Debugging and Optimizing Programs Accelerated with Intel® Xeon® Phi™ Coprocessors

Chris Gottbrath, Principal Product Manager

May 7th, 2013

CUG 2013, Napa, CA

Rogue Wave Today

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

Highlights

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

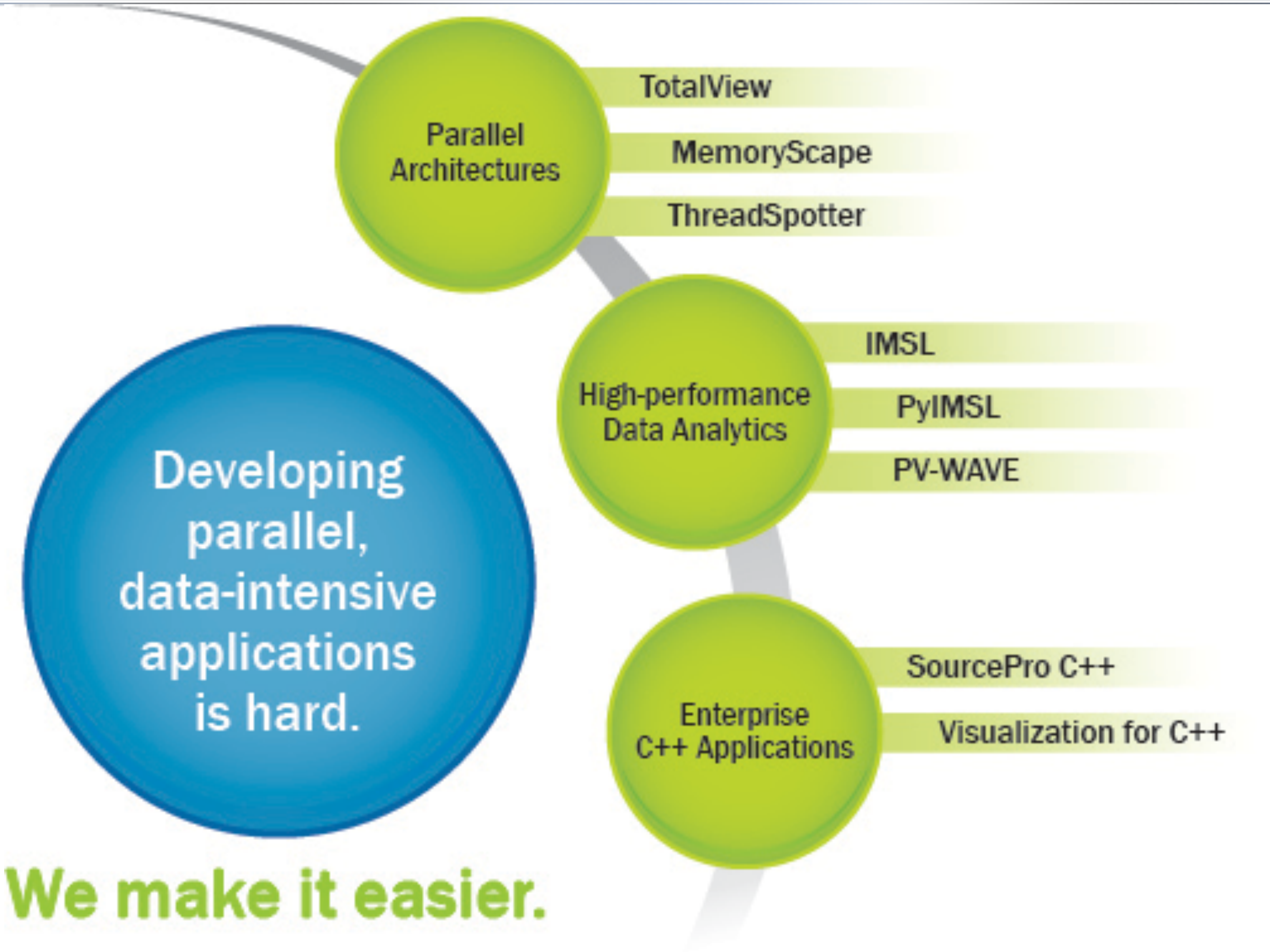
History

- Founded: 1989
- Acquired by Audax Group: 2012
- Acquired:
  - Visual Numerics: 2009
  - TotalView Technologies: 2009
  - Acumem: 2010
  - IBM ILOG Views C++: 2012
- 40 years of experience in HPC

Customers

- 3,000+ customers in 36 countries
- Multiple sectors:
  - Financial services
  - Telecom
  - Oil and gas
  - Government and aerospace
  - Research and academic

Rogue Wave Solution Portfolio



Developing parallel, data-intensive applications is hard.

Parallel Architectures

- TotalView
- MemoryScape
- ThreadSpotter

High-performance Data Analytics

- IMSL
- PyIMSL
- PV-WAVE

Enterprise C++ Applications

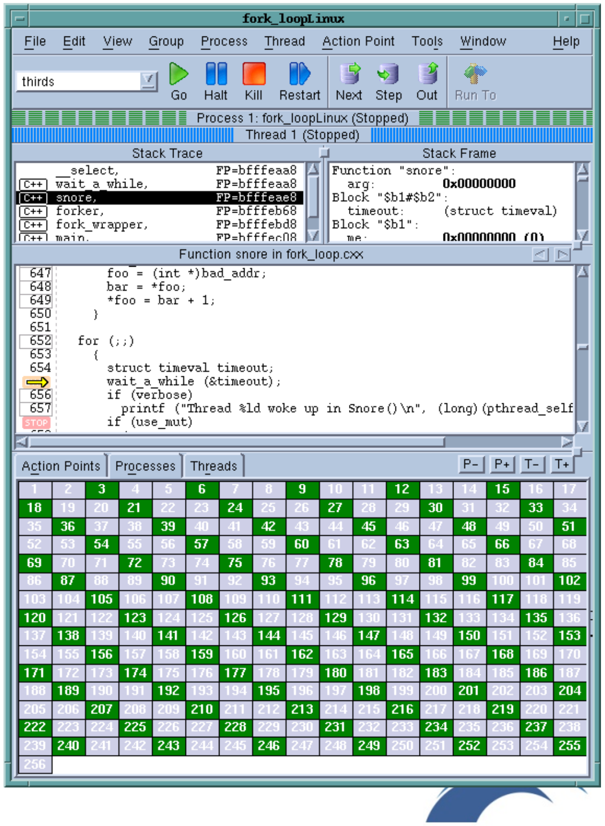
- SourcePro C++
- Visualization for C++

We make it easier.

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 (Cray, BG, BullX, etc..)**
  - **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, Xeon Phi (early access)**
  - **Includes a** Remote Display Client **freeing you to work from anywhere**
  - **Memory Debugging with** MemoryScape
  - **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**



Process 1: fork_loopLinux (Stopped)

Thread 1 (Stopped)

Stack Trace

```
      __select,              FP=bfffeaa8
C++   wait_a_while,          FP=bfffeaa8
C++   snore,                 FP=bfffeae8
C++   forker,                FP=bfffeb68
C++   fork_wrapper,          FP=bfffebd8
C++   main,                  FP=bfffec08
```

Stack Frame

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

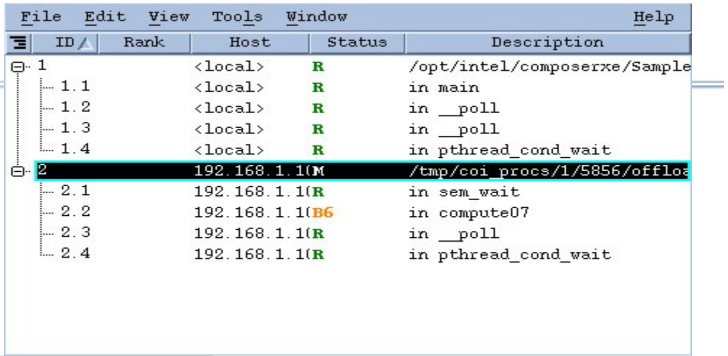
Function snore in fork_loop.cxx

```
        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
STOP    if (use_mut)
```

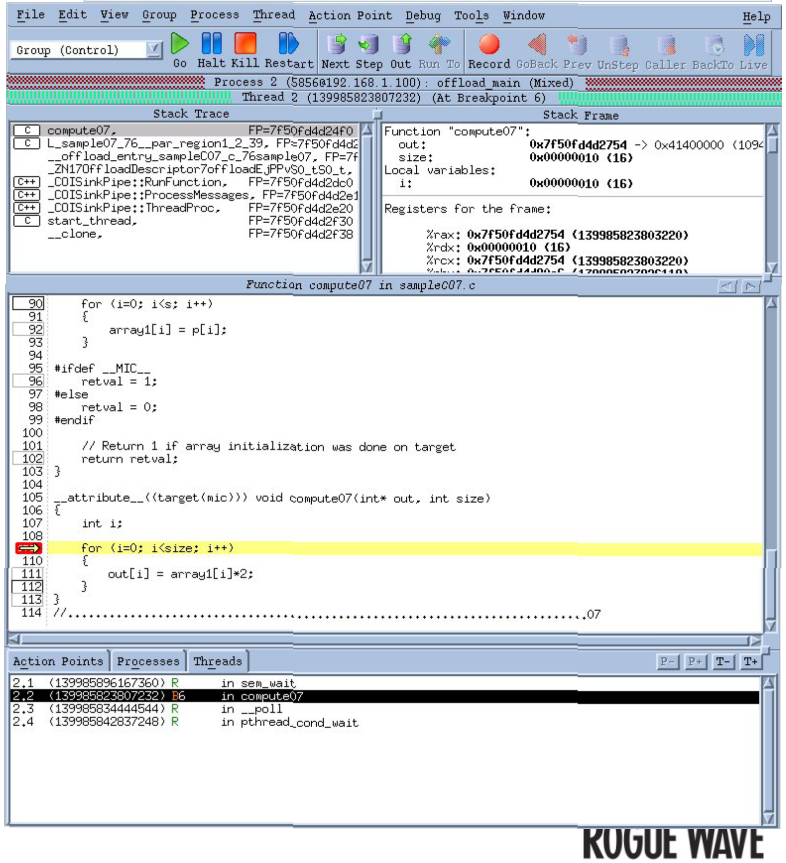
1	2	**3**	4	5	**6**	7	8	**9**	10	11	**12**	13	14	**15**	16	17
18	19	20	**21**	22	23	**24**	25	26	**27**	28	29	**30**	31	32	**33**	34
35	**36**	37	38	**39**	40	41	**42**	43	44	**45**	46	47	**48**	49	50	**51**
52	53	**54**	55	56	**57**	58	59	**60**	61	62	**63**	64	65	**66**	67	68
69	70	71	**72**	73	74	**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																

TotalView for Xeon Phi

- **Support Multiple Intel Xeon Phi configurations**
  - Native Mode
    - With MPI
  - Offload Directives
    - Similar to GPU
  - Multi-device
  - Multi-node
    - Certain configurations
  - CS300-AC, Future XC30
- **User Interface**
  - MPI Debugging Features
    - Process Control
    - View Across
    - Shared Breakpoints
  - Heterogeneous Debugging
    - Debug Both Xeon and Xeon-Phi Processes



```
ID    Rank  Host            Status  Description
1           <local>         R       /opt/intel/composerxe/Sample
  1.1       <local>         R       in main
  1.2       <local>         R       in __poll
  1.3       <local>         R       in __poll
  1.4       <local>         R       in pthread_cond_wait
2           192.168.1.1(    M       /tmp/coi_procs/1/5856/offloa
  2.1       192.168.1.1(    R       in sem_wait
  2.2       192.168.1.1(    B6      in compute07
  2.3       192.168.1.1(    R       in __poll
  2.4       192.168.1.1(    R       in pthread_cond_wait
```



```
Process 2 (5856@192.168.1.100): offload_main (Mixed)
Thread 2 (139985823807232) (At Breakpoint 6)
```

```
C    compute07,                                  FP=7f50fd4d24f0
C    L_sample07_76__par_region1_2_39,            FP=7f50fd4d2
     __offload_entry_sampleC07_c_76sample07,     FP=7f
     _ZN17OffloadDescriptor7offloadEjPPvS0_tS0_t,
C++  _COISinkPipe::RunFunction,                  FP=7f50fd4d2dc0
C++  _COISinkPipe::ProcessMessages,              FP=7f50fd4d2e1
C++  _COISinkPipe::ThreadProc,                   FP=7f50fd4d2e20
C    start_thread,                               FP=7f50fd4d2f30
     __clone,                                    FP=7f50fd4d2f38
```

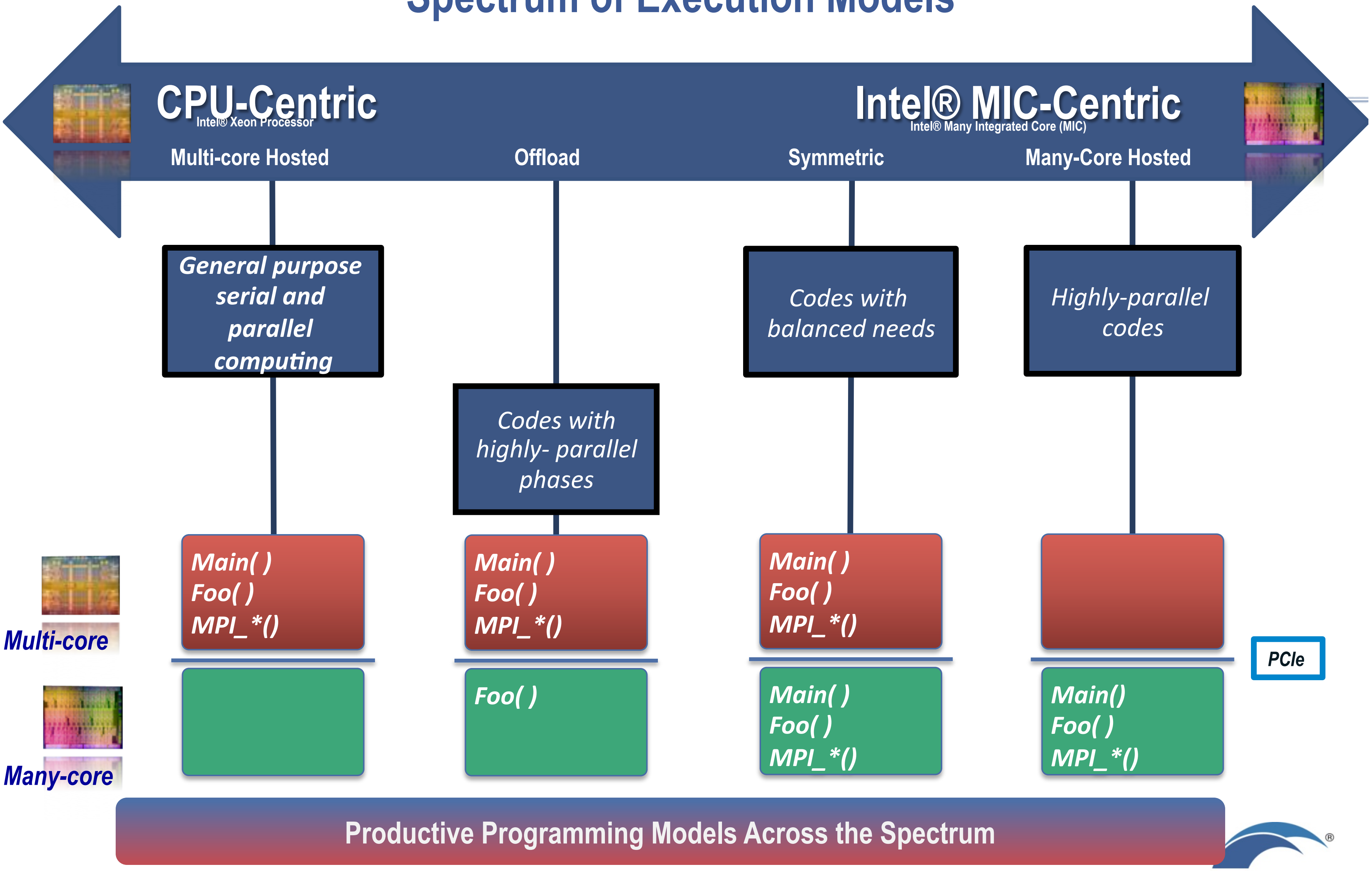
```
Function "compute07":
  out:            0x7f50fd4d2754 -> 0x41400000 (1094
  size:           0x00000010 (16)
Local variables:
  i:              0x00000010 (16)

Registers for the frame:

     %rax: 0x7f50fd4d2754 (139985823803220)
     %rdx: 0x00000010 (16)
     %rcx: 0x7f50fd4d2754 (139985823803220)
```

Function compute07 in sampleC07.c

```c
    for (i=0; i<s; i++)
    {
        array1[i] = p[i];
    }

#ifdef __MIC__
    retval = 1;
#else
    retval = 0;
#endif

    // Return 1 if array initialization was done on target
    return retval;
}

__attribute__((target(mic))) void compute07(int* out, int size)
{
    int i;

    for (i=0; i<size; i++)
    {
        out[i] = array1[i]*2;
    }
}
//.........................................................................07
```

```
2.1  (139985896167360) R     in sem_wait
2.2  (139985823807232) B6    in compute07
2.3  (139985834444544) R     in __poll
2.4  (139985842837248) R     in pthread_cond_wait
```

Spectrum of Execution Models



CPU-Centric

Intel® Xeon Processor

Intel® MIC-Centric

Intel® Many Integrated Core (MIC)

Multi-core Hosted

Offload

Symmetric

Many-Core Hosted

General purpose serial and parallel computing

Codes with highly- parallel phases

Codes with balanced needs

Highly-parallel codes

Multi-core

Many-core

Main( )
Foo( )
MPI_*()

Main( )
Foo( )
MPI_*()

Foo( )

Main( )
Foo( )
MPI_*()

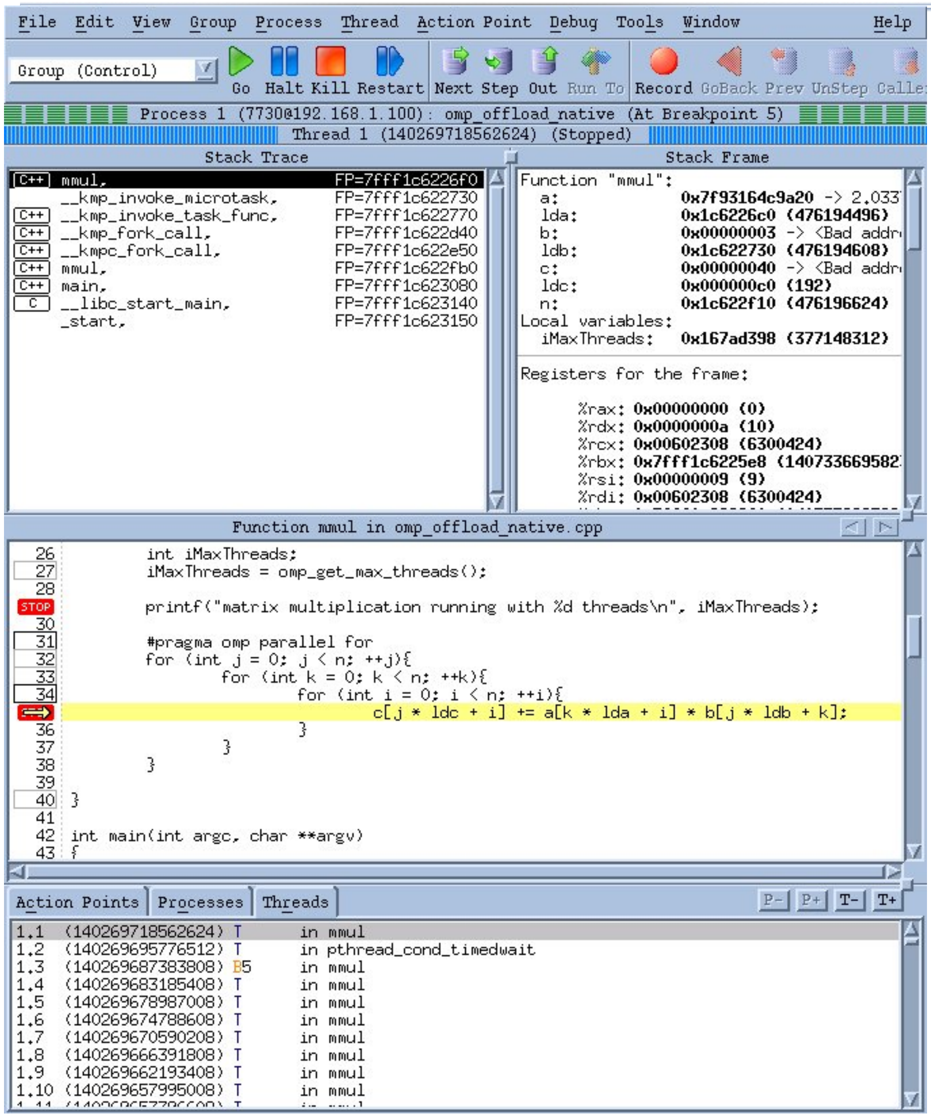
Main( )
Foo( )
MPI_*()

Main()
Foo( )
MPI_*()

PCIe

Productive Programming Models Across the Spectrum

Remote Debugging of Applications on Xeon Phi



```
Process 1 (7730@192.168.1.100): omp_offload_native (At Breakpoint 5)
Thread 1 (140269718562624) (Stopped)
```

```
C++  mmul,                      FP=7fff1c6226f0
     __kmp_invoke_microtask,    FP=7fff1c622730
C++  __kmp_invoke_task_func,    FP=7fff1c622770
C++  __kmp_fork_call,           FP=7fff1c622d40
C++  __kmpc_fork_call,          FP=7fff1c622e50
C++  mmul,                      FP=7fff1c622fb0
C++  main,                      FP=7fff1c623080
 C   __libc_start_main,         FP=7fff1c623140
     _start,                    FP=7fff1c623150
```

```
Function "mmul":
  a:              0x7f93164c9a20 -> 2.033
  lda:            0x1c6226c0 (476194496)
  b:              0x00000003 -> <Bad addr
  ldb:            0x1c622730 (476194608)
  c:              0x00000040 -> <Bad addr
  ldc:            0x000000c0 (192)
  n:              0x1c622f10 (476196624)
Local variables:
  iMaxThreads:    0x167ad398 (377148312)

Registers for the frame:

      %rax: 0x00000000 (0)
      %rdx: 0x0000000a (10)
      %rcx: 0x00602308 (6300424)
      %rbx: 0x7fff1c6225e8 (140733669582
      %rsi: 0x00000009 (9)
      %rdi: 0x00602308 (6300424)
```

Function mmul in omp_offload_native.cpp

```
        int iMaxThreads;
        iMaxThreads = omp_get_max_threads();

STOP    printf("matrix multiplication running with %d threads\n", iMaxThreads);

        #pragma omp parallel for
        for (int j = 0; j < n; ++j){
                for (int k = 0; k < n; ++k){
                        for (int i = 0; i < n; ++i){
                                c[j * ldc + i] += a[k * lda + i] * b[j * ldb + k];
                        }
                }
        }

}

int main(int argc, char **argv)
{
```

```
1.1  (140269718562624) T    in mmul
1.2  (140269695776512) T    in pthread_cond_timedwait
1.3  (140269687383808) B5   in mmul
1.4  (140269683185408) T    in mmul
1.5  (140269678987008) T    in mmul
1.6  (140269674788608) T    in mmul
1.7  (140269670590208) T    in mmul
1.8  (140269666391808) T    in mmul
1.9  (140269662193408) T    in mmul
1.10 (140269657995008) T    in mmul
```

- Just run as
  totalview –r mic0 <program>

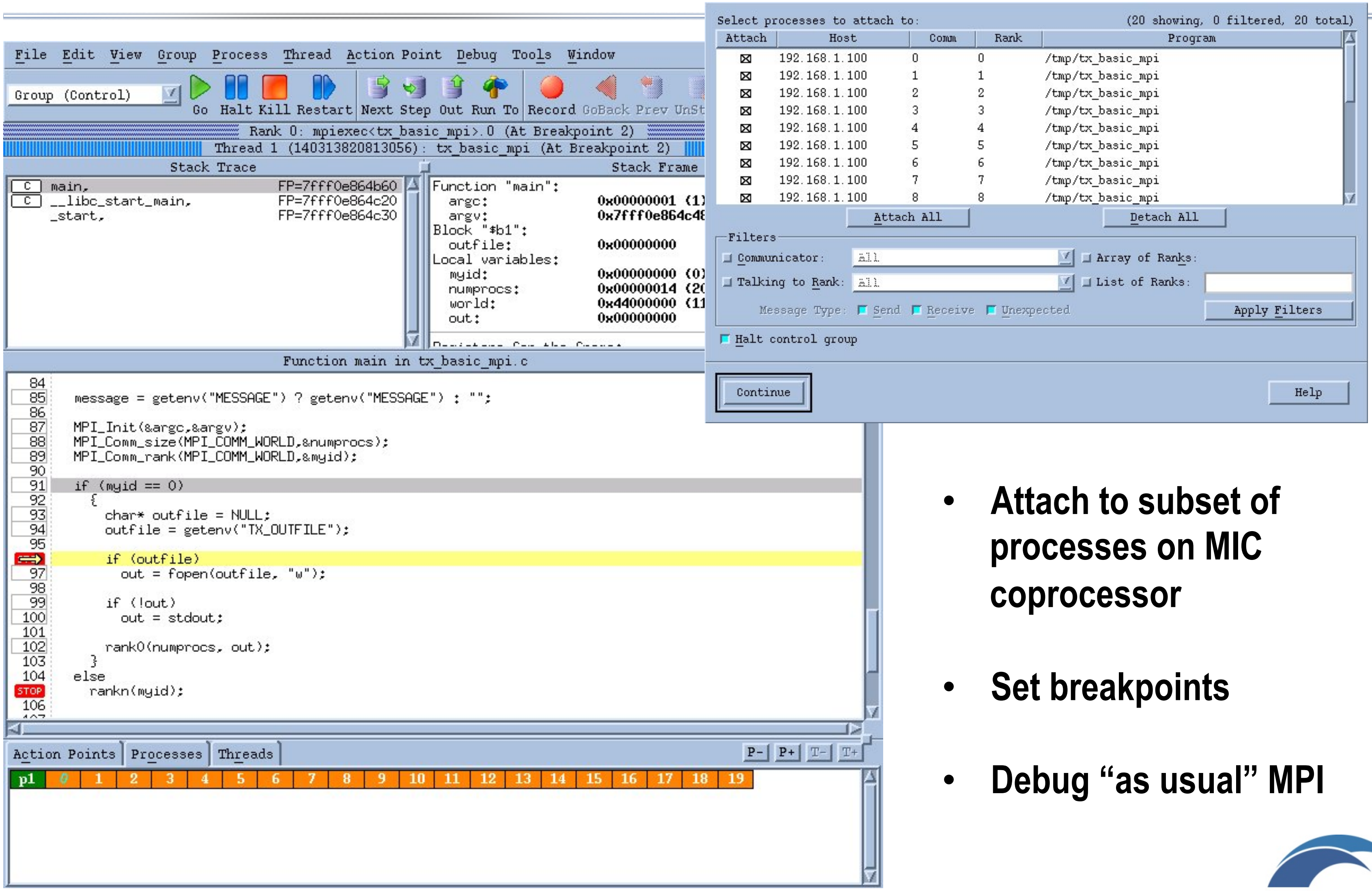
- Attach to running application

- See thread private data

- Investigate individual threads

- Kill stuck processes on MIC-coprocessor

Debugging MPI Applications



Rank 0: mpiexec<tx_basic_mpi>.0 (At Breakpoint 2)

Thread 1 (140313820813056): tx_basic_mpi (At Breakpoint 2)

```
C  main,                FP=7fff0e864b60
C  __libc_start_main,   FP=7fff0e864c20
   _start,              FP=7fff0e864c30
```

```
Function "main":
  argc:             0x00000001 (1)
  argv:             0x7fff0e864c48
Block "$b1":
  outfile:          0x00000000
Local variables:
  myid:             0x00000000 (0)
  numprocs:         0x00000014 (20
  world:            0x44000000 (11
  out:              0x00000000
```

Function main in tx_basic_mpi.c

```

  message = getenv("MESSAGE") ? getenv("MESSAGE") : "";

  MPI_Init(&argc,&argv);
  MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
  MPI_Comm_rank(MPI_COMM_WORLD,&myid);

  if (myid == 0)
    {
      char* outfile = NULL;
      outfile = getenv("TX_OUTFILE");

      if (outfile)
        out = fopen(outfile, "w");

      if (!out)
        out = stdout;

      rank0(numprocs, out);
    }
  else
    rankn(myid);

```

Select processes to attach to: (20 showing, 0 filtered, 20 total)

Attach	Host	Comm	Rank	Program
☒	192.168.1.100	0	0	/tmp/tx_basic_mpi
☒	192.168.1.100	1	1	/tmp/tx_basic_mpi
☒	192.168.1.100	2	2	/tmp/tx_basic_mpi
☒	192.168.1.100	3	3	/tmp/tx_basic_mpi
☒	192.168.1.100	4	4	/tmp/tx_basic_mpi
☒	192.168.1.100	5	5	/tmp/tx_basic_mpi
☒	192.168.1.100	6	6	/tmp/tx_basic_mpi
☒	192.168.1.100	7	7	/tmp/tx_basic_mpi
☒	192.168.1.100	8	8	/tmp/tx_basic_mpi

Filters

□ Communicator: All

□ Talking to Rank: All

□ Array of Ranks:

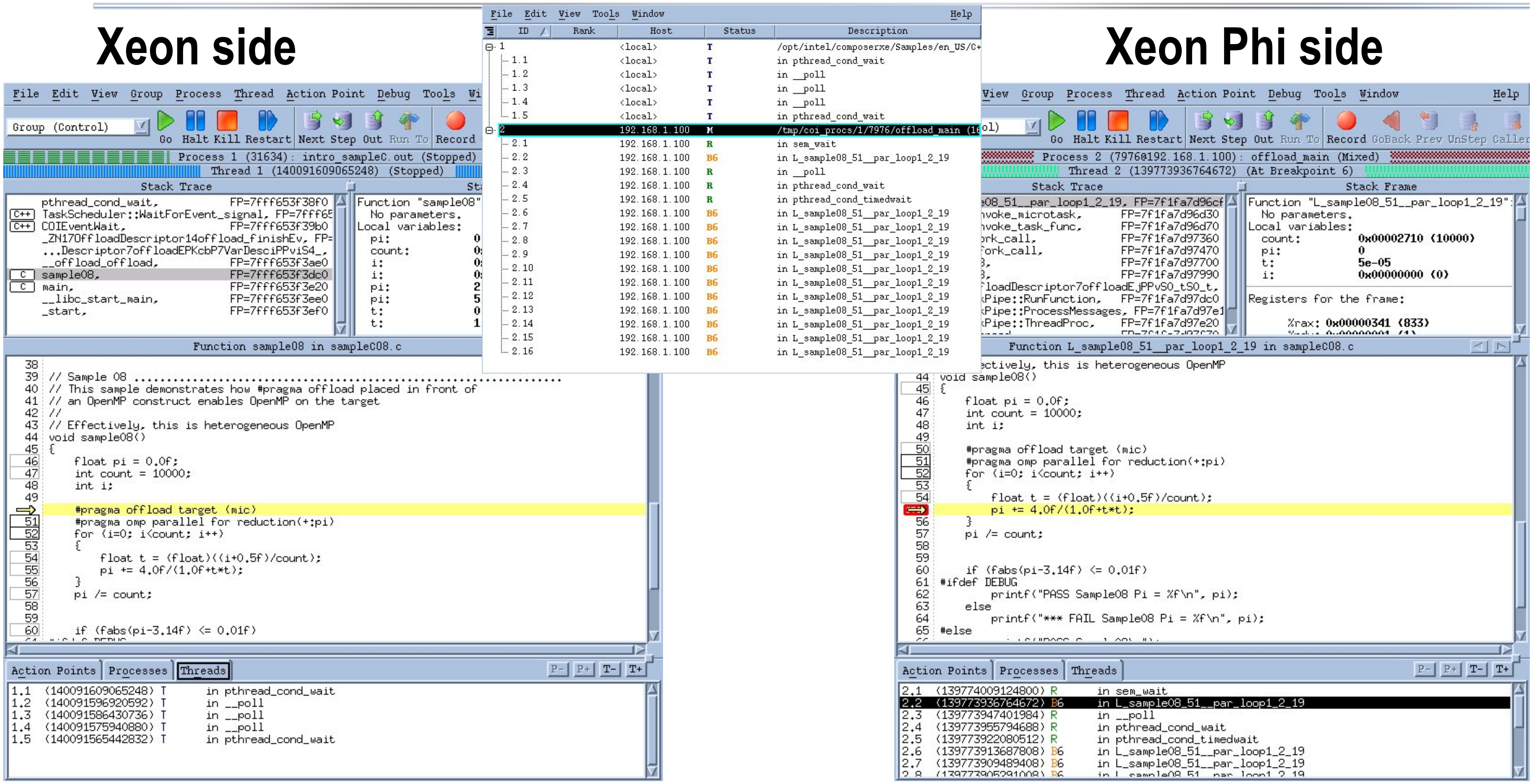
□ List of Ranks:

Message Type: ☑ Send ☑ Receive ☑ Unexpected

☑ Halt control group

- **Attach to subset of processes on MIC coprocessor**
- **Set breakpoints**
- **Debug "as usual" MPI**

Debugging Applications with Offloaded Code



Xeon side

Xeon Phi side

```
Process 1 (31634): intro_sampleC.out (Stopped)
Thread 1 (140091609065248) (Stopped)
```

```
    pthread_cond_wait,                FP=7fff653f38f0
C++ TaskScheduler::WaitForEvent_signal, FP=7fff65
C++ COIEventWait,                     FP=7fff653f39b0
    _ZN17OffloadDescriptor14offload_finishEv, FP=
    ...Descriptor7offloadEPKcbP7VarDesciPPviS4_,
    __offload_offload,                FP=7fff653f3ae0
C   sample08,                         FP=7fff653f3dc0
C   main,                             FP=7fff653f3e20
    __libc_start_main,                FP=7fff653f3ee0
    _start,                           FP=7fff653f3ef0
```

```
Function "sample08"
  No parameters.
Local variables:
  pi:        0
  count:     0:
  i:         0:
  i:         0:
  pi:        2
  pi:        5
  t:         0
  t:         1
```

Function sample08 in sampleC08.c

```c

// Sample 08 .............................................................
// This sample demonstrates how #pragma offload placed in front of
// an OpenMP construct enables OpenMP on the target
//
// Effectively, this is heterogeneous OpenMP
void sample08()
{
    float pi = 0.0f;
    int count = 10000;
    int i;

    #pragma offload target (mic)
    #pragma omp parallel for reduction(+:pi)
    for (i=0; i<count; i++)
    {
        float t = (float)((i+0.5f)/count);
        pi += 4.0f/(1.0f+t*t);
    }
    pi /= count;


    if (fabs(pi-3.14f) <= 0.01f)
```

```
1.1  (140091609065248) T      in pthread_cond_wait
1.2  (140091596920592) T      in __poll
1.3  (140091586430736) T      in __poll
1.4  (140091575940880) T      in __poll
1.5  (140091565442832) T      in pthread_cond_wait
```

```
ID      Rank  Host            Status  Description
1             <local>         T       /opt/intel/composerxe/Samples/en_US/C+
  1.1         <local>         T       in pthread_cond_wait
  1.2         <local>         T       in __poll
  1.3         <local>         T       in __poll
  1.4         <local>         T       in __poll
  1.5         <local>         T       in pthread_cond_wait
2             192.168.1.100   M       /tmp/coi_procs/1/7976/offload_main (16
  2.1         192.168.1.100   R       in sem_wait
  2.2         192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.3         192.168.1.100   R       in __poll
  2.4         192.168.1.100   R       in pthread_cond_wait
  2.5         192.168.1.100   R       in pthread_cond_timedwait
  2.6         192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.7         192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.8         192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.9         192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.10        192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.11        192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.12        192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.13        192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.14        192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.15        192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
  2.16        192.168.1.100   B6      in L_sample08_51__par_loop1_2_19
```

```
Process 2 (7976@192.168.1.100): offload_main (Mixed)
Thread 2 (139773936764672) (At Breakpoint 6)
```

```
e08_51__par_loop1_2_19, FP=7f1fa7d96cf
nvoke_microtask,       FP=7f1fa7d96d30
nvoke_task_func,       FP=7f1fa7d96d70
ork_call,              FP=7f1fa7d97360
fork_call,             FP=7f1fa7d97470
8,                     FP=7f1fa7d97700
8,                     FP=7f1fa7d97990
floadDescriptor7offloadEjPPvS0_tS0_t,
kPipe::RunFunction,    FP=7f1fa7d97dc0
kPipe::ProcessMessages, FP=7f1fa7d97e1
kPipe::ThreadProc,     FP=7f1fa7d97e20
```

```
Function "L_sample08_51__par_loop1_2_19":
  No parameters.
Local variables:
  count:      0x00002710 (10000)
  pi:         0
  t:          5e-05
  i:          0x00000000 (0)

Registers for the frame:

     %rax: 0x00000341 (833)
```

Function L_sample08_51__par_loop1_2_19 in sampleC08.c

```c
ectively, this is heterogeneous OpenMP
void sample08()
{
    float pi = 0.0f;
    int count = 10000;
    int i;

    #pragma offload target (mic)
    #pragma omp parallel for reduction(+:pi)
    for (i=0; i<count; i++)
    {
        float t = (float)((i+0.5f)/count);
        pi += 4.0f/(1.0f+t*t);
    }
    pi /= count;


    if (fabs(pi-3.14f) <= 0.01f)
#ifdef DEBUG
        printf("PASS Sample08 Pi = %f\n", pi);
    else
        printf("*** FAIL Sample08 Pi = %f\n", pi);
#else
```

```
2.1  (139774009124800) R      in sem_wait
2.2  (139773936764672) B6     in L_sample08_51__par_loop1_2_19
2.3  (139773947401984) R      in __poll
2.4  (139773955794688) R      in pthread_cond_wait
2.5  (139773922080512) R      in pthread_cond_timedwait
2.6  (139773913687808) B6     in L_sample08_51__par_loop1_2_19
2.7  (139773909489408) B6     in L_sample08_51__par_loop1_2_19
2.8  (139773905291008) B6     in L_sample08_51__par_loop1_2_19
```

One debugging session for MIC-accelerated code

ROGUE WAVE SOFTWARE

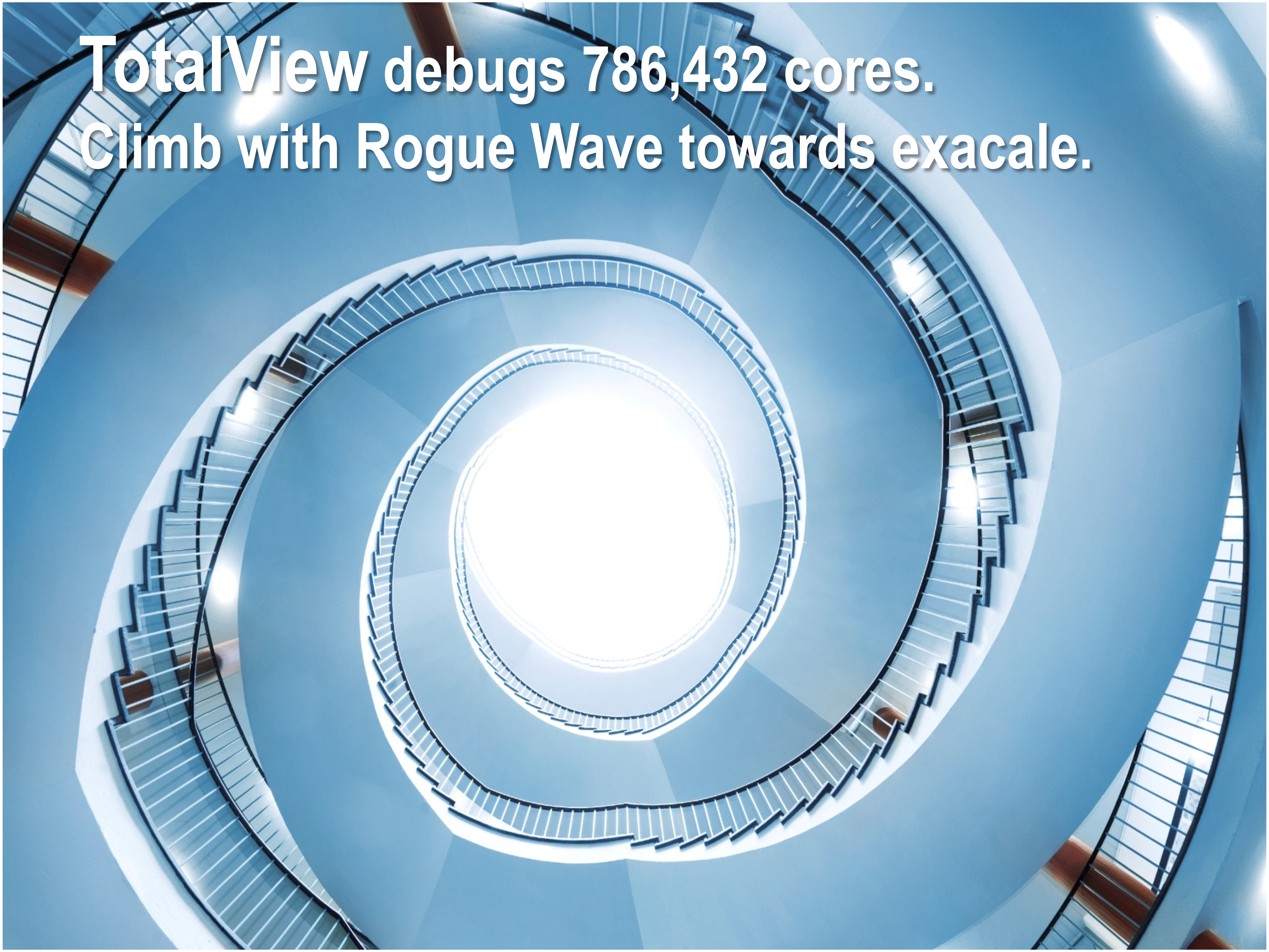
What’s New in TotalView 8.12

- Xeon Phi Support
- Formal support for Cray XC
- AVX Instruction Support (phase 1)
- Cray ATP Support

- Mac OS X Lion and Mountain Lion support
- Sessions Manager
- STL support for set, multi-set, multi-map
- Improvements for specifying addresses in C++ template breakpoints
- Updated OS and Compiler Support

Multi-phase R&D Projects Underway

- Massive Scalability
  - Collaboration with LLNL and Tri-lab partners
  - Targeting Cray, Blue Gene 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
- Working with customers through early access programs
  - Customer input is key to the success of both programs



TotalView debugs 786,432 cores.

Climb with Rogue Wave towards exacale.

Some more details on the 786,432 core test

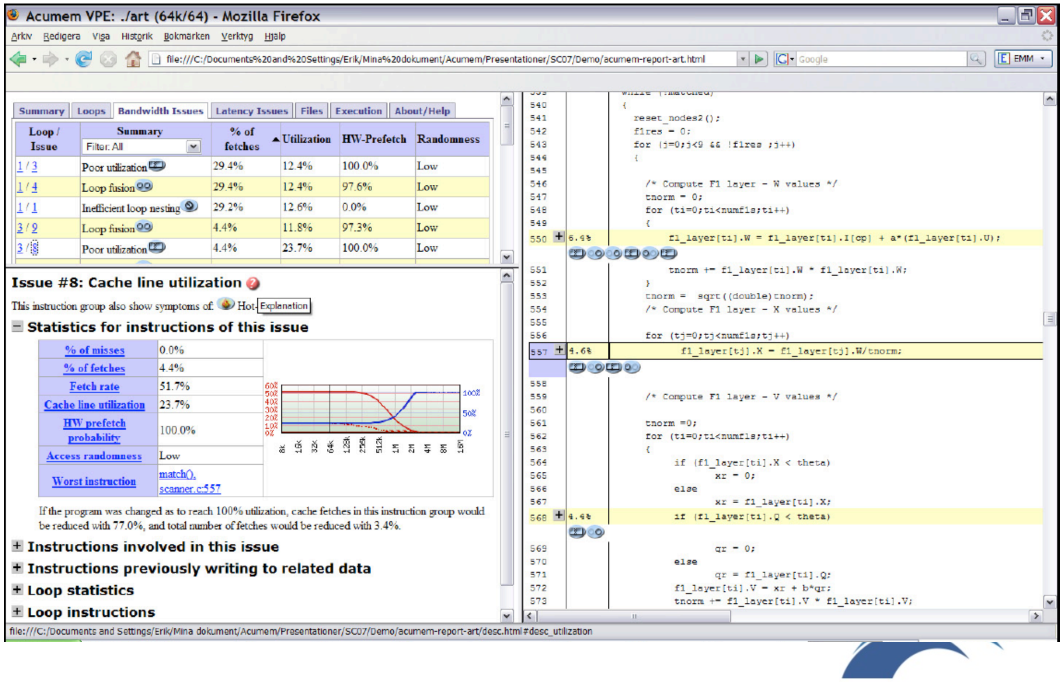
- The test was performed on 48 racks of Sequoia
- The test code
  - Implements a Jacobi Linear Equation Solver
  - The test code is a hybrid MPI + OpenMP code
  - 16 threads per process, one process per node
- The test operations
  - Start up
  - Setting breakpoints / removing breakpoints
  - Single stepping all threads
- Tests performed at a variety of scales to understand scalability

Second test - Oversubscription

- Same framework
  - same code
  - same machine
- Oversubscription
  - Scheduled more than one thread per physical core
  - This is a reasonable use case since the BG/Q supports 4 logical threads per core
- TotalView Debugged 1,048,576 threads

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 & Windows
  - Any compiled code
  - Runtime Analysis
    - Low overhead
  - Cache Modeling
    - Prioritizes Issues
    - Identifies Problem Lines of Code
  - Provides Advice
    - Explanations
    - Examples
    - Detailed statistics (if desired)


Acumem VPE: ./art (64k/64) - Mozilla Firefox
Arkiv Redigera Visa Historik Bokmärken Verktyg Hjälp
file:///C:/Documents%20and%20Settings/Erik/Mina%20dokument/Acumem/Presentationer/SC07/Demo/acumem-report-art.html
Google
EMM
Summary Loops Bandwidth Issues Latency Issues Files Execution About/Help
Filter: All
Explanation
60% 50% 40% 30% 20% 10% 0%
100% 50% 0%
8k 16k 32k 64k 128k 256k 512k 1M 2M 4M 8M 16M

Loop / Issue	Summary	% of fetches	Utilization	HW-Prefetch	Randomness
1 / 3	Poor utilization	29.4%	12.4%	100.0%	Low
1 / 4	Loop fusion	29.4%	12.4%	97.6%	Low
1 / 1	Inefficient loop nesting	29.2%	12.6%	0.0%	Low
3 / 9	Loop fusion	4.4%	11.8%	97.3%	Low
3 / 8	Poor utilization	4.4%	23.7%	100.0%	Low

Issue #8: Cache line utilization

This instruction group also show symptoms of: Hot-

Statistics for instructions of this issue

% of misses	0.0%
% of fetches	4.4%
Fetch rate	51.7%
Cache line utilization	23.7%
HW prefetch probability	100.0%
Access randomness	Low
Worst instruction	match(), scanner.c:557

If the program was changed as to reach 100% utilization, cache fetches in this instruction group would be reduced with 77.0%, and total number of fetches would be reduced with 3.4%.

Instructions involved in this issue

Instructions previously writing to related data

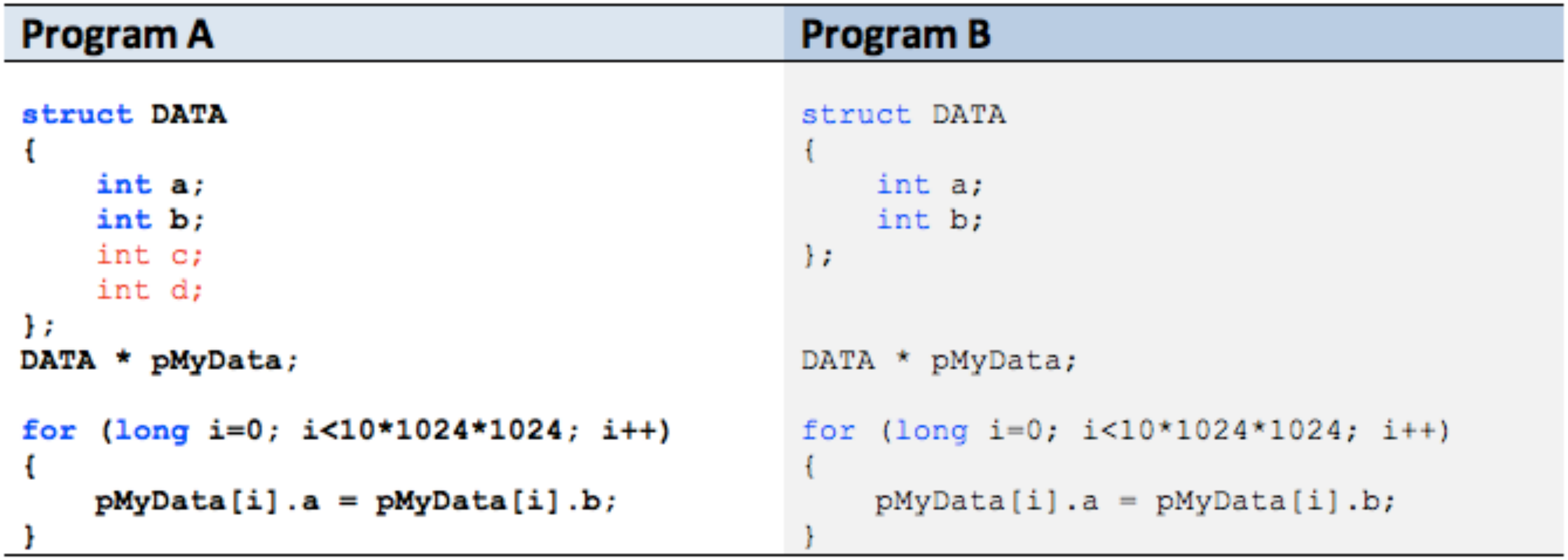
Loop statistics

Loop instructions

file:///C:/Documents and Settings/Erik/Mina dokument/Acumem/Presentationer/SC07/Demo/acumem-report-art/desc.html#desc_utilization

```
        while (!matched)
        {
          reset_nodes2();
          f1res = 0;
          for (j=0;j<9 && !f1res ;j++)
          {

            /* Compute F1 layer - W values */
            tnorm = 0;
            for (ti=0;ti<numf1s;ti++)
            {
6.4%            f1_layer[ti].W = f1_layer[ti].I[cp] + a*(f1_layer[ti].U);
                tnorm += f1_layer[ti].W * f1_layer[ti].W;
            }
            tnorm =  sqrt((double)tnorm);
            /* Compute F1 layer - X values */

            for (tj=0;tj<numf1s;tj++)
4.6%              f1_layer[tj].X = f1_layer[tj].W/tnorm;

            /* Compute F1 layer - V values */

            tnorm =0;
            for (ti=0;ti<numf1s;ti++)
            {
                 if (f1_layer[ti].X < theta)
                        xr = 0;
                 else
                        xr = f1_layer[ti].X;
4.4%             if (f1_layer[ti].Q < theta)
                        qr = 0;
                 else
                        qr = f1_layer[ti].Q;
                 f1_layer[ti].V = xr + b*qr;
                 tnorm += f1_layer[ti].V * f1_layer[ti].V;
```

Simple modifications can make a big difference



Program A

```
struct DATA
{
    int a;
    int b;
    int c;
    int d;
};
DATA * pMyData;

for (long i=0; i<10*1024*1024; i++)
{
    pMyData[i].a = pMyData[i].b;
}
```

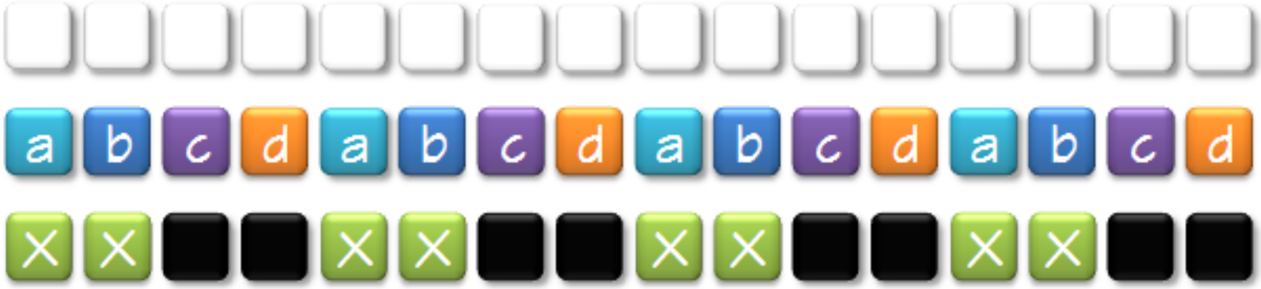
Program B

```
struct DATA
{
    int a;
    int b;
};


DATA * pMyData;

for (long i=0; i<10*1024*1024; i++)
{
    pMyData[i].a = pMyData[i].b;
}
```

Partially Used Structures

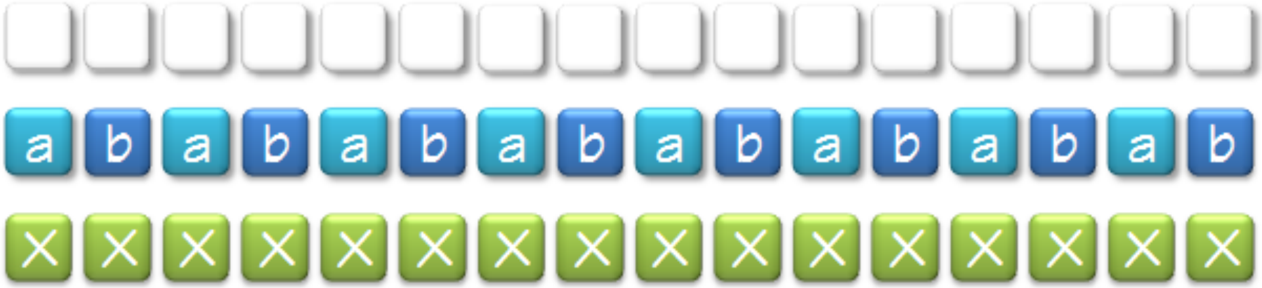
Partially Used Structures

Defined data structure includes a,b,c,d… but only uses a & b



50%

Redefined data structure includes a,b,a,b,a,b… c,d are elsewhere.



100%

Other opportunities for optimization include

- Alignment Problems
- False Sharing
- Excessive communication (cache coherence) traffic
- Temporal locality issues
- Spatial locality issues
- Loop fusion

Recent improvements to ThreadSpotter

- Improved parallel support
  - Support for sampling all MPI processes in an MPI job
  - Cray XT, XE, XK Support
    - ALPS, SLURM and Torque
  - Continued additions to the processor library
    - Including cross-processor analysis

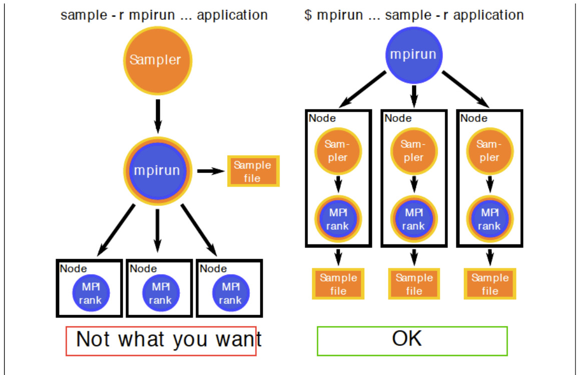

sample -r mpirun ... application
$ mpirun ... sample -r application
Sampler
mpirun
Sample file
Node
MPI rank
Sam-pler
Not what you want
OK

Figure A.1. MPI Sampling Principles

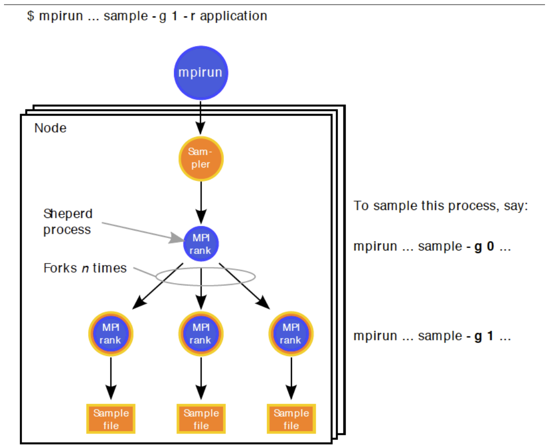

$ mpirun ... sample -g 1 -r application
mpirun
Node
Sam-pler
Sheperd process
Forks *n* times
MPI rank
Sample file
To sample this process, say:
mpirun ... sample - **g 0** ...
mpirun ... sample - **g 1** ...

Figure A.2. Message Passing Toolkit, runtime system and shepherd process

Next release: Improving ThreadSpotter MPI support

- Launchmon
  - Provides scalable mechanism for launching the tool in HPC clusters
  - Allows for coordination and synchronization of sampler activity
    - Will reduce “load balancing” bias that might otherwise be introduced by uncoordinated burst sampling with ThreadSpotter
  - Parallel framework can also be used for post-sampling processing
- Clustering Analysis
  - Some level of variability in sample results across the run
    - However the bulk of the results will be similar
  - Identify clusters of similar performance data
  - Present a small number (2-5) of reports that represent those clusters
  - Cluster analysis is done in parallel right after the sampling is completed

Rogue Wave Proprietary; plans and forward-looking statements subject to change without notice

ThreadSpotter work towards supporting the Xeon Phi

- Xeon Phi has an interesting cache architecture
  - L1 & L2 caches for each core
  - The set of all the L2 sometimes described as “shared”
  - L2 caches organized around a ring-shaped bus
    - Duplication of data referenced by more then one
  - Successful cache utilization is important to achieving performance
- Modeling and analysis of this cache architecture
- Sampler
  - Updated for Xeon Phi vector instructions
  - Scaling up the sampler for many-core thread parallelism
- Project is still ongoing

Rogue Wave Proprietary; plans and forward-looking statements subject to change without notice

Thanks!

- Talk to us here at CUG

- Contact me at: chris.gottbrath@roguewave.com
  - Sign up for the TotalVIew 8.12 beta (Xeon Phi)
  - Learn more about ThreadSpotter
  - Feedback, suggestions, use cases

- Learn more at: www.roguewave.com
  - White papers
  - Product Documentation
  - Videos
  - Product evaluation