allinea

Leaders in parallel software development tools

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

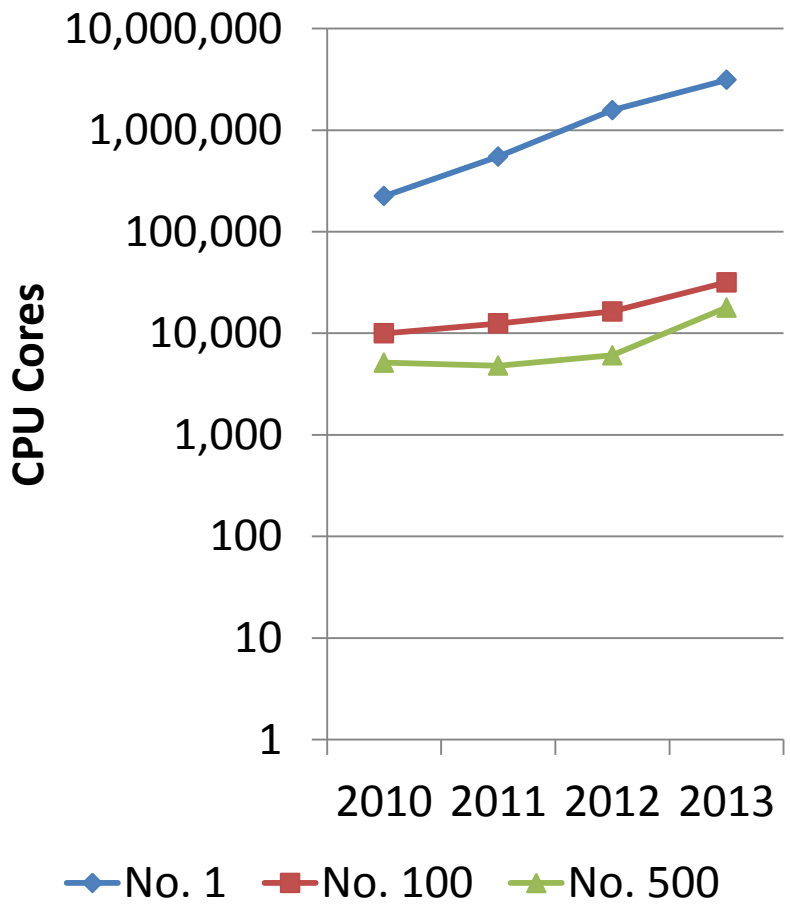
- **Designed for productivity**
  - Consistent integrated easy to use tools
  - Enables effective use of HPC resources and expertise

allinea
www.allinea.com

Major Supercomputing Centers

Oak Ridge National Laboratory
cea
Blue Waters – Sustained Petascale Computing
NERSC – National Energy Research Scientific Computing Center
HLRS
Argonne National Laboratory
NSF
TACC
Berkeley Lab
SHARCNET
University of California Lawrence Livermore National Laboratory
SNIC
Technische Universität Dresden
Los Alamos National Laboratory – Est. 1943
Jülich Forschungszentrum
GENCI
ONERA – The French Aerospace Lab
BSC Barcelona Supercomputing Center – Centro Nacional de Supercomputación
ICHEC – Irish Centre for High-End Computing
CINECA
lrz
Science & Technology Facilities Council
epcc
hpc Cymru Wales
NCHC 財團法人國家實驗研究院 國家高速網路與計算中心 National Center for High-performance Computing
جامعة الملك عبدالعزيز King Abdulaziz University
IFERC
ارامكو السعودية Saudi Aramco
allinea www.allinea.com

Extreme machines are everywhere

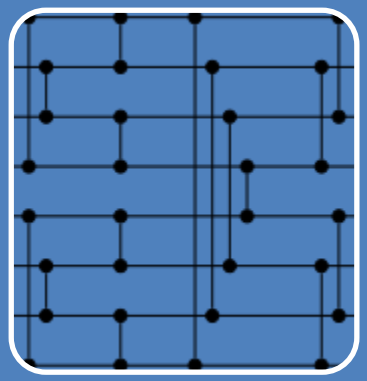


Machine sizes are exploding

Software scale grows as machines grow

allinea
www.allinea.com

Some Software Challenges for the Extreme



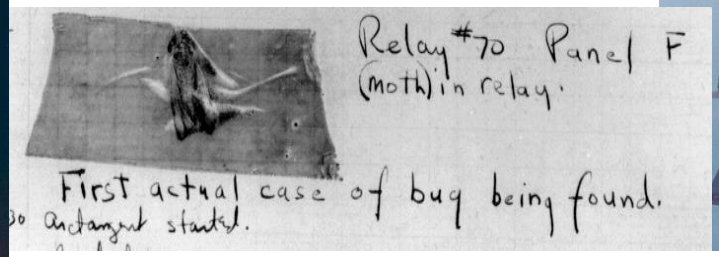
Algorithmic: Compilers are not enough!

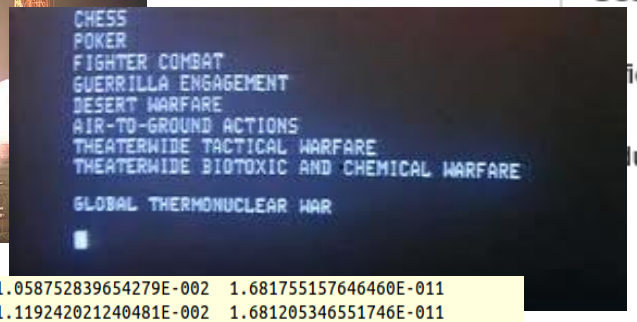
- Restructure for SIMD threads and vectorization
- Fundamental changes: Do we really need FFTs here?
- Rediscover PRAM and 0-1 Sorting Networks(!)

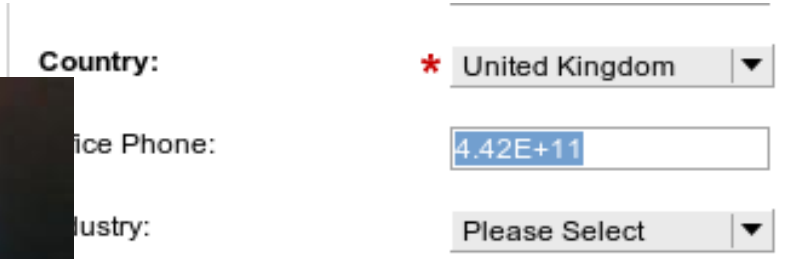
Programmer Efficiency

- MPI alone is not sufficient: Hybrid required
- Performance trade-offs harder to understand
- Software bugs harder to fix

Bugs in Practice







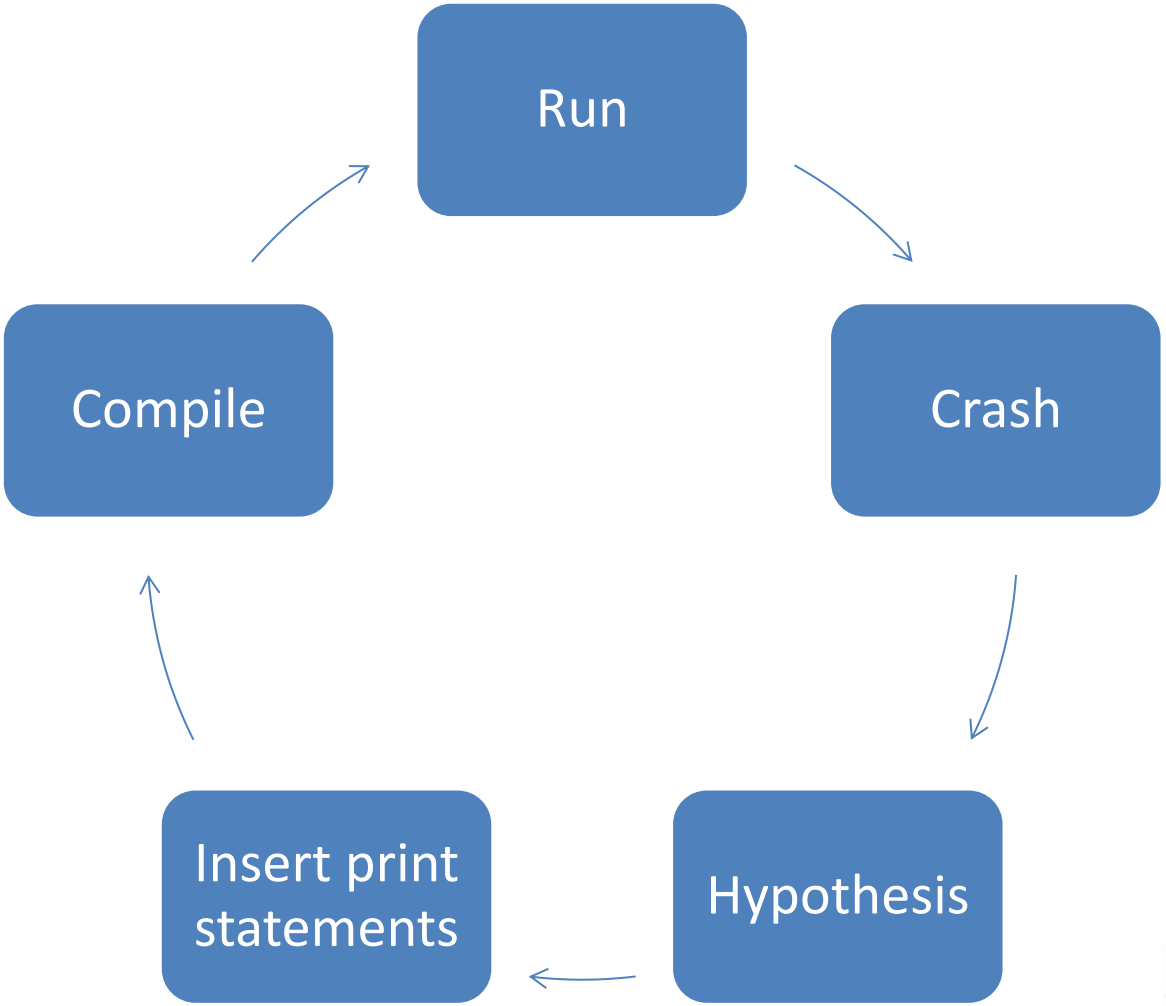

```
   124395.444928040       1.058752839654279E-002  1.681755157646460E-011
   124395.444323148       1.119242021240481E-002  1.681205346551746E-011
   124395.443701451       1.181411574161518E-002  1.680444969505865E-011
   124395.443062951       1.245261508079283E-002  1.679731384893576E-011
   124395.442407647       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

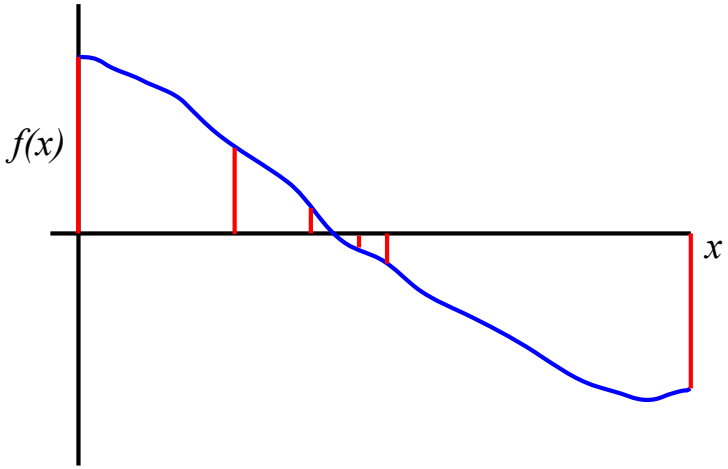
Bohrbug	Steady, dependable bug
Heisenbug	Vanishes when you try to debug (observe)
Mandelbug	Complexity and obscurity of the cause is so great that it appears chaotic
Schroedinbug	First occurs after someone reads the source file and deduces that it never worked, after which the program ceases to work

Debugging in practice…



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

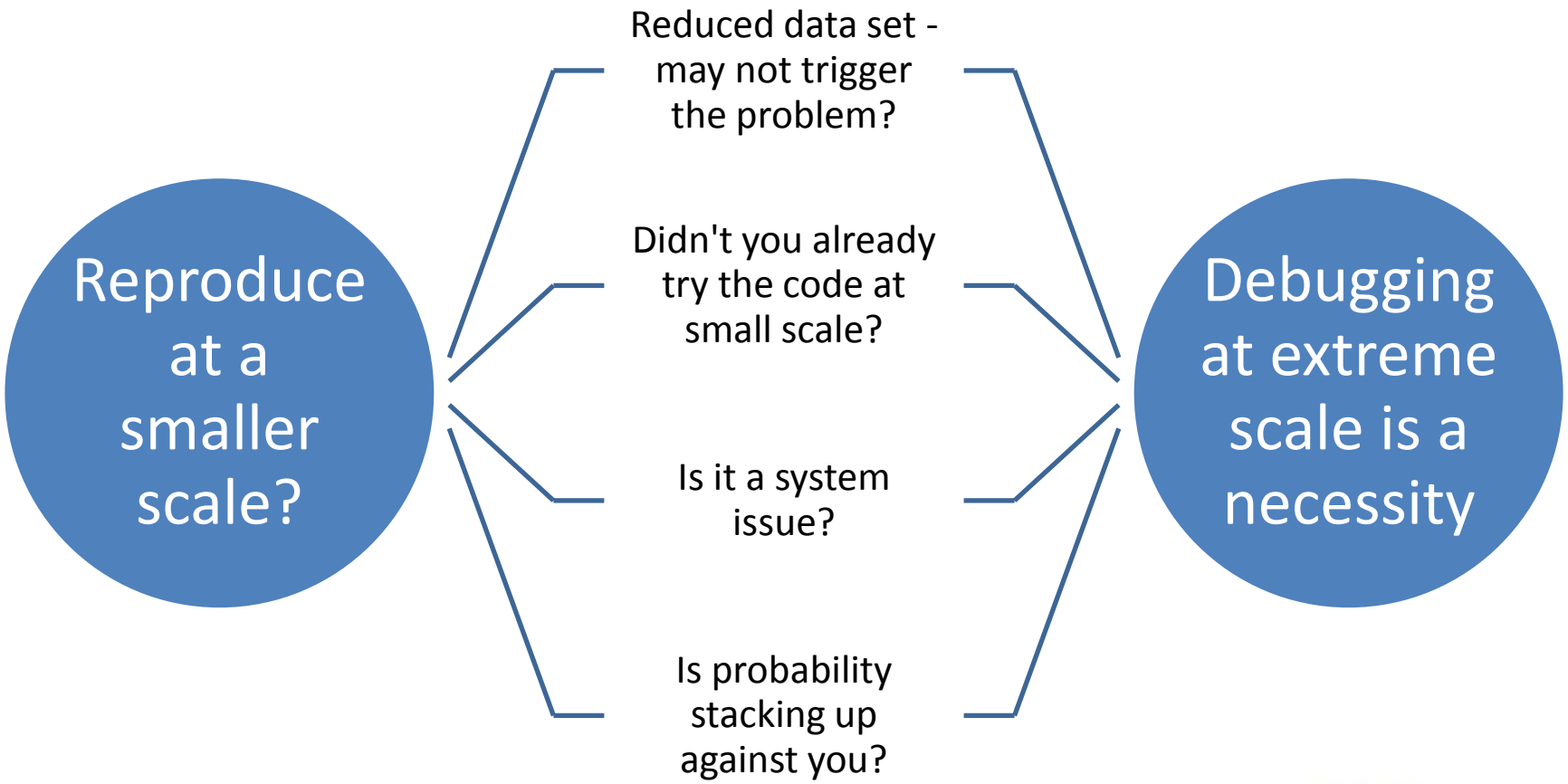
- 18,688 nodes
- 18,688 NVIDIA Kepler K20 GPUs
- 299,008 CPU cores
- 50,233,344 CUDA cores

Mira

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

Does the printf workflow “work”?

Bug fixing as scale increases



Three Challenges for tools

Scalability

- Speed and Simplification

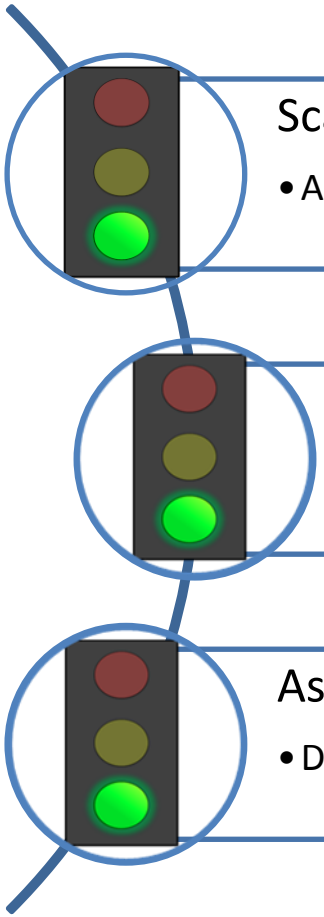
Heterogeneity

- 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

allinea
www.allinea.com

ALCF, OLCF and Allinea deliver

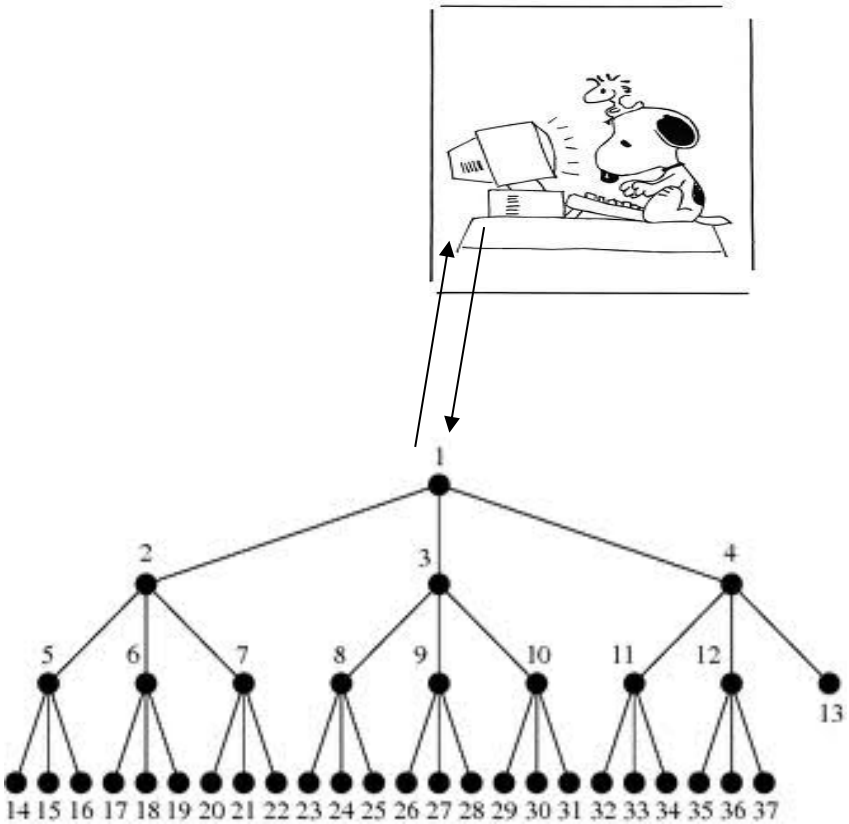
2009 - Allinea and Oak Ridge begin collaboration to provide super-Petascale debugging

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

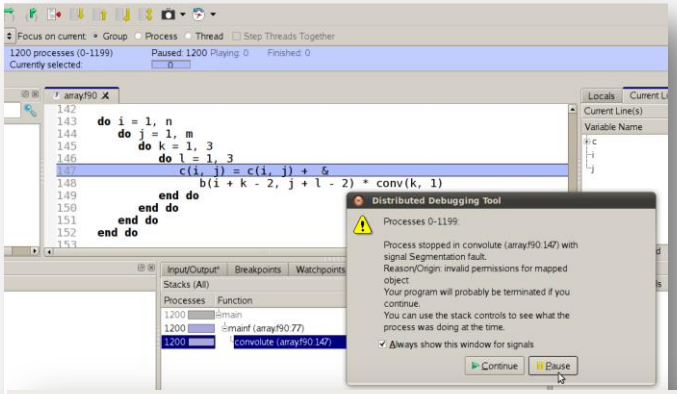
- **Unified interface with Allinea MAP :**
  - One interface eliminates learning curve
  - Spend more time on your results

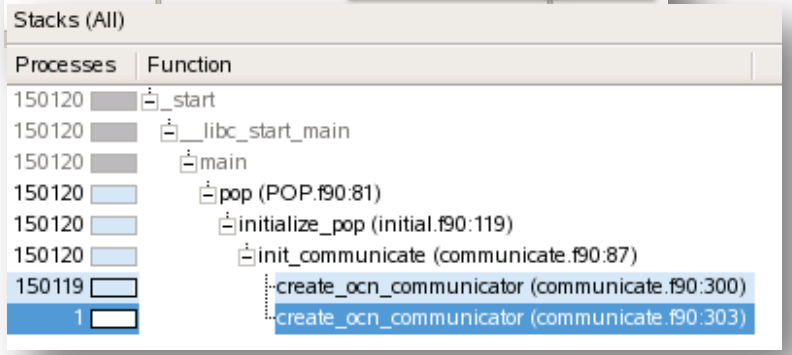
- **Slash your time to debug**
  - Reproduces and triggers your bugs instantly
  - Helps you easily understand where issues come from quickly
  - Helps you to fix them as swiftly as possible

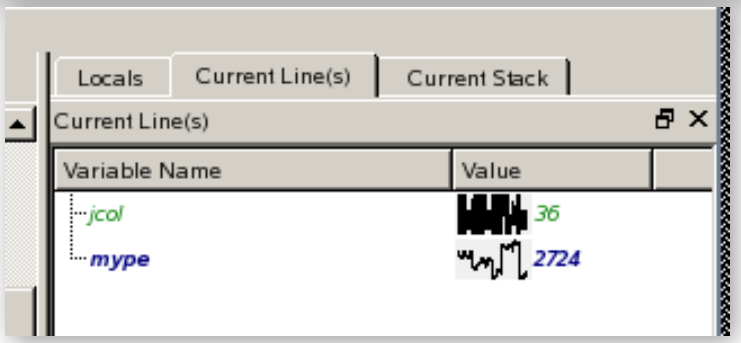
allinea
www.allinea.com

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

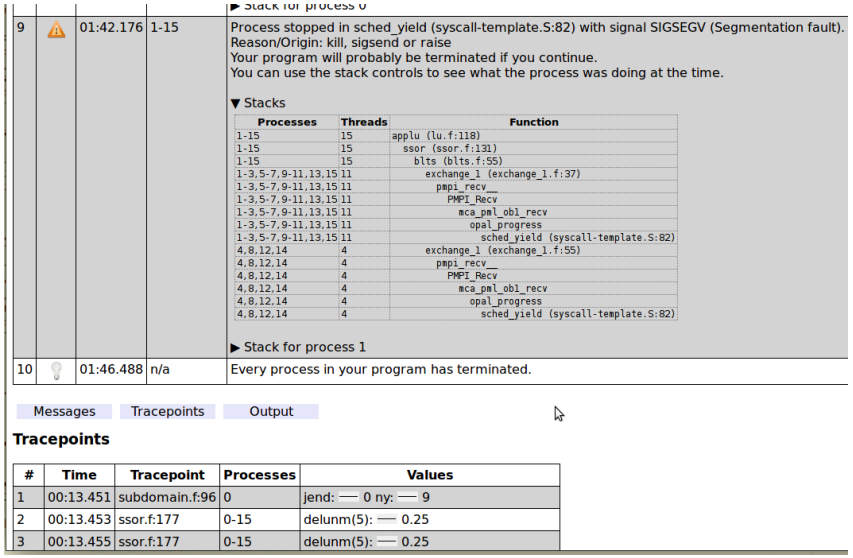
- HPC code fails on 16,384 cores
- Code abandoned – bug couldn't be fixed
- Machine too busy for interactive debugging
- Allinea DDT offline mode runs bug case overnight
- Found error in initialization

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



▶ Stack for process 0

9	⚠	01:42.176	1-15	Process stopped in sched_yield (syscall-template.S:82) with signal SIGSEGV (Segmentation fault). Reason/Origin: kill, sigsend or raise 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. ▼ Stacks

Processes	Threads	Function
1-15	15	applu (lu.f:118)
1-15	15	ssor (ssor.f:131)
1-15	15	blts (blts.f:55)
1-3,5-7,9-11,13,15	11	exchange_1 (exchange_1.f:37)
1-3,5-7,9-11,13,15	11	pmpi_recv__
1-3,5-7,9-11,13,15	11	PMPI_Recv
1-3,5-7,9-11,13,15	11	mca_pml_obl_recv
1-3,5-7,9-11,13,15	11	opal_progress
1-3,5-7,9-11,13,15	11	sched_yield (syscall-template.S:82)
4,8,12,14	4	exchange_1 (exchange_1.f:55)
4,8,12,14	4	pmpi_recv__
4,8,12,14	4	PMPI_Recv
4,8,12,14	4	mca_pml_obl_recv
4,8,12,14	4	opal_progress
4,8,12,14	4	sched_yield (syscall-template.S:82)

▶ Stack for process 1

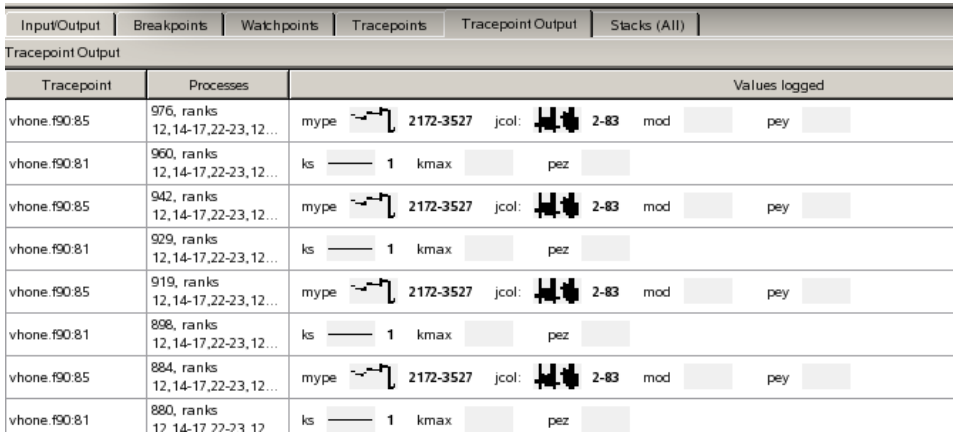
10	💡	01:46.488	n/a	Every process in your program has terminated.

Messages | Tracepoints | Output

Tracepoints

#	Time	Tracepoint	Processes	Values
1	00:13.451	subdomain.f:96	0	jend: — 0 ny: — 9
2	00:13.453	ssor.f:177	0-15	delunm(5): — 0.25
3	00:13.455	ssor.f:177	0-15	delunm(5): — 0.25

Tracepoints



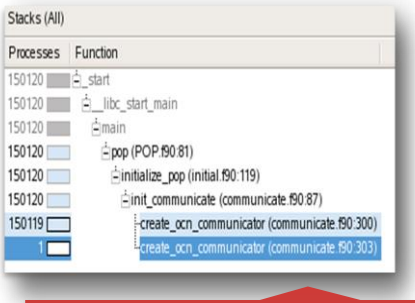
- A scalable print alternative
  - Merged print – with a sparkline graph showing distribution
  - No recompilation required

allinea
www.allinea.com

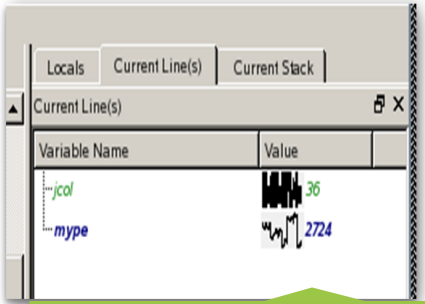
Mira Interlude

- Use offline debugging to full advantage

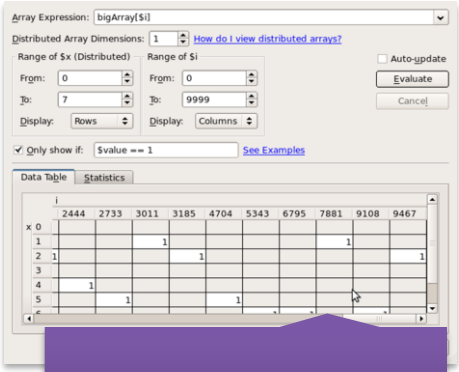
Top 5 features at scale



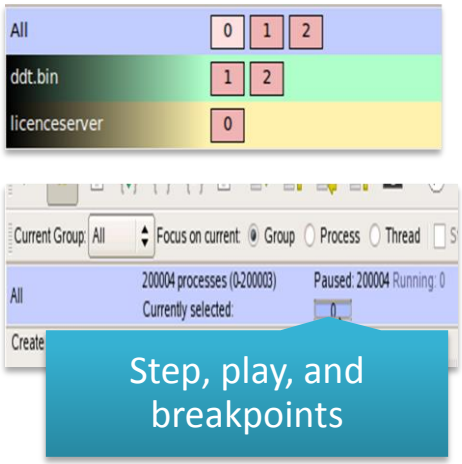
Parallel stack view



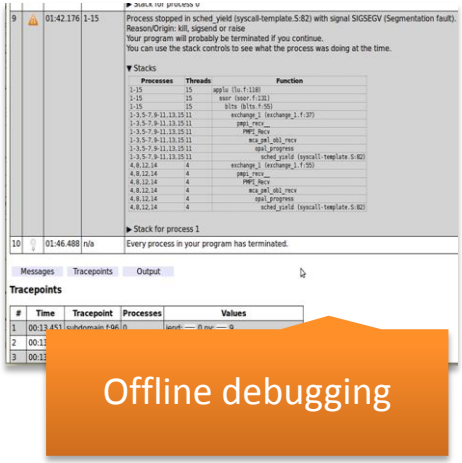
Automated data comparison: sparklines



Parallel array searching



Step, play, and breakpoints



Offline debugging

allinea
www.allinea.com

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

allinea
www.allinea.com