

allinea



Leaders in parallel software development tools

Alinea DDT

Debugging HPC Applications

Dirk Schubert

Senior Software Developer, Alinea Software

dschubert@allinea.com

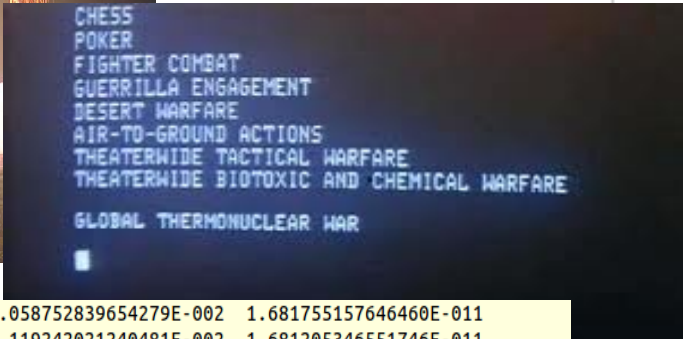
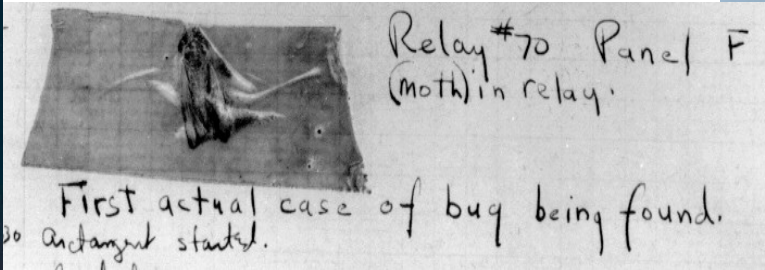
www.allinea.com

About Alinea



- HPC development tools company
 - Flagship product Alinea 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



Country:

* United Kingdom ▼

Phone:

4.42E+11

Industry:

Please Select ▼

```

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 000000000405400 Unknown Unknown Unknown
omp-break 000000000404B23 Unknown Unknown Unknown
libiomp5.so 00007F6E3A7C6B93 Unknown Unknown Unknown
Aborted (core dumped)
    
```



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

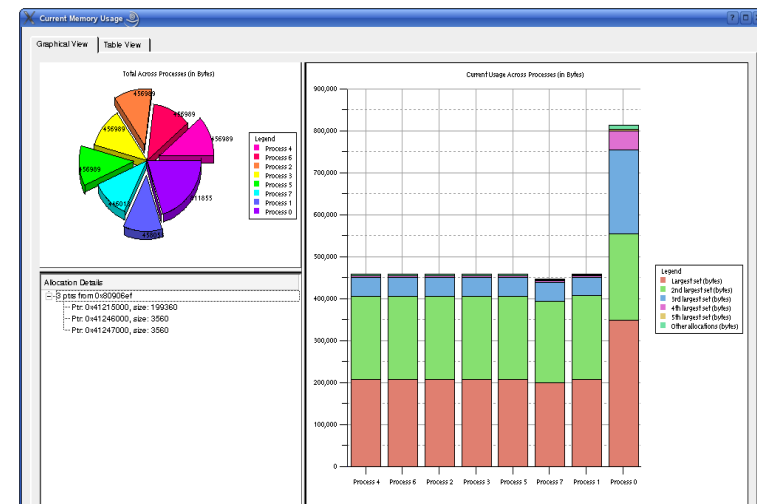
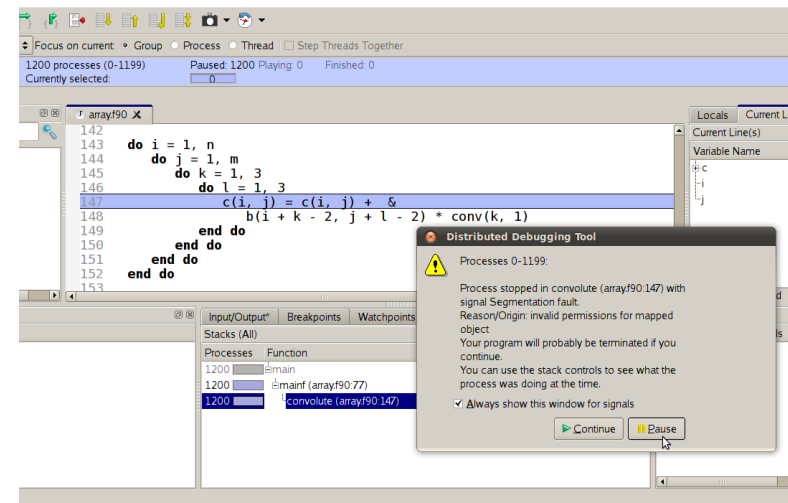
Debugging Parallel Applications

- The same need: observation, control, ...
 - A complex environment – with complex problems
 - More processes, more data
 - 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

Alinea DDT in a nutshell

- Graphical source level debugger for
 - Parallel, multi-threaded, scalar or hybrid code
 - C, C++, F90, Co-Array Fortran, UPC, CUDA, OpenACC
- Strong feature set
 - Memory debugging
 - Data analysis
- Managing concurrency
 - Emphasizing differences
 - Collective control

“Make as simple as possible, no more”



Fixing everyday crashes

- Typical crash scenario:
 - Threads/processes can be anywhere
 - Too many to manually examine individually
- A good overview is important
 - Allinea DDT merges stacks from processes and threads into a tree
 - Leap to source for crashes
 - Information scalably without overload
- Common fault patterns evident instantly
 - Divergence, deadlock

Processes	Function
150120	_start
150120	_libc_start_main
150120	main
150120	pop (POP.f90:81)
150120	initialize_pop (initial.f90:119)
150120	init_communicate (communicate.f90:87)
150119	create_ocn_communicator (communicate.f90:300)
1	create_ocn_communicator (communicate.f90:303)

arrayf90.x

```
142 do i = 1, n
143 do j = 1, m
144 do k = 1, 3
145 do l = 1, 3
146 c(i, j) = c(i, j) + &
147 b(i + k - 2, j + l - 2) * convolve(k, 1)
148 end do
149 end do
150 end do
151 end do
152
153
```

Stacks (All)

Processes	Function
1200	main
1200	mainf (arrayf90.77)
1200	convolute (arrayf90.147)

Distributed Debugging Tool

Processes 0-1199:

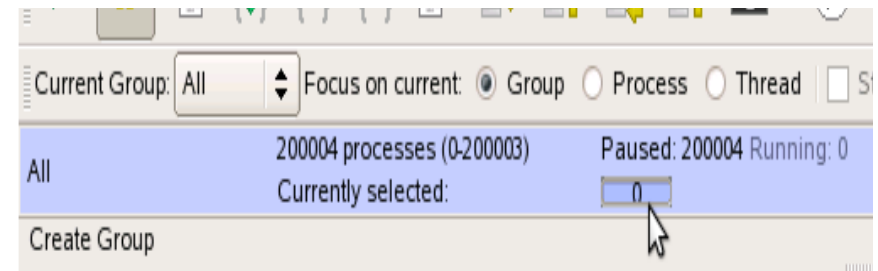
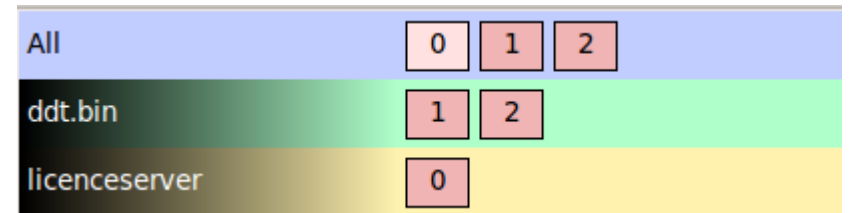
Process stopped in convolute (arrayf90.147) with signal Segmentation fault.
Reason/Origin: invalid permissions for mapped object
Your program will probably be terminated if you continue.
You can use the stack controls to see what the process was doing at the time.

Always show this window for signals

Continue Pause

Process Control

- Interacting with application progress is easy with DDT
 - Step, breakpoint, play, or set data watchpoints based on groups
 - Change interleaving order by stepping/playing selectively
- Group creation is easy
 - Integrated throughout Alinea DDT - eg. stack and data views

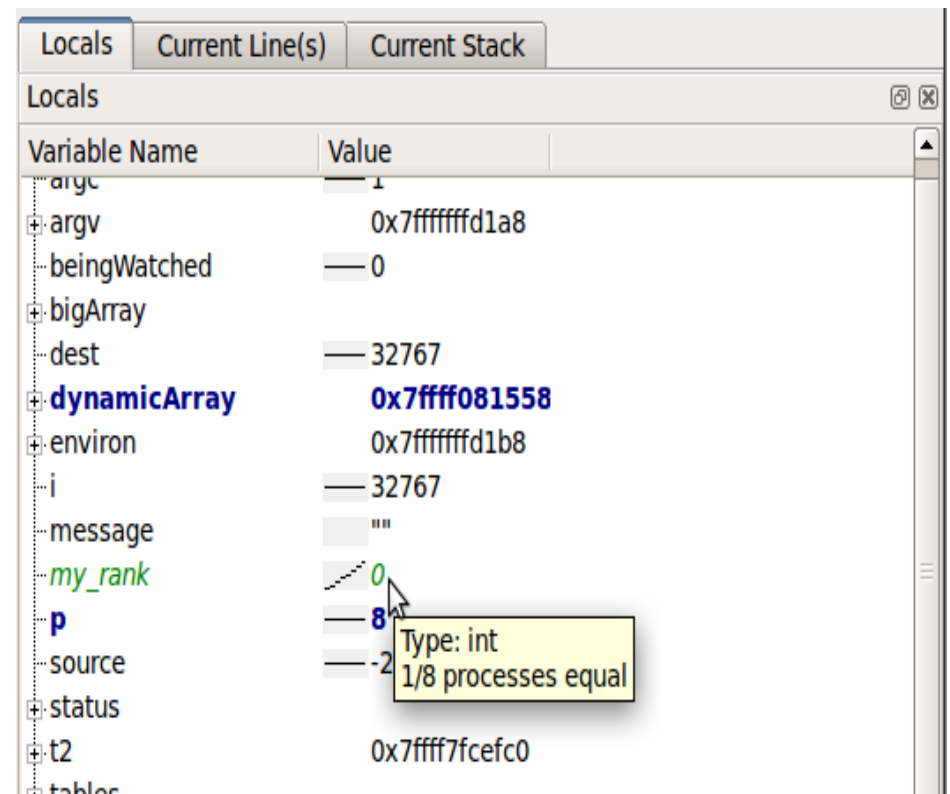


A screenshot of the DDT stack view showing a call stack for process 1. The stack is titled 'Stacks (All)' and has columns for 'Processes' and 'Function'.

Processes	Function
150120	_start
150120	__libc_start_main
150120	main
150120	pop (POP.f90:81)
150120	initialize_pop (initial.f90:119)
150120	init_communicate (communicate.f90:87)
150119	create_ocn_communicator (communicate.f90:300)
1	create_ocn_communicator (communicate.f90:303)

Simplifying data divergence









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



The screenshot shows the 'Locals' window in Allinea DDT. The window has tabs for 'Locals', 'Current Line(s)', and 'Current Stack'. The 'Locals' tab is active, displaying a table of local variables. The table has two columns: 'Variable Name' and 'Value'. The variables listed are: argc (1), argv (0x7fffffff1a8), beingWatched (0), bigArray, dest (32767), dynamicArray (0x7fff081558), environ (0x7fffffff1b8), i (32767), message (''), my_rank (0), p (8), source (-2), status, t2 (0x7fff7fcefc0), and tablec. The variable 'my_rank' is highlighted in green, and a tooltip is displayed over its value '0', showing 'Type: int' and '1/8 processes equal'.

Variable Name	Value
argc	1
argv	0x7fffffff1a8
beingWatched	0
bigArray	
dest	32767
dynamicArray	0x7fff081558
environ	0x7fffffff1b8
i	32767
message	""
my_rank	0
p	8
source	-2
status	
t2	0x7fff7fcefc0
tablec	

Tracepoints

Input/Output	Breakpoints	Watchpoints	Tracepoints	Tracepoint Output	Stacks (All)
Tracepoint Output					
Tracepoint	Processes	Values logged			
vhone.f90:85	976, ranks 12, 14-17, 22-23, 12...	mype  2172-3527	jcol:  2-83	mod <input type="checkbox"/>	pey <input type="checkbox"/>
vhone.f90:81	960, ranks 12, 14-17, 22-23, 12...	ks <input type="checkbox"/> 1	kmax <input type="checkbox"/>	pez <input type="checkbox"/>	
vhone.f90:85	942, ranks 12, 14-17, 22-23, 12...	mype  2172-3527	jcol:  2-83	mod <input type="checkbox"/>	pey <input type="checkbox"/>
vhone.f90:81	929, ranks 12, 14-17, 22-23, 12...	ks <input type="checkbox"/> 1	kmax <input type="checkbox"/>	pez <input type="checkbox"/>	
vhone.f90:85	919, ranks 12, 14-17, 22-23, 12...	mype  2172-3527	jcol:  2-83	mod <input type="checkbox"/>	pey <input type="checkbox"/>
vhone.f90:81	898, ranks 12, 14-17, 22-23, 12...	ks <input type="checkbox"/> 1	kmax <input type="checkbox"/>	pez <input type="checkbox"/>	
vhone.f90:85	884, ranks 12, 14-17, 22-23, 12...	mype  2172-3527	jcol:  2-83	mod <input type="checkbox"/>	pey <input type="checkbox"/>
vhone.f90:81	880, ranks 12, 14-17, 22-23, 12...	ks <input type="checkbox"/> 1	kmax <input type="checkbox"/>	pez <input type="checkbox"/>	

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

Large Array Support

Array Expression:

Distributed Array Dimensions: [How do I view distributed arrays?](#)

Range of \$x (Distributed) Range of \$i Auto-update

From: From: Evaluate

To: To: Cancel

Display: Display:

Only show if: [See Examples](#)

Data Table Statistics

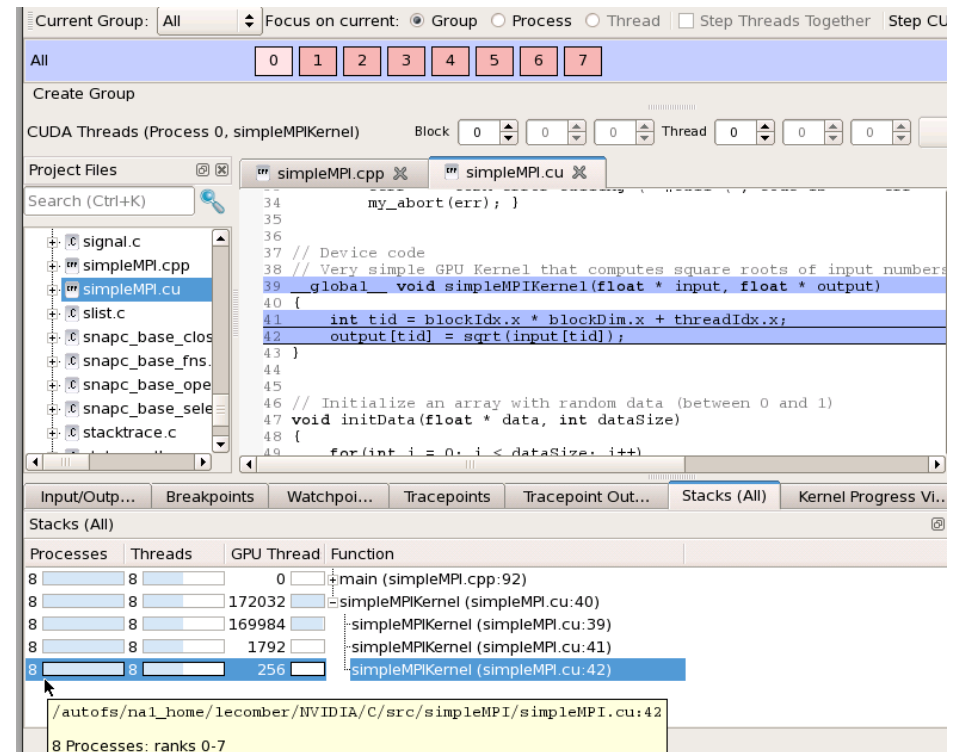
	i	2444	2733	3011	3185	4704	5343	6795	7881	9108	9467
x 0											
1				1					1		
2	1				1						1
3											
4		1									
5			1			1					
6											

Visualize in 3D Export to Spreadsheet/HDF5... Close

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

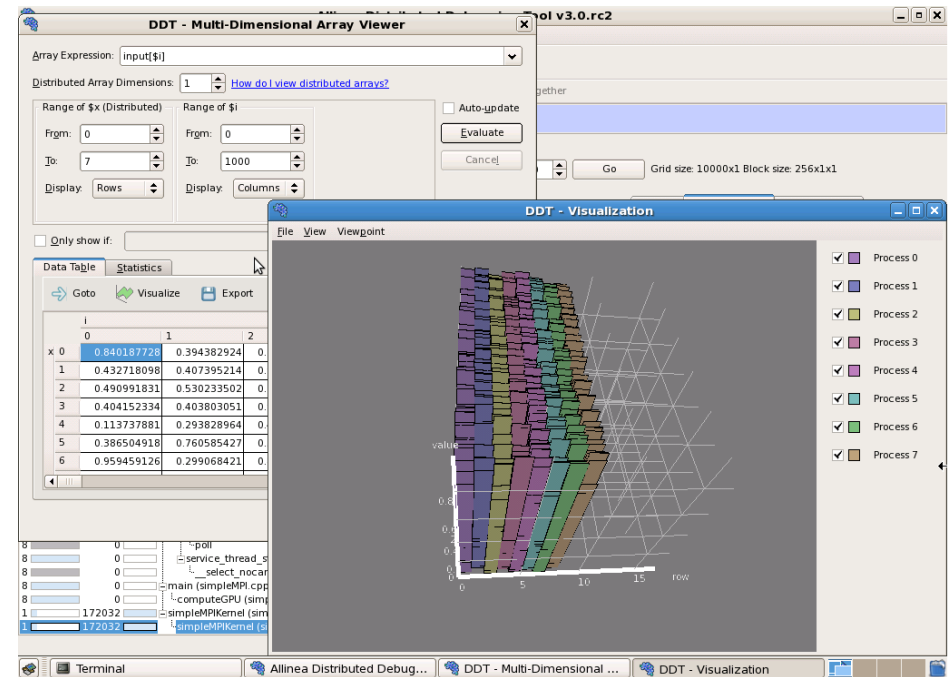
GPU debugging with Alinea DDT

- 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



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



Overviews of GPUs

Attribute Name	Value
Ranks 0,21,35,98	
gf100	2 Devices
IDs	0-1
Compute Capability	sm_20
Number of SMs	14
Warps per SM	48
Lanes per Warp	32
Registers per Lane	64
Ranks 1-20,22-34,36-55,57-97,99-119	No Device

- 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

Kernel	Progress
simpleM...	<div style="width: 10%;"><div style="width: 10%;"></div></div>

Kernels: 7
CUDA thread: <<<(1080,0,0),(0,0,0)>>>
Dimensions: <<<(10000,1,1),(256,1,1)>>>

User and administrator friendly Offline debugging

- Using a workload scheduler
 - Machines are available when the scheduler decides (by night ?)
 - Can be tricky to get a big cluster exactly when the developer wants it
- Offline debugging : printf replacement
 - Tracepoints and offline debugging
 - Job runs without debugger interface and record variables
- Worlds first scalable batch debugger
 - Set tracepoints, breakpoints, and run !
 - Memory debugging errors, crashes
 - Reports in HTML or plain text

Allinea DDT Off-line Log

Messages Tracepoints Output

Messages

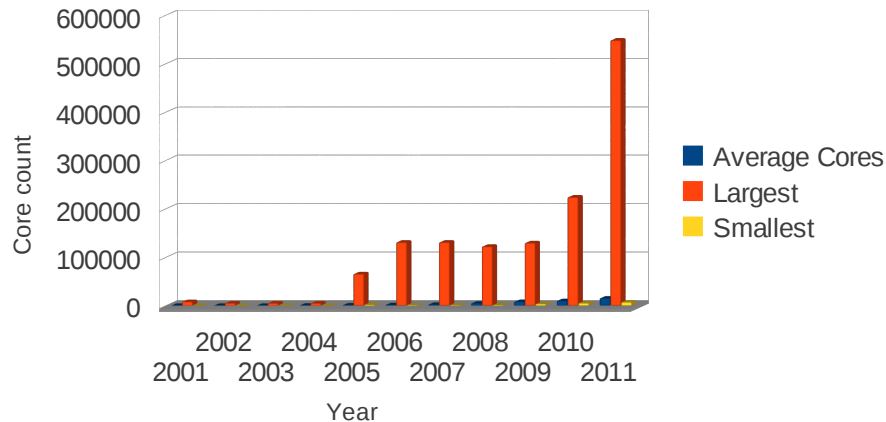
Type	Time	Processes	Message
	08:53:45.437	n/a	Launching program /home/david/Work/HEAD/code/ddt/libexec/ddt.bin.
	08:53:48.154	n/a	DDT could not find valid debug information in one or more of your processes. Source files, local variables and other features may be unavailable or inaccurate. Please check you are using the correct debug interface and have compiled your code with debug information.
	08:53:48.436	0	Memory error detected in FcPatternReference from /usr/lib/x86_64-linux-gnu/libfontconfig.so.1. Thread 1 attempted to dereference a null pointer or execute an SSE instruction with an incorrectly aligned memory. Tip: Use the stack list and the local variables to explore your program's current state and identify the source of the error. Threads,Function 1,main 1, QApplication::QApplication 1, QApplicationPrivate::construct 1, QApplicationPrivate::x11_apply_settings 1, FcInit 1, FcInitLoadConfigAndFonts 1, FcInitLoadConfig 1, FcConfigParseAndLoad 1, XML_ParseBuffer 1, 0x00007ffff0809e6b 1, 0x00007ffff08072e2 1, 0x00007ffff080a77e 1, 0x00007ffff0809e65 1, FcConfigParseAndLoad 1, FcConfigParseAndLoad 1, FcConfigParseAndLoad 1, FcConfigParseAndLoad 1, XML_ParseBuffer 1, 0x00007ffff0809e6b 1, 0x00007ffff08072e2 1, 0x00007ffff080a77e 1, 0x00007ffff0809e65 1, FcConfigParseAndLoad 1, FcValueSave 1, FcValueSave 1, FcValueSave 1, FcPatternReference
	08:53:48.477	n/a	Every process in your program has terminated.

Messages Tracepoints Output

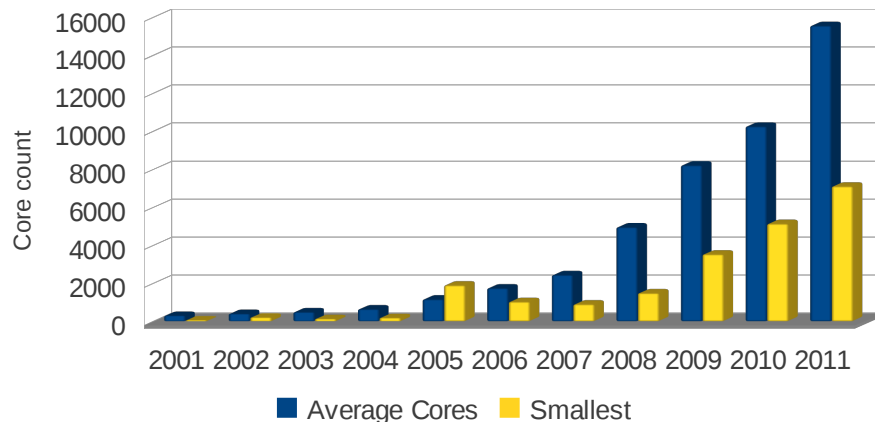
Tracepoints

Extreme machine sizes

Growth in HPC core counts



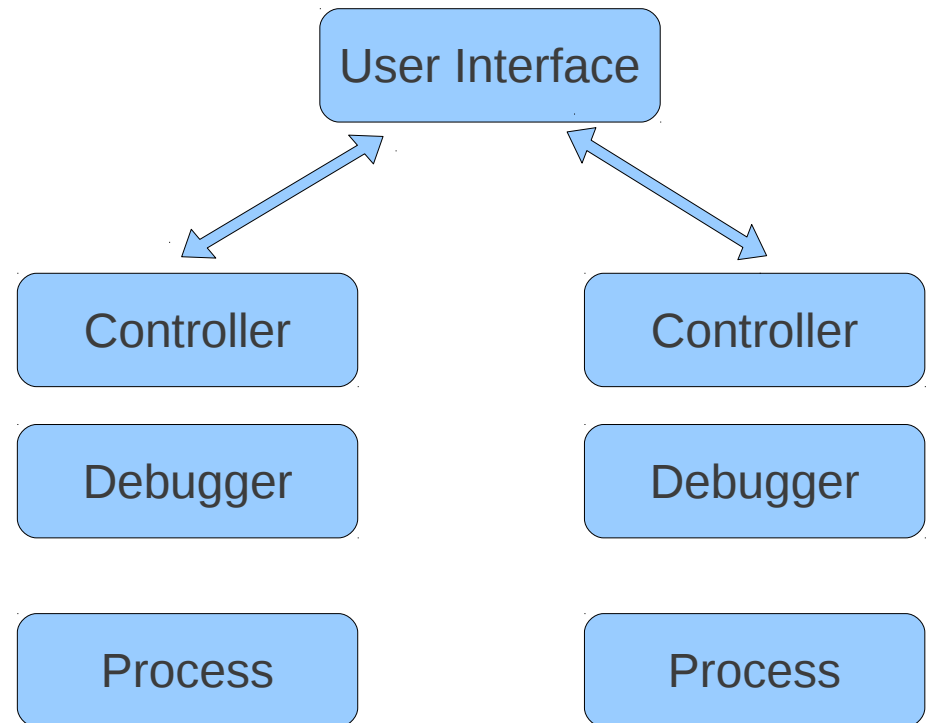
HPC core counts



- 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

A simple parallel debugger

- A basic parallel debugger
 - Aggregate scalar debuggers
 - _ They work: good starting point
 - _ 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
 - _ Direction connections - linear performance
 - Any per-process item is an eventual bottleneck
 - _ Operating system limitations
 - File handles on the GUI
 - Threads, processes
 - _ I/O limitations
 - Linear access counts on the best networked file systems are still linear
 - _ Memory and computation limitations
 - Machines still getting bigger...

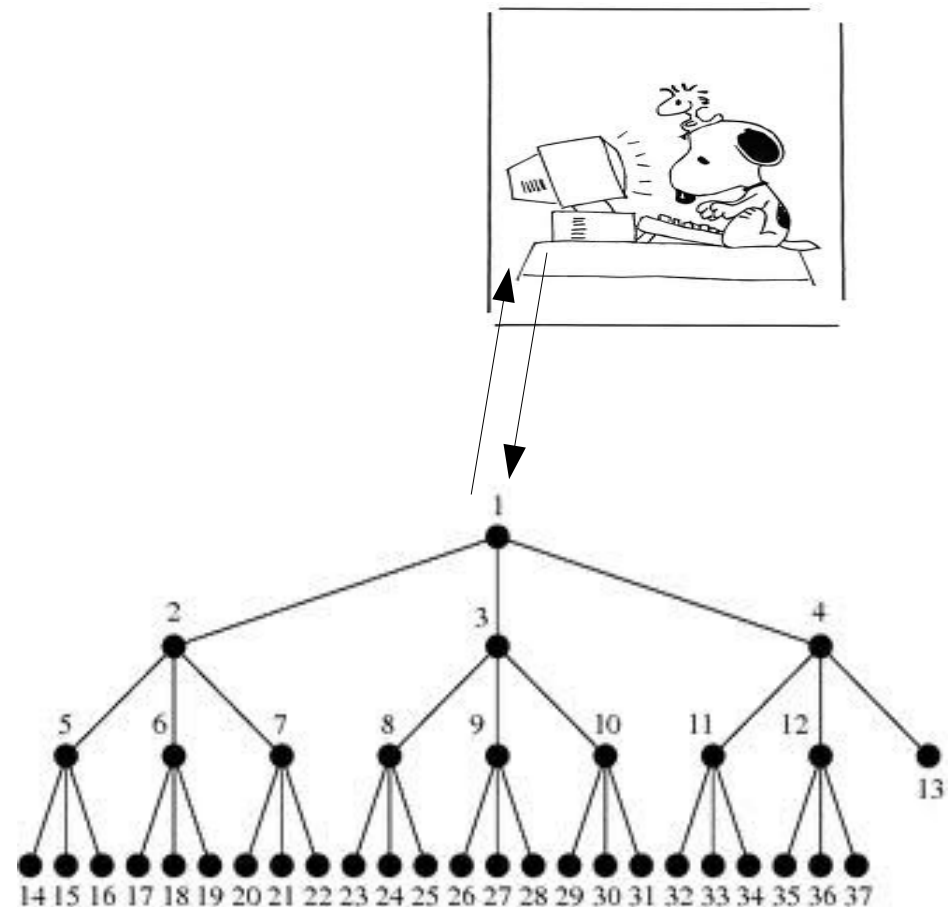


Bug fixing as scale increases

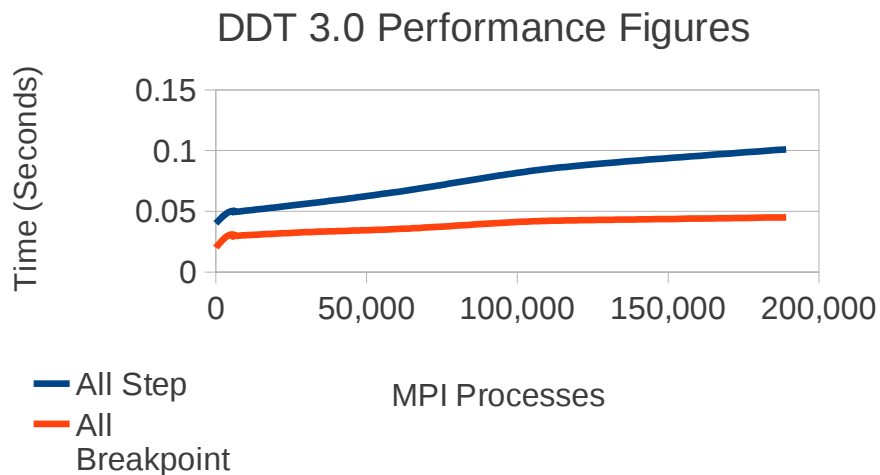
- Can we reproduce at a smaller scale?
 - Attempt to make problem happen on fewer nodes
 - Often requires reduced data set – the large one may not fit
 - Smaller data set may not trigger the problem
 - Does the bug even exist on smaller problems?
 - Didn't you already try the code at small scale?
 - 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

How to make a Petascale debugger

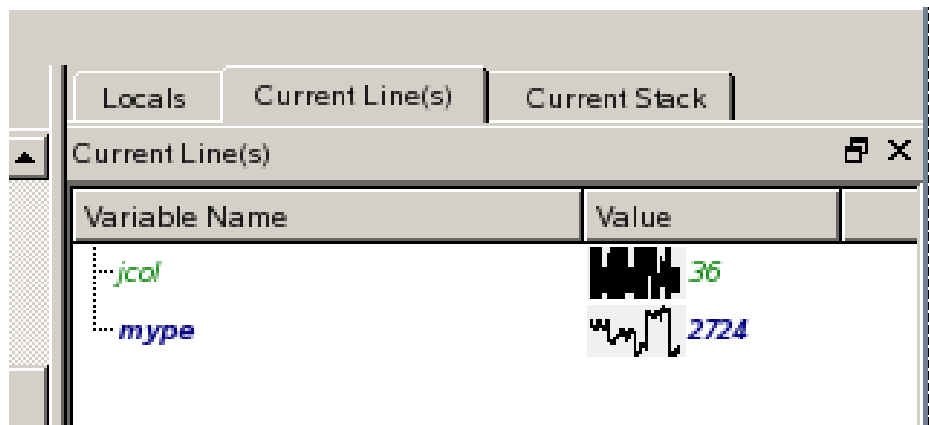
- A control tree is the solution
 - Ability to send bulk commands and merge responses
 - 100,000 processes in a depth 3 tree
 - Compact data type to represent sets of processes
 - eg. For message envelopes
 - An ordered tree of intervals?
 - Or a bitmap?
 - Develop aggregations
 - Merge operations are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



For Petascale and beyond



- Partnership with largest users
 - Oak Ridge National Laboratories
 - LLNL, ANL, CEA and others
- High performance debugging - even at 220,000 cores
 - Step all and display stacks: 0.1 seconds
 - Logarithmic
- Usability is a “Big Thing”
 - Scalable interface and features



Research

- Debugging is about observing the anomalies
 - Anomaly detection is easier when thousands of “trials” are running together
 - Parallel debugging could be easier than single threaded
 - Can past behaviour help users to identify current issues?
 - Can we make developing and debugging code easier with today's flux in programming models?
 - Heterogeneous systems/models: CUDA, OpenACC, OpenCL, for GPUs, Intel MIC, ...
 - PGAS languages - Coarray Fortran, UPC to take the pain of MPI away
 - Task parallel models – Cilk, OpenMP 3, and others ...
 - With multi/many-core mobile phones and tablets – what analogies work in embedded computing?

The Future

- Concurrency will become greater
 - 2012 or early 2013 – DDT will debug a million core system
 - International and national groups are preparing for **Exascale**
 - Systems expected 2018-2020
 - 100x more powerful than today's most powerful system
 - Orders of magnitude more components – fault tolerance issues
 - More hybrid environments expected to give better green credentials
 - Programming models will continue to change

Questions?
