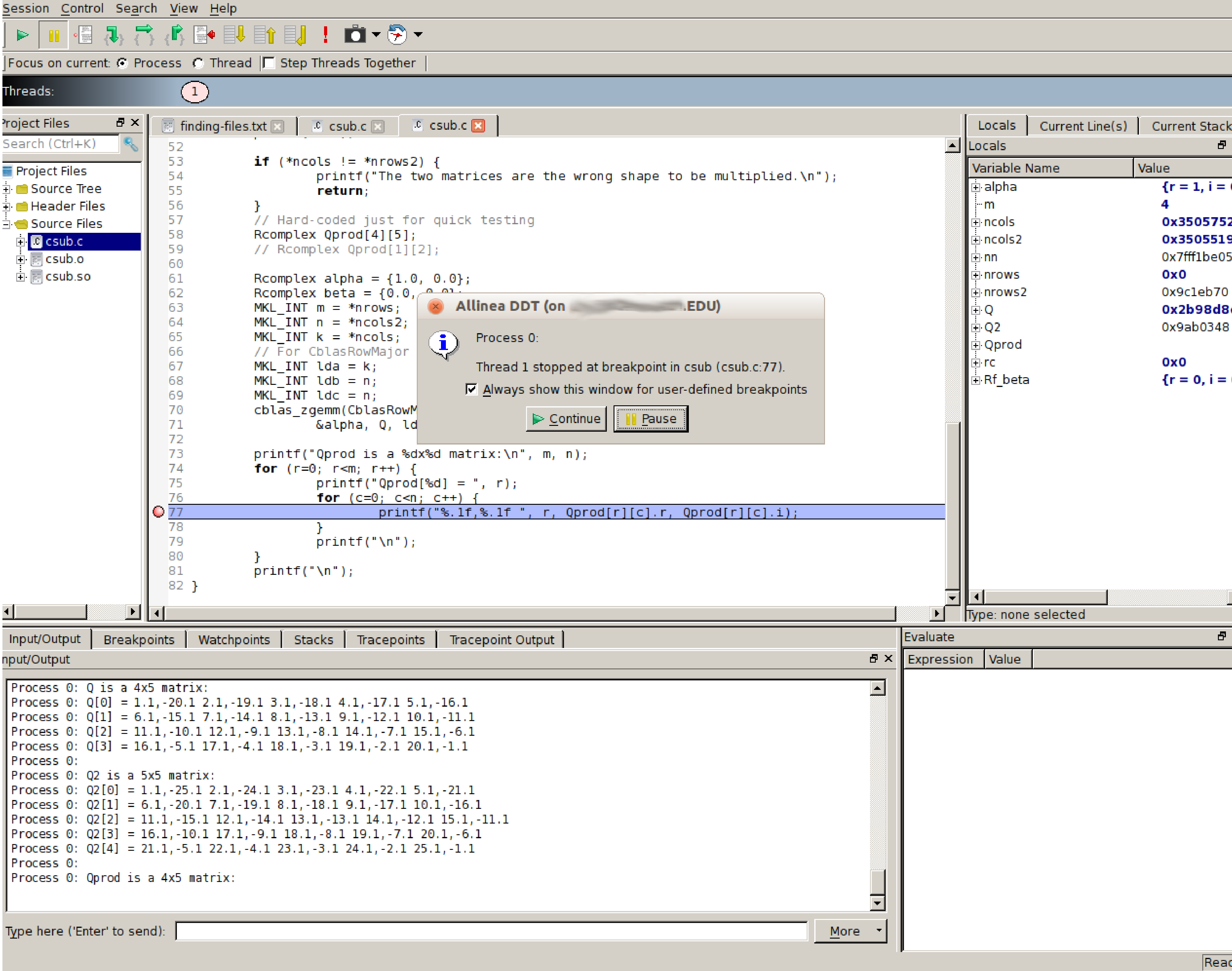
OpenMP Debugging Considerations

- Outside parallel regions
  - Disable “Step Threads Together”
- Control threads individually
  - Use “Focus on current: Thread” feature
- Shared OpenMP variables may appear twice in Locals window
  - Side effect of introducing parallelism

OpenMP Debugging Considerations

- Parallel regions displayed as new functions in stack views?
  - Implemented as automatically-generated “outline” functions
- Stepping often behaves unexpectedly inside parallel regions
- Some compilers optimize parallel loops
  - Ignore options specified on the command line


Session Control Search View Help
Current Group:
Focus on current: Group Process Thread
Step Threads Together
Create Group
Project Files
Search (Ctrl+K)
Project Files
Source Tree
Header Files
Source Files
DDT - Run (on .EDU)
Application: /usr/lib64/R/bin/exec/R -f /home/ian.lumb/test/R/csub/csub.R
Details...
Application: /usr/lib64/R/bin/exec/R
Arguments: -f /home/ian.lumb/test/R/csub/csub.R
Input File:
Working Directory:
MPI
OpenMP
CUDA
Memory Debugging
Environment Variables: R_HOME=/usr/lib64/R, DDT_ENTRY_POINT=_start
R_HOME=/usr/lib64/R
DDT_ENTRY_POINT=_start ddt
Plugins: none
Run
Cancel
Locals
Current Line(s)
Current Stack
Input/Output
Breakpoints
Watchpoints
Stacks
Processes
Function
Evaluate
Expression
Value
DDT v3-2-1-BRANCH


Session Control Search View Help
Focus on current: Process Thread Step Threads Together
Threads: 1
Project Files
Search (Ctrl+K)
Project Files
Source Tree
Header Files
Source Files
csub.c
csub.o
csub.so
finding-files.txt
csub.c
csub.c
if (*ncols != *nrows2) {
printf("The two matrices are the wrong shape to be multiplied.\n");
return;
}
// Hard-coded just for quick testing
Rcomplex Qprod[4][5];
// Rcomplex Qprod[1][2];
Rcomplex alpha = {1.0, 0.0};
Rcomplex beta = {0.0,
MKL_INT m = *nrows;
MKL_INT n = *ncols2;
MKL_INT k = *ncols;
// For CblasRowMajor
MKL_INT lda = k;
MKL_INT ldb = n;
MKL_INT ldc = n;
cblas_zgemm(CblasRowM
&alpha, Q, ld
printf("Qprod is a %dx%d matrix:\n", m, n);
for (r=0; r<m; r++) {
printf("Qprod[%d] = ", r);
for (c=0; c<n; c++) {
printf("%.1f,%.1f ", r, Qprod[r][c].r, Qprod[r][c].i);
}
printf("\n");
}
printf("\n");
}
Allinea DDT (on .EDU)
Process 0:
Thread 1 stopped at breakpoint in csub (csub.c:77).
Always show this window for user-defined breakpoints
Continue
Pause
Locals
Current Line(s)
Current Stack
Variable Name
Value
alpha {r = 1, i = 0}
m 4
ncols 0x35057529
ncols2 0x35055199
nn 0x7fff1be05c
nrows 0x0
nrows2 0x9c1eb70
Q 0x2b98d8c4
Q2 0x9ab0348
Qprod
rc 0x0
Rf_beta {r = 0, i = 0}
Type: none selected
Input/Output
Breakpoints
Watchpoints
Stacks
Tracepoints
Tracepoint Output
Process 0: Q is a 4x5 matrix:
Process 0: Q[0] = 1.1,-20.1 2.1,-19.1 3.1,-18.1 4.1,-17.1 5.1,-16.1
Process 0: Q[1] = 6.1,-15.1 7.1,-14.1 8.1,-13.1 9.1,-12.1 10.1,-11.1
Process 0: Q[2] = 11.1,-10.1 12.1,-9.1 13.1,-8.1 14.1,-7.1 15.1,-6.1
Process 0: Q[3] = 16.1,-5.1 17.1,-4.1 18.1,-3.1 19.1,-2.1 20.1,-1.1
Process 0:
Process 0: Q2 is a 5x5 matrix:
Process 0: Q2[0] = 1.1,-25.1 2.1,-24.1 3.1,-23.1 4.1,-22.1 5.1,-21.1
Process 0: Q2[1] = 6.1,-20.1 7.1,-19.1 8.1,-18.1 9.1,-17.1 10.1,-16.1
Process 0: Q2[2] = 11.1,-15.1 12.1,-14.1 13.1,-13.1 14.1,-12.1 15.1,-11.1
Process 0: Q2[3] = 16.1,-10.1 17.1,-9.1 18.1,-8.1 19.1,-7.1 20.1,-6.1
Process 0: Q2[4] = 21.1,-5.1 22.1,-4.1 23.1,-3.1 24.1,-2.1 25.1,-1.1
Process 0:
Process 0: Qprod is a 4x5 matrix:
Type here ('Enter' to send):
More
Evaluate
Expression
Value
Ready

The old quick way to debug...

- Logging – printf and write
  - If you have good intuition into the problem
    - Edit code, insert print, recompile and re-run
    - **Slow and iterative**
  - Logs grow too quickly
    - Hard establish real order of output of multiple processes
    - **Unscalable**

No longer a very effective way to solve bugs

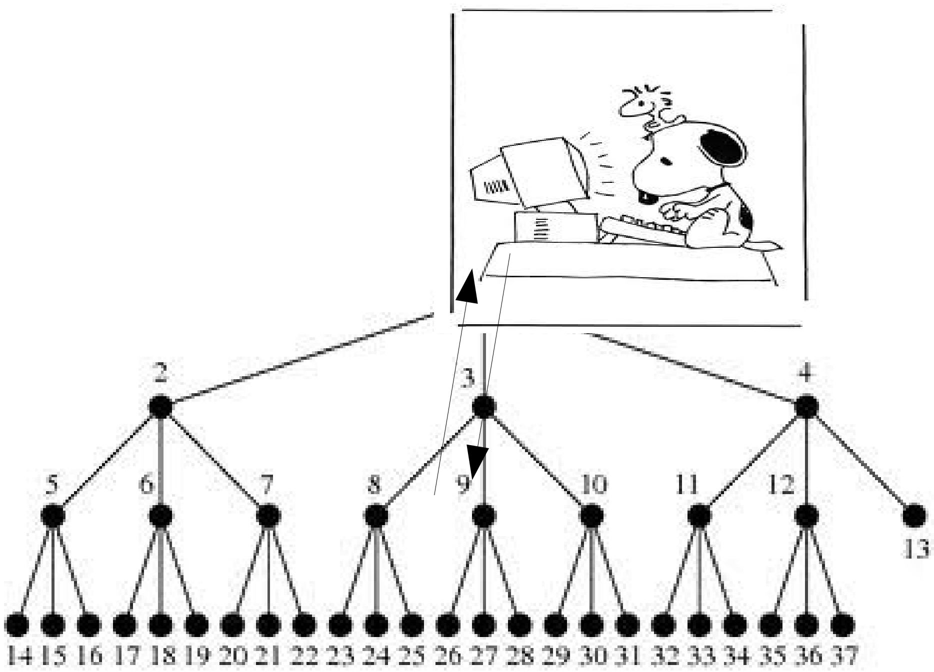
So, can we use a real debugger?

Why debug at scale?

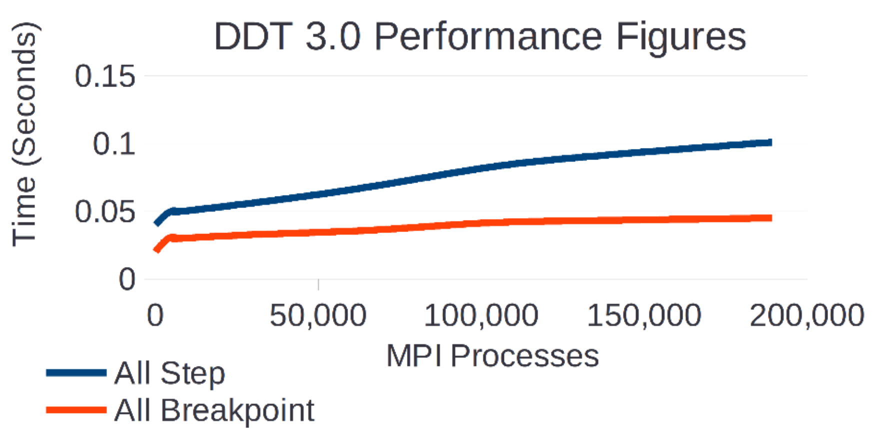
- Increasing job sizes leads to unanticipated errors
  - Regular bugs
    - Logic issues and control flow
    - Data issues from larger data sets – eg. garbage in..., overflow
  - Increasing probability of independent random error
    - Memory errors/exhaustion – “random” bugs!
    - System problems – MPI and operating system
  - Coded boundaries
    - Algorithmic (performance) or hard-wired limits (“magic numbers”)
  - Unknown unknowns
- Machine time is too expensive to ignore failures!

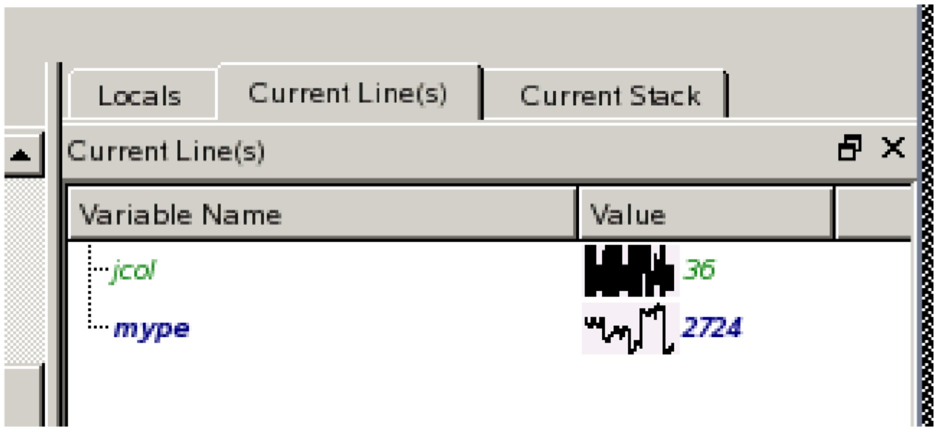
How to Make a Petascale Debugger

- A control tree gives scalability
- 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/should merge losslessly
  - Maintain the essence of the information: eg. min, max, distribution



Real Petascale Debugging

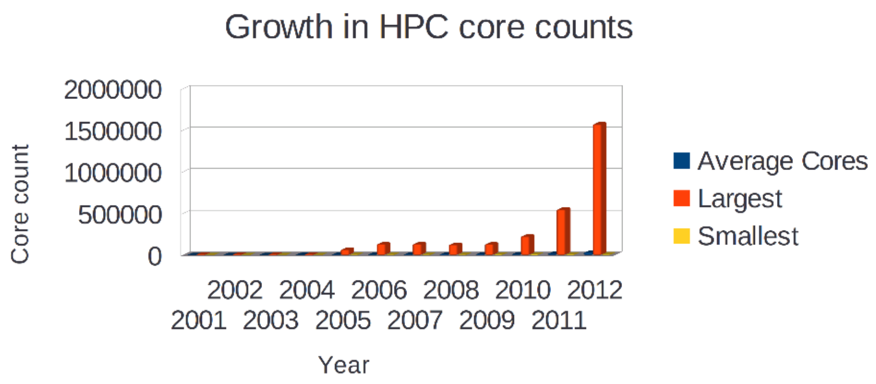


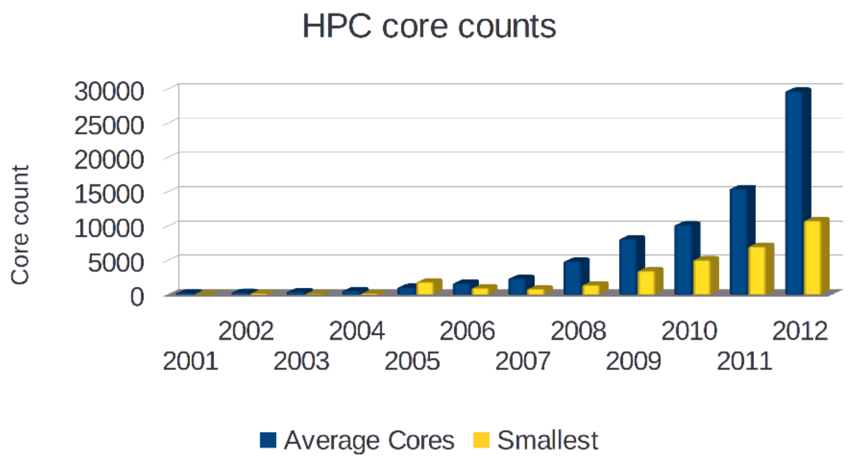


- Logarithmic performance with a tree network
- Still fast at 220,000 cores: step all and display stacks in 0.1 seconds
- Partnership with users
  - DoE Oak Ridge National Laboratories
  - LLNL, ANL, CEA and others
- Usability is a “Big Thing”: scalable interface and features

allinea
www.allinea.com

Scale is already here ...





- Machine sizes are exploding
  - Skewed by largest machines

    … but a common trend
  - Largest (Jun 2012) 1.5M cores
- Petaflop owners club is growing
  - 23 members!
- Petaflop users club?
  - How will you get your application there?

allinea
www.allinea.com

Extreme-Scale Endorsements

“My group routinely debugs parallel code at over 100,000 processes using Allinea DDT. No other debugger can even come close to its performance, so obviously it’s a hit with users.”

– Dr Richard Graham, Oak Ridge National Laboratory

“Allinea's experience and tools will make a big impact in the speed at which scientists can complete their research. We are looking to Allinea to help teams become more productive by more quickly moving codes to the new technologies, and improve the performance of their codes at the full scale of the entire system.”

– Dr Bill Kramer, Deputy Project Director of Blue Waters

Extreme-Scale Endorsements (2)

"This tool has already proven its value in the migration of our early science applications onto Mira," said Kalyan Kumaran, who manages ALCF's applications performance engineering team. "These projects cover the range of scientific fields, numerical methods, programming models and computational approaches expected to run on Mira, so accurate debugging is critical."

www.allinea.com

Allinea Strengths

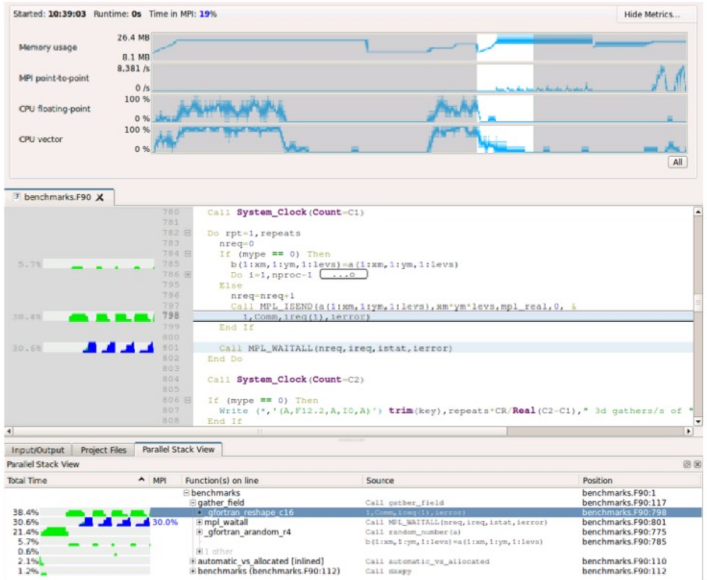
- Focus
  - Tools for HPC developers
- Design
  - User experience
  - Architecture
    - Integrated
    - Interoperable
    - Scalable

About Allinea

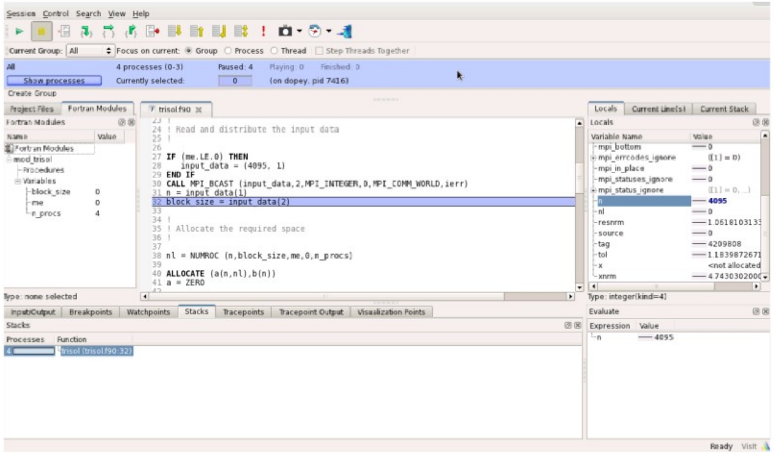
- HPC development tools company
  - Flagship product Allinea DDT
    - Now the leading debugger in parallel computing
    - The scalable debugger
      - Record holder for debugging software on largest machines
      - Production use at extreme scale … and desktop
    - Wide customer base
      - Blue-chip engineering, government and academic research
      - Strong collaborative relationships with customers and partners
  - Announced product Allinea MAP
    - The profiler you'll actually want to use!

A Unified Environment for HPC

Shared Graphical Interface



Shared Configuration Files



Shared Scalable Architecture

Intelligent data consolidation

allinea

www.allinea.com

Allinea DDT - Debugging++

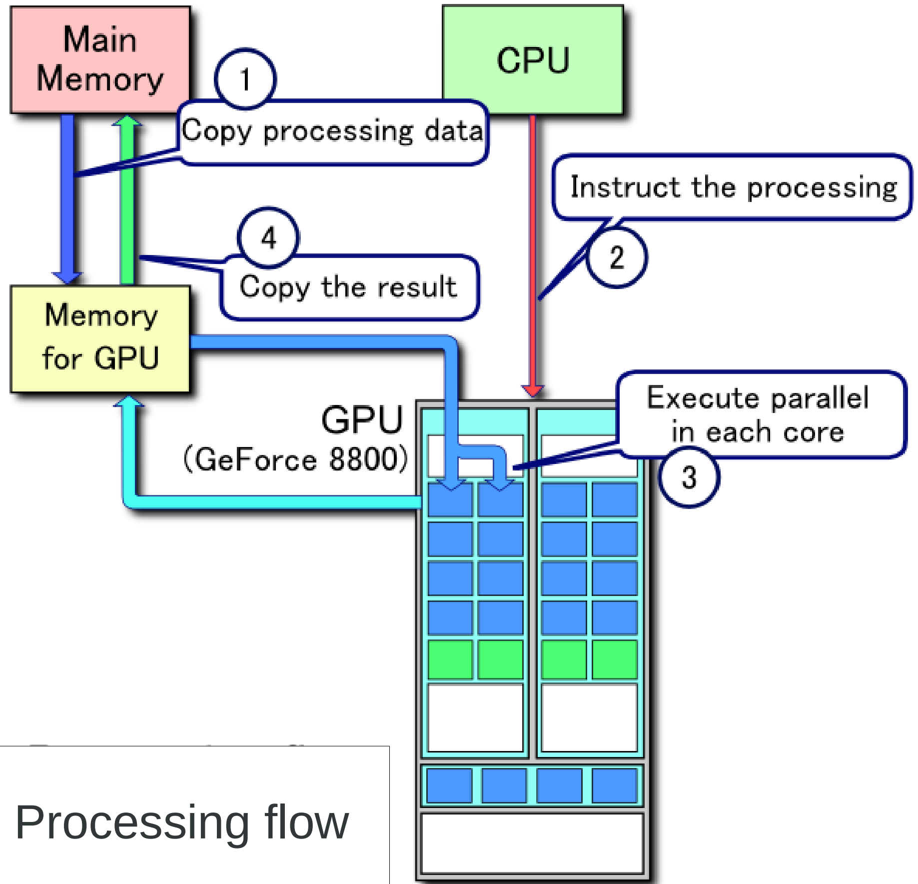
- Productively **debug** your parallel code
- Completely **understand** your parallel code
  - Interact with data, algorithms, codes, programs and applications in real time
- **Develop** parallel your code from scratch
- **Port** parallel algorithms, codes, programs and applications to X
- **Scale** your algorithms, codes, programs and applications

The First CUDA Bug

- 2007 – introduction of the CUDA programming model
  - Powerful, efficient and C-based
  - Understood and adopted by new groups of experts
  - Existing codes modified to extract SIMD parallelism and introduce CUDA kernels
  - Performance of codes is optimized
    - Overlapping device (GPU) and host (CPU), or
    - Rearranging memory usage inside device (GPU)
- The first CUDA bug is created ...

Embracing GPUs

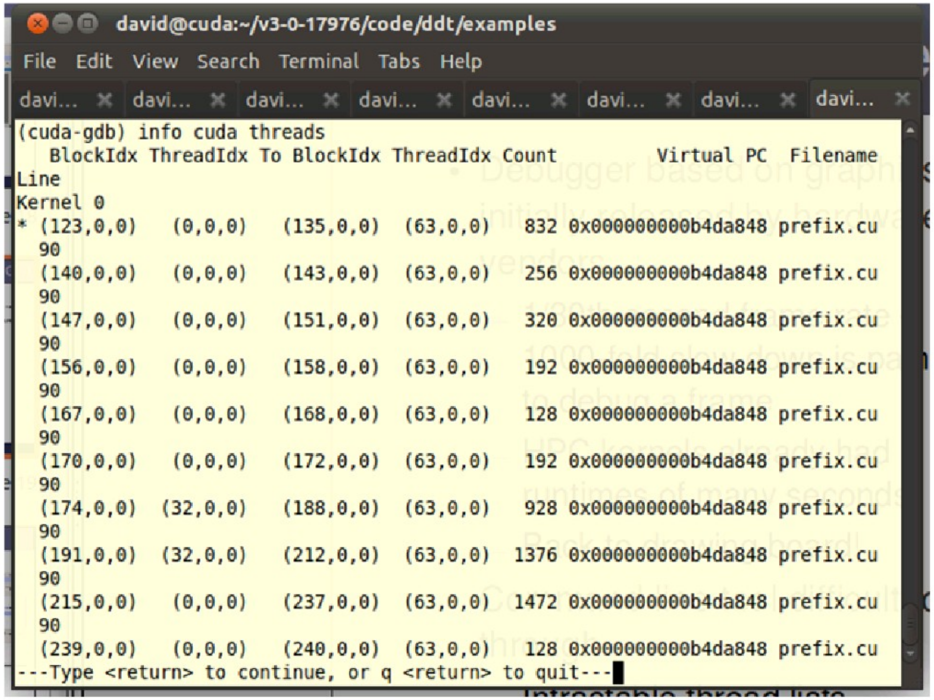
- GPUs – a rival to traditional processors
  - Great price/performance ratios
  - Offerings from AMD and NVIDIA
- New languages, compilers, standards
  - CUDA, OpenACC, OpenCL, ...
- HPC developers need to consider
  - Data transfer
  - Multiple memory levels
  - Grid/block layout and thread scheduling
  - Synchronization
- Bugs are **inevitable**



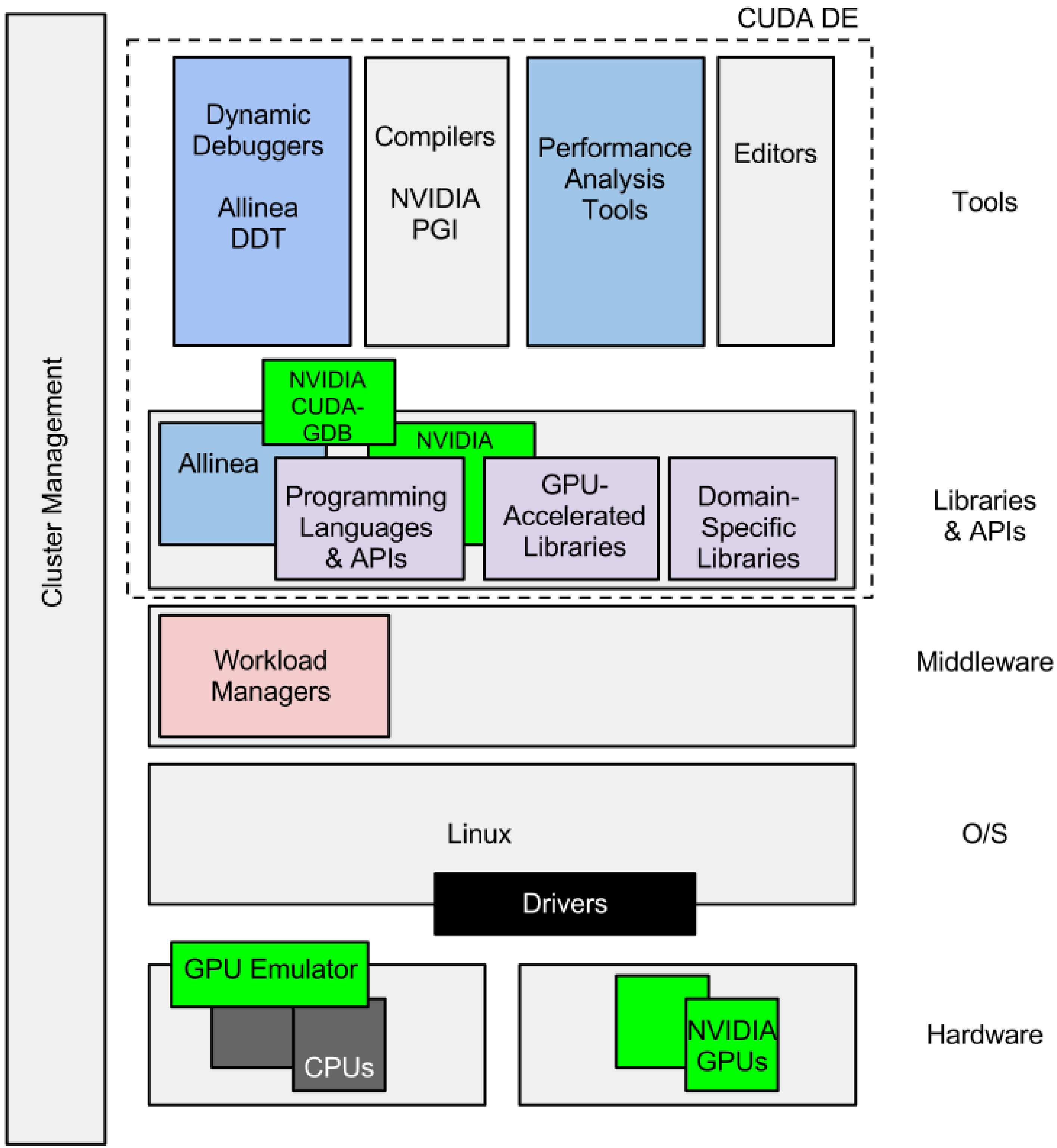
Processing flow

Debugging Parallel CUDA Applications

- Current status
  - Software complexity reflects hardware complexity
    - cuda-gdb
      - Direct use challenging
      - Indirect use via a debugger



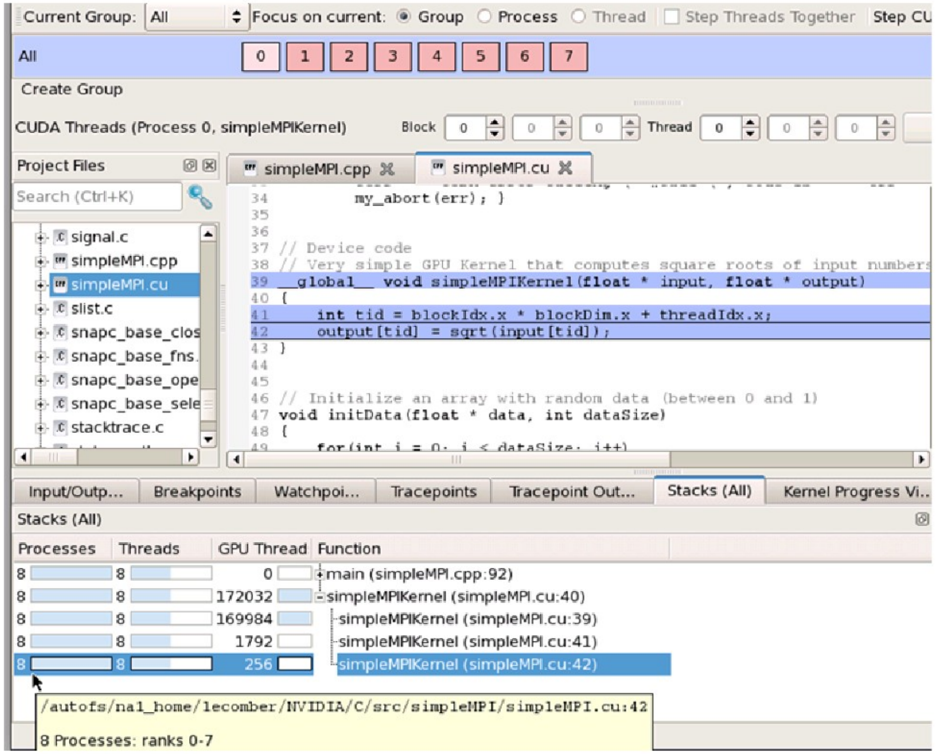
allinea
www.allinea.com



CUDA DE
Cluster Management
Dynamic Debuggers
Allinea DDT
Compilers
NVIDIA PGI
Performance Analysis Tools
Editors
Tools
NVIDIA CUDA-GDB
NVIDIA
Allinea
Programming Languages & APIs
GPU-Accelerated Libraries
Domain-Specific Libraries
Libraries & APIs
Workload Managers
Middleware
Linux
Drivers
O/S
GPU Emulator
CPUs
NVIDIA GPUs
Hardware

Allinea DDT and CUDA

- Supports
  - CUDA toolkits 3.1 -- 3.2 – 4.0 – 4.1 – 4.2 – 5.0
- 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



www.allinea.com

Allinea DDT and CUDA
Core Debugging Capabilities

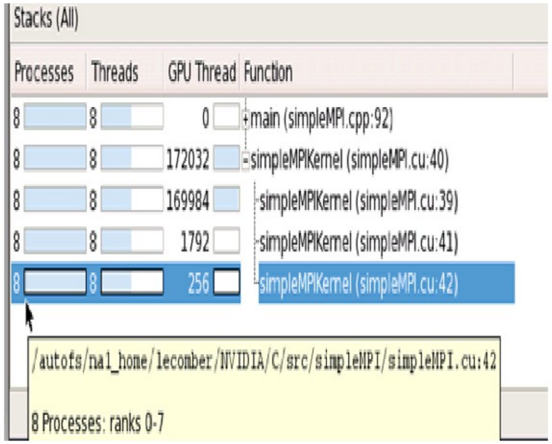
- The first graphical debugger for NVIDIA CUDA
  - Simple and easy to use
  - As easy as debugging ordinary (i.e., non-GPU) code

- 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

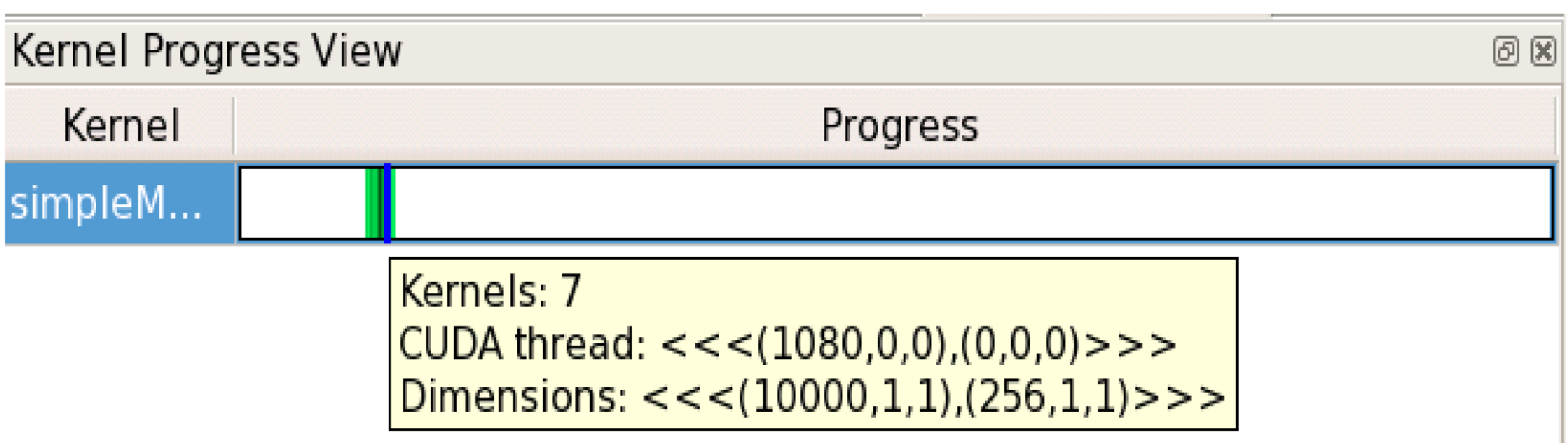


www.allinea.com

Allinea DDT and CUDA Kernel Progress

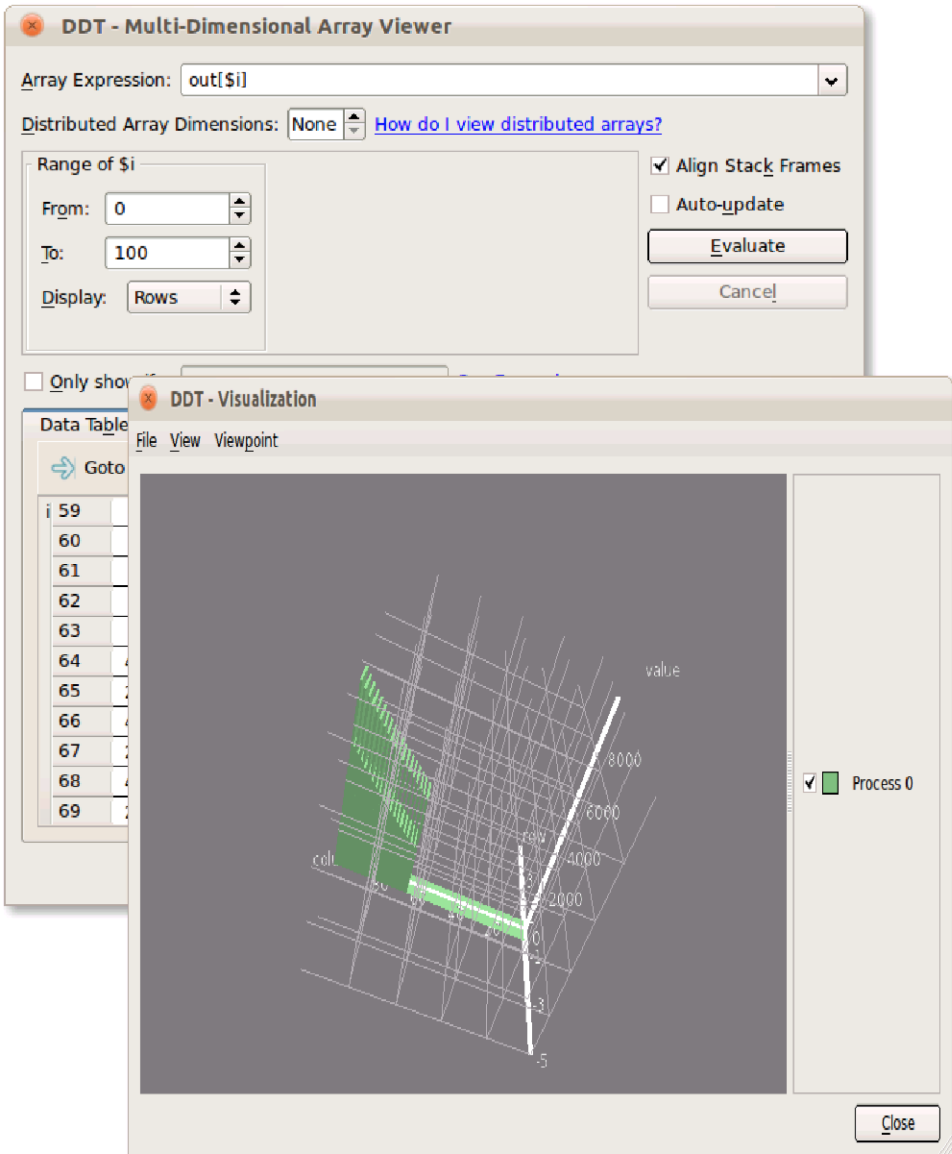
- Has my thread calculated the output yet ? Is it to be scheduled ?
  - Contrast with scalar programming

- Keep an eye on your kernel progress across processes



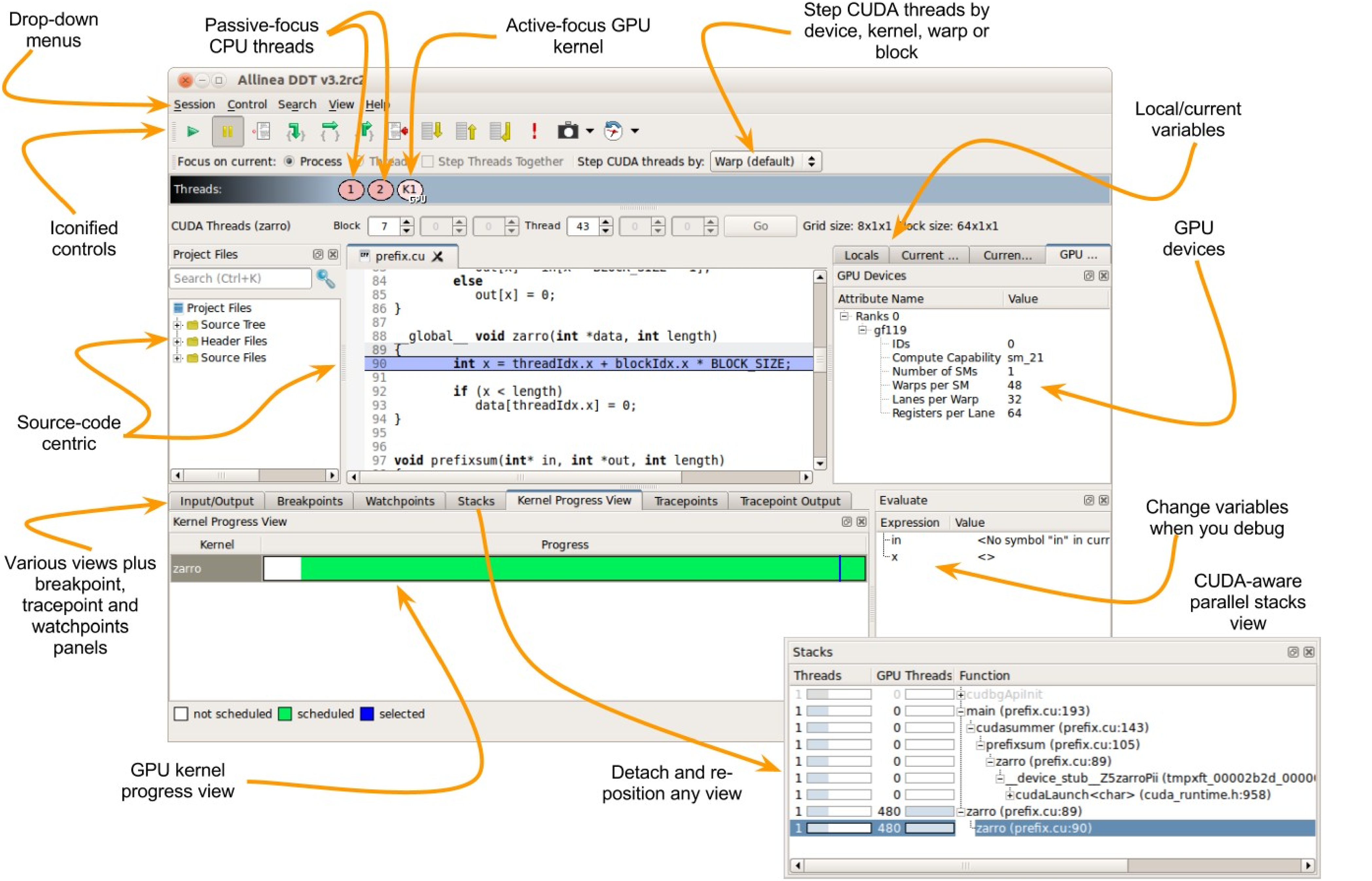
allinea
www.allinea.com

Array Visualization Support



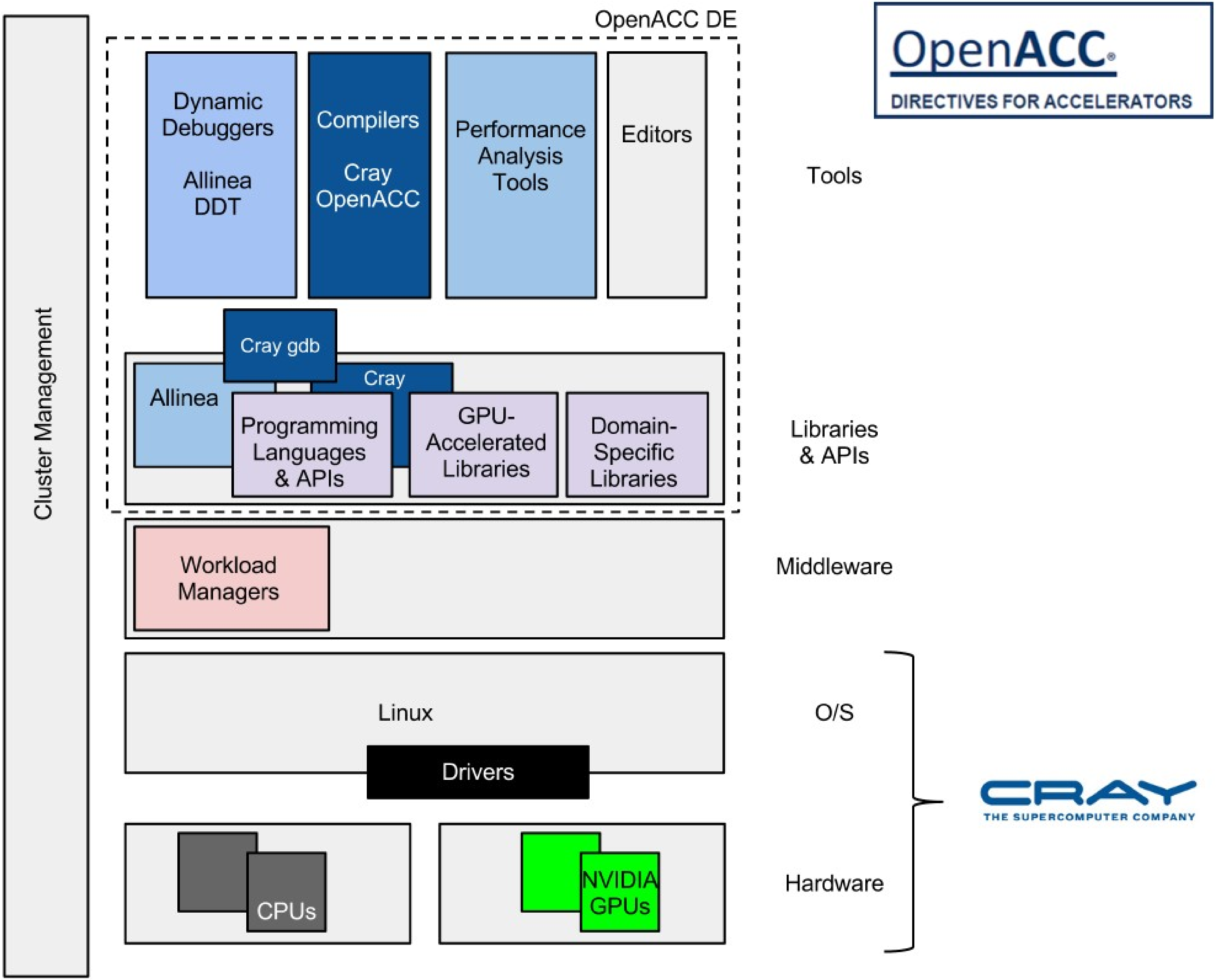
- Browse arrays
  - 1, 2, 3, … dimensions
  - Table view
- Filtering
  - Look for an outlier
- Export
  - Save to a spreadsheet

allinea
www.allinea.com


Drop-down menus
Passive-focus CPU threads
Active-focus GPU kernel
Step CUDA threads by device, kernel, warp or block
Local/current variables
Iconified controls
GPU devices
Source-code centric
Change variables when you debug
CUDA-aware parallel stacks view
Various views plus breakpoint, tracepoint and watchpoints panels
GPU kernel progress view
Detach and re-position any view
Allinea DDT v3.2rc2
Session Control Search View Help
Focus on current: Process Thread Step Threads Together Step CUDA threads by: Warp (default)
Threads: 1 2 K1
CUDA Threads (zarro) Block 7 0 0 Thread 43 0 0 Go Grid size: 8x1x1 Block size: 64x1x1
Project Files
Source Tree
Header Files
Source Files
prefix.cu
else
out[x] = 0;
__global__ void zarro(int *data, int length)
int x = threadIdx.x + blockIdx.x * BLOCK_SIZE;
if (x < length)
data[threadIdx.x] = 0;
void prefixsum(int* in, int *out, int length)
Locals Current ... Curren... GPU ...
GPU Devices
Attribute Name Value
Ranks 0
gf119
IDs 0
Compute Capability sm_21
Number of SMs 1
Warps per SM 48
Lanes per Warp 32
Registers per Lane 64
Input/Output Breakpoints Watchpoints Stacks Kernel Progress View Tracepoints Tracepoint Output
Kernel Progress View
Kernel Progress
zarro
not scheduled scheduled selected
Evaluate
Expression Value
in <No symbol "in" in curr
x <>
Stacks
Threads GPU Threads Function
cudbgApiInit
main (prefix.cu:193)
cudasummer (prefix.cu:143)
prefixsum (prefix.cu:105)
zarro (prefix.cu:89)
__device_stub__Z5zarroPii (tmpxft_00002b2d_0000
cudaLaunch<char> (cuda_runtime.h:958)
480 zarro (prefix.cu:89)
480 zarro (prefix.cu:90)

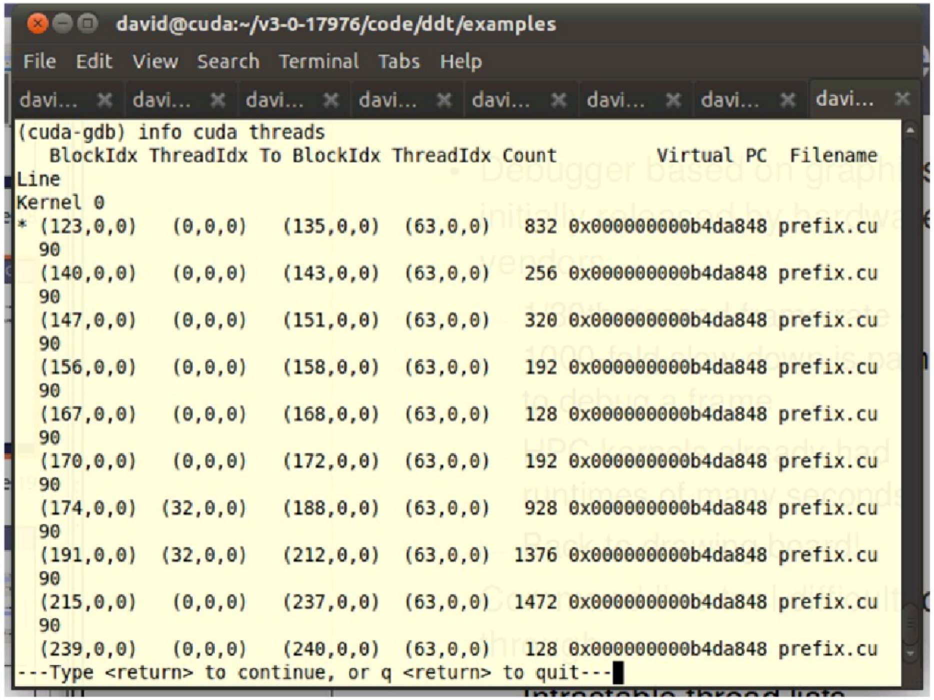
Introducing OpenACC

- **SC11** (Seattle, November 2011)
  - CAPS, Cray, NVIDIA and PGI announce new standard for accelerator programming
    - Easily realize the power of GPU computing
    - A common standard
  - Allinea supports debugging Cray OpenACC compiler
    - Others to follow


OpenACC DE
Dynamic Debuggers
Allinea DDT
Compilers
Cray OpenACC
Performance Analysis Tools
Editors
Tools
Cray gdb
Allinea
Cray
Programming Languages & APIs
GPU-Accelerated Libraries
Domain-Specific Libraries
Libraries & APIs
Workload Managers
Middleware
Linux
Drivers
O/S
CPUs
NVIDIA GPUs
Hardware
Cluster Management

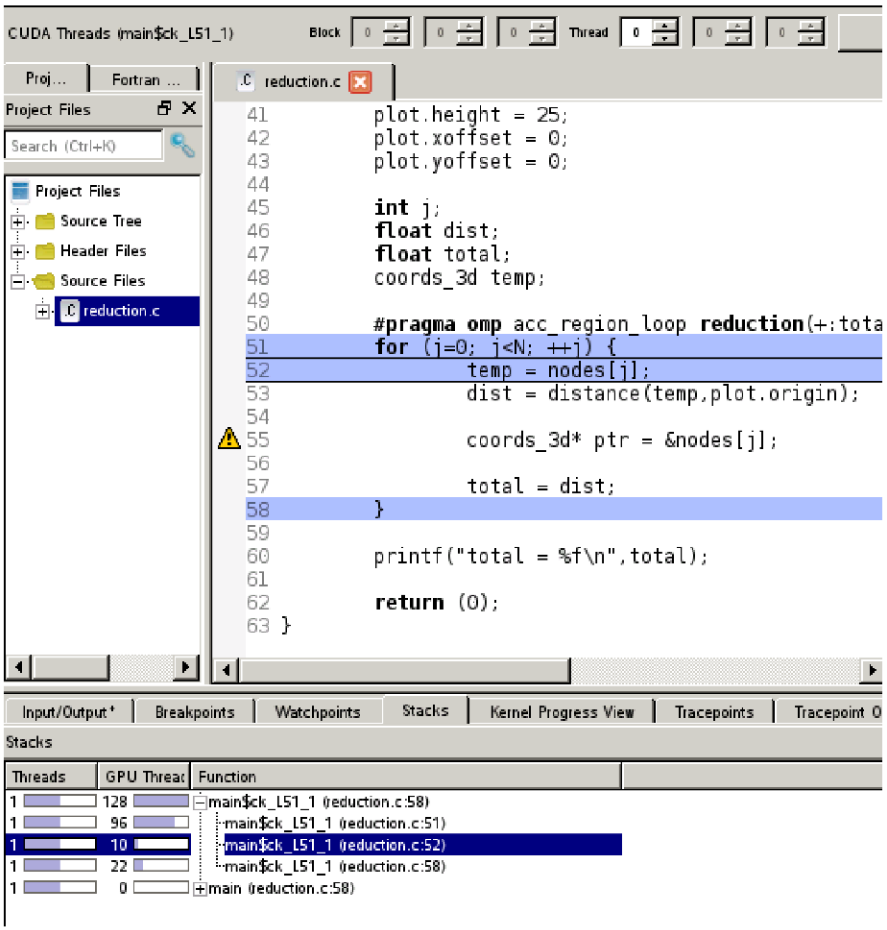
How do we fix OpenACC bugs?

- Use print statements?
  - Too intrusive
  - May alter runtime behaviour
- Command line debugger?
  - A good start:
    - Variables, source code
    - Large thread counts overwhelming
  - Too complex
- A graphical debugger is needed ...



allinea
www.allinea.com

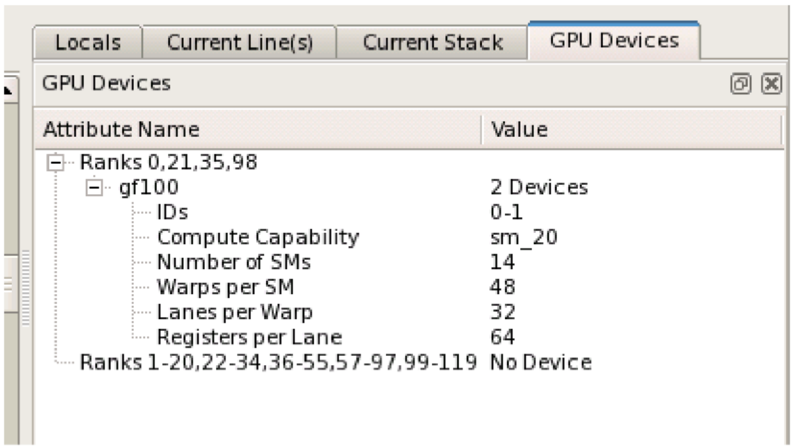
OpenACC Debugging on Cray Platforms

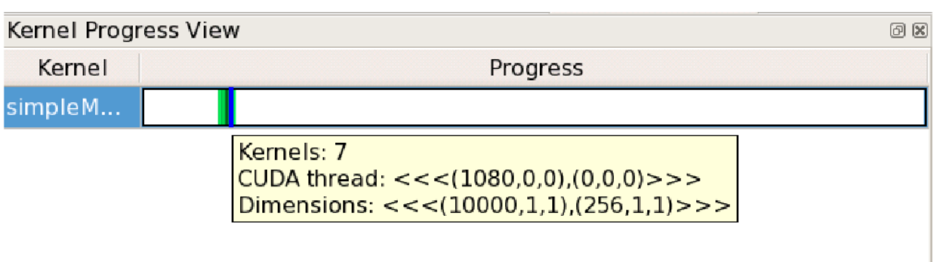


- Debug on GPUs with Allinea DDT
  - Variables – arrays, pointers, full F90 and C support
  - Set breakpoints and step warps and blocks
  - Consistent user experience with full warp/block/kernel controls
- Requires Cray compiler with OpenACC support
  - Other compilers to follow ...

allinea
www.allinea.com

Overviews of GPUs

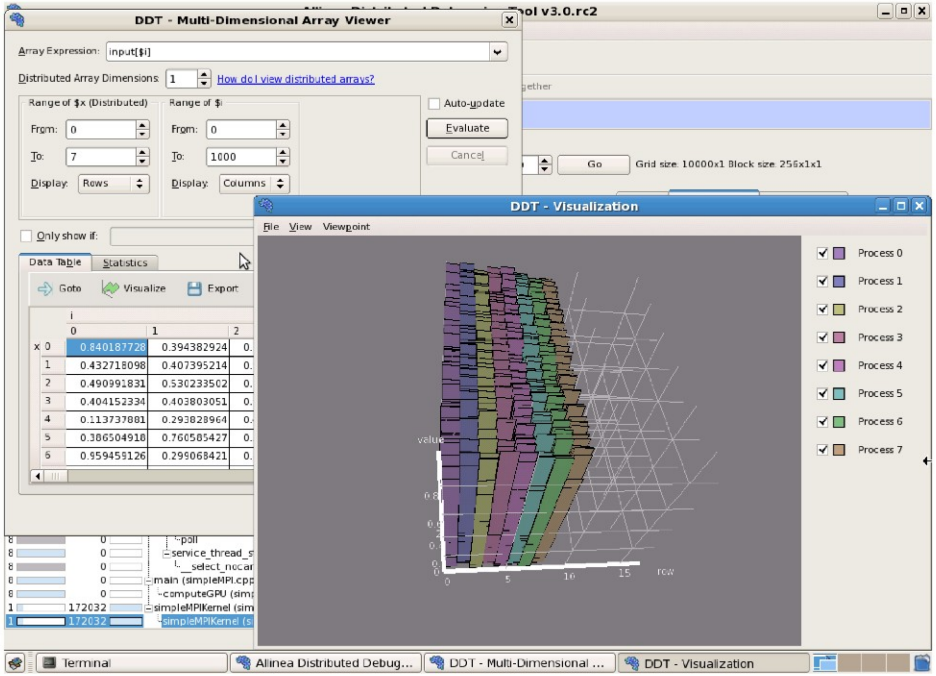




- GPU device overview shows system properties
  - Helps optimize grid sizes
  - Handy for bug fixing – and detecting hardware failure!
- Kernel progress view
  - Shows progress through kernels running on GPUs
  - Click to select a thread

Examining GPU data

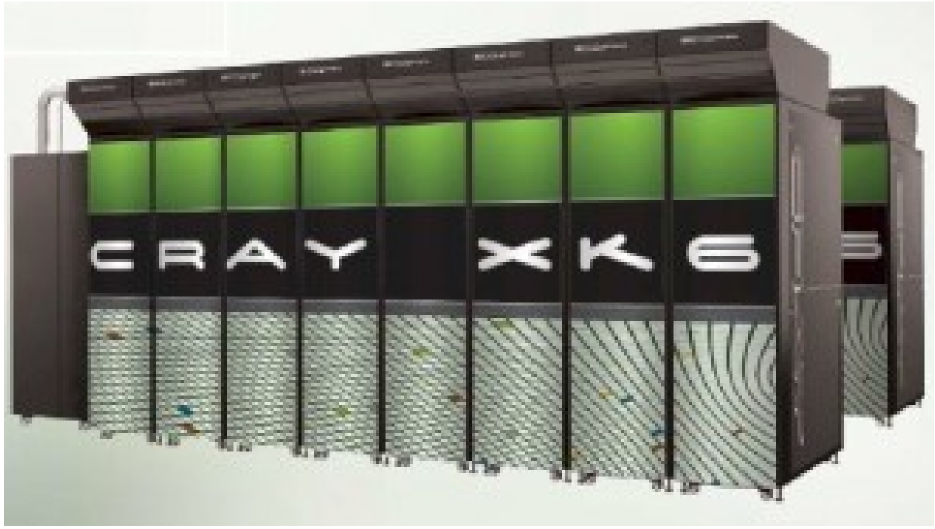
- Debugger reads host and device memory
  - Shows all memory classes: shared, constant, local, global, register..
  - Able to examine variables
  - … or plot larger arrays directly from device memory



allinea
www.allinea.com

Debugging OpenACC at Scale!

- Large Cray XK6 systems in (or almost in) place
  - ORNL Titan
    - MPI debugging proven at 220,000 CPU cores
      - Targeting 300,000 CPU cores
    - MPI-OpenACC hybrid codes *expected* to scale similarly
  - NCSA Blue Waters
    - Targeting 380,000 CPU cores



- Allinea DDT chosen for both systems – at scale

What to expect from performance tools

- Performance tools collect information during the execution of a program to enable the performance of an application to be understood, documented, and improved
- A wide variety of performance tools exist which collect different information in different ways
- It is up to the user to determine:
  - what tool to use
  - what information to collect
  - how to interpret the collected information
  - how to change code to improve the performance

https://www.alcf.anl.gov/sites/www.alcf.anl.gov/files/L2P_Scott_0.pdf

BG/Q Tools Development Status

Tool Name	Source	Provides	Q Status
bgpm	IBM	HPC	Available
gprof	GNU/IBM	Timing (sample)	Available
TAU	Unv. Oregon	Timing (inst, sample), MPI, HPC	Available
Rice HPCToolkit	Rice Unv.	Timing (sample), HPC (sample)	Available
IBM HPCT	IBM	MPI, HPC	In development. Beta available
mpiP	LLNL	MPI	Available
PAPI	UTK	HPC API	Available
Darshan	ANL	IO	Available
Open|Speedshop	Krell	Timing (sample), HCP, MPI, IO	In development. Beta available
Scalasca	Juelich	Timing (inst), MPI	Available
DynInst	UMD/Wisc/IBM	Binary rewriter	In development
ValGrind	ValGrind/IBM	Memory & Thread Error Check	Development pending
Jumpshot	ANL	MPI	Available

https://www.alcf.anl.gov/sites/www.alcf.anl.gov/files/L2P_Scott_0.pdf

Market analysis - customer quotes

Richard Gerber
NERSC, 2011

> Chris and I went back and forth over a number of months, trying to install and use various tools. All failed. My result: no success identifying and installing new tools for our entire user base.

Sergey Mashchenko
SHARCNET 2012

> I've tried all the tools; none of them were any use for real codes. Some made programs run 3 times slower - completely useless!

Ramses / Scott
SciNet 2012

> At least 50% of all the problems we see are simple, common mistakes made again and again, but we don't have a tool easy enough for users to run on their own.

Missing pieces

- Scalability
- Programmers new to MPI
- Long-running jobs
- Ease-of-use

Who needs performance tools?

Scientists and researchers	Domain experts first, programmers second. Bottlenecks are easy to add, difficult to find. Frustrating, inefficient.
HPC analysts	Tool experts, support above full-time. No tools to recommend to 'normal' users. Time wasted on trivial errors.
Cluster 'owners'	Must show return on investment - that cluster utilization is high. Best served by lots of well-scaling codes.

What is a performance tool worth?

Scientists and researchers	Get results faster. More efficient use of their time and cluster allocation. Alternative: email analyst and wait.
HPC analysts	Increases their job performance and satisfaction. Alternative: continue supporting every user "by hand".
Cluster 'owners'	Scalable programs justify cluster purchase. Alternative: training courses, upgrade hardware.

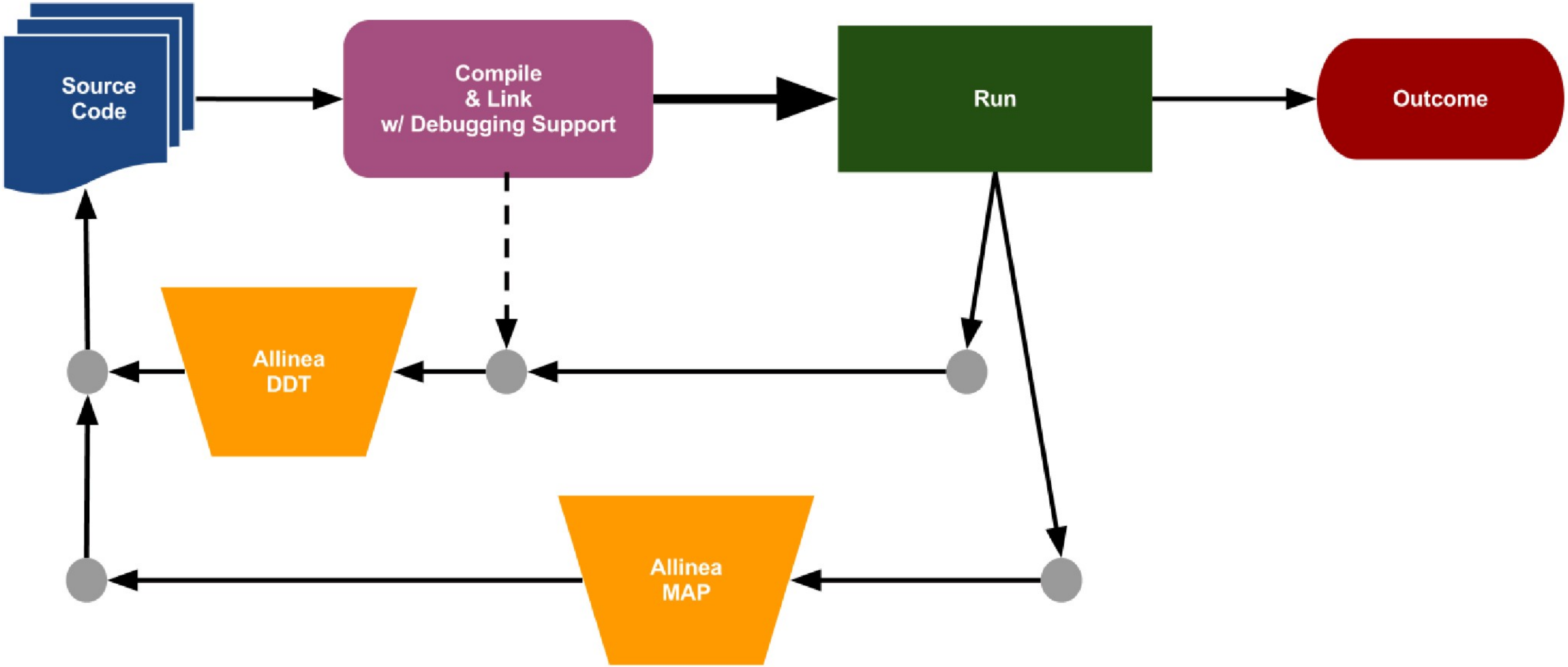
allinea

Leaders in parallel software development tools

Built on DDT's infrastructure

- Proven at over 200k cores
- Easy to configure
- Data aggregated on the cluster
- Visually scalable UI

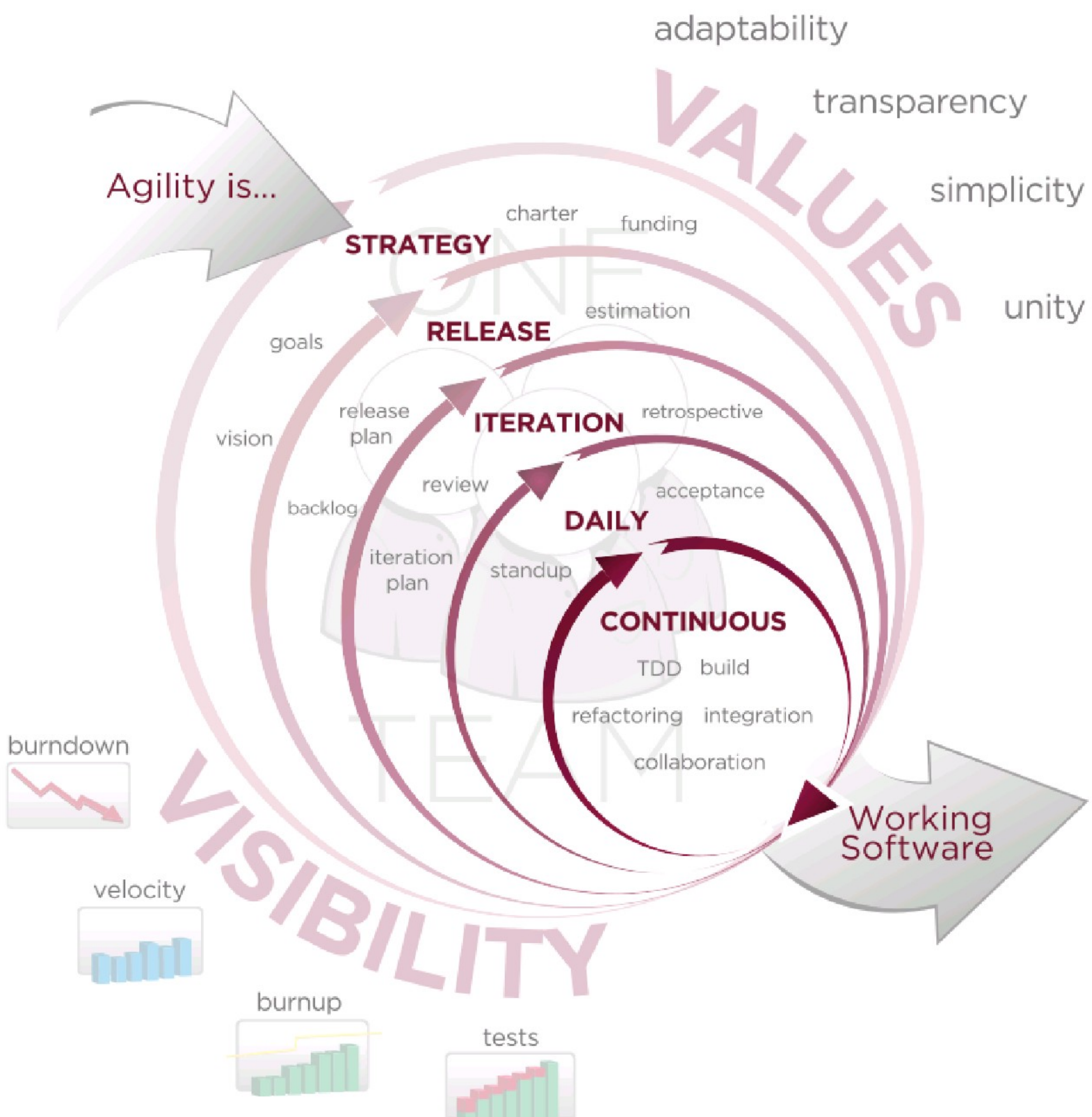
Dev. Process + Allinea Env.



www.allinea.com

http://agilemanifesto.org/

AGILE DEVELOPMENT



ACCELERATE DELIVERY

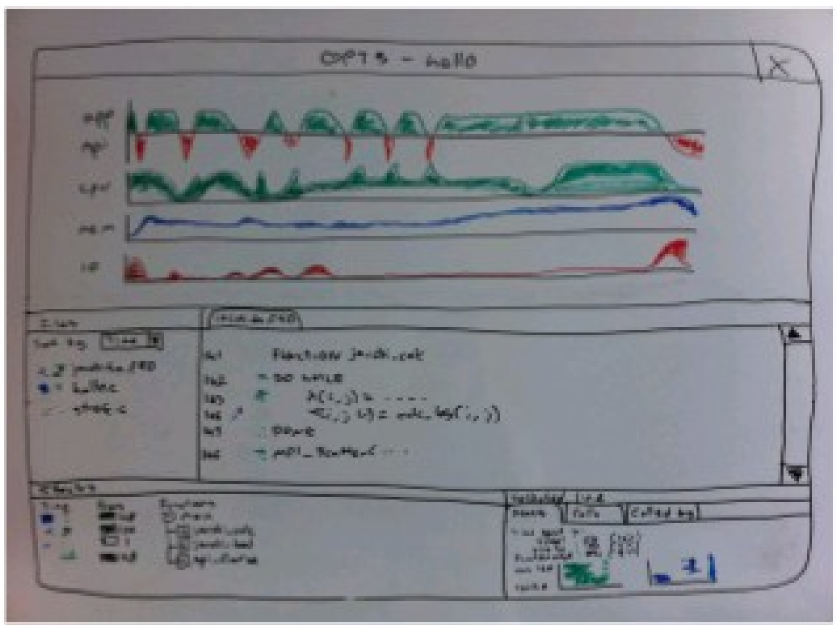
Agile Manifesto: 12 Principles

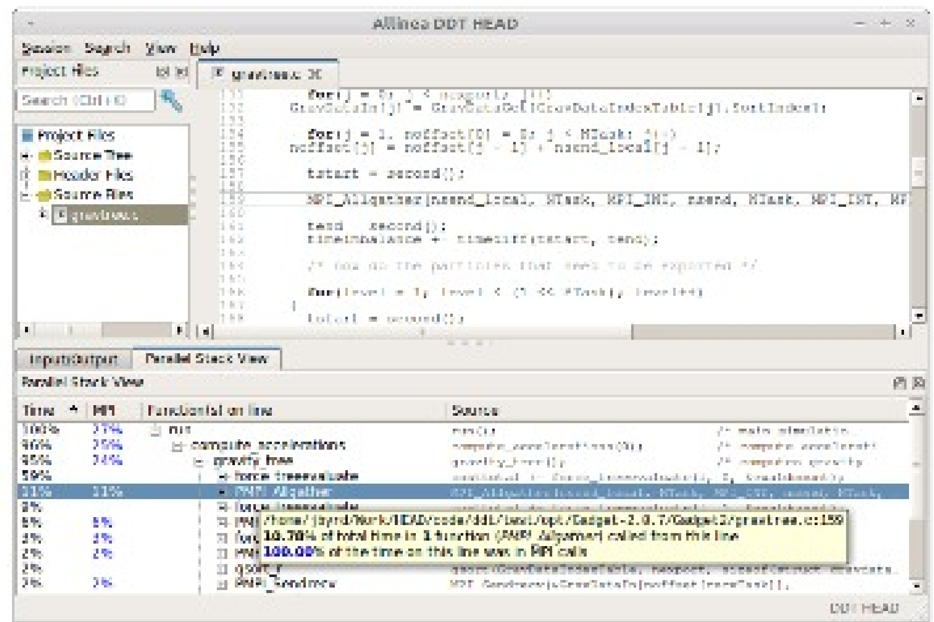
- Customer satisfaction by rapid delivery of useful software
- Welcome changing requirements, even late in development
- Working software is delivered frequently (weeks rather than months)
- Working software is the principal measure of progress
- Sustainable development, able to maintain a constant pace
- Close, daily co-operation between business people and developers
- Face-to-face conversation is the best form of communication (co-location)
- Projects are built around motivated individuals, who should be trusted
- Continuous attention to technical excellence and good design
- Simplicity- The art of maximizing the amount of work not done - is essential
- Self-organizing teams
- Regular adaptation to changing circumstances

allinea

Leaders in parallel software development tools

Feedback-driven development

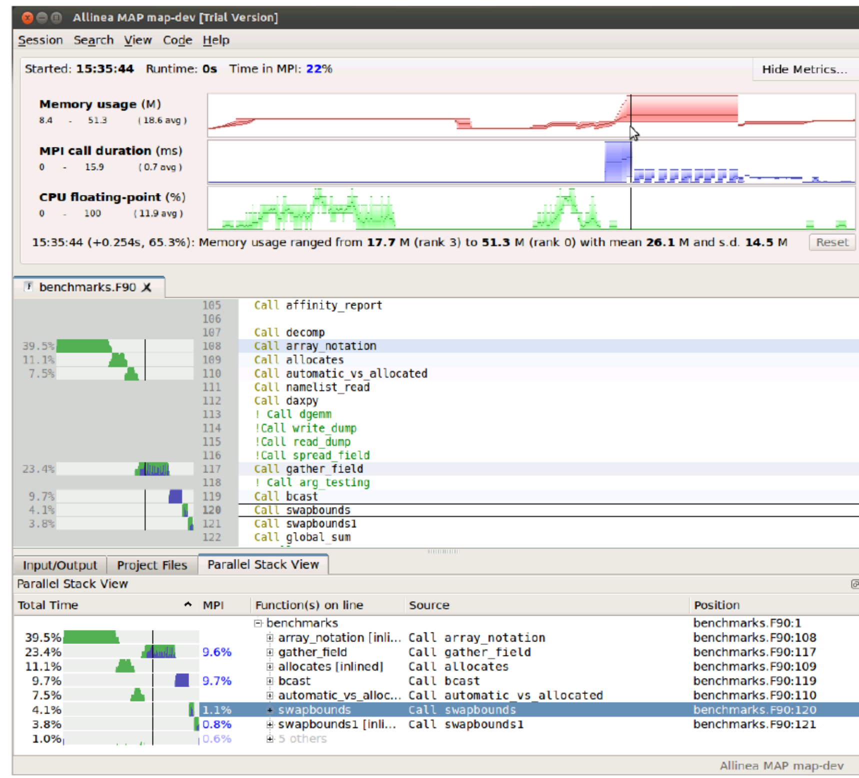




www.allinea.com

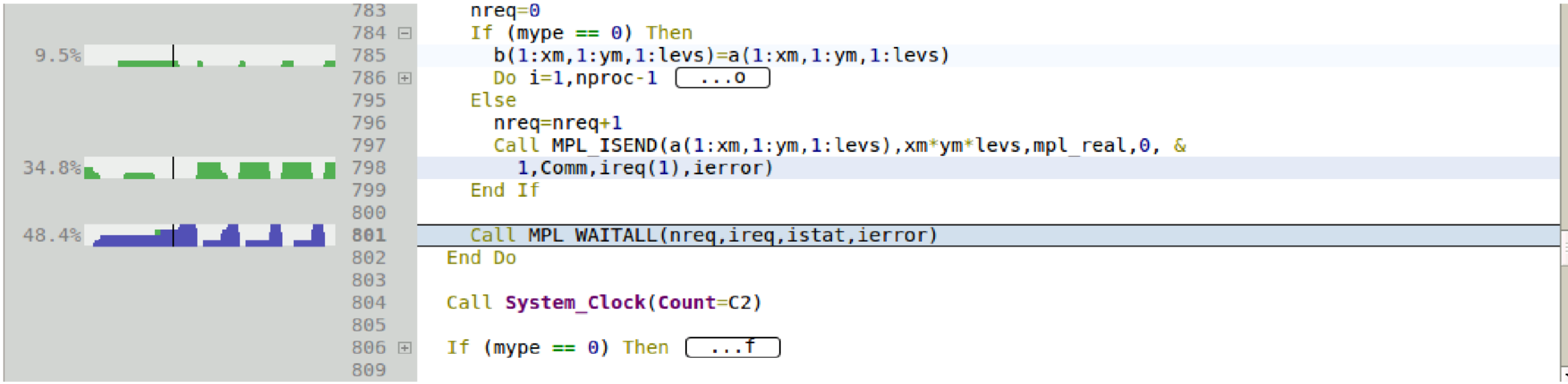
Allinea MAP: The profiler you'll want to use

- Works first time, every time
- From one process to tens of thousands
  - 5% slowdown
  - No instrumentation and no huge data files
- No need to recompile



allinea
www.allinea.com

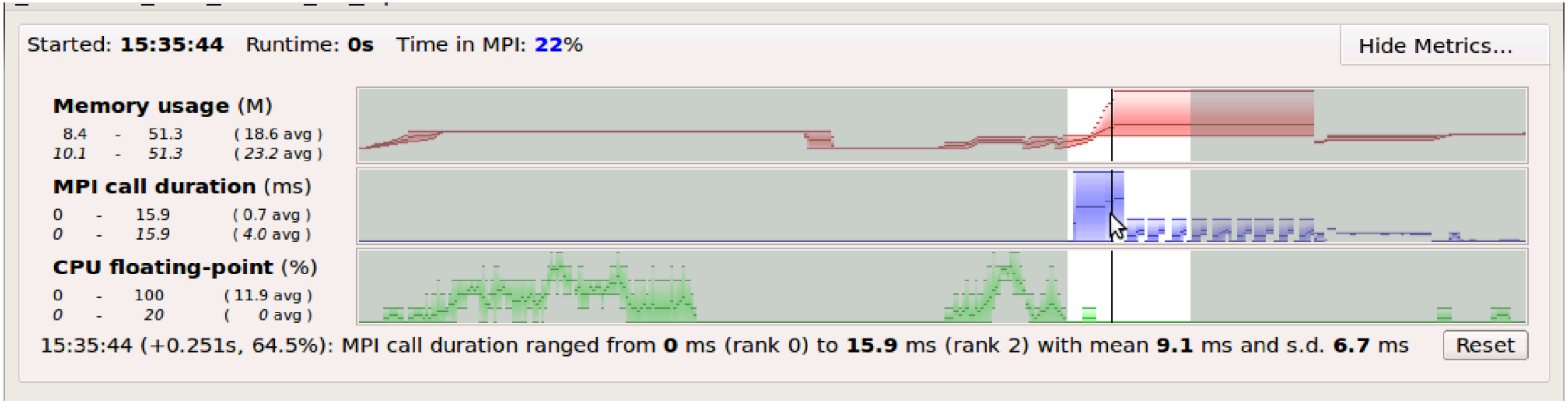
Allinea MAP: Refreshing simple



- See where time is spent in your source code.
- Visualize the entire run
- Zoom in to explore iterations, functions and loops.

www.allinea.com

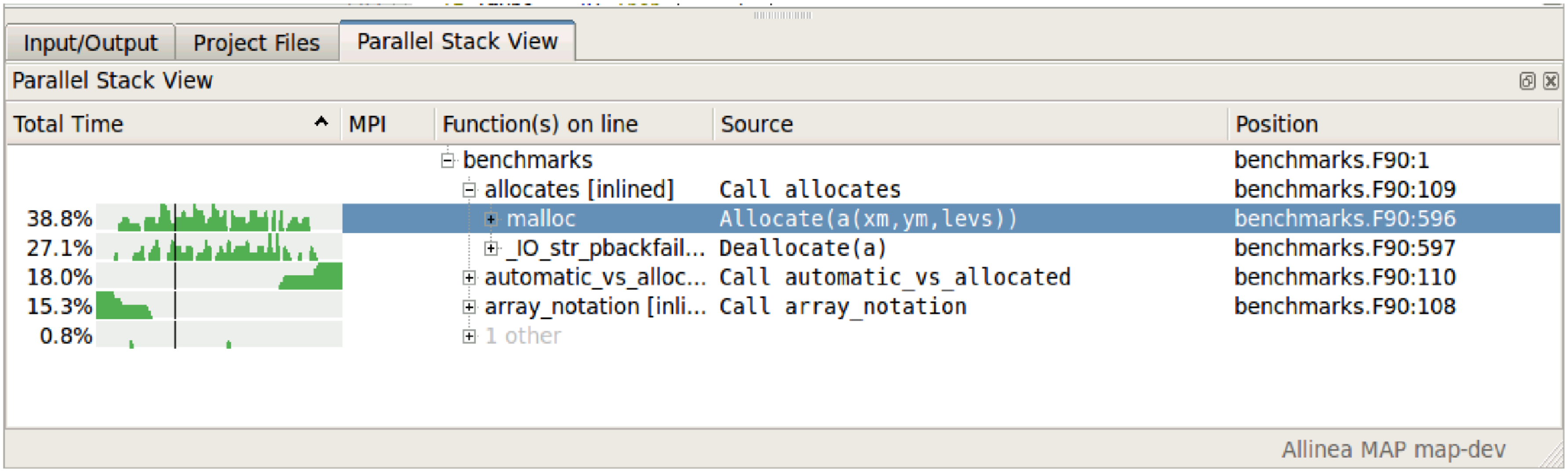
Allinea MAP: Surprisingly deep



- See where time is spent in your source code
- Visualize the entire run
- Zoom in to explore iterations, functions and loops

allinea
www.allinea.com

Allinea MAP: Built on strength



- World-class scalability
  - Shares Allinea DDT tree architecture – proven beyond Petascale
  - Data is merged on the cluster: no huge files.

allinea
www.allinea.com