



Debugging and Optimizing Scalable Applications on the Cray

Chris Gottbrath, Principal Product Manager
May 2nd, 2012



Agenda

- **Who are Rogue Wave?**
- **High Performance Computing Challenges**
- **Debugging: TotalView Updated for the Cray Platform**
- **Optimization: Introduction to ThreadSpotter**
- **Future Development Tool Needs**

Rogue Wave Today

The largest independent provider of cross-platform software development tools and embedded components for the next generation of HPC applications.



- **History**

- Founded: 1989
- Acquired by Battery Ventures: 2007
- Acquired:
 - Visual Numerics: 2009
 - TotalView Technologies: 2010
 - Acumem: 2010

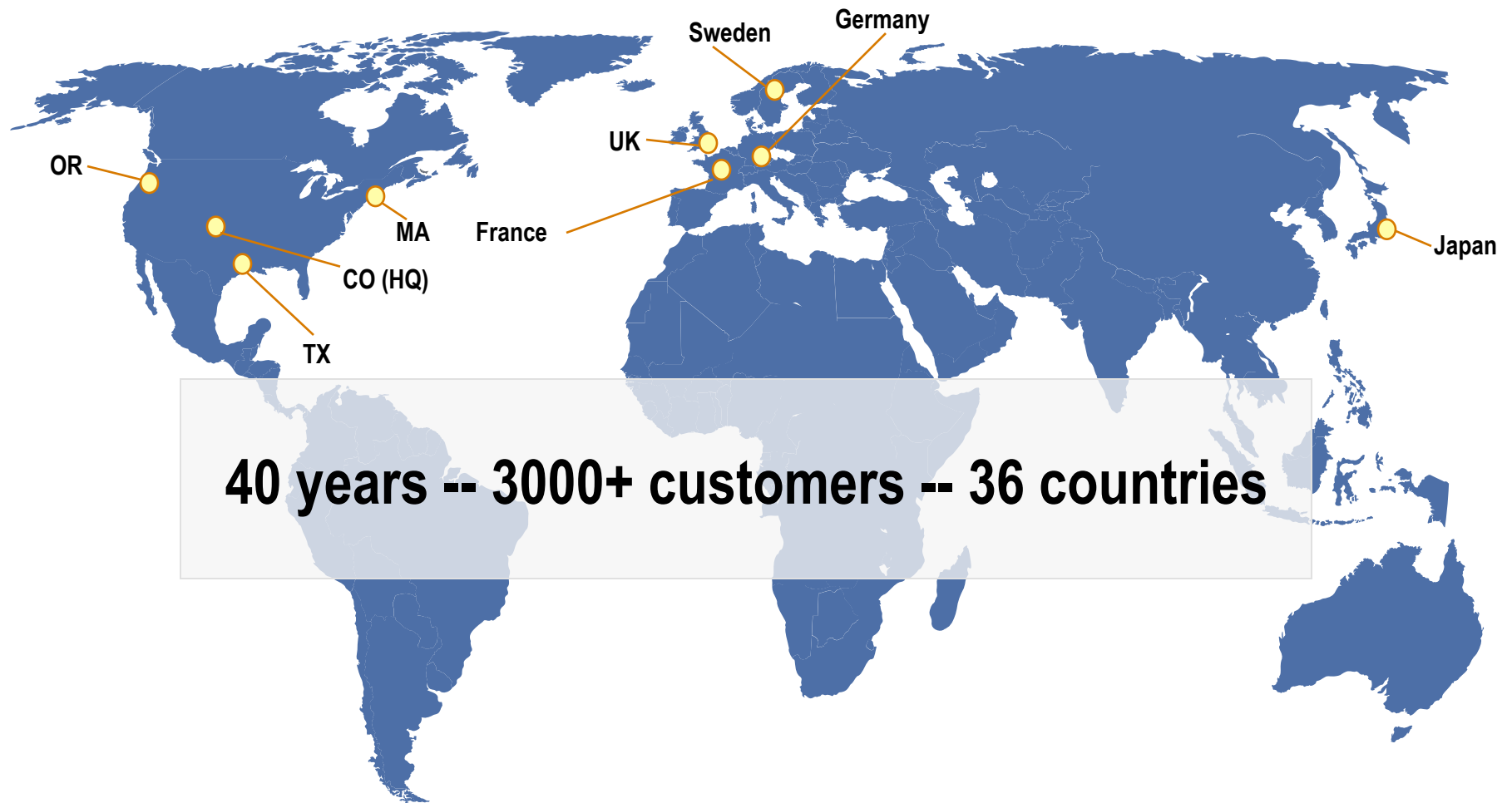
- **Pioneers in C++/object-oriented development**
- **Leading the way in cross-platform, parallel development**

- **Customers**

- 3,000+ in 36 countries
- Financial services, telecoms, oil and gas, government and aerospace, research and academic



Global Reach



Representative Customers



TotalView and Cray give developers the tools they need for application development



Agenda

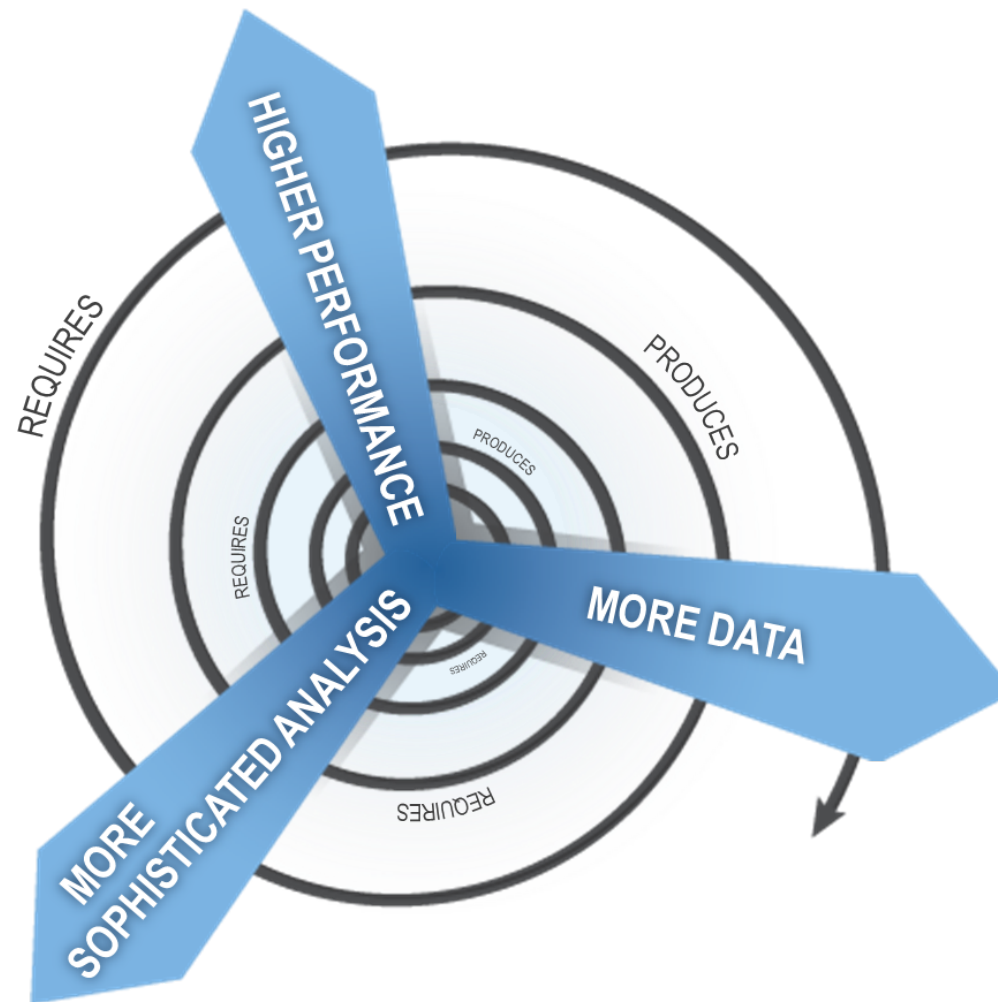
- **Who are Rogue Wave?**
- **High Performance Computing Challenges**
- **Debugging: TotalView Updated for the Cray Platform**
- **Optimization: Introduction to ThreadSpotter**
- **Future Development Tool Needs**

Challenges of HPC Software Development

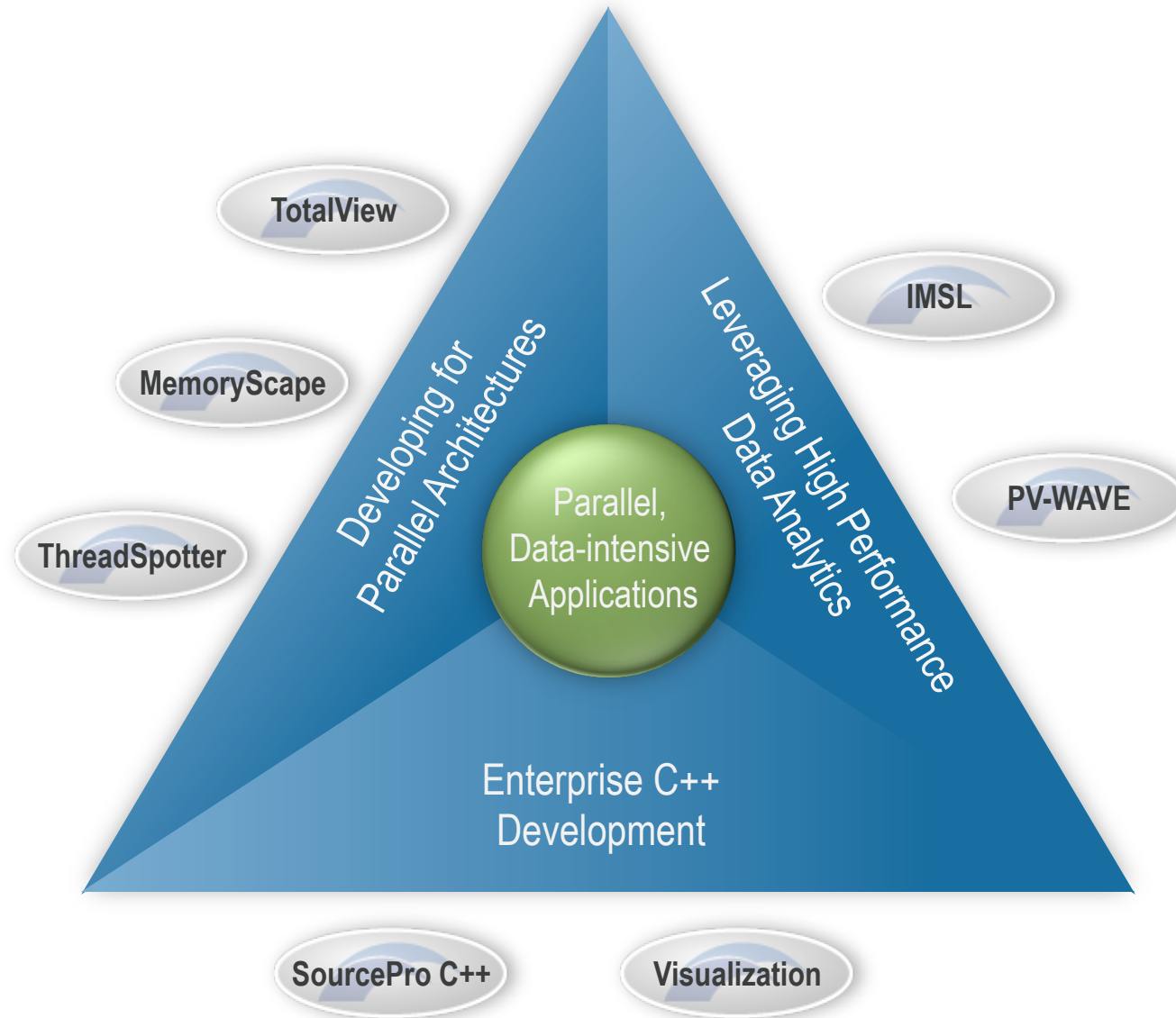
- **Large Scale Clusters with Many-Core**
 - Massive number of total cores
 - Hybrid (MPI + Threading)
 - Balance (memory per core, bandwidth per core)
- **Accelerators**
 - Fundamentally heterogeneous architecture
- **Bandwidth bottlenecks**
- **Rapid development in paradigms, languages and runtimes**
- **Code complexity**
 - Collaboration
 - Software engineering and maintainability
- **Rising cost of system operations (power, cooling)**
- **Increasing diversity of users**

High Performance Computing

Scalability: A Vicious/Virtuous Cycle



Rogue Wave Solution Portfolio



Agenda

- **Who are Rogue Wave?**
- **High Performance Computing Challenges**
- **Debugging: TotalView Updated for the Cray Platform**
- **Optimization: Introduction to ThreadSpotter**
- **Future Development Tool Needs**

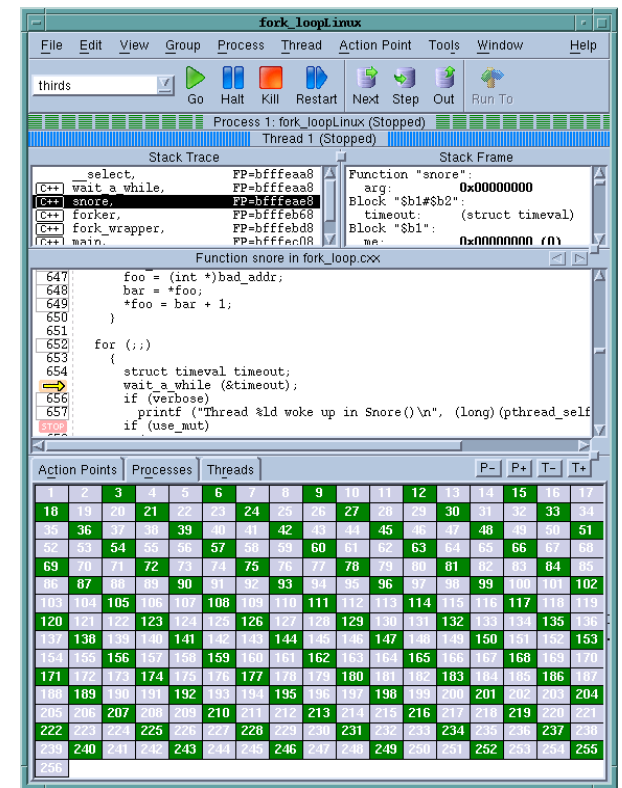
What is TotalView?

- **Application Analysis and Debugging Tool: Code Confidently**

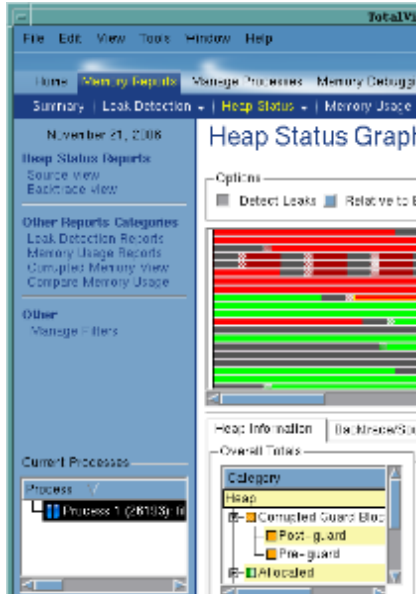
- Debug and Analyze C/C++ and Fortran on Linux, Unix or Mac OS X
- Laptops to supercomputers (BG, Cray)
- Makes developing, maintaining and supporting critical apps easier and less risky

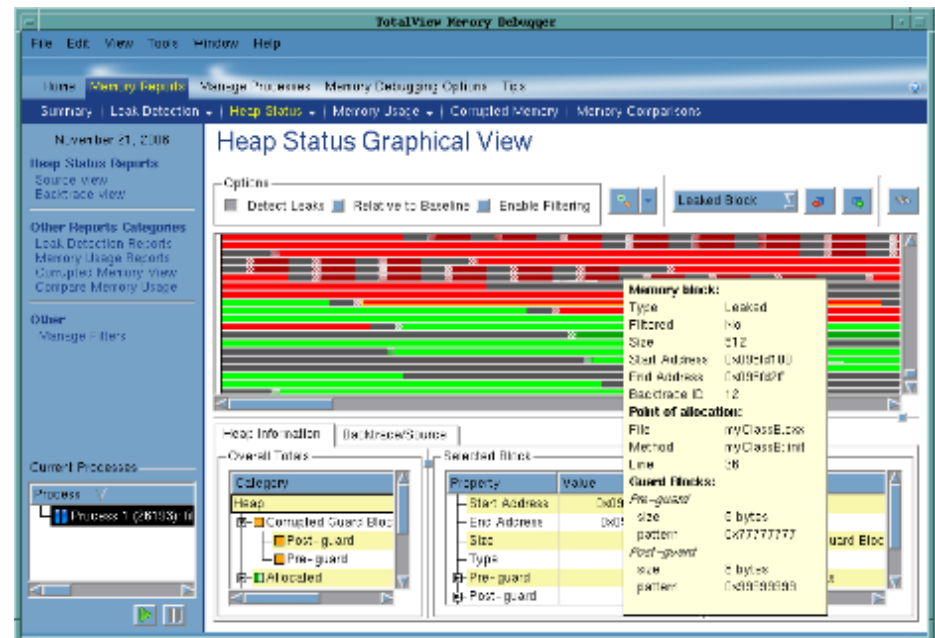
- **Major Features**

- Easy to learn graphical user interface with data visualization
- Parallel Debugging
 - MPI, Pthreads, OpenMP, GA, UPC
 - CUDA and OpenACC (early access)
- Includes a Remote Display Client freeing you to work from anywhere
- Memory Debugging with MemoryScope
- Deterministic Replay Capability Included on Linux/x86-64
- Non-interactive Batch Debugging with TVScript and the CLI
- TTF & C++View to transform user defined objects



What Is MemoryScape?

- **Runtime Memory Analysis : Eliminate Memory Errors**
 - Detects memory leaks *before* they are a problem
 - Explore heap memory usage with powerful analytical tools
 - Use for validation as part of a quality software development process
 - **Major Features**
 - Included in TotalView, or Standalone
 - Detects
 - Malloc API misuse
 - Memory leaks
 - Buffer overflows
 - Supports
 - C, C++, Fortran
 - Linux, Unix, and Mac OS X
 - MPI, pthreads, OMP, and remote apps
 - Low runtime overhead
 - Easy to use
 - Works with vendor libraries
 - No recompilation or instrumentation
- 
- The screenshot shows the TotalView application window. The title bar says 'TotalView'. The menu bar includes 'File', 'Edit', 'View', 'Tools', 'Window', and 'Help'. Below the menu bar is a toolbar with icons for 'Home', 'View by Heap file', 'Manage Processes', 'Memory Debugger', 'Summary', 'Link Detection', 'Heap Status', and 'Memory Usage'. The main window is divided into several panes. On the left, there's a 'Heap Status Reports' section with links for 'Source view' and 'Backtrace view'. Below that is 'Other Reports: Categories' with links for 'Link Detection Reports', 'Memory Usage Reports', 'Compiled Memory view', and 'Compare Memory Usage'. Further down is 'Other' with a 'Manage Filters' link. At the bottom left is 'Current Processes' showing 'Process 1 (20113)'. The right pane is titled 'Heap Status Graph' and contains an 'Options' section with 'Detect Leaks' and 'Relative to' checkboxes. Below the options is a graph showing memory usage over time with red and green bars. At the bottom right is 'Heap Information' and 'Overall Times' sections. The 'Overall Times' section shows a tree view with 'Category' and 'Heap' nodes, with sub-nodes for 'Committed Guard Block', 'Post-guard', 'Pre-guard', and 'Allocated'.



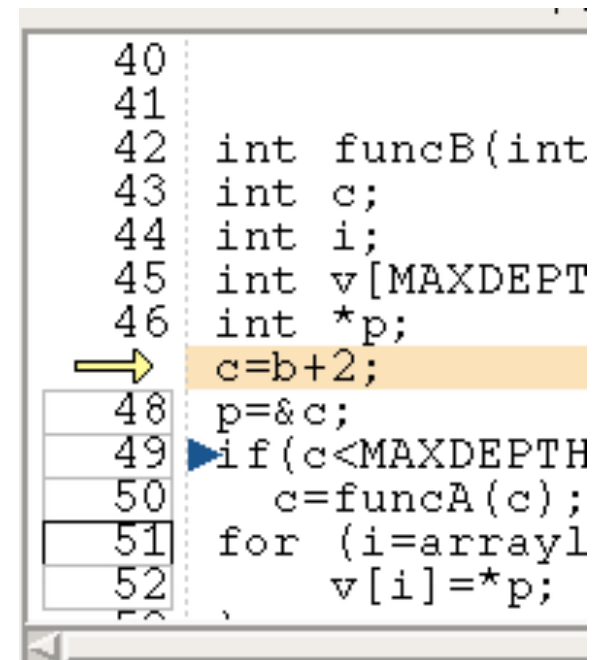
Deterministic Replay Debugging

- **Reverse Debugging: Radically simplify your debugging**

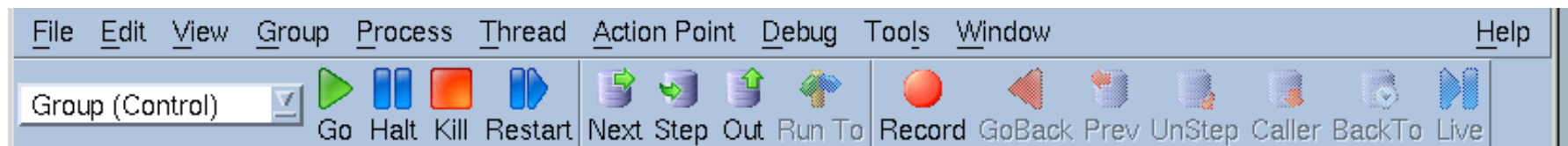
- Captures and Deterministically Replays Execution
 - Not just “checkpoint and restart”
- Eliminate the Restart Cycle and Hard-to-Reproduce Bugs
- Step Back and Forward by Function, Line, or Instruction

- **Specifications**

- A feature included in TotalView on Linux x86 and x86-64
 - No recompilation or instrumentation
 - Explore data and state in the past just like in a live process, including C++View transformations
- Replay on Demand: enable it when you want it
- Supports MPI on Ethernet, Infiniband, Cray XE Gemini
- Supports Pthreads, and OpenMP



```
40
41
42 int funcB(int
43 int c;
44 int i;
45 int v[MAXDEPT
46 int *p;
47 c=b+2;
48 p=&c;
49 if(c<MAXDEPTH
50 c=funcA(c);
51 for (i=array1
52 v[i]=*p;
```



New Capabilities in TotalView 8.10

- **CUDA 4.1**
- **Visual Dive Indicator**
- **Cray-specific enhancements**
 - Improved Cray Compiler Edition Support
 - Replay on Cray XE
 - CUDA support on Cray XK
 - Early Access Preview for OpenACC on the Cray with CCE 8
- **Reverse Debugging**
 - Replay on Demand
 - C++View and ReplayEngine interoperability
- **TVScript Scalability Improvements**

```
30 |
31 |
32 |
33 | int funcA(int a){
34 |     int b;
35 |     b=a+2;
36 |     b=functB(b);
37 |     return b;
38 | }
39 |
40 | .. - .. ..
```



```

n15765@vista:~$ gcc -x cuda -c ACC_tests/ftn_test.c
... Done.
Mapping 581 bytes of ELF string data from '@TEMP@CUDA@man.8df39e39'...done
Digesting 77 ELF symbols from '@TEMP@CUDA@man.8df39e39'...done
Library @TEMP@CUDA@man.49d7eed, with 1 asects, was linked at 0x00003d5, and i
nitially loaded at 0xff000000a3c43100
INFO: Copying library "@TEMP@CUDA@man.49d7eed" into the local file cache ...
... Done.
Mapping 13 bytes of ELF string data from '@TEMP@CUDA@man.49d7eed'...done
Digesting 11 ELF symbols from '@TEMP@CUDA@man.49d7eed'...done
Library @TEMP@CUDA@man.54051f77, with 5 asects, was linked at 0x0000124c, and i
nitially loaded at 0xff000000a3c43200
INFO: Copying library "@TEMP@CUDA@man.54051f77" into the local file cache ...
... Done.
Mapping 167 bytes of ELF string data from '@TEMP@CUDA@man.54051f77'...done
Digesting 47 ELF symbols from '@TEMP@CUDA@man.54051f77'...done
Indexing 2040 bytes of DWARF '.debug_frame' symbols from '@TEMP@CUDA@man.54051f
77'...done
Skimming 1611 bytes of DWARF '.debug_info' symbols from '@TEMP@CUDA@man.54051f7
7'...done
Incorporating 1600 bytes of DWARF '.debug_info' information for man.f90.i (linen
umber)...done
Incorporating 1600 bytes of DWARF '.debug_info' information for man.f90.i (sybo
ls)...done
0

```

TotalView 8X.10.0-6A

ID	Rank	Host	Status	Description
1	<local>		T	aprun (2 active threads)
2	0 nid00130		T	aprun<man>:0 (3 active threads)

b - test_openacc_\$ck_L11_1 - 2-1

Expression: b Address: 0x200361c00

Slice: (:) Filter:

Type: INTEGER*4(100000), @global

Field	Value
(1)	1 (0x00000001)
(2)	2 (0x00000002)
(3)	3 (0x00000003)
(4)	4 (0x00000004)
(5)	5 (0x00000005)
(6)	6 (0x00000006)
(7)	7 (0x00000007)
(8)	8 (0x00000008)
(9)	9 (0x00000009)

aprun<man>:0

File Edit View Group Process Thread Action Point Debug Tools Window Help

Group (Control) Go Halt Kill Restart Next Step Out Run To Record GoBack Prev UnStep Caller BackTo Live

Physical Device: 0 SM: 0 Warp: 0 Lane: 0

Rank 0: aprun<man>:0 (Stopped)

Thread -1 (<<<(0,0,0),(0,0,0)>>>): @TEMP@CUDA@man.54051f77 (Stopped) <Trace Trap>

Stack Trace

Function	Stack Frame
[f90] test_openacc_\$ck_L11_1, PP=fffc0	Function "...nacc_\$ck_L11_1<<<(782,1,1),(128,1,1)>>> Device: 0/1 SM/WP/LN: 0/0/0 of 16/48/32 \$\$arg_ptr_acc_a_t16: 8593080320 (0x0000000200300c \$\$arg_ptr_acc_b_t17: 8593480704 (0x0000000200361c \$\$arg_ptr_acc_c_t18: 8594128896 (0x0000000200400c Block "\$b1": j: 1 (0x00000001) a: (INTEGER+4(100000) @global) b: (INTEGER+4(100000) @global) c: (INTEGER+4(100000) @global)

Function test_openacc_\$ck_L11_1 in man.f90

```

1 PROGRAM test_openacc
2 IMPLICIT NONE
3 INTEGER, PARAMETER :: M=100000
4 INTEGER :: a(M),b(M),c(M)
5 INTEGER :: j,total,expected
6
7 !!$ For simple cases, use parallel loop as a shortcut for
8 !!$ parallel and loop
9 !!$ Set a,b,c
10
11 !$acc parallel loop
12 DO j = 1,M
13   a(j) = j
14   b(j) = j
15   c(j) = j
16 ENDDO
17 !$acc end parallel loop
18
19 !!$ Set b, copy it to host
20 !$acc parallel copyout(b)
21 !$acc loop
22 DO j = 1,M
23   b(j) = j
24 ENDDO
25 !$acc end loop
26 !$acc end parallel
27
28 !!$ Set c, copy it to host
29 !$acc parallel copyout(c)
30 !$acc loop
31 DO j = 1,M
32   c(j) = -j
33 ENDDO
34 !$acc end loop
35 !$acc end parallel

```

Action Points Processes Threads

STOP	1	man.f90#14	test_openacc_\$ck_L11_1+0x1f8
STOP	2	man.f90#23	test_openacc_\$ck_L20_2+0x118
STOP	3	man.f90#32	test_openacc_\$ck_L29_3+0x118

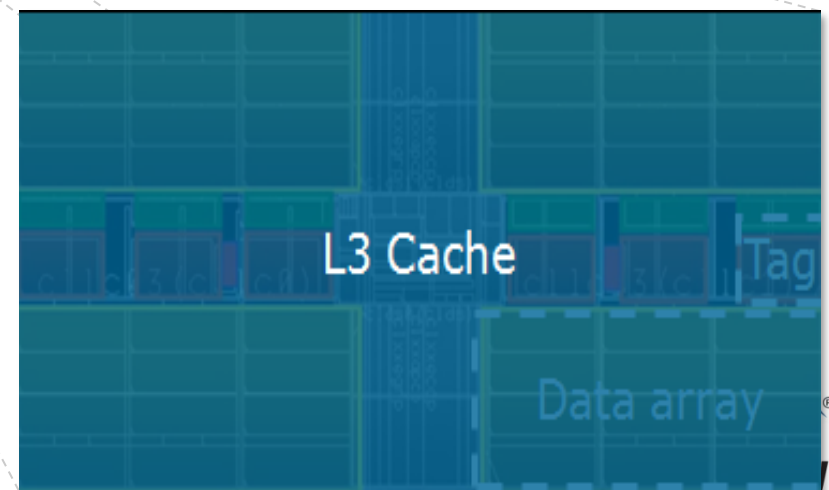
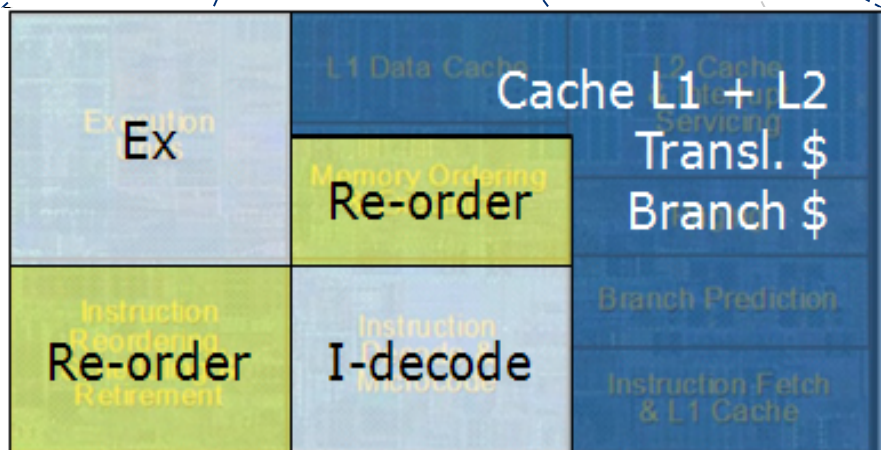
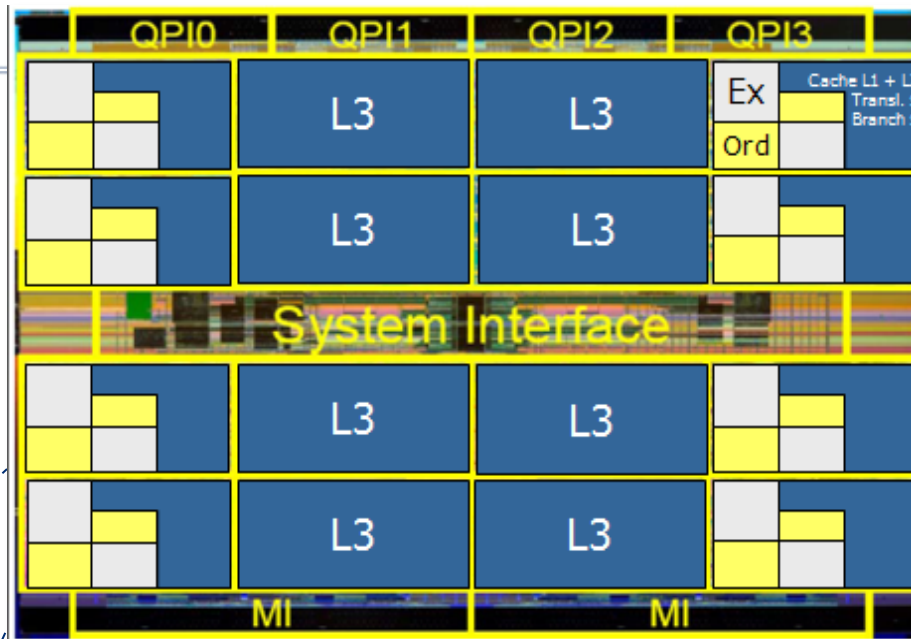
Coming Soon

- **Massive Scalability**
 - Collaboration with LLNL and Tri-lab partners
 - Targeting the Cielo, Sequioa and Linux Clusters
- **Shiny new GUI**
 - Sleek, Modern and Fast
 - Configurable
 - Improved Usability
 - Provides aggregation capabilities for big data and scale
 - Leveraging math and stat expertise from IMSL
- **Intel MIC**
 - Demo at ISC
- **Talk to us about participation in early access programs**

Agenda

- **Who are Rogue Wave?**
- **High Performance Computing Challenges**
- **Debugging: TotalView Updated for the Cray Platform**
- **Optimization: Introduction to ThreadSpotter**
- **Future Development Tool Needs**

How is the silicon used (i7-Ex)?



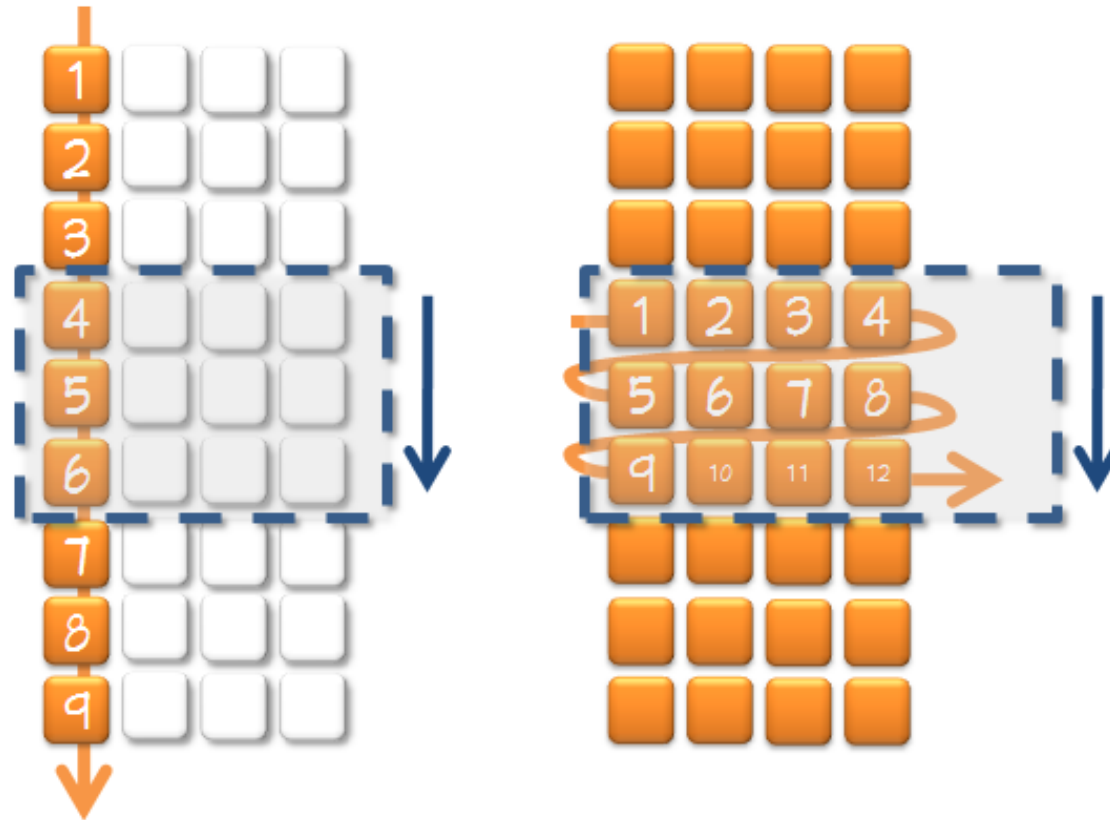
A rule of thumb

Memory system level	Relative latency
L1 cache	1x
Higher cache levels	10x
Main memory	100x

Source: AMD, Michael Wall

Inefficient Loop Nesting

Explanation



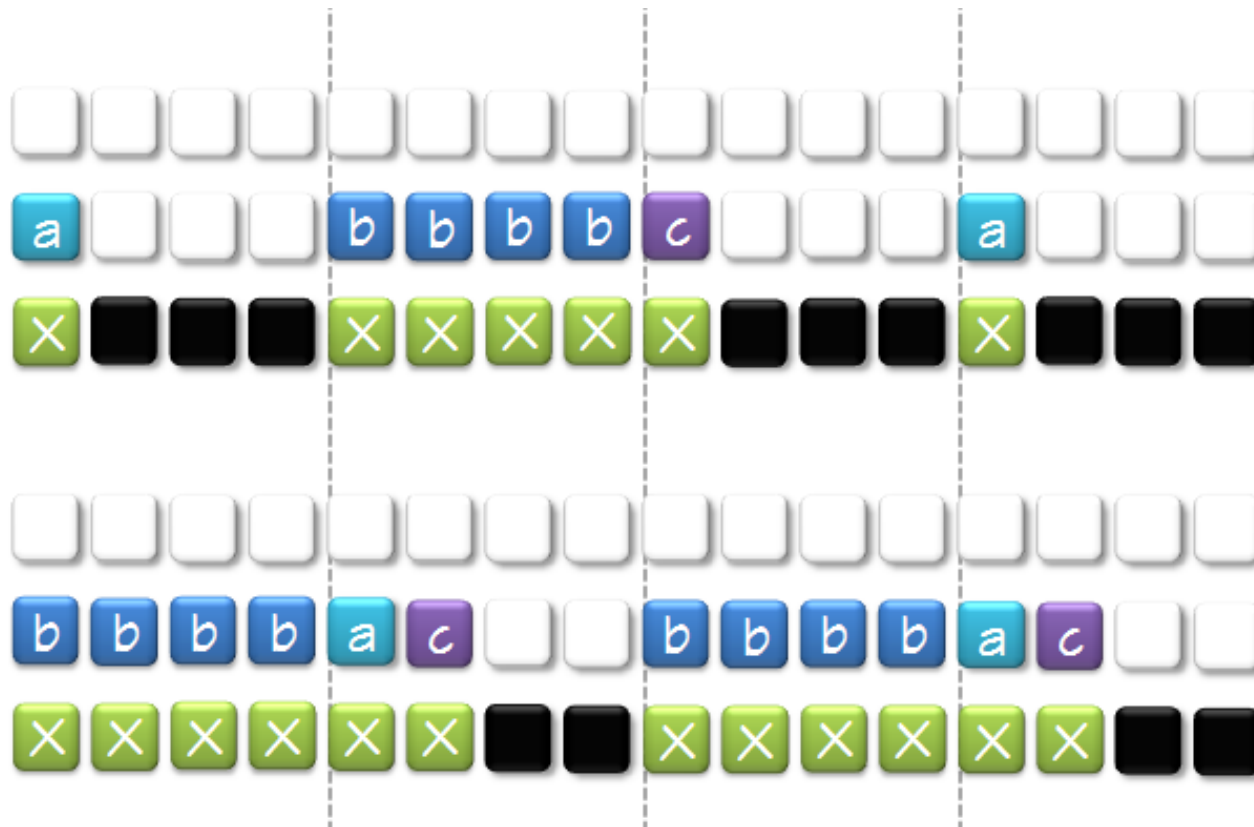
Partially Used Structure

Explanation



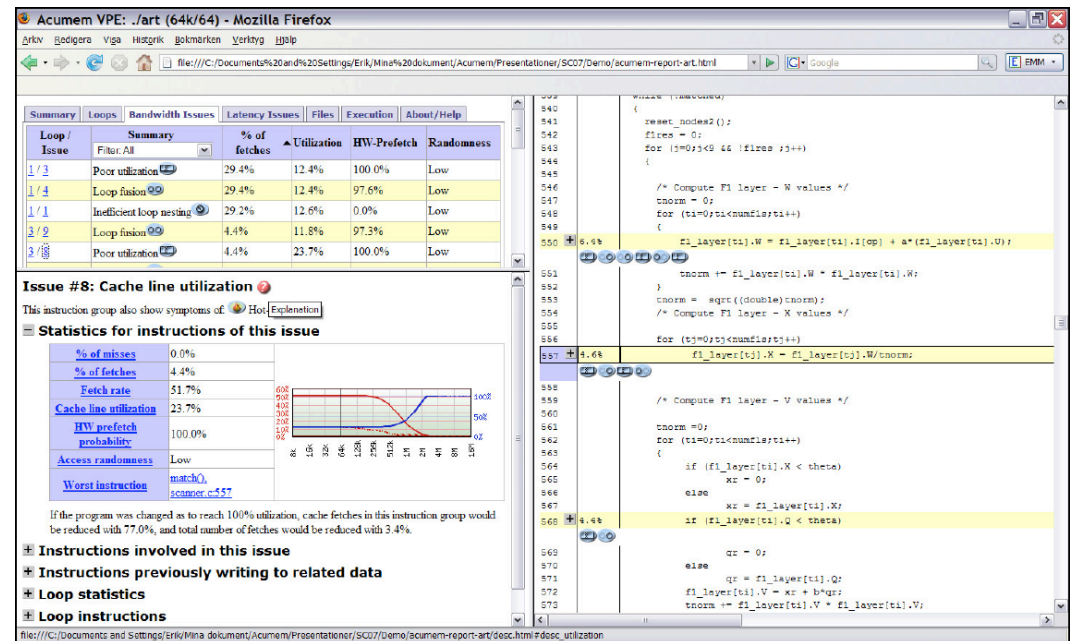
Alignment Problem

Explanation



What is ThreadSpotter?

- **Runtime Cache Performance Optimization Tool: Tune into the Multi-Core Era**
 - Realize More of the Performance Offered by Multi/Many-Core Chips
 - Quickly Detects and Prioritizes Issues -- and then Provides Usable Advice!
 - Brings Cache Performance Into Reach for Every Developer
 - Makes Experienced Cache Optimizers Hyper-Efficient
- **Features**
 - Supports Linux x86/x86-64
 - Any compiled code
 - Runtime Analysis
 - Low overhead
 - Cache Modeling
 - Prioritizes Issues
 - Identifies Problem Lines of Code
 - Provides Advice
 - Explanations
 - Examples
 - Detailed statistics (if desired)



Agenda

- **Who are Rogue Wave?**
- **High Performance Computing Challenges**
- **Debugging: TotalView Updated for the Cray Platform**
- **Optimization: Introduction to ThreadSpotter**
- **Future Development Tool Needs**

Feedback on debugging and optimization tool priorities

- **How key are the following potential enhancements**
 - **C++ 0x (2011)**
 - **Co-Array Fortran**
 - **OpenCL**
 - **Improved support for comparative debugging**
 - **Cray Fast Track**
 - **ATP**
 - **Thread race condition detection**

Thank you!

Rogue Wave Software

**Developing parallel, data-intensive applications is hard.
We make it easier.**

sales@roguewave.com