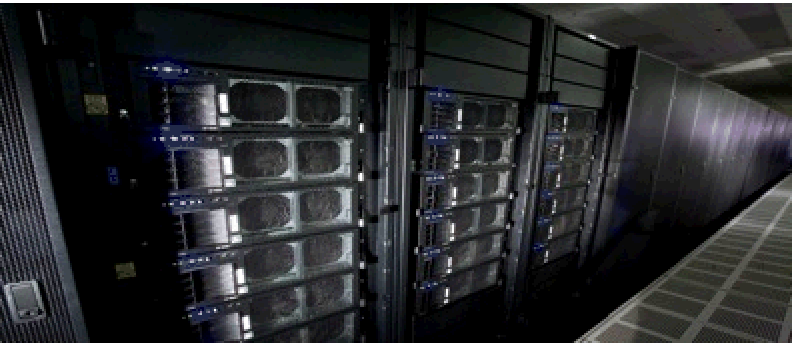
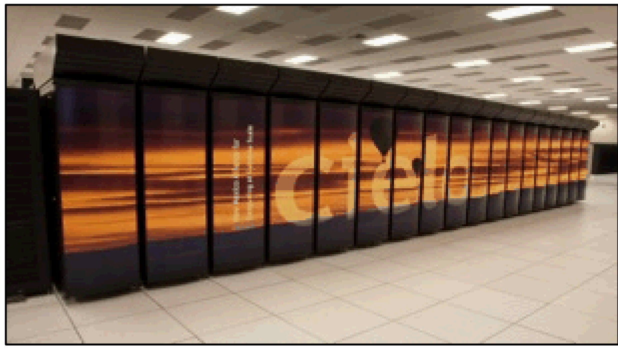
SAND2014-0622P

Exceptional service in the national interest

Sandia National Laboratories





Application Memory Analysis

D. Joseph
IBM, Austin

E. Cooper-Balis
Micron

S.D. Hammond, B.J. Moore, A.F. Rodrigues and D.R. Resnick
Scalable Computer Architectures
Sandia National Laboratories, NM

Outline

- Overview of the Research Problem
  - Describing why are we trying to generate traces of memory?
- Toolkit and Simulation Discussion
- Application Work
- Results and Analysis – Doug Joseph will discuss
- Wrap Up and Questions?

Problem Context

- Want to understand the performance of memory
  - Note – this is not the full memory hierarchy or memory subsystem
  - New memory technologies – think HMC, AMC, HBM *etc*
  - Opportunities for optimization (buffering, queuing, coalescing *etc*)
- Means we need to extract addresses arriving at the actual memory components
  - Filtered for cache efficiencies
  - Potentially augmented with memory subsystem prefetching
  - Aligned with the coherency protocol used in the system
  - Trace the addresses and save for replaying in optimization study
- Need a fair amount of addresses to make this exercise worth it

Typical Methods

- "Roofline" estimates – approximate guesses from counters
- Trace all of an application's memory requests (e.g. PIN, SunShade, Valgrind *etc*)
  - But… this means you don't get cache filtering, problem for busy-wait loops, highly cache friendly codes, blocking, prefetching *etc*
  - Hard to work out cost of coherency/coherency effects
  - Means your results can lose accuracy, especially for "optimized" codes
- Apply a simple cache filter to a memory trace
  - Better accuracy but still can yield poor results for timing, coherency *etc*

- **Ideal:** a fast cycle accurate simulator
  - Requirement for long duration of simulation makes full cycle accurate simulation very expensive – we need something cheaper

APPLICATION MEMORY ANALYSIS

Lightweight Processor Core

- **Ideal:** a <u>fast</u> cycle accurate simulator
  - Requirement for long duration of simulation makes full cycle accurate simulation very expensive – we need something cheaper

- This implies a lightweight processor core emulator
  - Gives reasonable approximation to an out-of-order core behavior
  - Requires limited instruction decode (sufficient to extract information)
  - Works will existing binaries – want (well) optimized applications
  - Supports multi-threaded OpenMP execution
  - Supports approximated virtual to physical address translation

- Events from processor core drive a more accurate model of the memory subsystem

Lightweight Processor Core

- We utilize a set of custom tools based on Intel's PIN library and XED2 X86 decoder ("Ariel" front-end)
  - Enables extraction of memory requests and virtual addresses
  - Enables blocking of requests into instructions so we can control issue rate
- Ariel then offloads output of application analysis into the SST simulation toolkit
- The Ariel back-end then:
  - Maps virtual addresses to physical locations
  - Issues memory requests into the L1 caches
  - Causes back-pressure into the front-end when core resources are all busy
  - Control issues rates, queues *etc* to approximate performance

Cache Models

- The SST Toolkit includes a variety of cache models:
  - Simple caches with limited to no coherency (for emulation)
  - Traditional Cache models which implement MESI protocol
  - Advanced research caches – different caching schemes, eviction policies, flexible coherency, cache filtering mechanisms *etc*

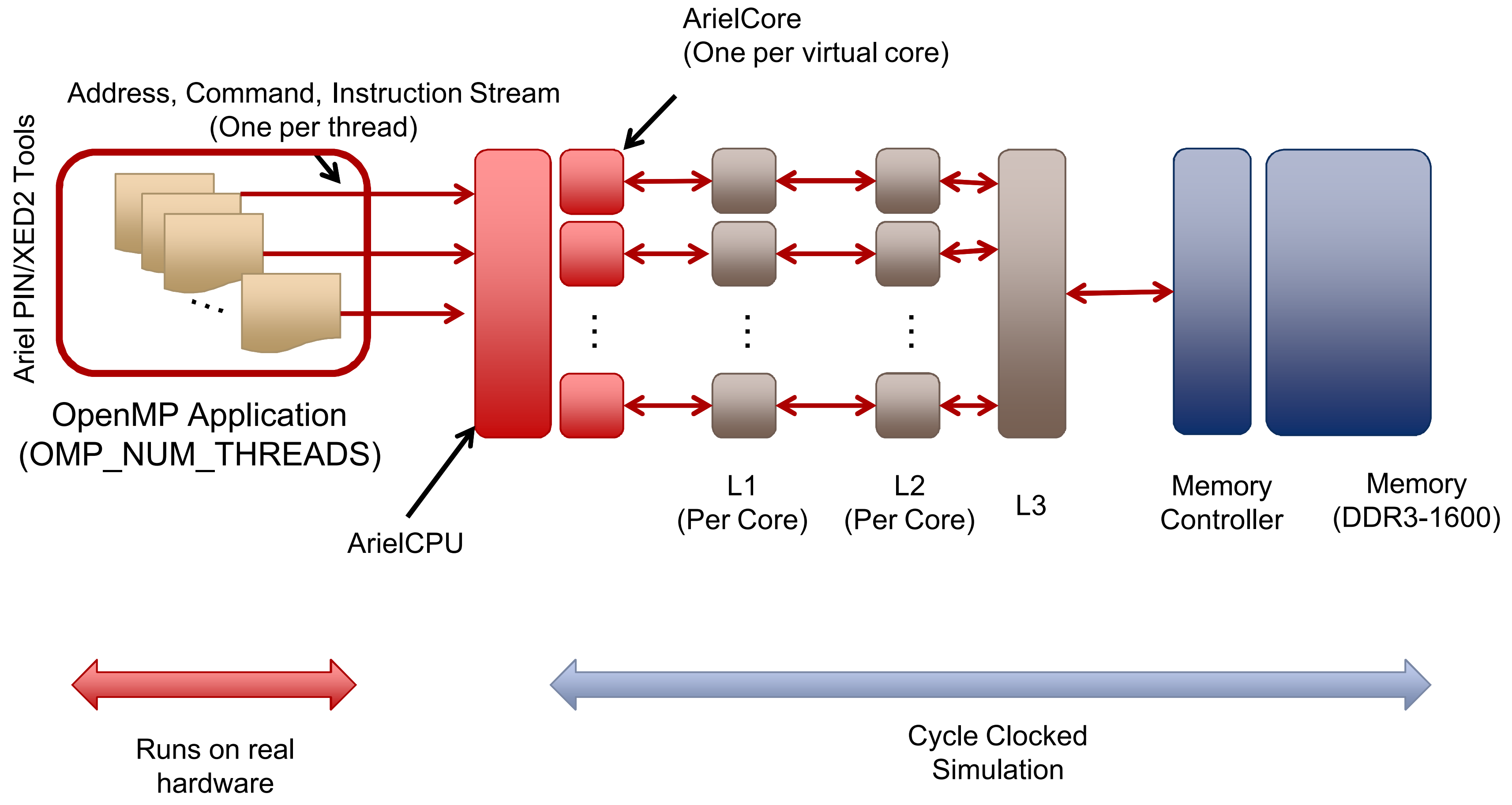
- For these experiments we are using the traditional cache models

- Caches can be connected using:
  - Traditional buses
  - Point-to-point on-chip networking (“NoC”)

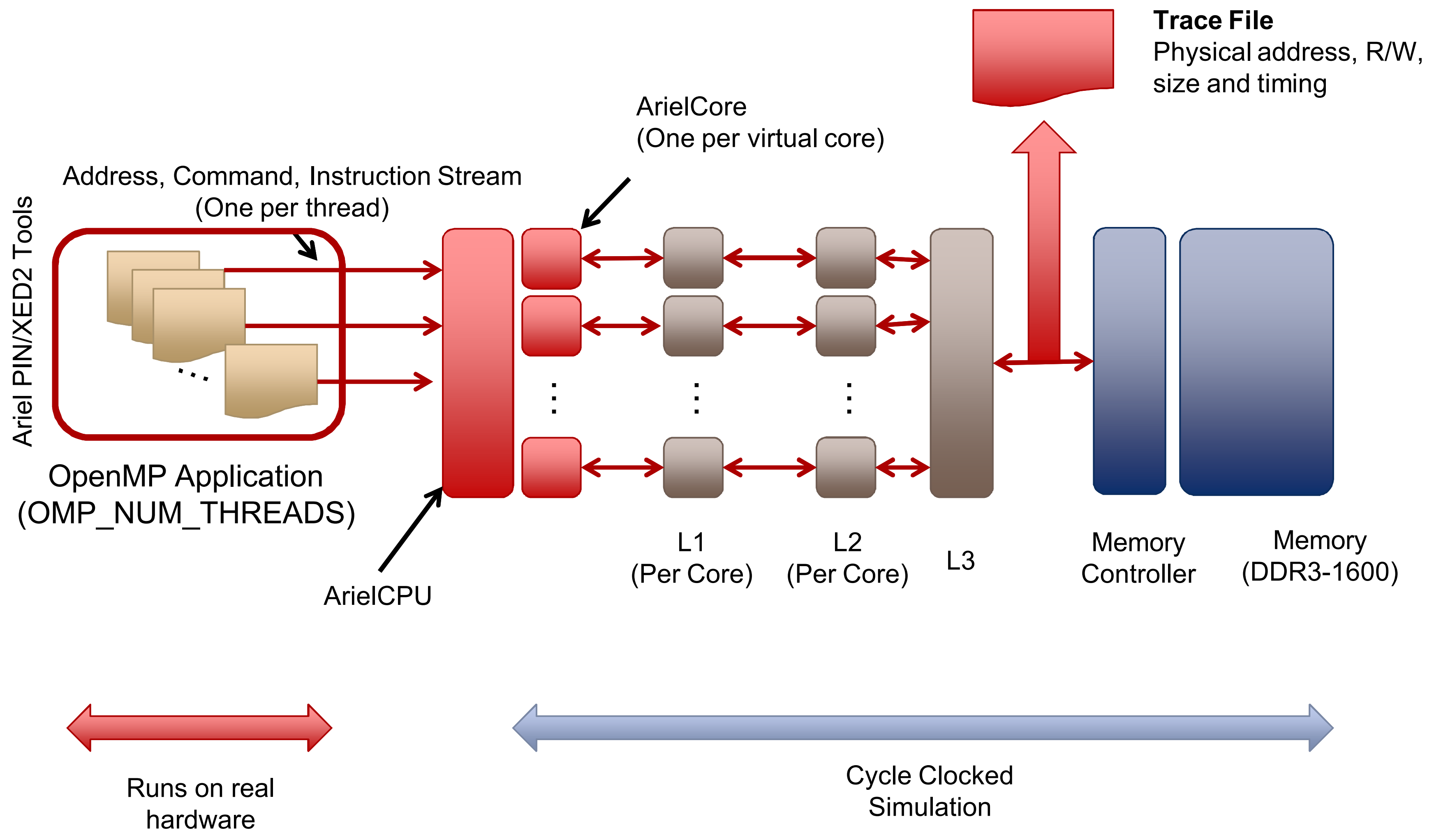
Memory Models

- SST implements several memory models:
  - Simple memories which have a fixed latency before returning requests, bandwidth dictated by frequency of processing
  - Cycle-accurate models of DRAM (using DRAMSim)
  - Models for advanced memory technologies including NVRAM, GDDR and others
- For these experiments we utilize a cycle-accurate DRAM model set to perform at DDR3-1600
  - Easy to change and we can do coarser analysis using simple memory models when required

Tracing Configuration



Tracing Configuration



Using SST cache and memory listening interfaces we can trap traces at any point during execution (to file, screen, UNIX pipes, etc)

Ariel Cores

- Provide an approximate page allocation mechanism so we can investigate virtual-to-physical layout challenges
  - "Basic mode" – trap virtual addresses you have not seen before, allocate and page and then map
  - "Intermediate mode" – use a special memory allocation on a per-data structure basis to run "special" page allocations (application can include hints in request)
  - "Full mode" – intercept malloc, calloc, realloc, posix_memalign *etc*, and then use these to produce a mapping in memory

- Generally basic mode is sufficient for this kind of study
  - More advanced work in our research labs investigating "smarter" strategies using the above

Ariel CPU

- Receives stream of instructions, hints etc from the running program via a UNIX pipe (which causes back pressure)
- Aggregates and schedules to improve performance
- Allocates instructions and requests to the various CPU cores ensuring core resources remain respect (e.g. queues, pending requests *etc*)
- Handles special function communication (e.g. “malloc”)

Application Work

- Using NNSA and Office of Science mini-applications and benchmarks as a first cut
  - AMG 2013
  - UMT 2013
  - MiniFE
  - LULESH
  - CoMD
  - SNAP

- Traces are for 8-core OpenMP runs (approximately one Sandy Bridge)
- Replicate those trace instances with address shifts to approximate many MPI ranks on same memory system

Analysis Results

- Doug Joseph (IBM) and Elliott Cooper-Balis (Micron) are working with the traces and will give an overview of initial analysis

- Plans are to increase scale of simulation and complexity of processor design
  - Considering approximation of a POWER8 node
  - Possible heavily multi-core chip
  - Increase application complexity

- http://www.sst-simulator.org

Sandia National Laboratories

Exceptional service in the national interest