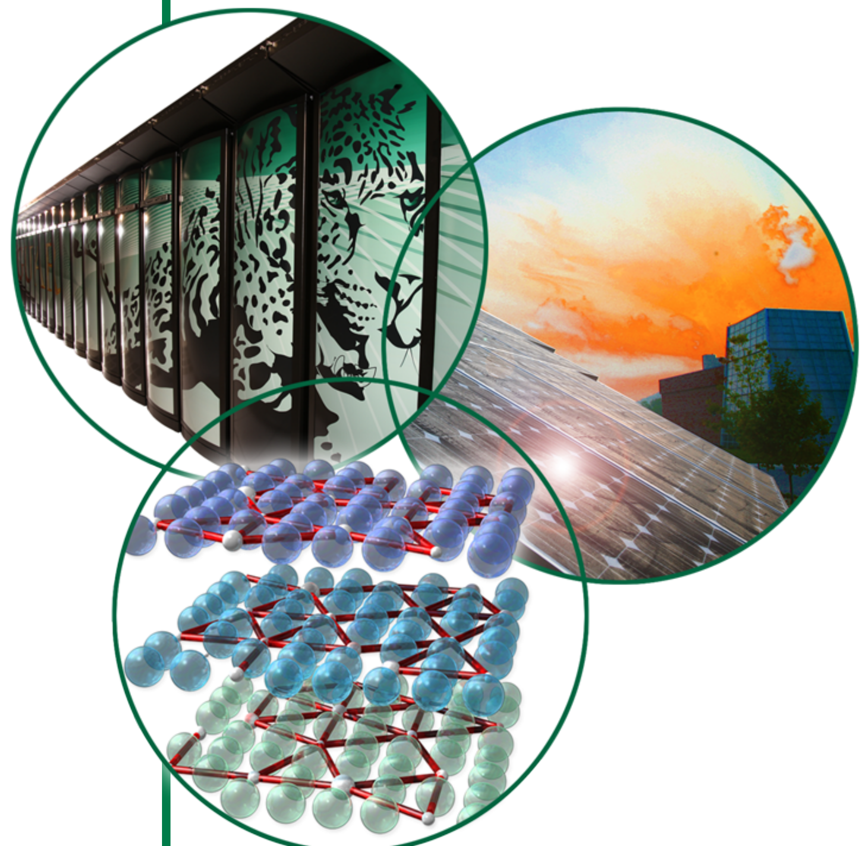
Recent Performance Analysis with Memphis

Collin McCurdy

Future Technologies Group



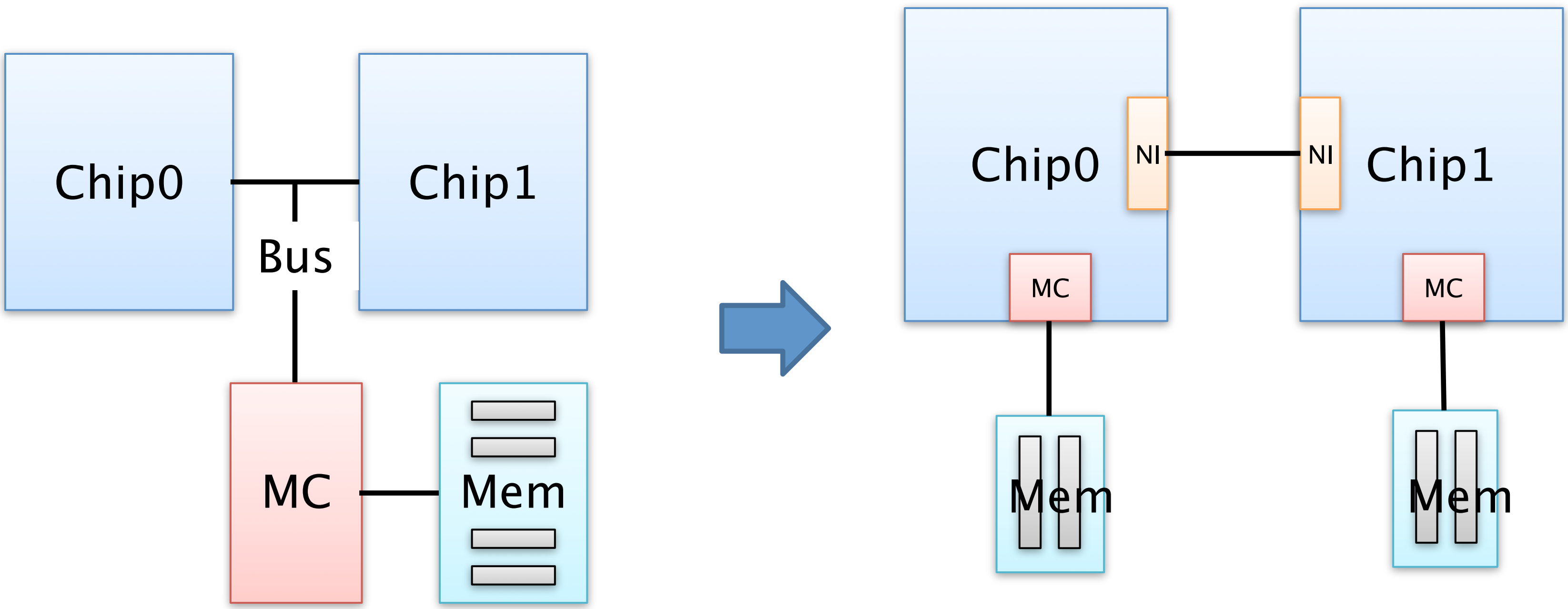
U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

Motivation

- Current projections call for each chip in an Exascale system to contain 100s to 1000s of processing cores
  - Already (~10 cores/chip) memory limitations and performance considerations are forcing scientific application teams to consider alternatives to “MPI-everywhere”
  - At the same time, trends in micro-processor design are pushing memory performance problems associated with Non-Uniform Memory Access (NUMA) to ever-smaller scales
- Memphis uses sampling-based hardware performance monitoring extensions to pinpoint the sources of memory system

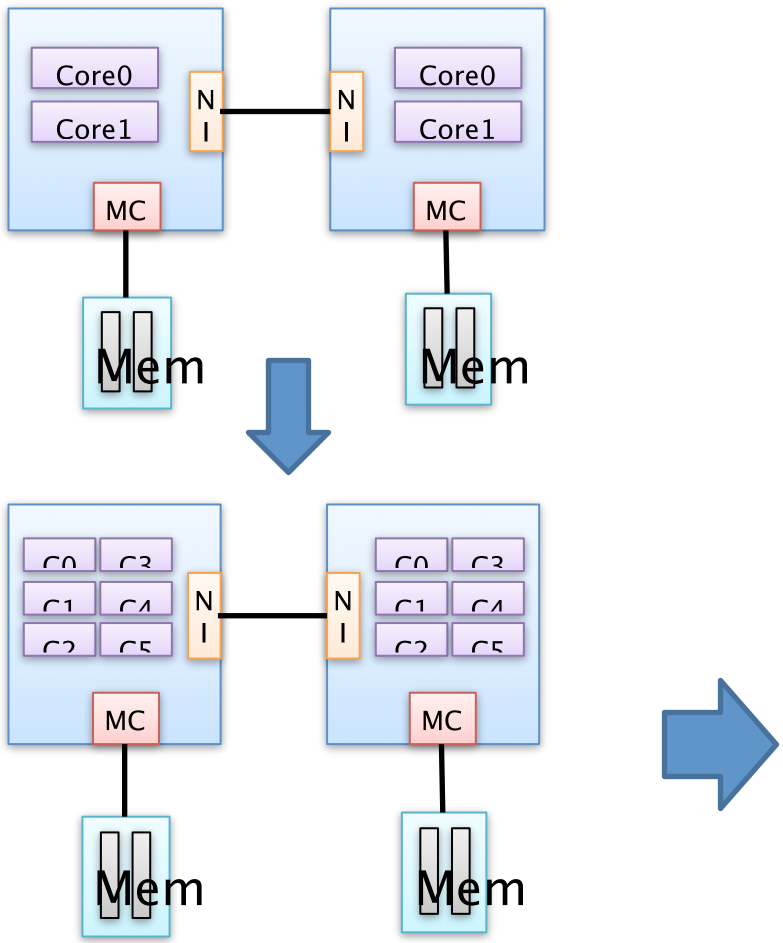
Why NUMA on 'SMP'?

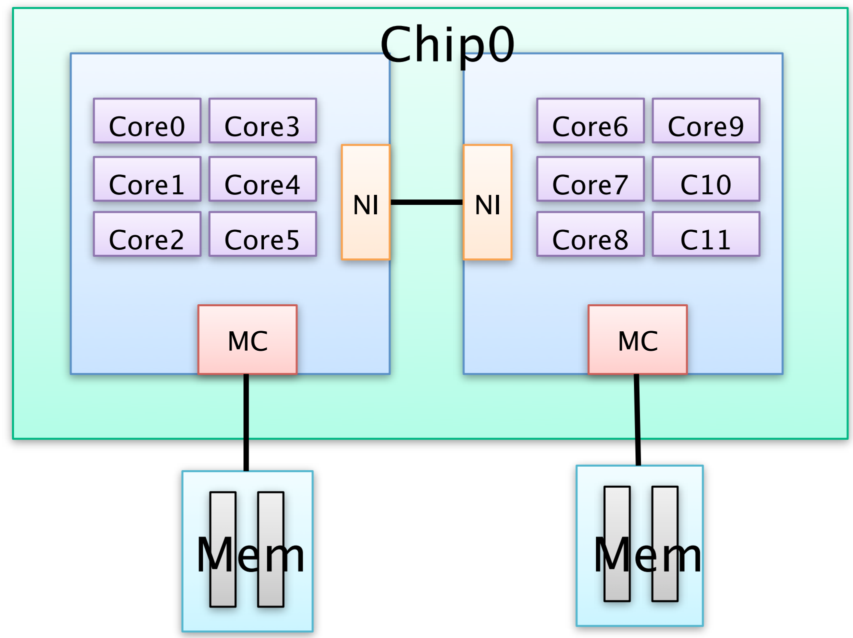


Multi-chip SMP systems used to be bus-based, limiting scalability.

On-chip memory controllers improve performance for local data, but non-local data requires communication.

Why NUMA on 'SMP'?





NUMA within socket.

More and more pressure on shared resources until eventually...

NUMA Performance Problems

- Typical performance problems associated w/ NUMA:
  - Hot-spotting
    - Due to poor initialization, memory not distributed across nodes
  - Computation/Data-partition mismatch
    - Memory distributed, but not appropriately
- NUMA can also amplify small performance bugs, turning them into significant problems
  - Example: contention for locks and other shared variables
    - NUMA can significantly increase latency (and thus waiting time), increasing possibility of further contention.

So, more for programmers to worry about, but there is Good News…

1. Mature infrastructure already exists for handling NUMA from software level
   - NUMA-aware operating systems, compilers and runtime
   - Based on years of experience with distributed shared memory platforms like SGI Origin/Altix
2. New access to performance counters that help identify problems and their sources
   - NUMA performance problems caused by references to remote data
   - Counters naturally located in Network Interface

Instruction-Based Sampling

- Hardware-based performance monitoring extensions
  - AMD -> IBS
  - Intel -> PEBS-LoadLatency extensions
- Similar to ProfileMe hardware introduced in DEC Alpha 21264
- Like event-based sampling, interrupt driven; but not due to cntr overflow
  - HW periodically interrupts, follows the next instruction through pipeline
  - Keeps track of what happens to and because of the instruction
  - Calls handler upon instruction retirement
- Provides the following data useful for finding NUMA problems:
  - Precise program counter of instruction
  - Virtual address of data referenced by instruction
  - Where the data came from: i.e., DRAM, another core's cache
  - Whether the agent was local or remote

Memphis

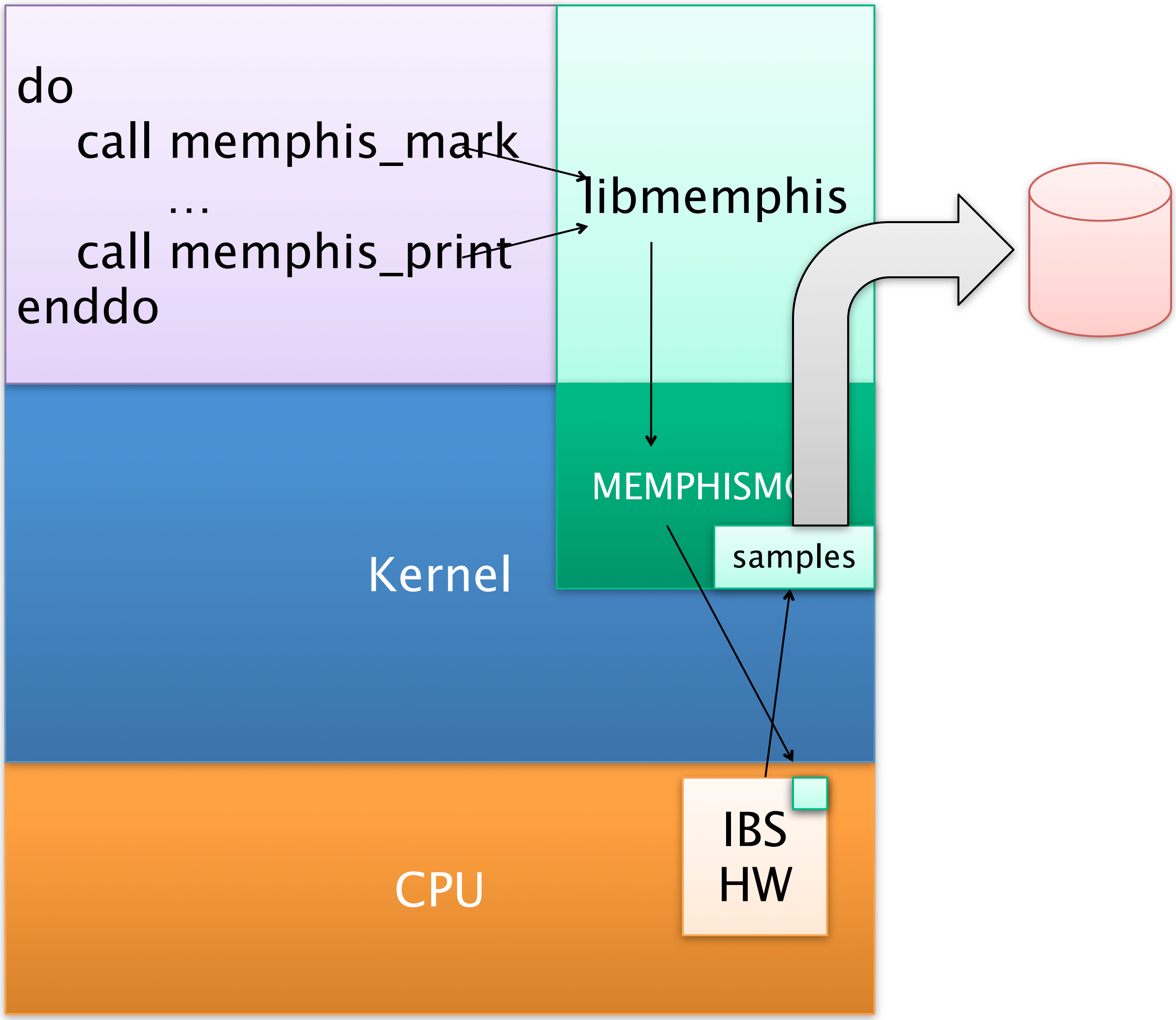
- Uses IBS hardware to pinpoint NUMA problems at source
- Data-centric approach
  - Sampling-based tools typically associate info w/

Key Insight: The source of a NUMA problem is not necessarily where it's evidenced

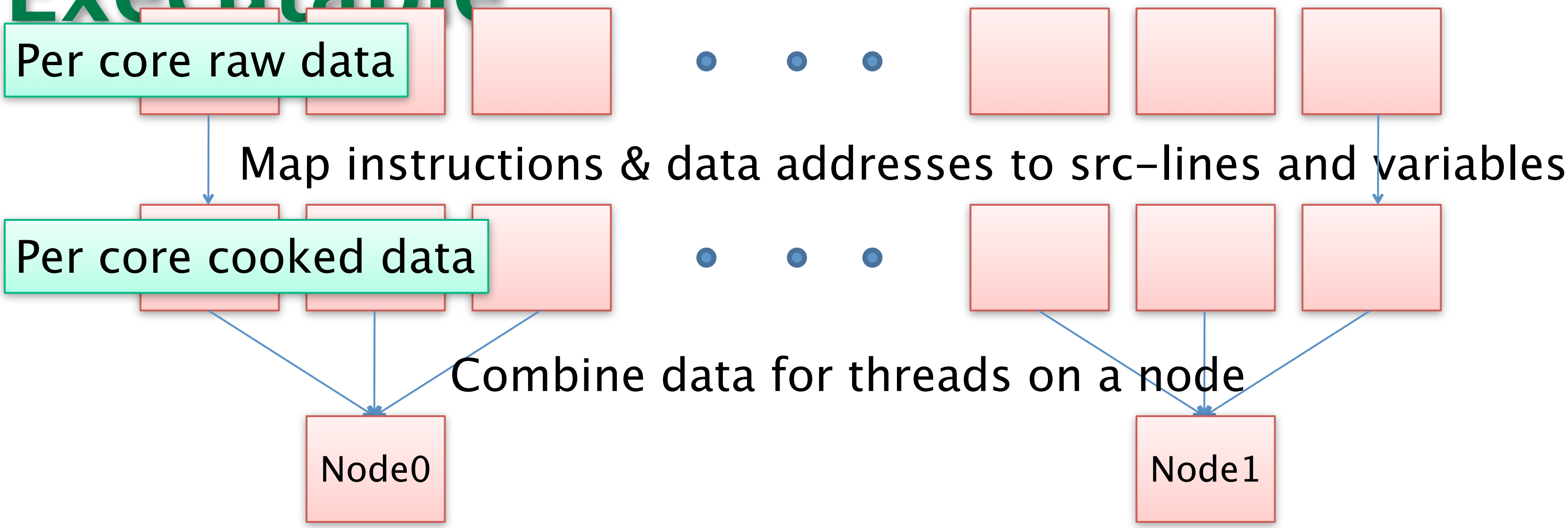
Key insight: The source of NUMA problem is not necessarily where it's evidenced

  - Example: Hot spot cause is variable init, problems evident at use
  - Programmers want to know
    - 1st what variable is causing problems
    - 2nd where (likely multiple sites)
- Consists of three components
  - Kernel module interface with IBS hardware

Memphis Runtime Components



Memphis Post-processing Executable



Challenges:

1) Instructions -> src-line mapping
   - Depends on quality of debug info; more likely to find loop-nest than line
2) Address -> variable mapping
   - Dynamic data (local vars in Fortran, global heap vars)

IBS Kernel Module (AMD)

- Most code stolen from Oprofile kernel module
- Differences in interrupt handler
  - Filter
    - Only interested in samples that went to Northbridge
  - User-level signaling
    - Currently used to implement watch-point addresses
  - Per-core sample buckets
    - Oprofile puts samples from all threads in a single bucket
  - Fixed-sized buffers
    - No handler for overflow

Recent Extensions

- Mapping addresses to dynamically allocated variables
- Port to Cray CNL
- Eclipse-based GUI

Allocation Instrumentation Tool

- Adds capability to map addresses to dynamically allocated variables
- Based on a Tau tool, built on top of Program Database Toolkit from University of Oregon
- Easily integrated into build process
  - Extra step in the rule to compile F90 files in Makefile
- At runtime, each dynamic allocation dumps variable-to-address-range mapping for use by post-processing tool
- Potential drawbacks
  - Adds overhead to each dynamic allocation
  - Requires access to source (i.e., cannot instrument libraries)

Memphis on Cray Platforms

- Compute Node Linux (CNL) is Linux-based
  - many components of Memphis work on Cray platforms without modification
- One exception: the kernel module
  - Several predefined kernel constants and functions not contained in the CNL distribution
  - Required finding and hard-coding values into calls that set configuration registers
- Kernel module port complicated by the black-box nature of CNL (not open-source)
  - Required the help of a patient Cray engineer (John Lewis) to perform first half of each iteration of the compile-install-test-modify loop
- Also required: mechanism for making Memphis available to jobs that want to use it

Runtime Policy and Configuration

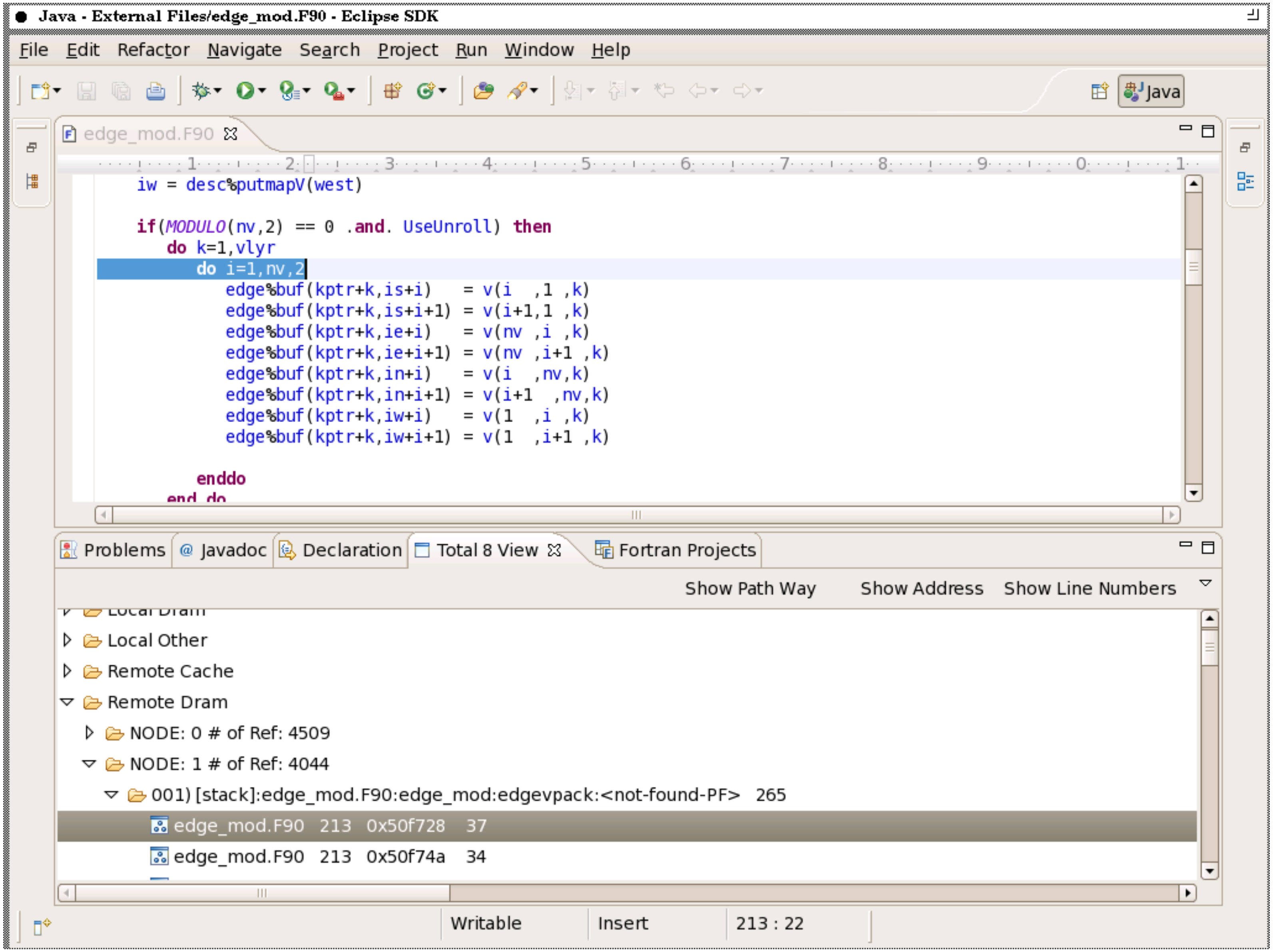
- Goal:
  - Maximize the availability of Memphis for selected users, while minimizing impact of a bleeding-edge kernel module on others
- Policy:
  - Kernel module is always available on a single, dedicated node of the system
    - On system reboots the kernel module is installed on the dedicated node and a device entry created in /dev
  - Users that want to access Memphis have a 'reservation' on that node
    - Realized as a Moab standing reservation
- Only one node provides sample data
  - We have found that this is sufficient for our needs
  - Intra-node performance is typically uniform across

Eclipse GUI

```
NODE: 0  total: 14
000) ~/apps/cesm1_0/cam-homme-ne2np4/cam:<sem2>  [ 0x1d00ea8 - 0x1d00eb0 ]  10
  ~/apps/cesm1_0/cam-homme-ne2np4/cam:<omp_set_lock>:0xaa022b    [ 0x1d00ea8 - 0x1d00eb0 ]  10
001) [map-anon-0]:<x_rbx>  [ 0x1fb0dd8 - 0x1fb0de0 ]  2
  ~/apps/cesm1_0/cam-homme-ne2np4/cam:<_mp_penter64>:0xaa0388    [ 0x1fb0dd8 - 0x1fb0de0 ]  2
002) ~/apps/cesm1_0/cam-homme-ne2np4/cam:<bar>  [ 0x1cc0540 - 0x1ccc708 ]  1
  ~/apps/cesm1_0/cam-homme-ne2np4/cam:<_mp_barrier>:0xa9ecb2    [ 0x1cc0540 - 0x1ccc708 ]  1
003) [heap]:<elem>  [ 0x51728b8 - 0x554dcb8 ]  1
  ~/apps/cesm1_0/cam-homme-ne2np4/./stepon.F90:262:0x97376a    [ 0x5492e40 - 0x5492e48 ]  1

NODE: 1  total: 914
000) [heap]:<edge%buf>  [ 0x5561ba0 - 0x56e4b48 ]  265
  ~/apps/cesm1_0/cam-homme-ne2np4/./edge_mod.F90:212:0x56081a    [ 0x55657c0 - 0x5694e88 ]  20
  ~/apps/cesm1_0/cam-homme-ne2np4/./edge_mod.F90:212:0x560825    [ 0x5566b40 - 0x56e39c8 ]  19
  ~/apps/cesm1_0/cam-homme-ne2np4/./edge_mod.F90:212:0x56084a    [ 0x55666c0 - 0x56c06a8 ]  19
  ~/apps/cesm1_0/cam-homme-ne2np4/./edge_mod.F90:212:0x56080a    [ 0x5563380 - 0x56db348 ]  17
  ~/apps/cesm1_0/cam-homme-ne2np4/./edge_mod.F90:212:0x560821    [ 0x5563b00 - 0x56c4888 ]  16
  ...
001) [heap]:<elem>  [ 0x51728b8 - 0x554dcb8 ]  242
  ~/apps/cesm1_0/cam-homme-ne2np4/./prim_advance_mod.F90:1648:0x7a3c3d    [ 0x5173eb8 - 0x5502450 ]  16
  ~/apps/cesm1_0/cam-homme-ne2np4/./prim_advance_mod.F90:2150:0x7a88f0    [ 0x5172a40 - 0x552a730 ]  12
  ~/apps/cesm1_0/cam-homme-ne2np4/./prim_advance_mod.F90:2150:0x7a88e5    [ 0x519ded8 - 0x5500b18 ]  11
  ~/apps/cesm1_0/cam-homme-ne2np4/./prim_advance_mod.F90:1798:0x7a585b    [ 0x5218100 - 0x54b0888 ]  10
  ~/apps/cesm1_0/cam-homme-ne2np4/./prim_advection_mod.F90:1911:0x7b848d    [ 0x5193538 - 0x5548ea8 ]  7
  ~/apps/cesm1_0/cam-homme-ne2np4/./derivative_mod.F90:1983:0x5226dc    [ 0x5242b40 - 0x54d87c8 ]  6
  ~/apps/cesm1_0/cam-homme-ne2np4/./prim_advection_mod.F90:1301:0x7b0ef0    [ 0x51e5fe8 - 0x551fdd0 ]  6
  ~/apps/cesm1_0/cam-homme-ne2np4/./prim_advance_mod.F90:1648:0x7a3c44    [ 0x5173278 - 0x5502710 ]  5
  ...
```

Eclipse GUI


Java - External Files/edge_mod.F90 - Eclipse SDK
File Edit Refactor Navigate Search Project Run Window Help
Java
edge_mod.F90
iw = desc%putmapV(west)
if(MODULO(nv,2) == 0 .and. UseUnroll) then
do k=1,vlyr
do i=1,nv,2
edge%buf(kptr+k,is+i) = v(i ,1 ,k)
edge%buf(kptr+k,is+i+1) = v(i+1,1 ,k)
edge%buf(kptr+k,ie+i) = v(nv ,i ,k)
edge%buf(kptr+k,ie+i+1) = v(nv ,i+1 ,k)
edge%buf(kptr+k,in+i) = v(i ,nv,k)
edge%buf(kptr+k,in+i+1) = v(i+1 ,nv,k)
edge%buf(kptr+k,iw+i) = v(1 ,i ,k)
edge%buf(kptr+k,iw+i+1) = v(1 ,i+1 ,k)
enddo
Problems
Javadoc
Declaration
Total 8 View
Fortran Projects
Show Path Way
Show Address
Show Line Numbers
Local Other
Remote Cache
Remote Dram
NODE: 0 # of Ref: 4509
NODE: 1 # of Ref: 4044
001) [stack]:edge_mod.F90:edge_mod:edgevpack:<not-found-PF> 265
edge_mod.F90 213 0x50f728 37
edge_mod.F90 213 0x50f74a 34
Writable
Insert
213 : 22

Memphis Evaluation

- Quick demonstration of two aspects of ‘performance’
  - Runtime overhead
  - Usefulness
    - Application performance improvements

Runtime Overhead

	IBS Off,	IBS On,
Base	40.69	41.18
Mod1	36.29	36.63
Mod2	35.90	36.31

- Even with allocation statements instrumented, overhead is ~1%.

Performance Improvements: CESM

- Memphis-directed changes to one file (of many).
- Performance of 12 threads (two NUMA nodes)

Current Work

- Problem with IBS: refs to outstanding misses
  - Secondary references to blocks serviced from the Northbridge are marked as L1 hits, albeit with extremely long latency
- Can lead to false negatives
  - Apparent ‘fix’, indicated by lower remote reference counts, doesn’t improve performance as expected.
- Exploring modifications to filtering mechanism in kernel module
  - Let through long-latency L1 hits
  - Unfortunately, latency can have other causes
    - Resource contention

Conclusion

- NUMA is already a problem, and it will only get worse...but there is hope.
  - Memphis is a toolset that uses sampling-based hardware performance monitoring extensions to pinpoint the sources of memory performance problems
  - Memphis is now available on Cray platforms
  - We have used Memphis to find and fix significant problems in several large-scale production applications
- Want us to look at an application? Let us know!