SCALE TO NEW HEIGHTS

SCALE TO NEW HEIGHTS

Simplifying complex software development environments at scale

David Lecomber
CTO Allinea

www.allinea.com

Coming up...

- **Introduction**
  - Allinea and tools in HPC
- **DDT**
  - Brief overview, scalability focus and update
- **OPT**
  - Brief overview and update
- **Questions**

Parallel Software is Complicated

- **Multithreaded, multiprocess code**
  - The usual issues: bugs, speed ...
  - ... now add communication, synchronization, race conditions, deadlock, scalability ….
  - ... unpredictability of behaviour between systems, and within same system
- **Now more complex.. several architectures**
  - Hybrid Cell and Opteron
- **Hybrid GPU and x86_64**
  - Homogeneous heavyweight-kernel clusters
  - Homogeneous lightweight-kernel clusters
  - Large SMP machines
  - Desktop SMP via multicore
- **No clear winner yet**
  - Development nightmare!
  - Can you do it without tools?

Three Challenges Today

- **Languages and compilers**
  - What do I use currently?
  - What will I use on my next system?
- **Making a code that works right**
  - How do I debug today?
  - How will I debug my next system?
- **Making a code that works fast**
  - How do I optimize today?
  - How will I optimize my next system?

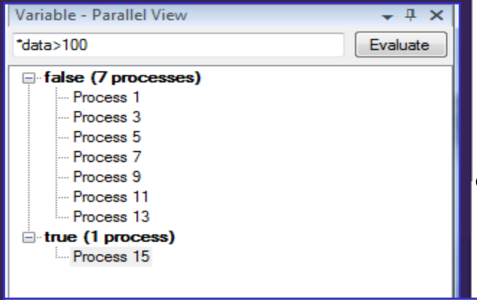
- **Lots of choice for the language – but few for the rest**

Allinea Software

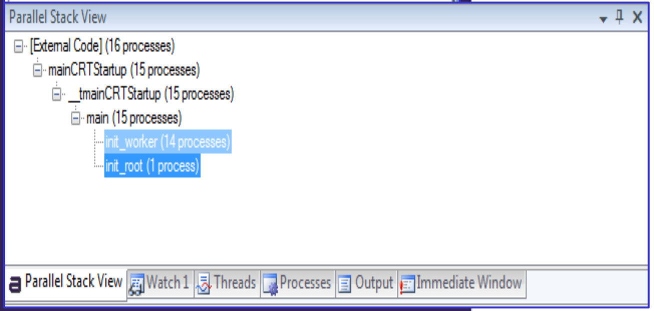
- **HPC tools company since 2001**

- **Core products**
  - **DDT** - Debugger for MPI, threaded/OpenMP and scalar applications
  - **OPT** - Optimizing and profiling tool for MPI and non-MPI applications

- **New product**
  - **DDTLite** - Plugin for Microsoft Visual Studio 2008
    - Adds parallel and multi-threaded components to user interface
    - Real parallel debugging for Windows!
    - Released September 22nd 2008



```
*data>100
false (7 processes)
    Process 1
    Process 3
    Process 5
    Process 7
    Process 9
    Process 11
    Process 13
true (1 process)
    Process 15
```



```
[External Code] (16 processes)
  mainCRTStartup (15 processes)
    __tmainCRTStartup (15 processes)
      main (15 processes)
        init_worker (14 processes)
        init_root (1 process)
```

High Profile Clients (extract)

- **Grids**
  - SHARCNet, ICHEC, North West Grid
- **Universities**
  - FZ Jülich, Karlsruhe, Dresden, HLRS Stuttgart, LRZ Munich
  - Oxford, Cambridge, Warwick, Manchester
  - Vanderbilt, TACC, Michigan, Oregon, Indiana, Penn State, Wisconsin, Alberta
- **Aerospace research**
  - DLR, EADS CCR, CIRA, MBDA, CERFACS, Dassault, BAe Systems
- **Commercial research**
  - Airbus, Fujitsu, CGG, CGG Veritas, Total, IFP, OHM, AVL, MTEM, Intel
- **Research centres**
  - CEA, NERSC, IDRIS, BSC, ONERA, RAL, HLRS, CASPUR, CINECA, NERSC, LLNL
- **Weather/Climate**
  - Met Office, BGS, Proudman, Ifremer, Mercator

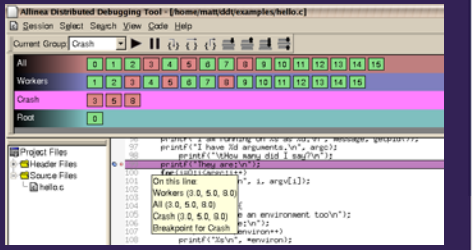
SCALE TO NEW HEIGHTS

DDT - Distributed Debugging Tool

- **A mature, powerful and highly intuitive tool**
  - Traditional focus has been HPC
- **Cross-platform support**
  - Linux, Solaris (Sparc, x86-64), CLE, AIX
  - GNU, Absoft, IBM, Intel, PGI, PathScale, Sun compilers
  - Blue Gene, Cell, x86-64, ia64, Power, UltraSparc, NEC
- **Across all MPI and OpenMP implementations**
  - From low end to high end
- **Support for all scheduling systems**
  - SGE, PBS, LSF, MOAB, ...
  - Flexible, powerful, easy to use queue submission

DDT: Basic Principles

- Sophisticated GUI helps the user to control parallel execution and helps to find and focus on potential problems
- User controls actions by groups..
  - Set breakpoints, lock step, align stacks etc
  - But can focus in on individual threads / processes when necessary
- Create groups both
  - Manually: select processes via drag and drop
  - Automatically: by process stack, values etc


Allinea Distributed Debugging Tool - [/home/matt/ddt/examples/hello.c]
Session Select Search View Code Help
Current Group: Crash
All 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Workers 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
Crash 3 5 8
Root 0
Project Files
Header Files
Source Files
hello.c

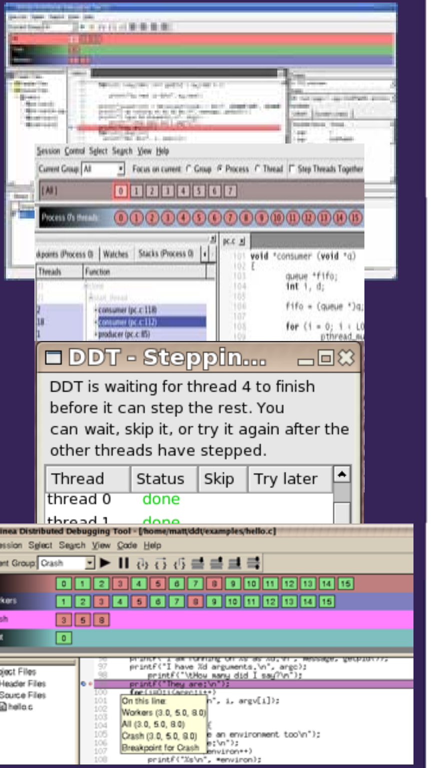
```
    printf("I have %d arguments.\n", argc);
        printf("\tHow many did I say?\n");
    printf("They are:\n");
    for(i=0;i<argc;i++)
                        n", i, argv[i]);


                        {
                        e an environment too\n");
                        e:\n");
                        environ++)
        printf("%s\n", *environ);
```

On this line:
Workers (3.0, 5.0, 8.0)
All (3.0, 5.0, 8.0)
Crash (3.0, 5.0, 8.0)
Breakpoint for Crash

Features for every model

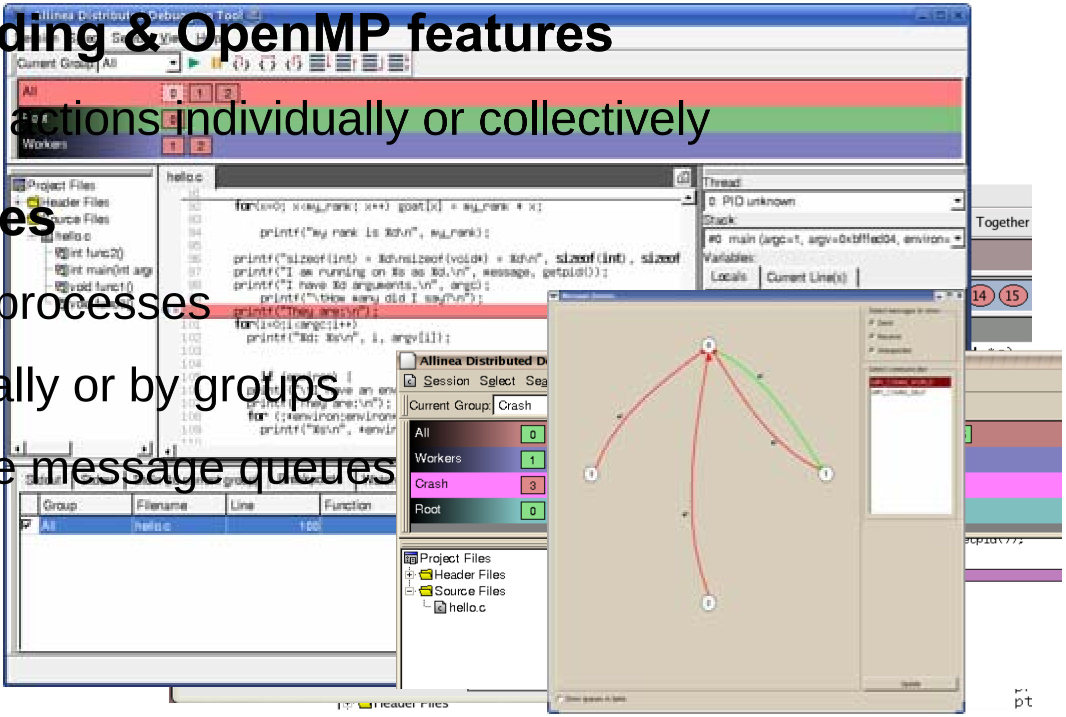
- **Scalar features**
  - Advanced C++ support including STL, namespaces, virtual functions and templates
  - Advanced Fortran 90, 95 and 2003 support including modules, allocatable data, pointers and derived types
- **Multithreading & OpenMP features**
  - Perform actions individually or collectively
- **MPI features**
  - Control processes
  - individually or by groups
  - Visualize message queues



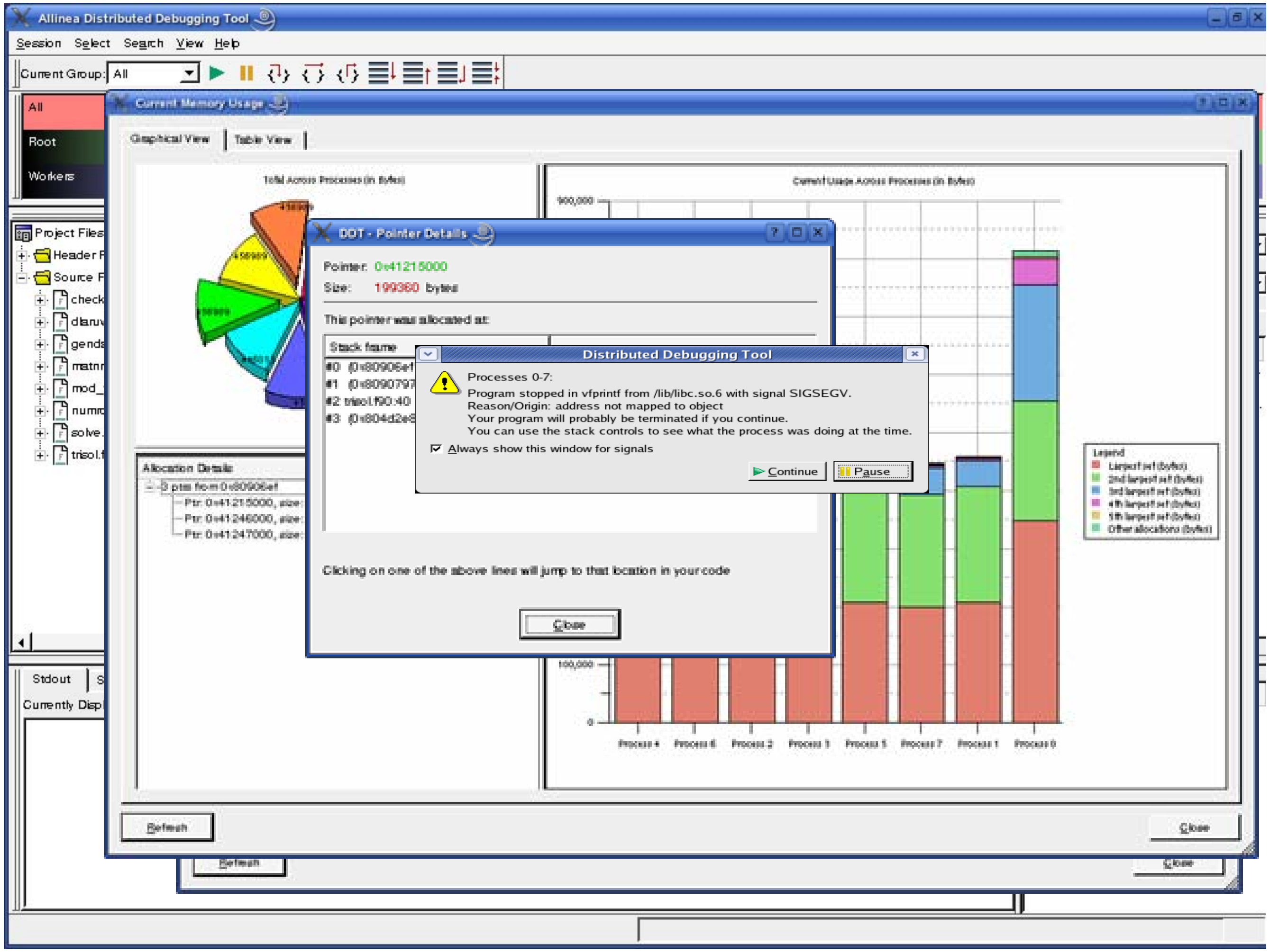
DDT - Steppin...

DDT is waiting for thread 4 to finish before it can step the rest. You can wait, skip it, or try it again after the other threads have stepped.

Thread	Status	Skip	Try later
thread 0	done		
thread 1	done		



Memory Debugging



Pointer: 0x41215000
Size: 199360 bytes

This pointer was allocated at:

```
#0 (0x80906ef
#1 (0x8090797
#2 trisol.f90:40
#3 (0x804d2e8
```

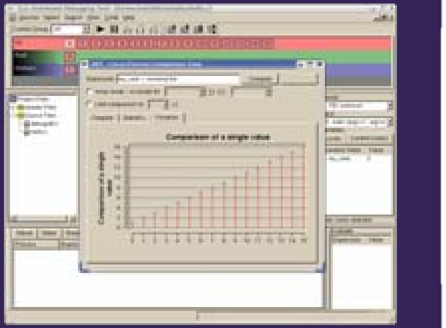
```
3 ptrs from 0x80906ef
  Ptr: 0x41215000, size:
  Ptr: 0x41246000, size:
  Ptr: 0x41247000, size:
```

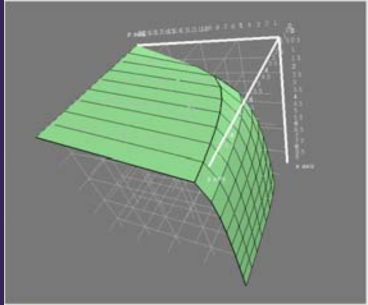
Clicking on one of the above lines will jump to that location in your code

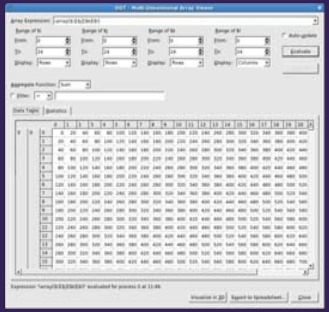
Processes 0-7:
Program stopped in vfprintf from /lib/libc.so.6 with signal SIGSEGV.
Reason/Origin: address not mapped to object
Your program will probably be terminated if you continue.
You can use the stack controls to see what the process was doing at the time.

☑ Always show this window for signals

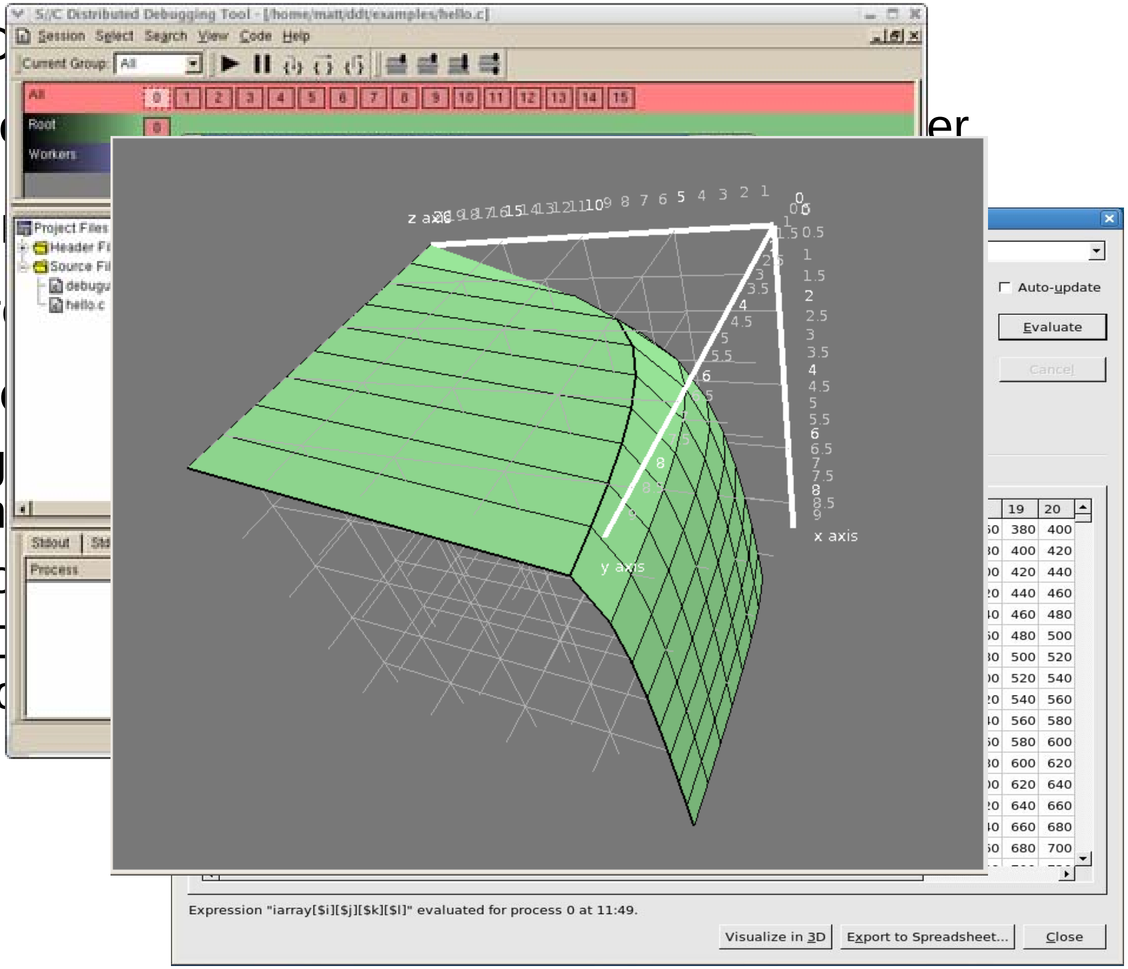
... and more







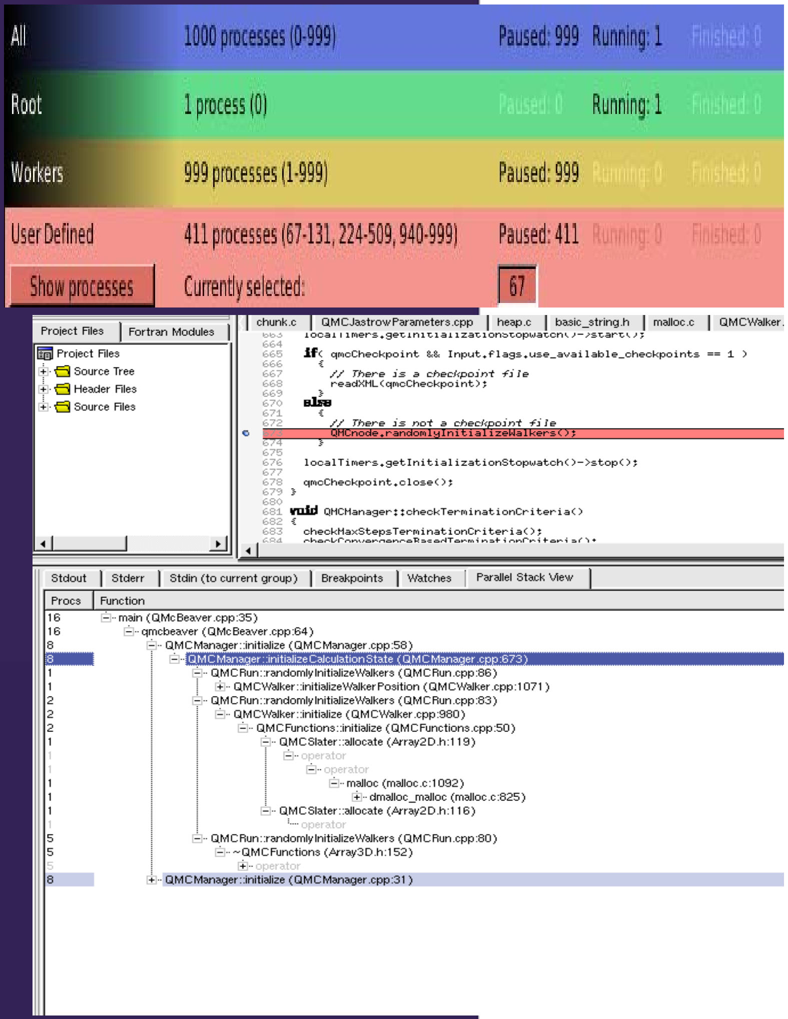
- Cross process/thread comparison
- Visualize multidimensional data
  - 3D
  - Fr er
- Adva
  - Pr
- Singl
  - eg an
  - Co
  - DD mo



19	20
380	400
400	420
420	440
440	460
460	480
480	500
500	520
520	540
540	560
560	580
580	600
600	620
620	640
640	660
660	680
680	700

Expression "iarray[$i][$j][$k][$l]" evaluated for process 0 at 11:49.

Scalable Debugging



All	1000 processes (0-999)	Paused: 999	Running: 1	Finished: 0
Root	1 process (0)	Paused: 0	Running: 1	Finished: 0
Workers	999 processes (1-999)	Paused: 999	Running: 0	Finished: 0
User Defined	411 processes (67-131, 224-509, 940-999)	Paused: 411	Running: 0	Finished: 0
Show processes	Currently selected:	67		

```
  localTimers.getInitializationStopwatch()->start();

  if( qmcCheckpoint && Input.flags.use_available_checkpoints == 1 )
    {
      // There is a checkpoint file
      readXML(qmcCheckpoint);
    }
  else
    {
      // There is not a checkpoint file
      QMCnode.randomlyInitializeWalkers();
    }

  localTimers.getInitializationStopwatch()->stop();

  qmcCheckpoint.close();
}

void QMCManager::checkTerminationCriteria()
{
  checkMaxStepsTerminationCriteria();
  checkConvergenceBasedTerminationCriteria();
```

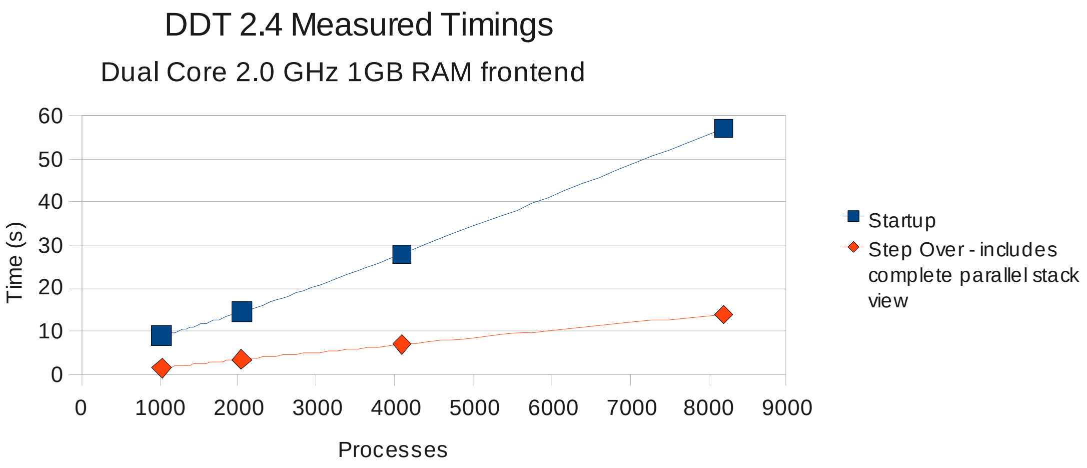
```
Procs  Function
16     ⊟ main (QMcBeaver.cpp:35)
16       ⊟ qmcbeaver (QMcBeaver.cpp:64)
8          ⊟ QMCManager::initialize (QMCManager.cpp:58)
8            ⊟ QMCManager::initializeCalculationState (QMCManager.cpp:673)
1              ⊟ QMCRun::randomlyInitializeWalkers (QMCRun.cpp:86)
1                ⊞ QMCWalker::initializeWalkerPosition (QMCWalker.cpp:1071)
2              ⊟ QMCRun::randomlyInitializeWalkers (QMCRun.cpp:83)
2                ⊟ QMCWalker::initialize (QMCWalker.cpp:980)
2                  ⊟ QMCFunctions::initialize (QMCFunctions.cpp:50)
1                    ⊟ QMCSlater::allocate (Array2D.h:119)
1                      ⊟ operator
1                        ⊟ operator
1                          ⊟ malloc (malloc.c:1092)
1                            ⊞ dmalloc_malloc (malloc.c:825)
1                    ⊟ QMCSlater::allocate (Array2D.h:116)
1                      operator
5              ⊟ QMCRun::randomlyInitializeWalkers (QMCRun.cpp:80)
5                ⊟ ~QMCFunctions (Array3D.h:152)
5                  ⊞ operator
8          ⊞ QMCManager::initialize (QMCManager.cpp:31)
```

- **Control Processes by Groups**
  - Set breakpoints, step, play, stop etc. using user-defined groups of processes
  - Scalable process groups view
- **Parallel Stack View**
  - Finds rogue processes faster
  - Identifies classes of process behaviour easily
  - Allows rapid grouping of processes
- **Parallel Variable View**
  - Find rogue data faster
  - Integrated with process groups

Current Scalability

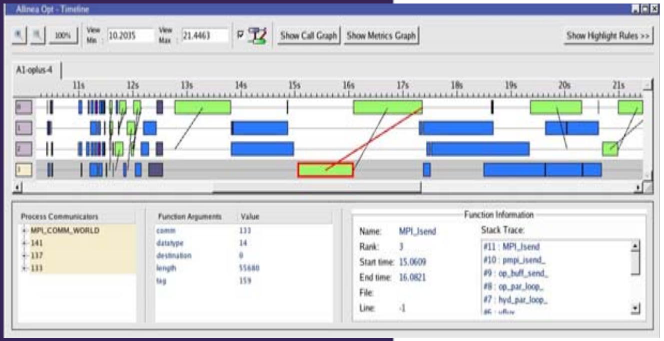
- **What has been achieved in last 12 months?**
- **Scalable GUI**
  - For the first time debug 5,000 with same ease as 100
  - At a glance full stack and status of all processes
- **10x improvement in scale limits**
  - Iterative improvement has brought benefit
  - Debugging 5,000 processes is comfortable
  - Regular users at 4096 cores
  - Test rig emulating 16,000 cores at native speed
- **High end platforms**
  - BlueGene/P support added to list Q3/08
  - Cray XT4, XT5 users at scale
  - Ranger at TACC – Infiniband Sun Constellation cluster

Latest results with DDT



- **Good for all of most of today's systems**
  - Highly parallel architecture has served us well
    - All process debugging done on parallel nodes
    - GUI interprets and displays results
    - Some system architectures better than others..
  - GUI already scales for presentation
    - Permits new tools – eg. plugin MPI checkers
- **Tomorrow: Need to beat linear performance**
  - “Infinitely scalable” performance via multi-level network

Optimizing in a Parallel Universe…



View Min: 10.2035
View Max: 21.4463

- MPI_COMM_WORLD
- 141
- 137
- 133

Function Arguments	Value
comm	133
datatype	14
destination	0
length	55680
tag	159

Name: MPI_Isend
Rank: 3
Start time: 15.0609
End time: 16.0821
File:
Line: -1

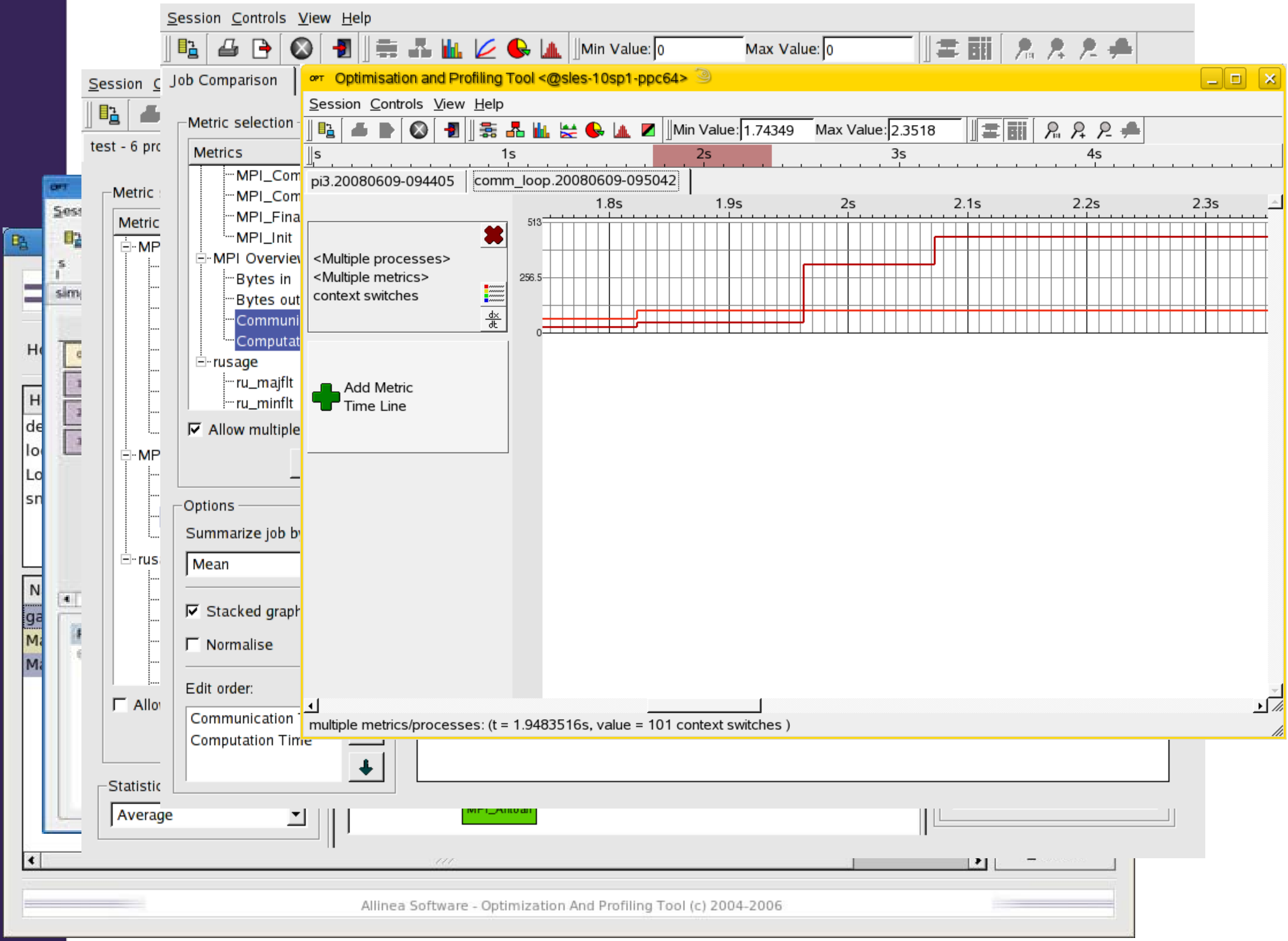
#11 : MPI_Isend
#10 : pmpi_isend_
#9 : op_buff_send_
#8 : op_par_loop_
#7 : hyd_par_loop_
#6 : uflux

- **Traditional tracers**
  - Timelines:
    - Good for watching messages and memory accesses to pick out problems visually
  - But not easily scalable!
- **Can log everything but…**
  - Vast quantities of data are generated
  - Is it really necessary?
  - Analysis becomes an expert task
- **Is MPI the only game in parallel computing?**
  - Of course not...
  - Cell, GPU, desktop multi-core
  - New programming models, new challenges

...Keep It Simple

- **Focus is the key!**
  - Too much visual information can be confusing
  - Good parallel tools should simplify things
  - Tools should target the areas which cause problems
  - Directing the user towards the problem points...
- **Allinea OPT**
  - A 'top-down' focused approach:
    - See the “big picture” first – call graph
    - Drill down successively for more information..
  - Don’t drown users (or system!) in too much data
    - Mixture of sampling and selective tracing
  - Supports most cluster flavours, and IBM/Sony Cell
    - New: IBM BlueGene/P support

OPT – Making Optimization Easier



Min Value: 0 Max Value: 0

test - 6 pro

Min Value: 1.74349 Max Value: 2.3518

pi3.20080609-094405 | comm_loop.20080609-095042

<Multiple processes>
<Multiple metrics>
context switches

multiple metrics/processes: (t = 1.9483516s, value = 101 context switches)

Options

Summarize job by: Mean

☑ Stacked graph

☐ Normalise

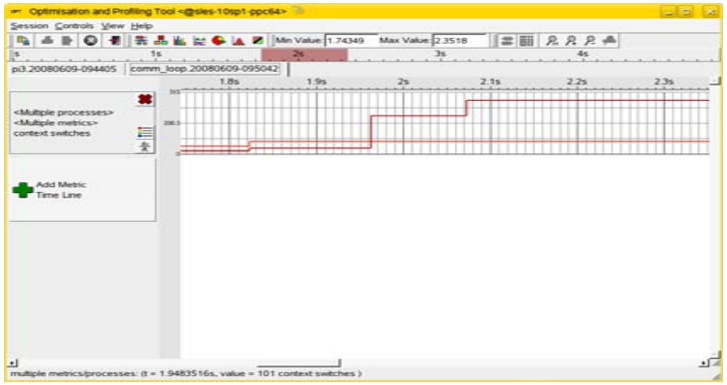
Edit order:
Communication Time
Computation Time

Statistic: Average

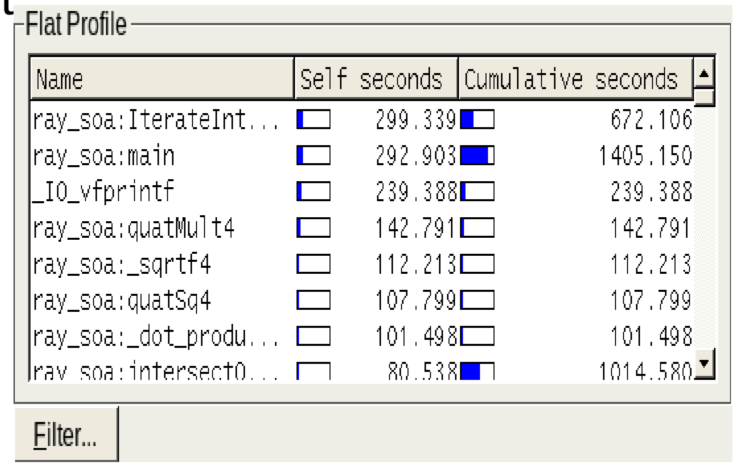
Allinea Software - Optimization And Profiling Tool (c) 2004-2006

Optimizing Hybrids

- **How do we optimize hybrids?**
  - Can traditional products extend to GPU, or Cell?
- **Need to show core behaviour**
  - It's where the most computation happens..
  - No simple “gprof” support yet for most hybrids
- **Need to measure data transfer costs and offloaded op times**
- **OPT now available for Cell**
  - Shows SPU and PPU actual processor usage per function



multiple metrics/processes: (t = 1.9483516s, value = 101 context switches)



Name	Self seconds	Cumulative seconds
ray_soa:IterateInt...	299.339	672.106
ray_soa:main	292.903	1405.150
_IO_vfprintf	239.388	239.388
ray_soa:quatMult4	142.791	142.791
ray_soa:_sqrtf4	112.213	112.213
ray_soa:quatSq4	107.799	107.799
ray_soa:_dot_produ...	101.498	101.498
ray_soa:intersect0...	80.538	1014.580

Summary

- **Architectural complexity is already here**
- **Scale is coming**
  - Top 500 June 2008 – 30 systems with > 10,000 cores
- **Debugging and optimizing at scale**
  - Some problems appear only at scale
  - Need scalable debugging performance
  - Need a scalable GUI: the brain is the bottleneck
- **Yet, we must continue to innovate at a lower scale**
  - Most problems are solved at lower scale – even on the larger systems
  - How many systems are < 10,000 cores?
  - Persistent and reverse debugging MPI and scalar codes in DDT 2.4
- **Our goal**
  - Make picking up a tool easier, more instinctive, than printf
  - Bugs get fixed faster with debuggers

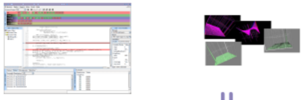
allinea
SCALE TO NEW HEIGHTS

allinea

SCALE TO NEW HEIGHTS

Thank you

allinea ddt
the distributed debugging tool



sales@allinea.com
support@allinea.com

allinea
SCALE TO NEW HEIGHTS

allinea opt
the optimization and profiling tool

sales@allinea.com
support@allinea.com

allinea
SCALE TO NEW HEIGHTS

www.allinea.com