# Blines Scale to New Heights

### **Debugging at Scale**

**Lindon Locks** 

llocks@allinea.com

www.allinea.com



- At scale debugging from 100 cores to 250,000
  - Problems faced by developers on real systems
  - Alternative approaches to debugging and how they stack up
  - How Allinea makes debugging at scale work



- HPC tools since 2001
  - Allinea DDT Scalable parallel debugger
  - Allinea OPT Optimisation tool for MPI and non-MPI
  - Allinea DDTLite Parallel debugging plugin for Microsoft Visual Studio
- Large customer base
  - Ease of use means tools get used
  - Users debugging regularly at all scales: 1 to 100,000 cores
  - World's only Petascale debugger

### **Some Clients and Partners**

- ANL, CEA, EPCC, GENCI, IDRIS, Juelich, NERSC, ORNL











Microso

AMD The future is fusion



Sgi



Energy

Academic

- CGG Veritas, IFP, Total

Over 200 universities

Major research centres

Aviation and Defence

- EDA
  - Cadence, Intel, Synopsys
- Climate and Weather
  - UK Met Office, Meteo France, NOAA

- Airbus, AWE, BAE, Dassault, DLR, EADS



### Background

- Processor counts growing rapidly
- GPUs entering HPC
- Large hybrid systems imminent
- But what happens when software doesn't work?



Systems in Top 500

Year (June & November Lists)

- Increasing job sizes leads to unanticipated errors
  - Regular bugs
    - Data issues from larger data sets eg. garbage in..., overflow
    - Logic issues and control flow
  - Increasing probability of independent random error
    - Memory errors/exhaustion "random" bugs!
    - System problems MPI and operating system
  - Pushing coded boundaries
    - Algorithmic (performance)
    - Hard-wired limits ("magic numbers")
  - Unknown unknowns

• ....

- Improved coding standards
  - Unit tests, assertions and consistency checks
    - Good practice but tend to be single-process checks
    - Parallel checks also valid and good practice
  - Only checks for things you predict when developed
    - Coverage is rarely perfect
      - Unexpected problems particularly random/system issues often missed
    - Debugger still required
  - Combines well with debuggers
    - Find why a failure occurs not just a pass/fail

- Logging printf and write
  - The oldest debugger still in active use
    - · Tried and tested as easy as "hello world"
    - If you have good intuition into the problem
       Edit code, insert print, recompile and re-run
    - Slow and iterative
  - Use to log exceptions, progress or state
    - Post-mortem analysis only
      - Hard to establish real causal order of output of multiple processes
      - Output can be lost by process termination
    - Rapid growth in log output size
    - Unscalable

- Reproduce at a smaller scale
  - Attempt to make problem happen on fewer nodes
    - Often requires reduced data set the large one may not fit
      - Didn't you already try the code at small scale?
      - Smaller data set may not trigger the problem
    - Does the bug even exist on smaller problems?
    - Is it a system issue eg. an MPI problem?
  - Is probability stacking up against you?
    - Example: 1 in 10,000 independent probability of error?
    - Unlikely to spot on smaller runs without many many runs
    - But near guaranteed to see it on a 10,000 core run
  - What can a parallel debugger do to help?
    - Debug at the scale of the problem. **Now**.

- 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  $\rightarrow$  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

Locals	Curren	t Line(s)	Current Sta	ack
Locals				ð×
Variable N	lame	Value		
- beingW ⊕ bigArra		0		
-dest -dynamicArray -environ -i -message -my rank		0 0x7f1f73 0x7fff7c6 32767 "" 2087064	61468	=
my_ran p	K	2087064	375	





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
  - Visualise message queues

### **Scalable Process Control**

- Stacks (AII) Processes Function 🖹 start 150120 Ė\_\_libc\_start\_main 150120 ḋmain 150120 150120 pop (POP.f90:81) 150120 ⊨initialize\_pop (initial.f90:119) 150120 init\_communicate (communicate.f90:87) 150119 create ocn communicator (communicate.f90:300) create ocn communicator (communicate f90:303)
- Parallel Stack View
  - Find rogue processes quickly
  - Identify classes of process behaviour
  - Rapid grouping of processes
- Control Processes by Groups
  - Set breakpoints, step, play, stop for groups
  - Scalable groups view: compact group display





### Handling Regular Bugs

200 processes (0-1199)         Paused: 1200 Playing: 0         Finished: 0           Currently selected:         0		
© 8 1 array.190 × 142	Locals Cur Current Line(s)	rent l
143       do i = 1, n         144       do j = 1, m         145       do k = 1, 3         146       do l = 1, 3         147       c(i, j) = c(i, j) + &         148       b(i + k - 2, j + l - 2)	× conv(k 1)	
149 end do 150 end do	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  Cont	d

- Immediate stop on crash
  - Segmentation fault, or other memory problems
  - Abort, exit, error handlers
  - CUDA errors
- Scalable handling of error messages
- Leaps to the problem
  - Source code highlighted
  - Affected processes shown
  - Process stacks displayed clearly in parallel

- Full class/structure browsing
  - Local variables and current line(s)
    - Show variables relevant to current position
    - Drag in the source code to see more
  - C, C++, F90: object members, static members, derived types
- Automatic comparison and change detection
  - Scalable and fast

Locals	Current L	.ine(s)	Current Stack		
Locals				Ø	×
Variable N	lame	Value			-
argc		1			
🖃 argv		0x7fff3b	b6c32e8		=
	V)				
being₩	latched	0			
⊕bigArra	у				
··· dest		32767			
🖻 dynami	-	0x219e	890		
<sup>i</sup> *(dyr	namicArray)	0			
+ er viron	1	0x7fff3b	b6c3520		
	e changed			•	-
Type: 1/12	processes	equal			

Expression: ie	rr				~
Processes in c	urrent group (All, 8196 pro	cs)		<b>✓ A</b> lign stac <u>k</u> fr	ames
Limit comp	arison to 🛽 📥 s.f.			Compare	
Only show	if:	See Examples		C <u>a</u> ncel	
Values	Process(es)		Statistics		
-1 0	2195		Count:	8196	
0	0-2194,2196-8195		Not shown:	0	
			Errors:	0	
			Aggregate:	0	
			Numerical:	8196	
			Sum:	-1	
			Minimum:	-1	] =
			Maximum:	0	
			Range:	1	
		\$	Mean:	-0.000122011	
			Variance:	0.000122011	
			nan:	0	
			-nan:	0	
			inf:	0	
			-inf:	0	•
	Use as MPI F	Rank Create Gro	ups Export to S	Spreadsheet	

- Easy to find where the differences are...
  - Cross process comparison of data
    - Fetches values from every process, compares and then groups by value
    - Summary of NaN, Inf and statistics
  - Easy to spot rogues
- Use to group processes
  - Define a process group and control en-masse

### Large Array Support

Array Expression: bigArray[\$i]									•		
Distributed Array Dimensions: 1 🚔 How do I view distributed arrays?											
Range of \$x (Distributed) Range of \$i Auto-update											
Fr <u>o</u> m:	0	<b>A</b>	Fr <u>o</u> m	n: 0		-				<u>E</u> valuat	e
<u>T</u> o:	7	<b></b>	<u>T</u> o:	9999	9	<b>+</b>				Cance	
Display	Display: Rows 🗢 Display: Columns 🗢										
✓ Only	show if:	\$value =	== 1			See Exa	mples				
Data T	ā <u>b</u> le <u>S</u> t	atistics									
	i										
	2444	2733	3011	3185	4704	5343	6795	7881	9108	9467	
× 0			1					1			
2	1		1	1				-		1	
3											
4	1	1			1				2		
-						- 1			-		•
			(	Visualize	e in <u>3</u> D	E <u>x</u> port	to Sprea	adsheet/H	IDF5	Clos	e

- Browse arrays
  - 1, 2, 3, ... dimensions
  - Table view
- Filtering
  - Look for an outlying value
- Export
  - Save to a spreadsheet
- View arrays from multiple processes
  - Search terabytes for rogue data: in parallel with [v3.0]

### **Memory Debugging**





**DDT 3.0 Performance Figures** 



- DDT is delivering Petascale debugging today
  - Collaborations with ORNL on Jaguar Cray XT and CEA
  - Logarithmic performance
  - Many operations now faster at 220,000 than previously at 1,000 cores
  - ~1/10<sup>th</sup> of a second to step and gather all stacks at 220,000 cores



- Debuggers are recognised as the right tools to fix bugs quickly: other methods have limited success, and major issues at scale
- Debugging interfaces must scale to help the user understand what is happening
- Allinea DDT scales in performance and interface breaking all records and making problems manageable
- See Allinea at Supercomputing 2010: Booth 2305



## **Questions?**

SCALE TO NEW HEIGHTS

www.allinea.com