ROGUE WAVE SOFTWARE ®

ECMWF Workshop on High Performance Computing in Meteorology

3rd November 2010

Dean Stewart

Agenda

- Company Overview
- Rogue Wave Product Overview
  - IMSL Fortran
  - TotalView Debugger
  - Acumem ThreadSpotter

Rogue Wave

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

- **History**
  - Founded: 1989
  - Named after the founder’s boat
  - Acquisitions:
    - Visual Numerics: 2009
    - TotalView Technologies: 2010
    - Acumem: 2010

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

- **Global Locations**
  - HQ: Boulder, CO
  - NA: Houston, TX; Corvallis, OR; Natick MA
  - EMEA: France, Germany, Sweden, UK
  - APAC: Japan

Rogue Wave Today

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

Leading provider of enterprise class C++ components and infrastructure for high performance applications.

Leader in embeddable math and statistics algorithms and visualization software for data-intensive applications.

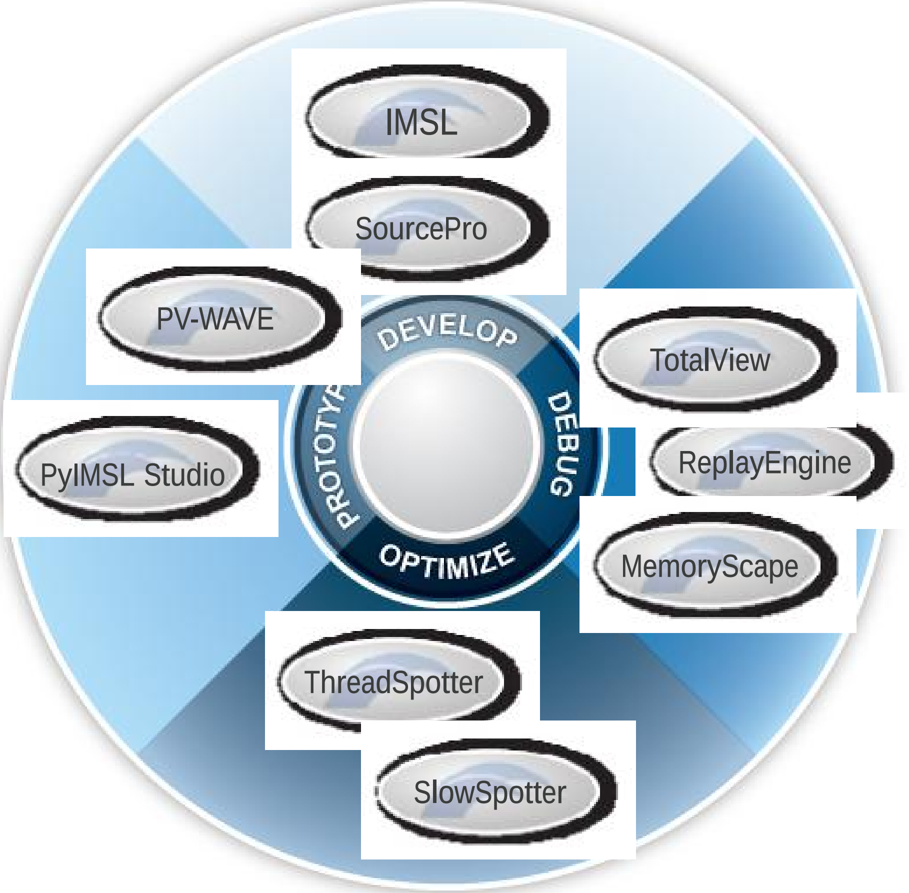
Industry-leading interactive analysis and debugging tools for the world's most sophisticated software applications.

Leading provider of intelligent software technology which analyzes and optimizes the computing performance in single- and multi-core environments.

Agenda

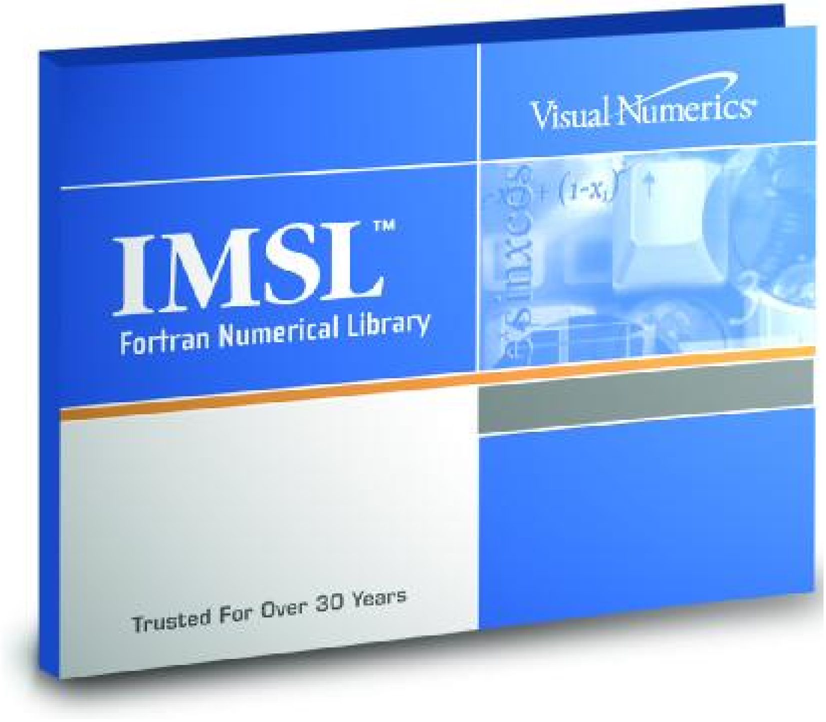
- Company Overview
- Rogue Wave Product Overview
  - IMSL Fortran
    - CUDA support
  - TotalView Debugger
    - Remote Display Client
    - TVScript for Batch Debugging
    - MemoryScape and ReplayEngine
  - Acumem ThreadSpotter

Rogue Wave Product Family



PROTOTYPE è DEVELOP è DEBUG è OPTIMIZE

IMSL Fortran Library



Now supports NVIDIA GPUs

Parallelism Using the IMSL Fortran Library

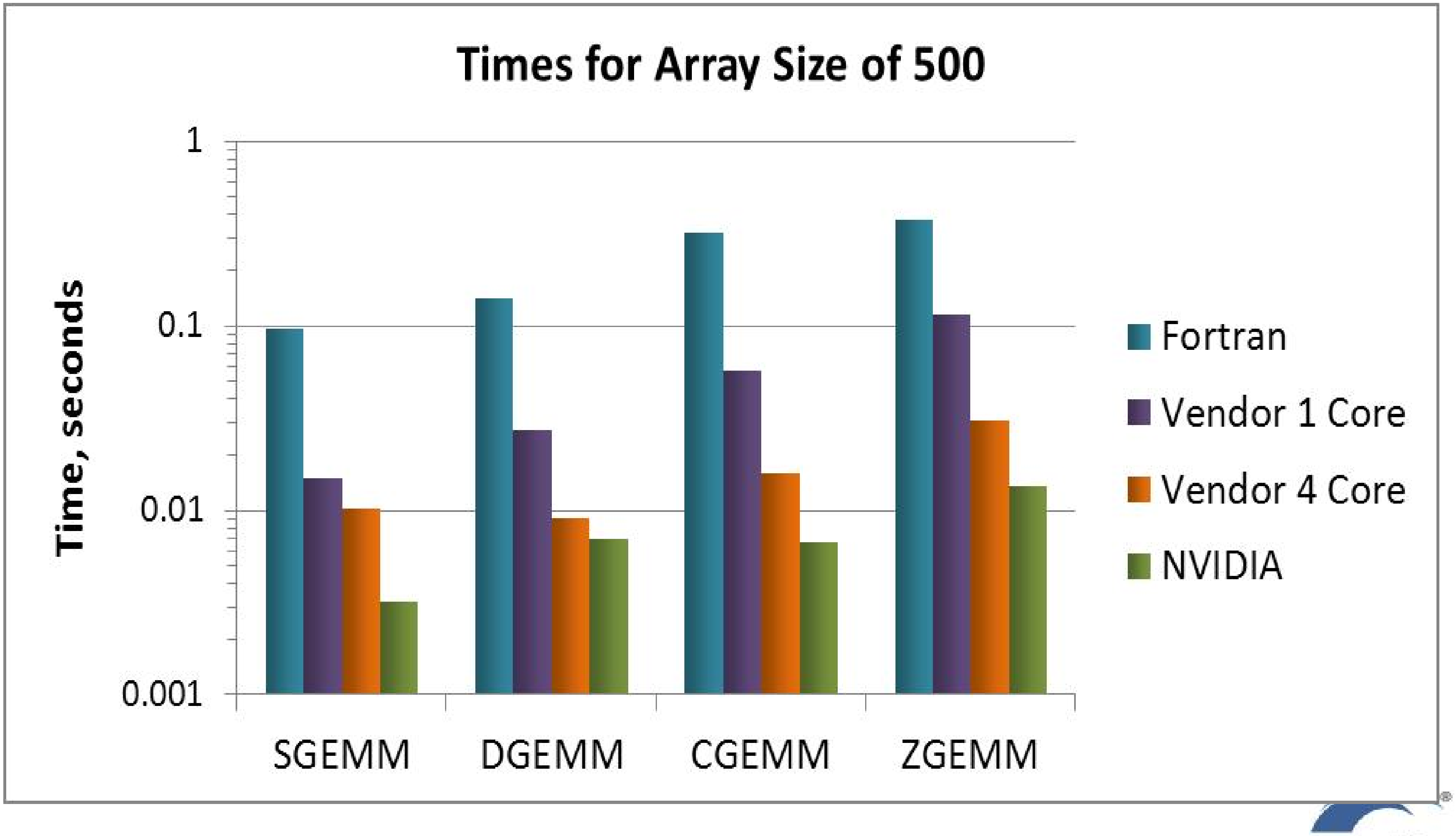
- OpenMP
  - Linear systems, Eigensystem Analysis, Transforms, etc.
  - The corresponding routines differ by environment
  - IMSL uses the OpenMP directive to parallelize internally; users do not need to write parallel program source

- MPI
  - Linear Algebra Operator and Generic Function, Linear systems, etc.
  - Corresponding routines are the same in all environments
  - Users need to call the IMSL function MP_SETUP()

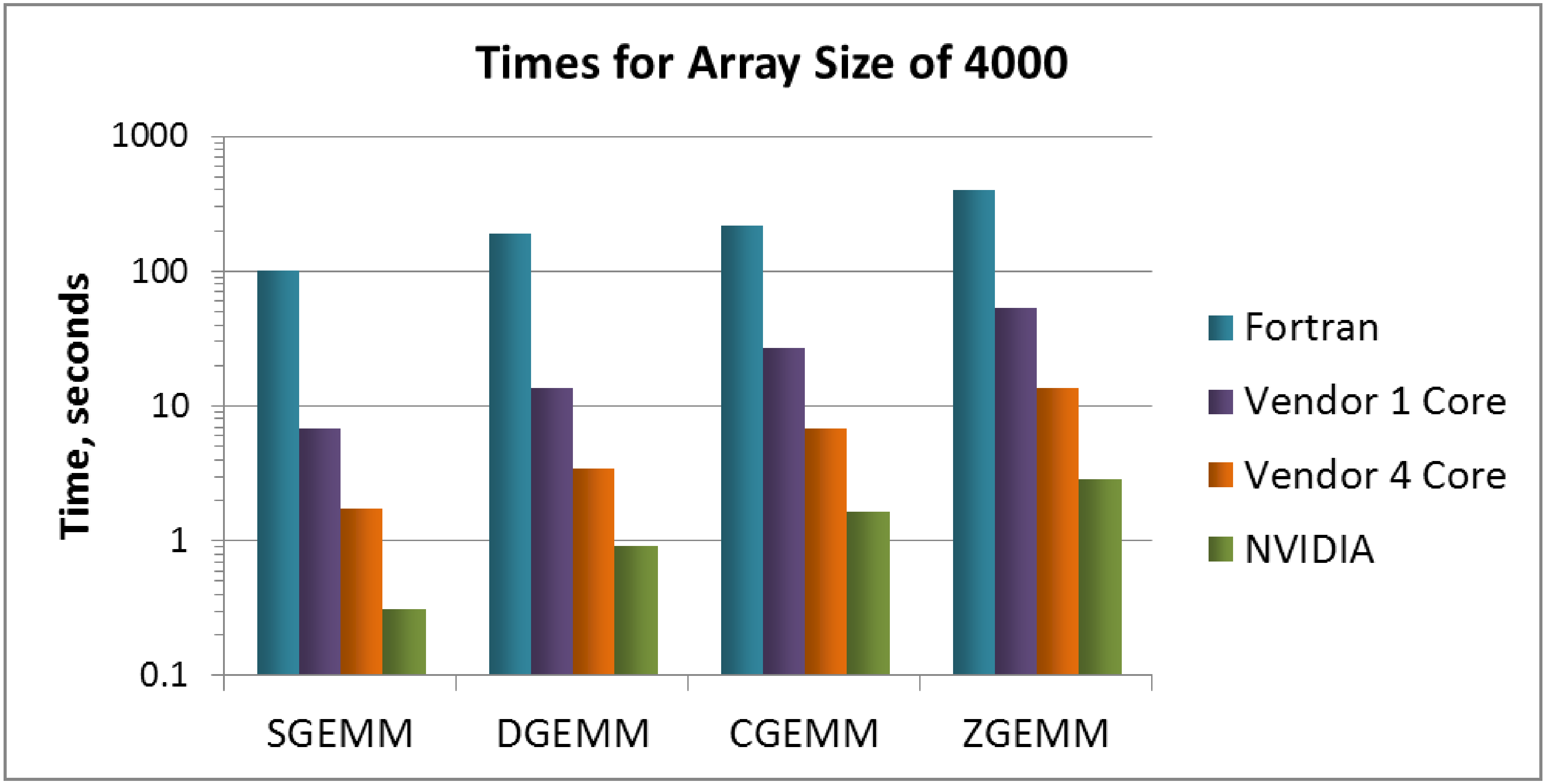
IMSL Fortran Library 7.0

- Integration of NVIDIA CUDA BLAS
  - Support NVIDIA GPU hardware (Tesla, Fermi)
  - Call CUDA Level 2 and 3 BLAS functions
    - Problem size must be large enough to see the benefit
    - Use IMSL functions as usual, calls NVIDIA’s library behind the scenes
    - If no GPU hardware or smaller problem, execute on CPU
  - Double and Single precision
  - Benchmark white paper available
    http://www.vni.com/products/imsl/fortran/

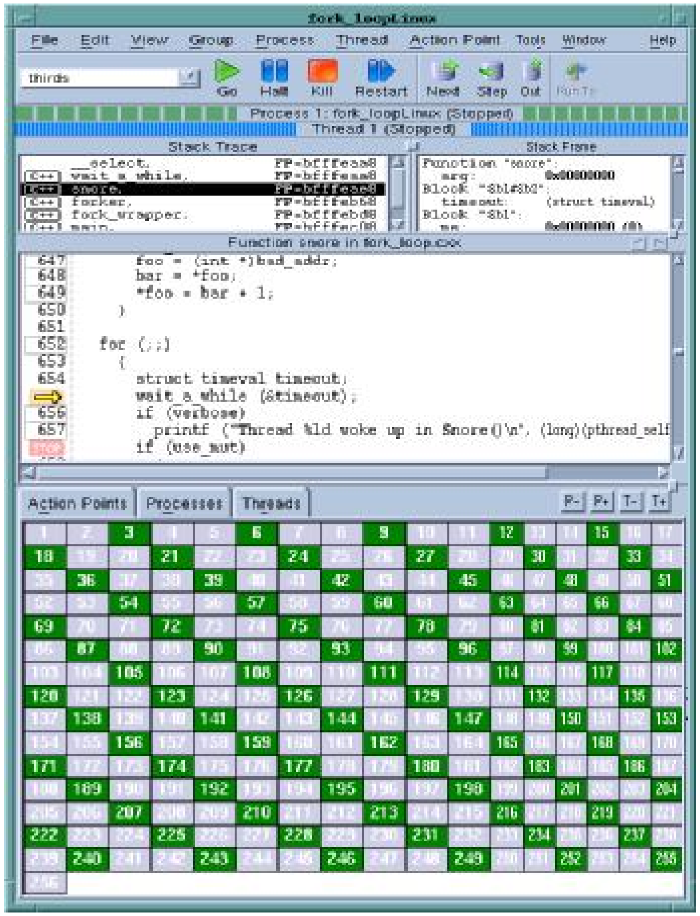
Throughput Results



Throughput Results (cont.)



TotalView Debugging Technology



```
        __select,            FP=bfffeaa8
C++     wait_a_while,        FP=bfffeaa8
C++     snore,               FP=bfffeac8
C++     forker,              FP=bfffeb68
C++     fork_wrapper,        FP=bfffebd8
C++     main,                FP=bfffec08
```

```
Function "snore":
  arg:            0x00000000
Block "$b1#$b2":
  timeout:        (struct timeval)
Block "$b1":
  ms:             0x00000000 (0)
```

```
        foo = (int *)bad_addr;
        bar = *foo;
        *foo = bar + 1;
      }

    for (;;)
      {
        struct timeval timeout;
        wait_a_while (&timeout);
        if (verbose)
          printf ("Thread %ld woke up in Snore()\n", (long)(pthread_self
        if (use_mut)
```

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51
52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68
69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85
86	87	88	89	90	91	92	93	94	95	96	97	98	99	100	101	102
103	104	105	106	107	108	109	110	111	112	113	114	115	116	117	118	119
120	121	122	123	124	125	126	127	128	129	130	131	132	133	134	135	136
137	138	139	140	141	142	143	144	145	146	147	148	149	150	151	152	153
154	155	156	157	158	159	160	161	162	163	164	165	166	167	168	169	170
171	172	173	174	175	176	177	178	179	180	181	182	183	184	185	186	187
188	189	190	191	192	193	194	195	196	197	198	199	200	201	202	203	204
205	206	207	208	209	210	211	212	213	214	215	216	217	218	219	220	221
222	223	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238
239	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255
256																

- What is TotalView?
  - Parallel and Multithreaded Debugging and Analysis Tool
  - For scientists and engineers working with C/C++ and Fortran
  - Makes developing, maintaining and supporting critical and cutting edge applications easier and less risky
- Major Features
  - Supports Linux, Unix and Mac OS X
  - Parallel Debugging
    - MPI, Pthreads, OMP, UPC
  - Includes a Remote Display Client freeing users to work from anywhere
  - Memory Debugging with MemoryScape
  - Optional Reverse Debugging with ReplayEngine
  - Batch Debugging with TVSCript and the CLI
- Advantages
  - Easy to learn graphical user interface with data visualization
  - Wide variety of features so users can tackle unexpected bugs
  - Consistent functionality and look and feel across a wide range of platforms
  - Works robustly with open source and vendor compilers
  - Native debugger core is highly scalable to large clusters, large code and massive datasets

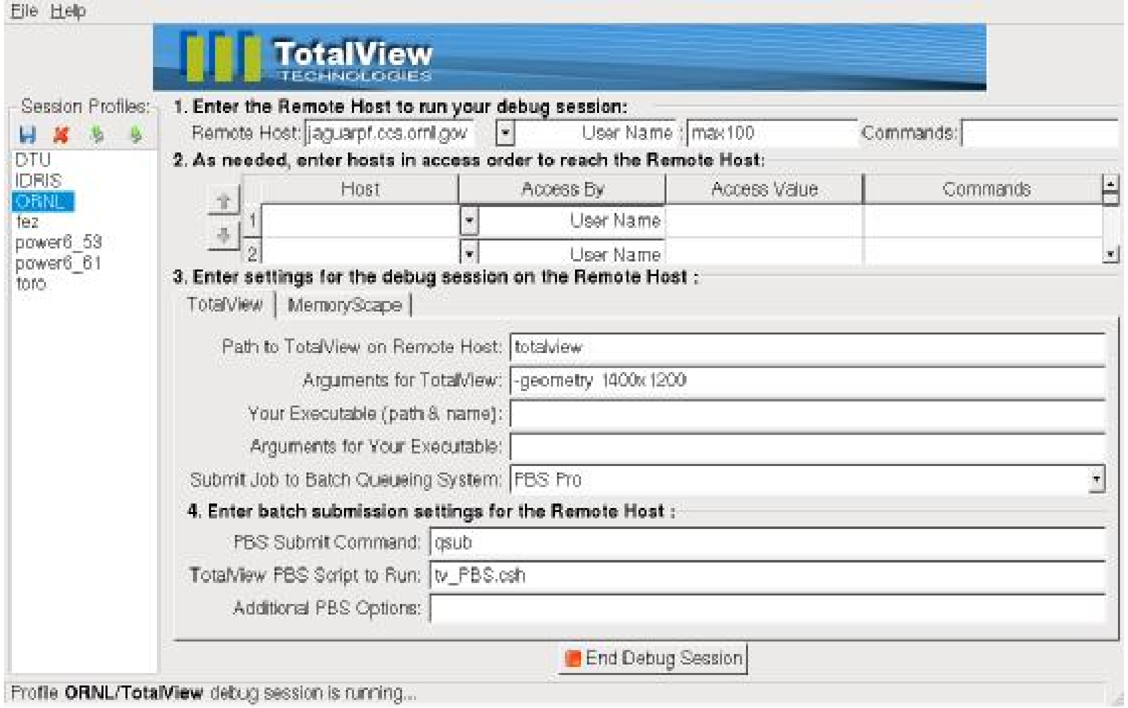
How can TotalView help you ?

Debugging means examining a specific controlled instance of program execution

Provides an answer to the question : “What is my program *really* doing?”

- **Threads and/or MPI**
  - When you have
    - Deadlocks and hangs
    - Race conditions
  - It provides
    - Asynchronous thread control
    - Barrier points
    - Powerful group mechanism
- **Fortran and/or C++**
  - Complex data structures
    - Diving and recursive dive
  - STL Collection Classes
    - STLView
  - Rich class hierarchies
    - Powerful type-casting features
- **Memory Analysis**
  - Leaks and Bounds Errors
    - Automatic error detection tools
  - Out of Memory Errors
    - Analysis of heap memory usage by file function and line
- **Data Analysis**
  - Numerical errors
    - Extensible data visualization
    - Slicing and filtering of arrays
    - Powerful expression system
    - Conditional watchpoints

TotalView Remote Display Client



Session Profiles:

DTU
IDRIS
ORNL
tez
power6_53
power6_61
toro

1. Enter the Remote Host to run your debug session:

Remote Host: jaguarpf.ccs.ornl.gov User Name : max100 Commands:

2. As needed, enter hosts in access order to reach the Remote Host:

	Host	Access By	Access Value	Commands
1		User Name		
2		User Name		

3. Enter settings for the debug session on the Remote Host :

TotalView | MemoryScape

Path to TotalView on Remote Host: totalview
Arguments for TotalView: -geometry 1400x1200
Your Executable (path & name):
Arguments for Your Executable:
Submit Job to Batch Queueing System: PBS Pro

4. Enter batch submission settings for the Remote Host :

PBS Submit Command: qsub
TotalView PBS Script to Run: tv_PBS.csh
Additional PBS Options:

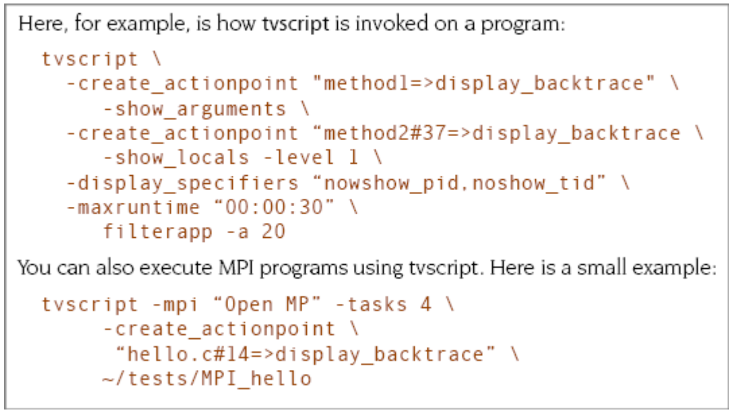
Profile **ORNL/TotalView** debug session is running...

- The Remote Display Client offers users the ability to easily set up and operate a TotalView debug session that is running on another system.
- Provides for a connection that is
  - Easy
  - Fast
  - Secure
- The Remote Display Client is available for:
  - Linux x86
  - Linux x86-64
  - Windows XP
  - Windows Vista
  - Mac OS X Leopard and Snow Leopard
- The Client also provides for submission of jobs to batch queuing systems PBS Pro and LoadLeveler

ROGUE WAVE®
SOFTWARE

Batch Debugging with TVScript

- TVScript
  - Defines events
    - Breakpoints, memory errors, etc..
  - Actions to take in response to these events
    - Print variables or create memory reports
  - Runs a serial or MPI program towards completion
    - With no user interaction
- More powerful and flexible than Printf-style debugging
  - Use to prepare and guide interactive debugging
  - Use whenever jobs need to be submitted into a managed environment
  - Can be used to automate test/verify environments



Here, for example, is how tvscript is invoked on a program:

```
tvscript \
  -create_actionpoint "methodl=>display_backtrace" \
     -show_arguments \
  -create_actionpoint “method2#37=>display_backtrace \
     -show_locals -level 1 \
  -display_specifiers “nowshow_pid.noshow_tid” \
  -maxruntime “00:00:30” \
     filterapp -a 20
```

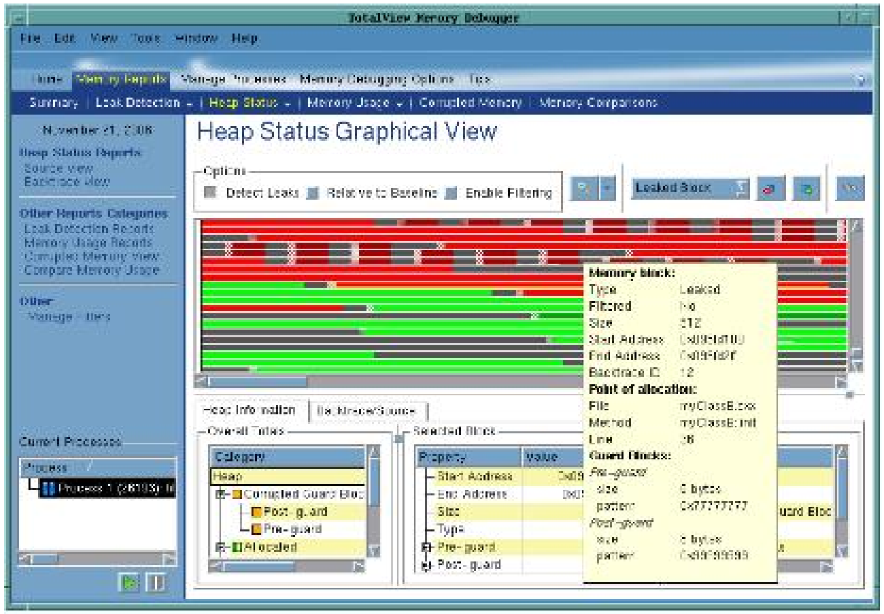
You can also execute MPI programs using tvscript. Here is a small example:

```
tvscript -mpi “Open MP” -tasks 4 \
     -create_actionpoint \
      “hello.c#14=>display_backtrace” \
     ~/tests/MPI_hello
```

MemoryScape

Simple to use, intuitive memory debugging

- What is MemoryScape?
  - Streamlined
  - Lightweight
  - Intuitive
  - Collaborative
  - Memory Debugging
- Features
  - Shows
    - Memory errors
    - Memory status
    - Memory leaks
    - Buffer overflows
  - MPI memory debugging
  - Remote memory debugging



Memory block:

Type	Leaked
Filtered	No
Size	512
Start Address	0x09f6f000
End Address	0x09f6f2ff
Backtrace ID	12

Point of allocation:

File	myClassB.cxx
Method	myClassB::init
Line	36

Guard Blocks:

Pre-guard

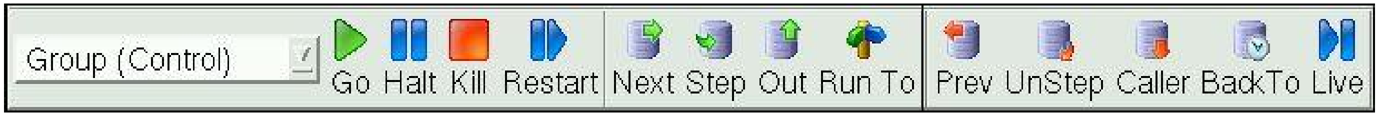
size	8 bytes
pattern	0x77777777

Post-guard

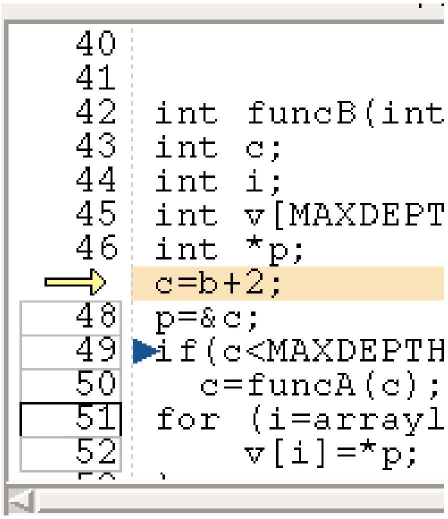
size	8 bytes
pattern	0x99999999

- Technical Advantages
  - Low overhead
  - No Instrumentation
- Interface
  - Inductive
  - Collaboration
  - Multi-process

ReplayEngine



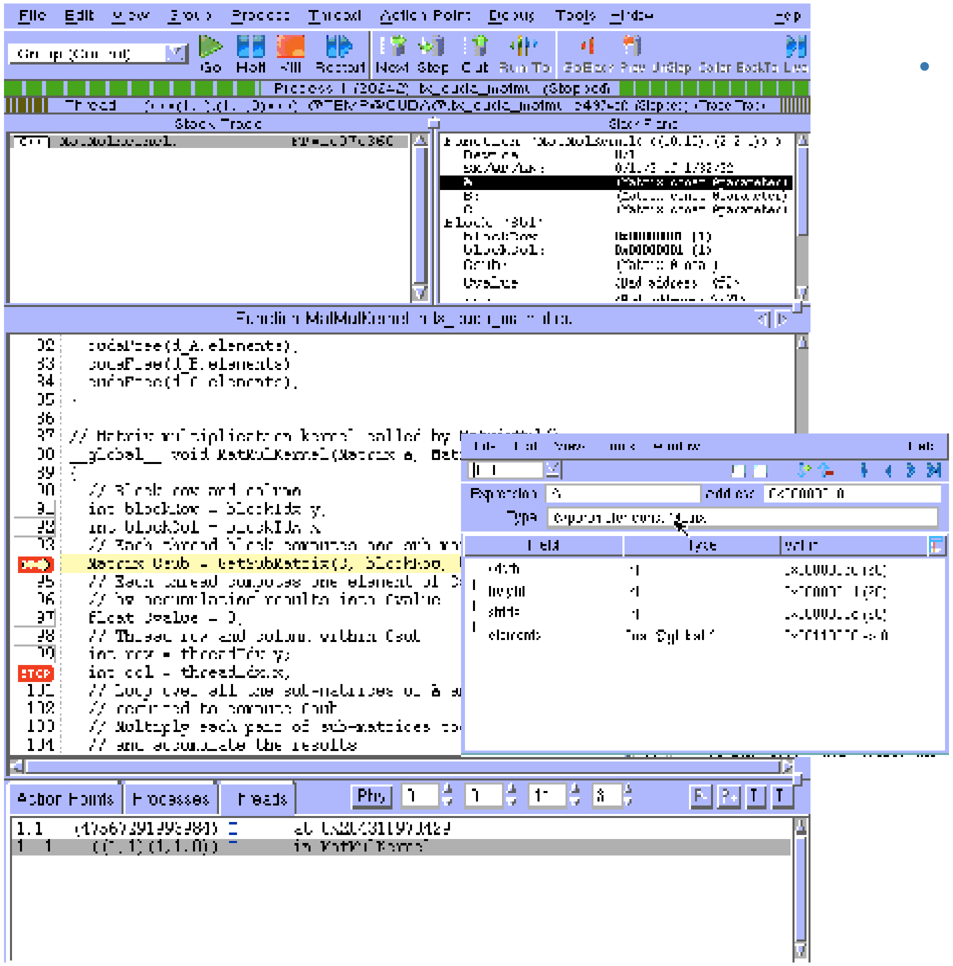
- Enhances debugging experience
  - Add-on to TotalView
- Captures execution history
  - Record all external input to program
  - Records internal sources of non-determinism
- Replays execution history
  - Examine any part of the execution history
  - Step as easily back through code as you do forwards
  - Jump to points of interest
- Simple extension to TotalView
  - No recompilation or instrumentation
  - The user just says where they want to go
  - Explore data and state in the past just like a live process
- Supported on Linux x86 and x86-64
- Supports MPI, Pthreads, and OpenMP



```


int funcB(int
int c;
int i;
int v[MAXDEPT
int *p;
c=b+2;
p=&c;
if(c<MAXDEPTH
  c=funcA(c);
for (i=arrayl
    v[i]=*p;
```

TotalView for CUDA


File Edit View Group Process Thread Action Point Debug Tools Window Help
Go Halt Kill Restart Next Step Out Run To
Stack Trace
Stack Frame
Action Points Processes Threads
Phy

```
  cudaFree(d_A.elements);
  cudaFree(d_B.elements);
  cudaFree(d_C.elements);
}

// Matrix multiplication kernel called by Matri
__global__ void MatMulKernel(Matrix A, Mat
{
  // Block row and column
  int blockRow = blockIdx.y;
  int blockCol = blockIdx.x;
  // Each thread block computes one sub-matri
  Matrix Csub = GetSubMatrix(C, blockRow,
  // Each thread computes one element of C
  // by accumulating results into Cvalue
  float Cvalue = 0;
  // Thread row and column within Csub
  int row = threadIdx.y;
  int col = threadIdx.x;
  // Loop over all the sub-matrices of A a
  // required to compute Csub
  // Multiply each pair of sub-matrices to
  // and accumulate the results
```

- Characteristics
  - Debugging of application running on the GPU device (not in an emulator)
  - Full visibility of both Linux threads and GPU device threads
    - Device threads shown as part of the parent Unix process
    - Correctly handle all the differences between the CPU and GPU
  - Fully represent the hierarchical memory
    - Display data at any level (registers, local, block, global or host memory)
    - Making it clear where data resides with type qualification
  - Thread and Block Coordinates
    - Built in runtime variables display threads in a warp, block and thread dimensions and indexes
    - Displayed on the interface in the status bar, thread tab and stack frame
  - Device thread control
    - Warps advance synchronously
  - Handles CUDA function inlining
    - Step into or over inlined functions
    - Functions show on stack trace
  - Reports memory access errors
    - CUDA memcheck
  - Multi-Device Support
  - Can be used with MPI

Acumem ThreadSpotter

- Optimization tool that provides real performance analysis of your running code
- Focus attention to the code and the problems that matter and can be solved
- What-if analysis from a single sampling
- The tool is your mentor – learn as you go, become proficient
- Performance regression test tool:
  - because resource contention is a global concern
  - Problems don't always show up at intuitive places

Cache Optimization Technology

- Analyze binary behavior
  - Collect sparse runtime data
  - Model off-line using math

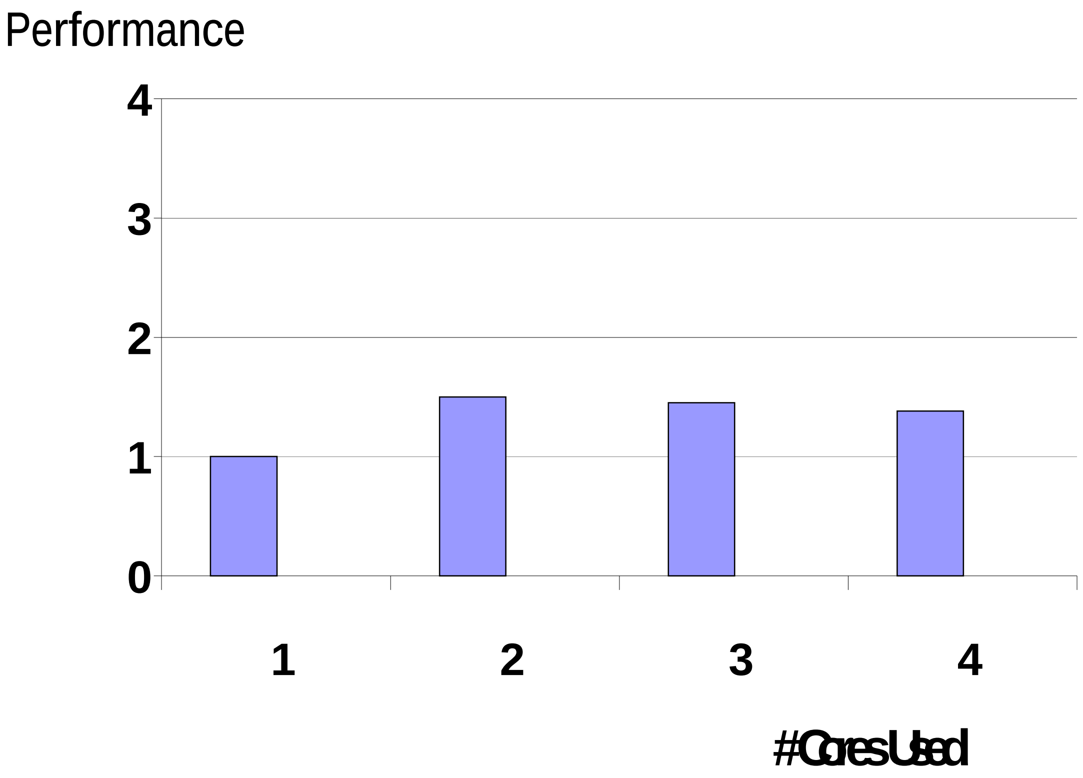
- ThreadSpotter
  - Application performance analysis for programmers
  - We tell what the problem is, where it is and how to fix it

Makes the performance experts more productive

Enables non-experts to optimize code

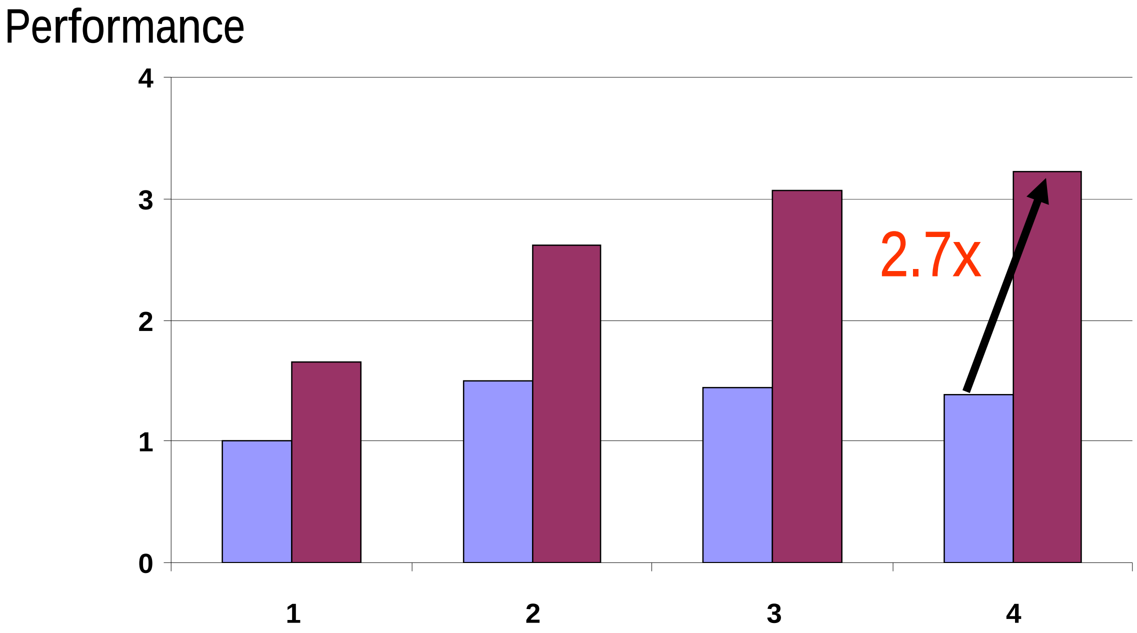
Example: Poor parallelism? (LBM)

Lattice Boltzmann methods – Fluid Dynamics



- Actually, it is “embarrassingly parallel”
- Poor memory usage è super-linear slowdown

The same application optimized (LBM)

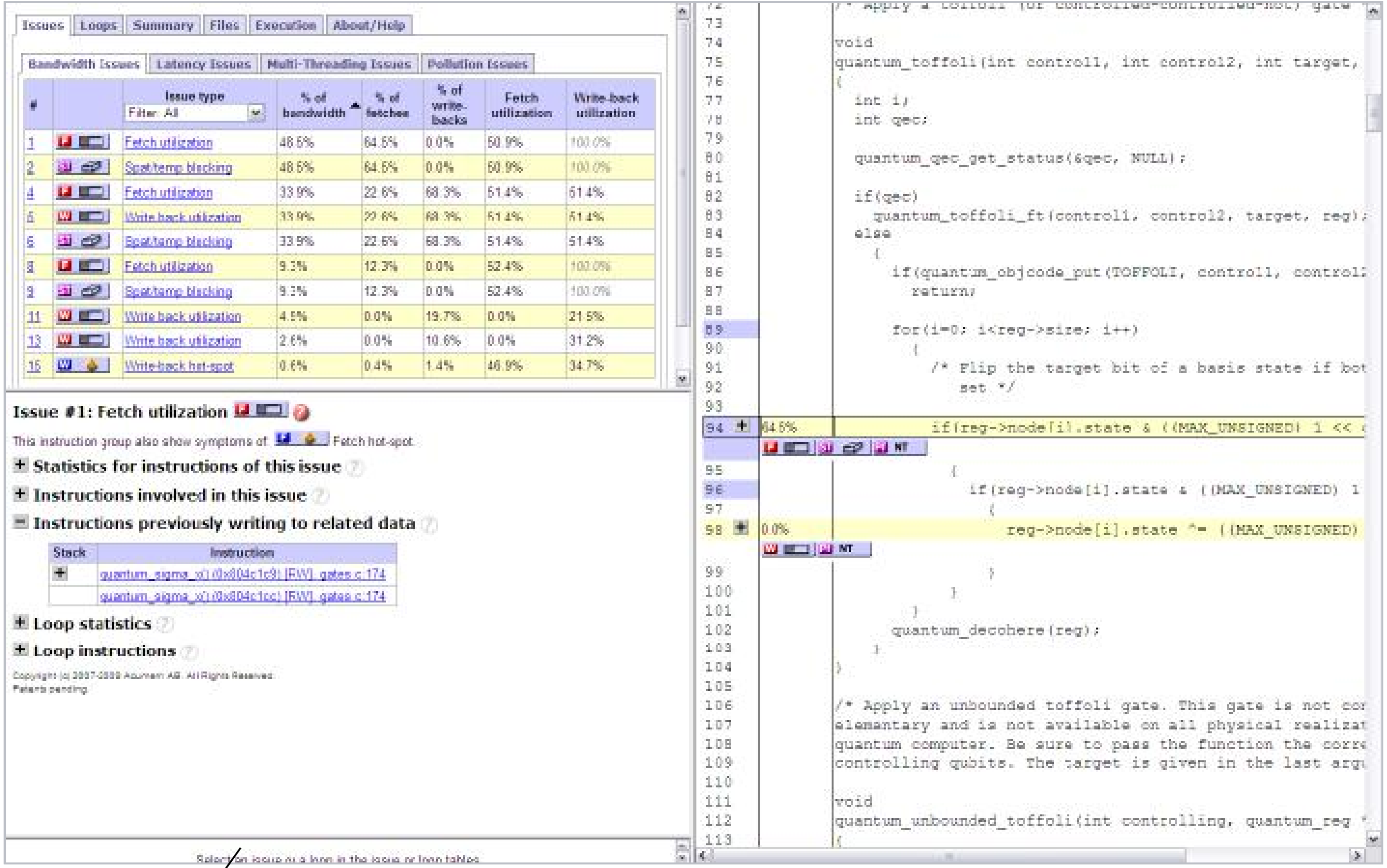


- Optimization can be rewarding, but costly…
  - Requires expert knowledge
  - Weeks of wading through performance data
- ThreadSpotter one-click advice: Change one line

ThreadSpotter Screenshot

What?

Where?



#		Issue type Filter: All	% of bandwidth	% of fetches	% of write-backs	Fetch utilization	Write-back utilization
1	F	Fetch utilization	48.5%	64.5%	0.0%	50.9%	100.0%
2	ST	Spat/temp blocking	48.5%	64.5%	0.0%	50.9%	100.0%
4	F	Fetch utilization	33.9%	22.6%	68.3%	51.4%	51.4%
5	W	Write back utilization	33.9%	22.6%	68.3%	51.4%	51.4%
6	ST	Spat/temp blocking	33.9%	22.6%	68.3%	51.4%	51.4%
8	F	Fetch utilization	9.3%	12.3%	0.0%	52.4%	100.0%
9	ST	Spat/temp blocking	9.3%	12.3%	0.0%	52.4%	100.0%
11	W	Write back utilization	4.5%	0.0%	19.7%	0.0%	21.5%
13	W	Write back utilization	2.6%	0.0%	10.6%	0.0%	31.2%
16	W	Write-back hot-spot	0.6%	0.4%	1.4%	46.9%	34.7%

Issue #1: Fetch utilization

This instruction group also show symptoms of F Fetch hot-spot

+ Statistics for instructions of this issue

+ Instructions involved in this issue

− Instructions previously writing to related data

Stack	Instruction
+	quantum_sigma_x() (0x804c1c9) [RW] gates.c:174
	quantum_sigma_x() (0x804c1cc) [RW] gates.c:174

+ Loop statistics

+ Loop instructions

Copyright (c) 2007-2009 Acumem AB. All Rights Reserved.
Patents pending

Select an issue or a loop in the issue or loop tables

```
/* Apply a toffoli (or controlled-controlled-not) gate

void
quantum_toffoli(int control1, int control2, int target,
{
  int i;
  int qec;

  quantum_qec_get_status(&qec, NULL);

  if(qec)
    quantum_toffoli_ft(control1, control2, target, reg);
  else
    {
      if(quantum_objcode_put(TOFFOLI, control1, control2
        return;

      for(i=0; i<reg->size; i++)
        {
          /* Flip the target bit of a basis state if bot
             set */

64.5%         if(reg->node[i].state & ((MAX_UNSIGNED) 1 << (
F ST NT
            {
              if(reg->node[i].state & ((MAX_UNSIGNED) 1
                {
0.0%                reg->node[i].state ^= ((MAX_UNSIGNED)
W NT
                }
            }
        }
      quantum_decohere(reg);
    }
}

/* Apply an unbounded toffoli gate. This gate is not cor
elementary and is not available on all physical realizat
quantum computer. Be sure to pass the function the corre
controlling qubits. The target is given in the last argu

void
quantum_unbounded_toffoli(int controlling, quantum_reg *
{
```

How?

Resources

Rogue Wave Software website

- www.roguewave.com

Demystifying debugging whitepaper

- http://www-931.ibm.com/bin/newsletter/tool/landingPage.cgi?lpId=3022

TotalView video tutorials

- http://www.totalviewtech.rsvp1.com/support/videos.html#0

Acumem ThreadSpotter On-line Demo

- https://acumem.webex.com/acumem/ldr.php?AT=pb&SP=MC&rID=19140117&rKey=CD1662E760EA59FE

Visiting SC10?

Visit Rogue Wave at Booths 2431 and 2432 and Acumem at 3749

Contact me if you would like us to schedule a meeting with our senior product and management staff at the event .

Dean Stewart dean.stewart@roguewave.com
Tel: +44 (0)6450 549957, Mob: +44 (0) 7713 150529

ROGUE WAVE®
SOFTWARE

Thank You