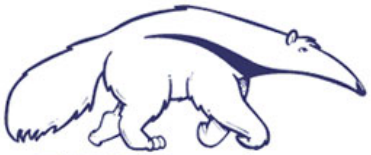
Parallel gem5 Simulation of Many-Core Systems with Software-Programmable Memories

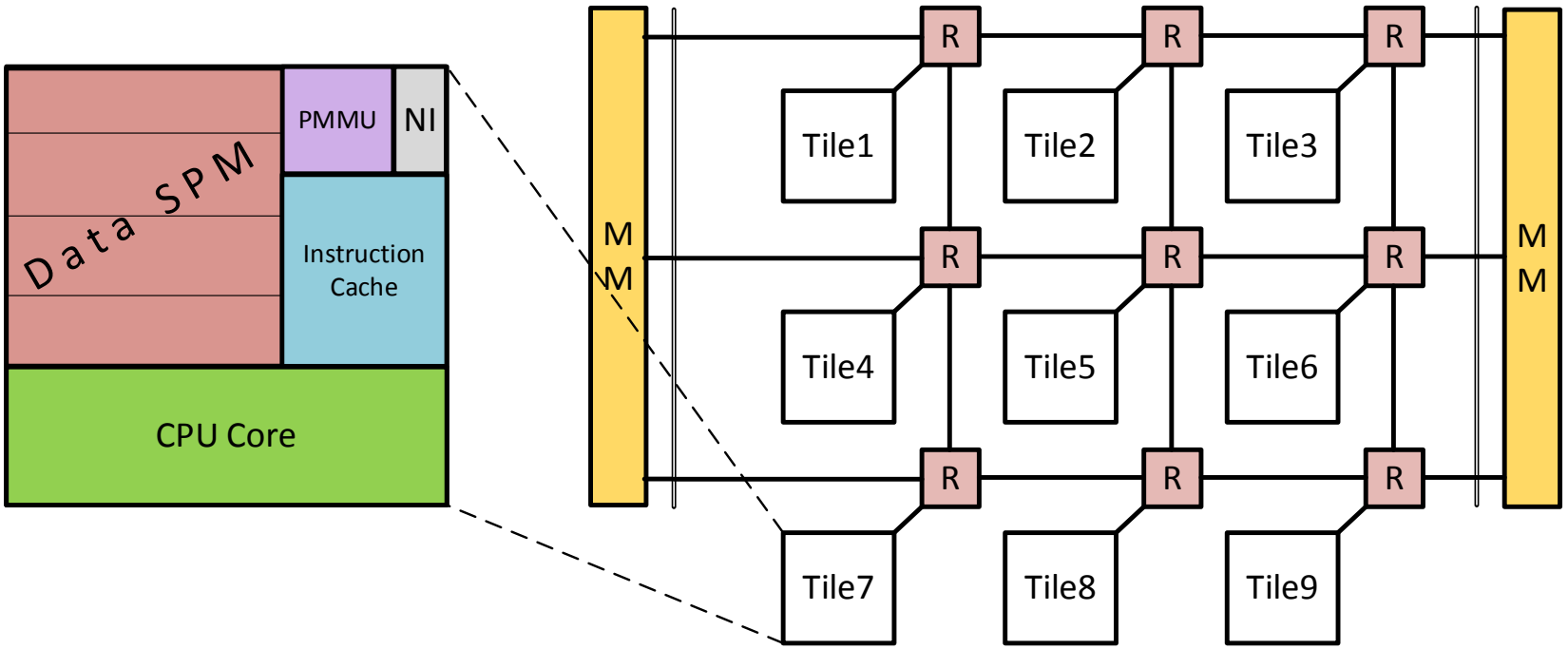
Bryan Donyanavard*
Tiago Muck*
Majid Shoushtari*
Nikil Dutt



Computer Science

*all listed authors are equal contributors

Target Platform



- A mesh-based many core architecture with distributed data software programmable memories

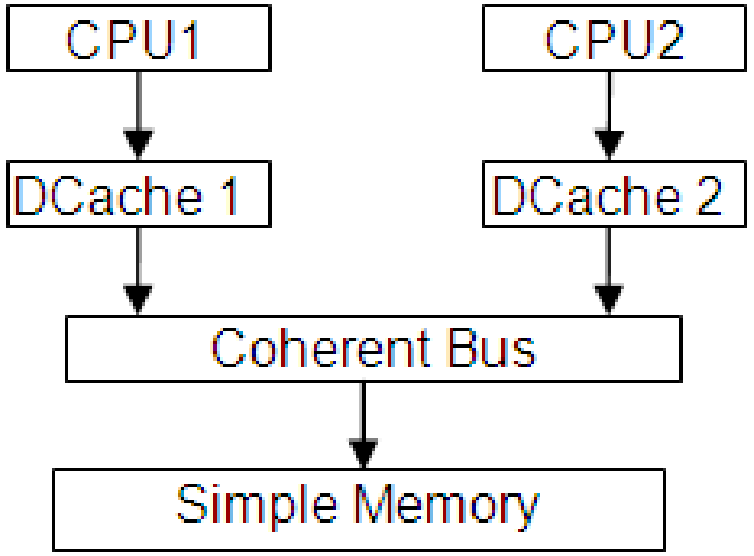
Enabling Simulation of Target Platform

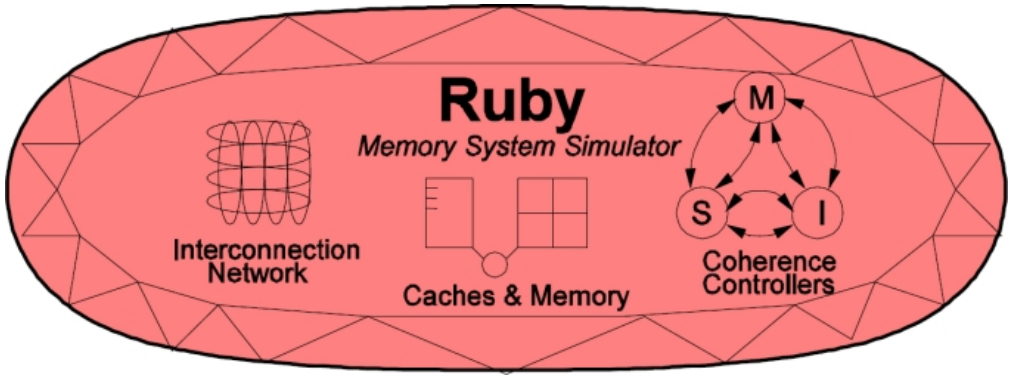
- Software Programmable On-chip Memories (SPMs)
  - SPM Architectural Components
  - SPM Programming API
  - Communication Infrastructure for Distributed SPMs
- Simulator Speedup via Parallelization
  - Event Queues

SPM Integration for Many-cores

Existing Memory Hierarchies

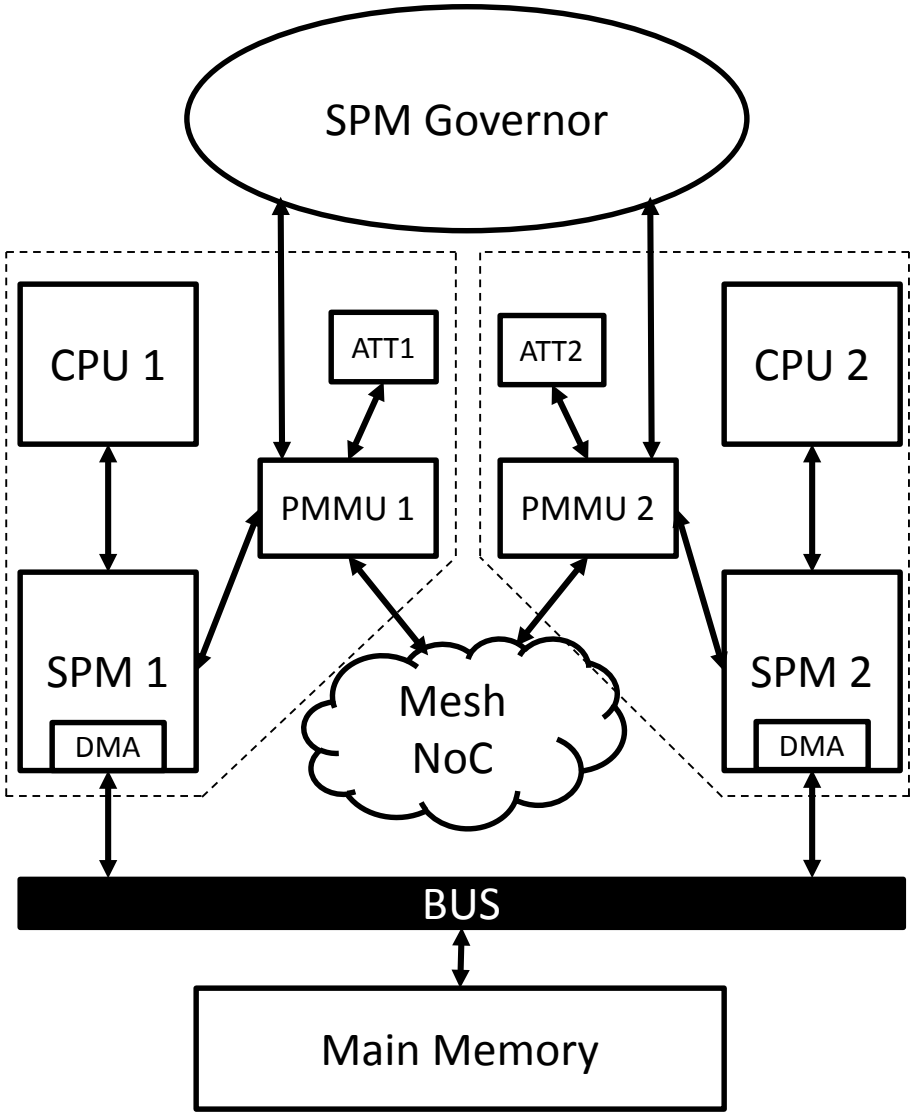
- Classic Memory Model
  - No explicit memory controller
  - No NoC
- Ruby
  - Protocol dependent





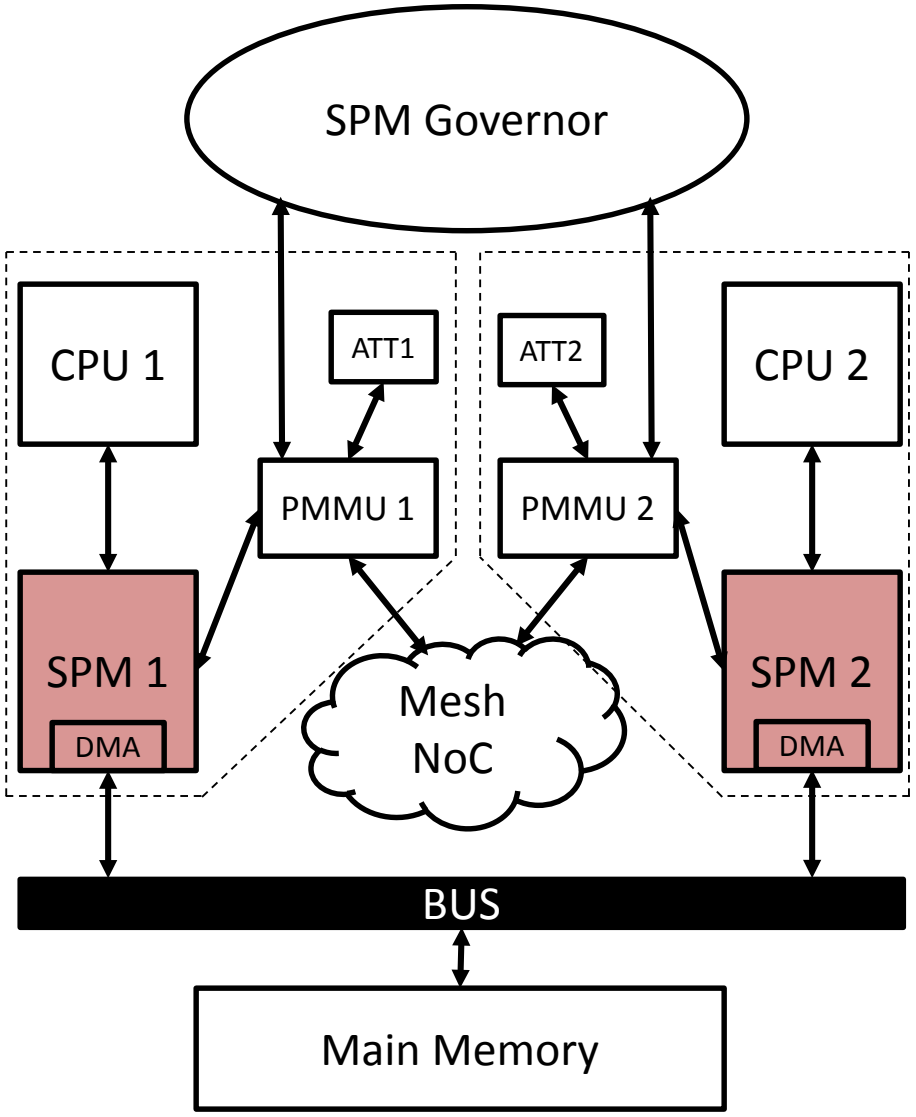
New Memory Hierarchy

- We add the following infrastructure to the Classic Memory Model
  - SPM
  - Paged Memory Management Unit (PMMU)
  - Address Translation Table (ATT)
  - SPM Governor



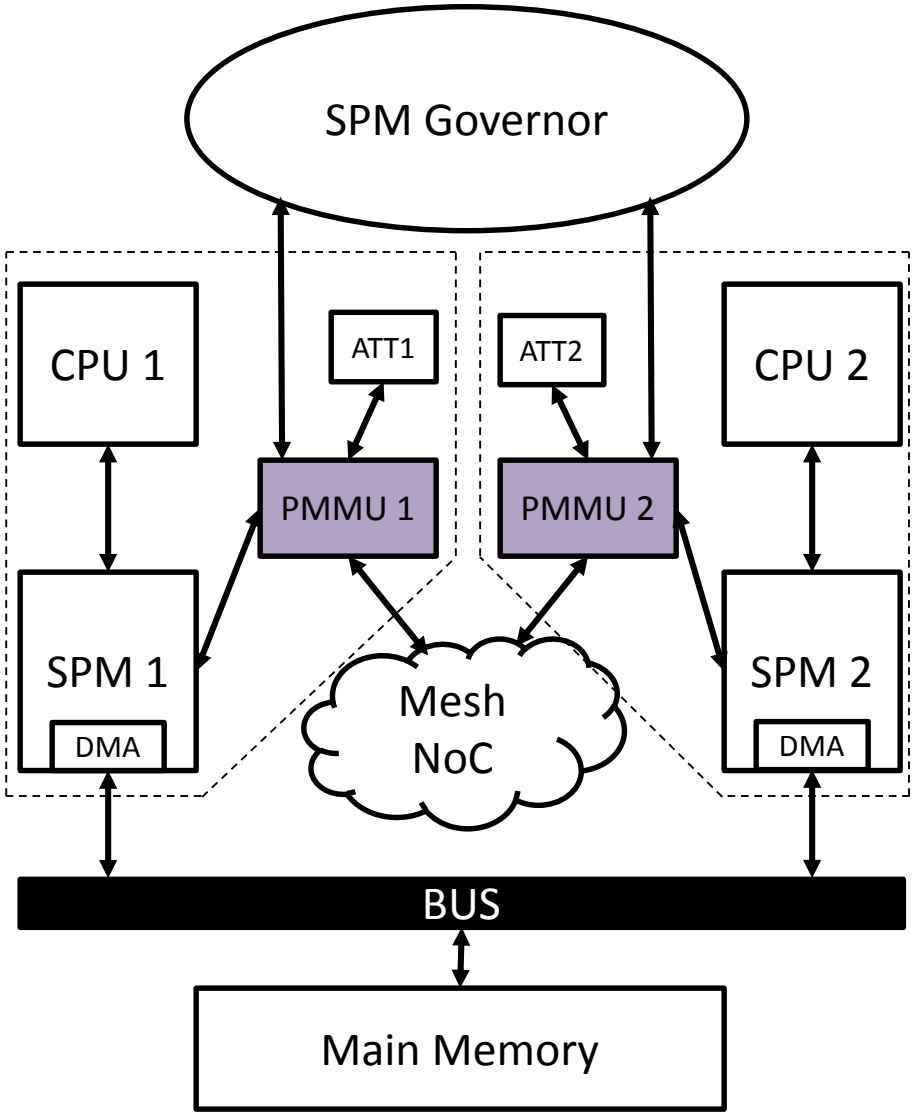
New Memory Hierarchy

- SPM
  - Receives and responds to memory requests from CPU (in place of old Cache)
  - Receives and responds to memory requests from PMMU (remote requests over NoC)
  - Forwards all main memory requests to memory bus (mimicking a DMA)



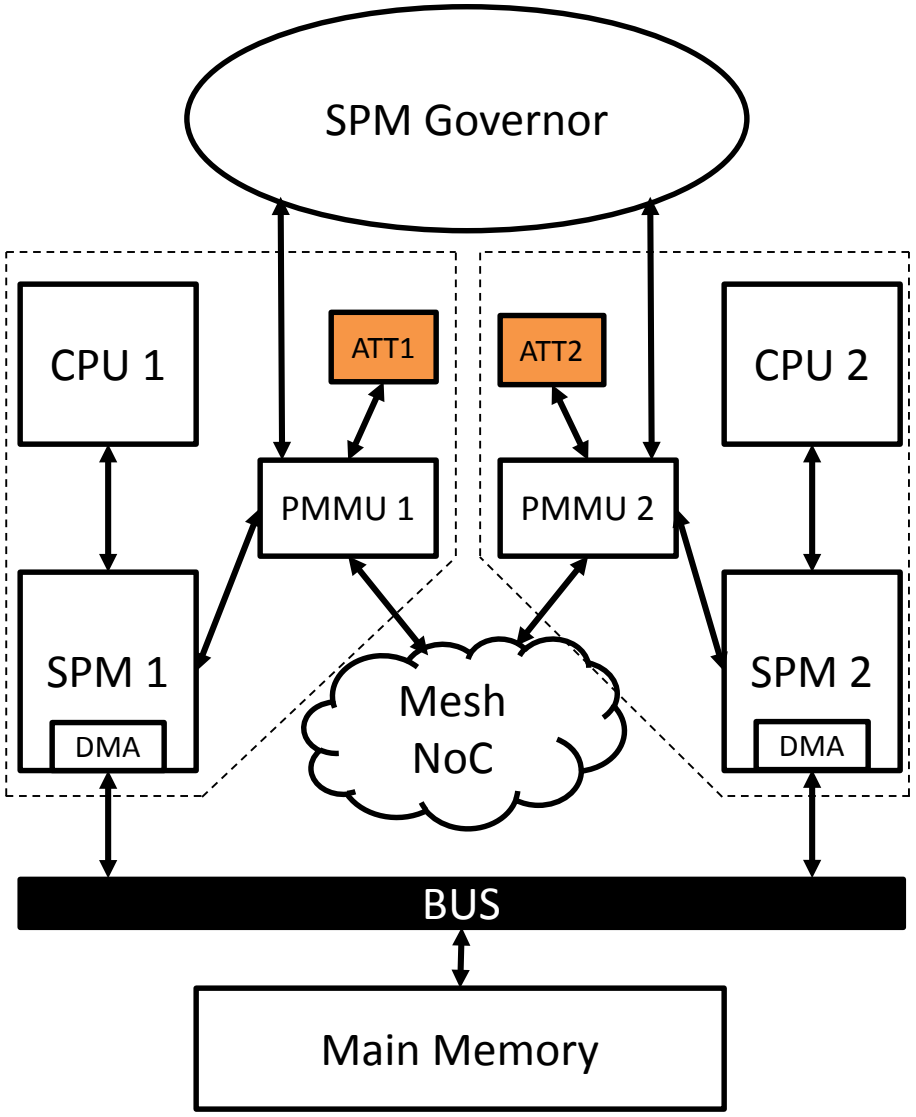
New Memory Hierarchy

- PMMU
  - Receives and responds to memory requests from SPM (local CPU)
  - Receives and responds to memory requests from NoC (remote CPU)
  - Processes SPM allocate and free requests from SPM Governor



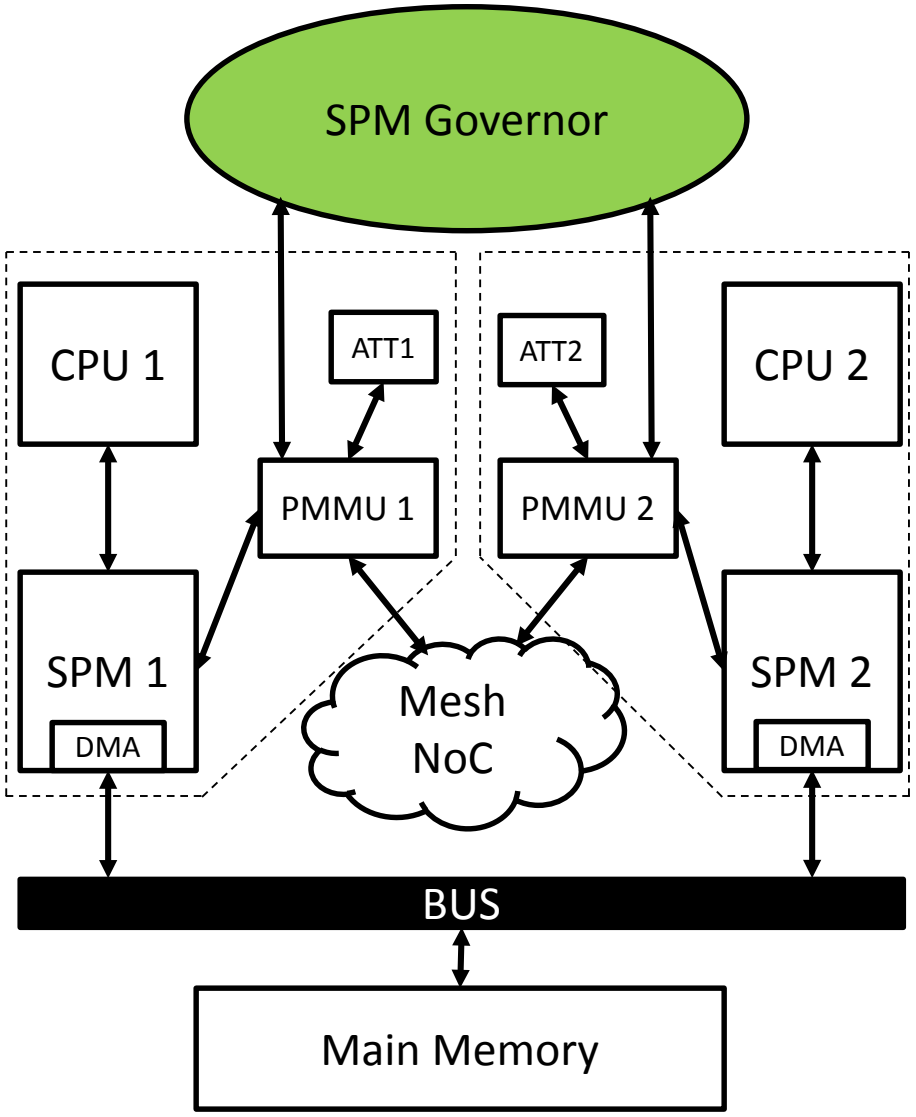
New Memory Hierarchy

- ATT
  - Holds translation of thread's virtual to SPM physical address mapping



New Memory Hierarchy

- Governor
  - Maintains global state of all memory mapped to SPMs
  - Receives SPM alloc and free requests from all executing threads (via pseudo instructions)
  - Determines memory mapping based on gem5-user-defined policy and system state



SPM Programming API

- Programmer's Interface
  - SPM_ARRAY_ALLOC (BASE_PTR, LENGTH, DATA_TYPE)
  - SPM_ARRAY_FREE (BASE_PTR, LENGTH, DATA_TYPE)

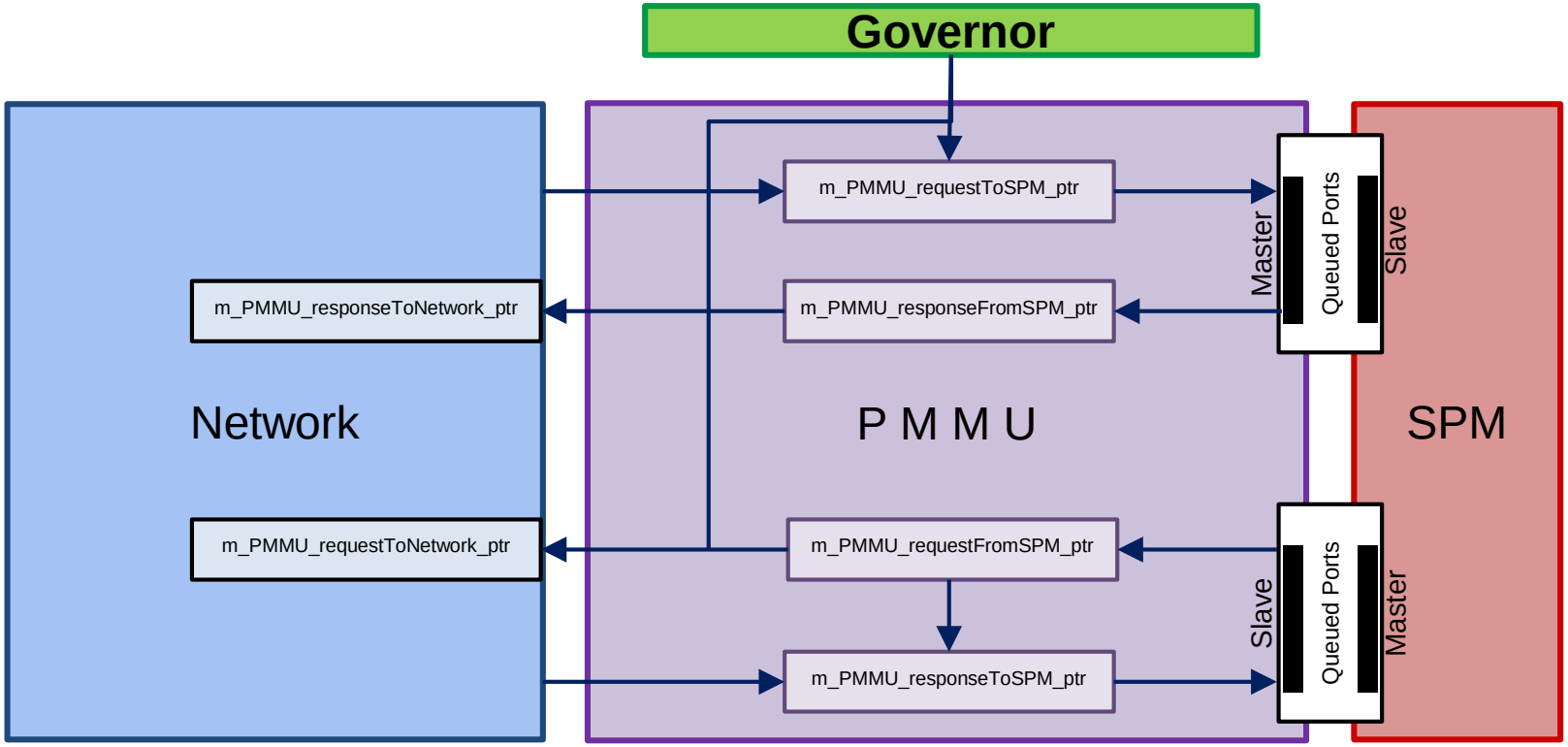
```
int *arr1 = (int*) malloc (LENGTH*sizeof(int));
...
SPM_ARRAY_ALLOC (arr1, LENGTH, int);

for (i = 0; i<LENGTH; i++) {
        arr1 [i] = i;
}

SPM_ARRAY_FREE (arr1, LENGTH, int);
...
free(arr1);
```

NoC Integration

- Integrated simple network mesh NoC from Ruby into Classic Memory Model
- PMMU handles crossover from QueuedPorts to MessageBuffers, and acts as network node



SPM Communication Protocol

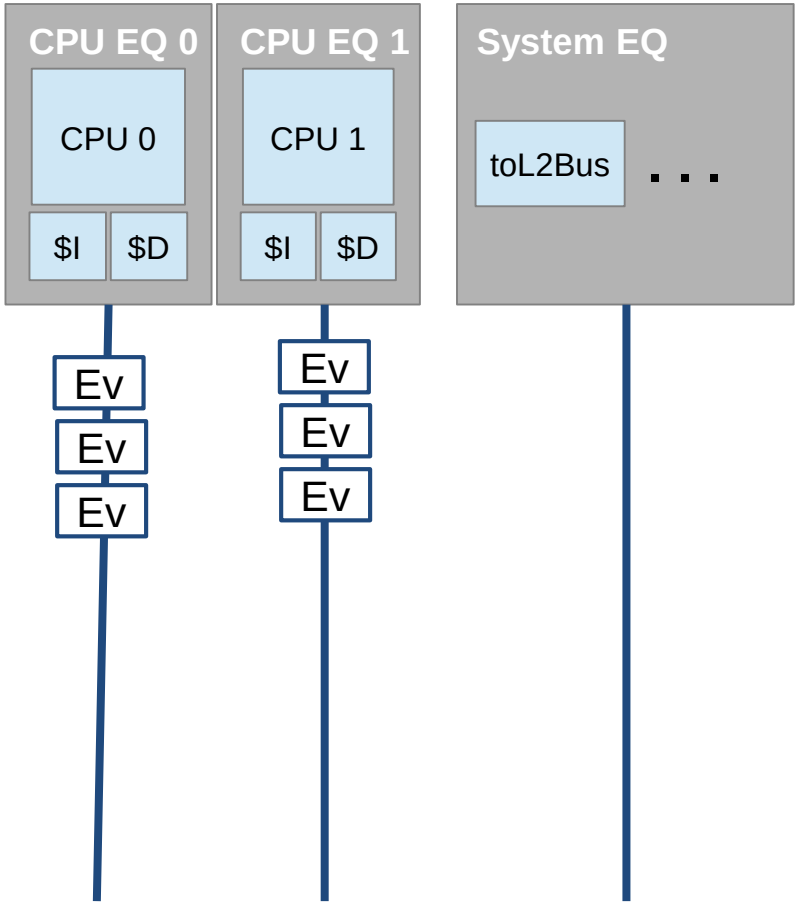
- Protocol to enable SPM
  - SPMRequestMsg : NetworkMessage
    - SPMRequestType_READ
    - SPMRequestType_WRITE
    - SPMRequestType_ALLOC
    - SPMRequestType_DEALLOC
  - SPMResponseMsg : NetworkMessage
    - SPMResponseType_WRITE_ACK
    - SPMResponseType_DATA
    - SPMResponseType_GOV_ACK

Simulator Speedup

Simulation Speedup via Parallelization

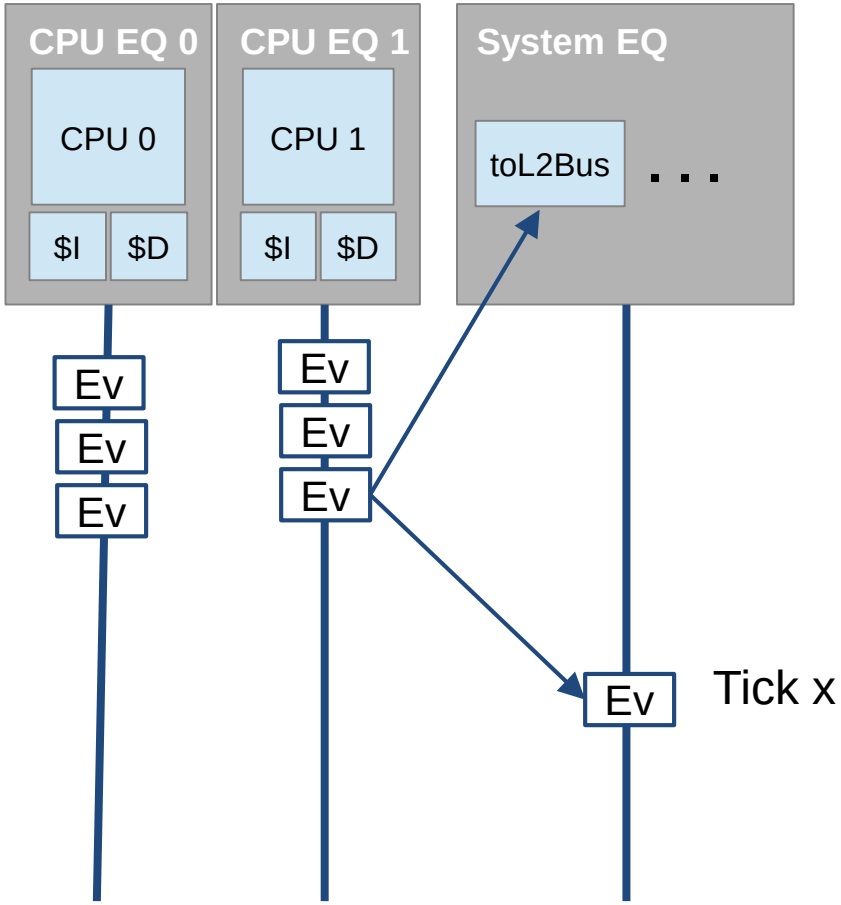
- Built on top of current multiple queue infrastructure
- Main changes
  - Assigning a separate queue for each CPU and its "private" SimObjects (e.g. I$, D$) by setting the *eventq_index* param. All other SimObjects associated to a global queue
  - Quantum-based synchronization replaced by event-based synchronization

Event Synchronization



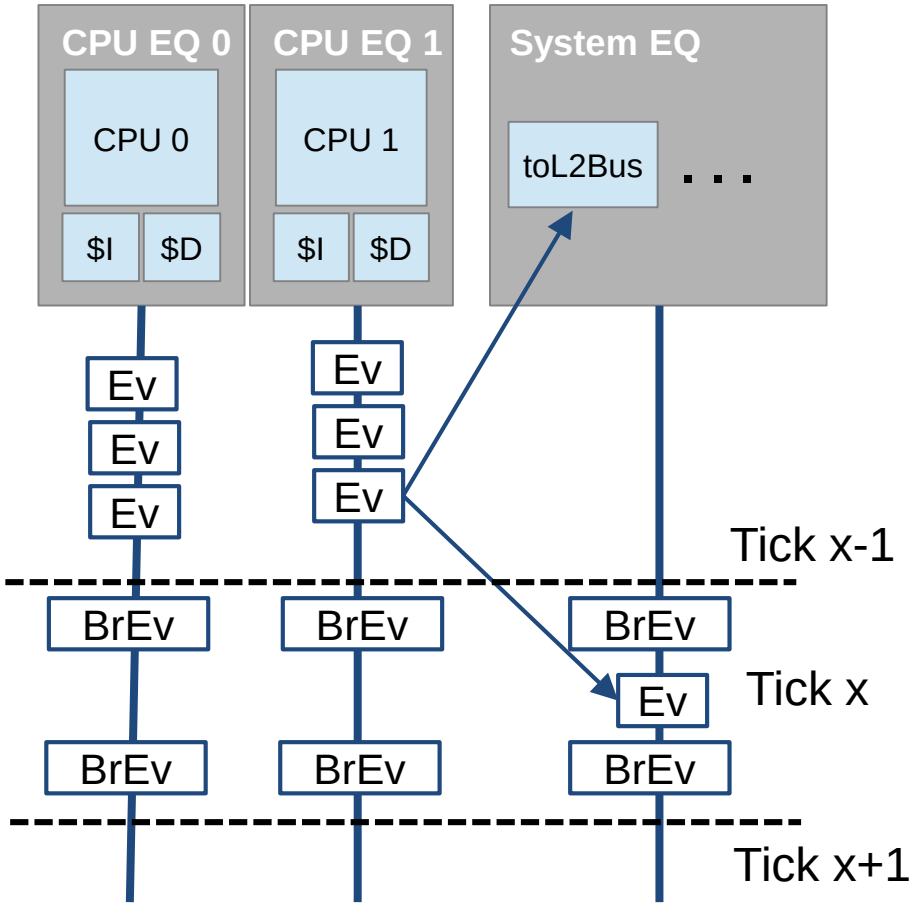
- Event on the system queue executes after all CPU queues
- CPU queues block until shared event is handled
- Using synchronization events to create a barrier

Event Synchronization



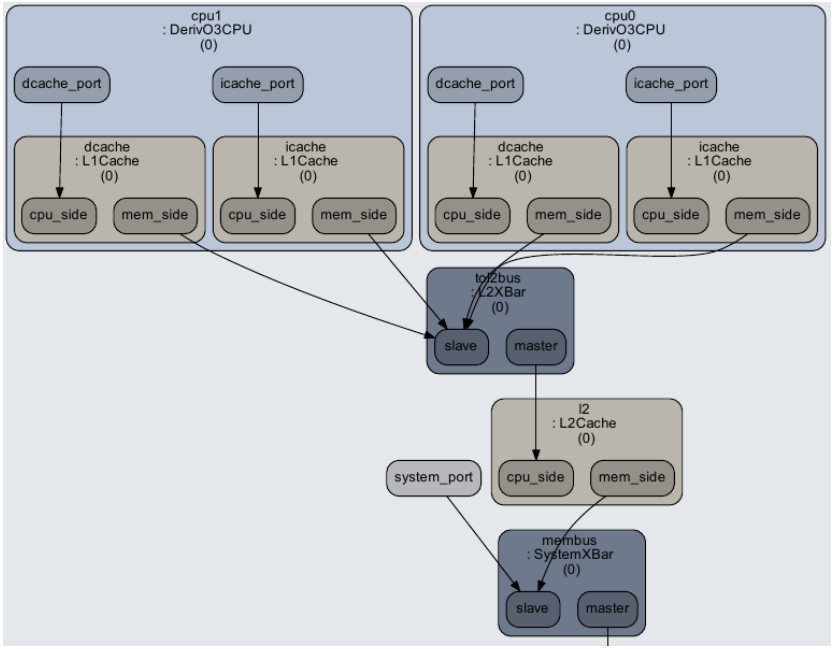
- Event on the system queue executes after all CPU queues
- CPU queues block until shared event is handled
- Using synchronization events to create a barrier

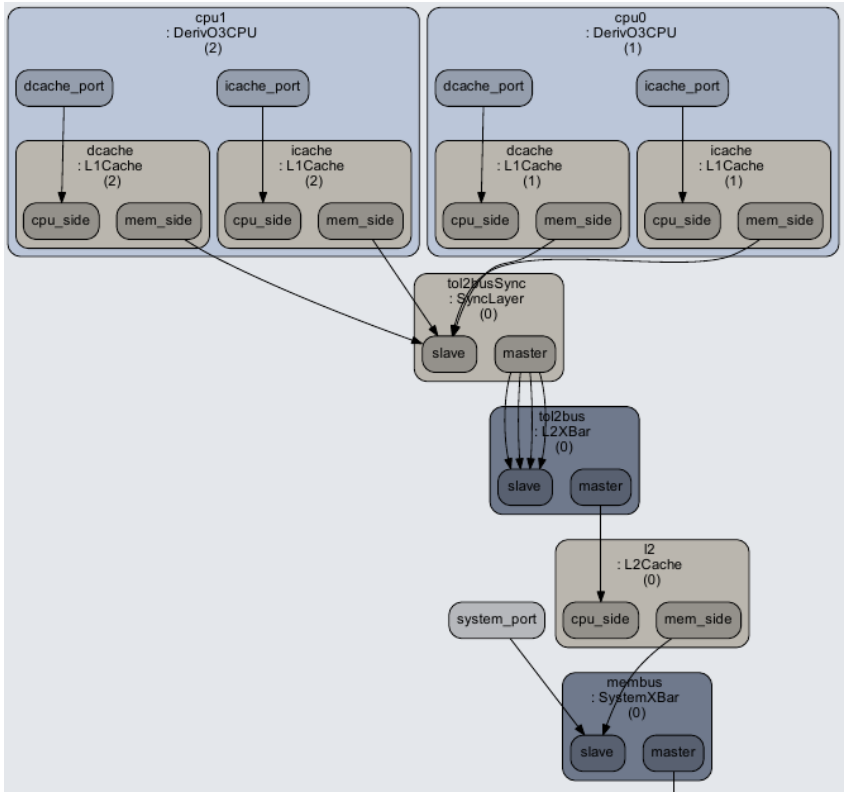
Event Synchronization



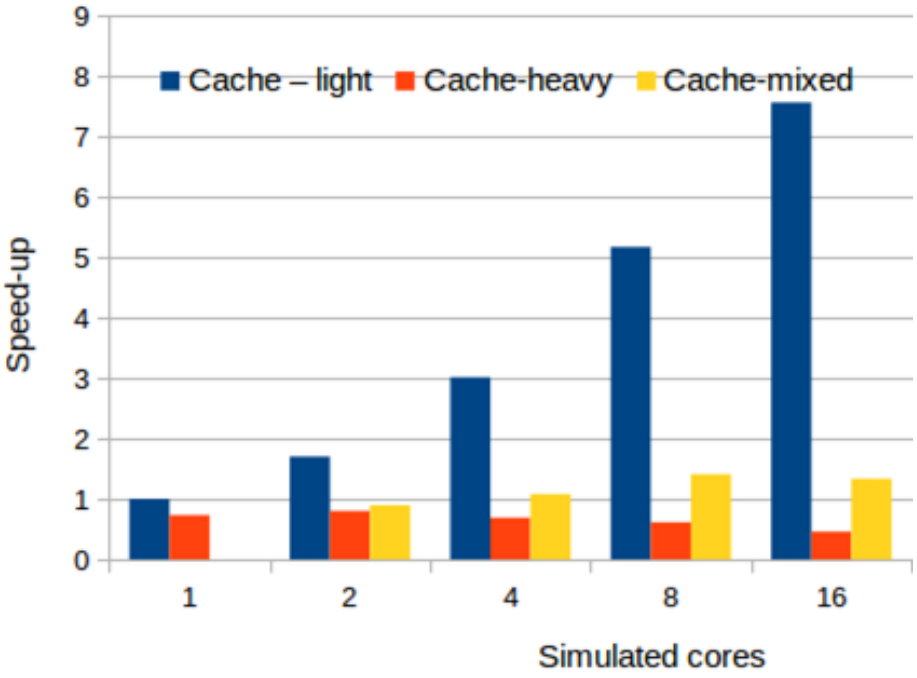
- Event on the system queue executes after all CPU queues
- CPU queues block until shared event is handled
- Using synchronization events to create a barrier

Synchronization Layers for Race Conditions



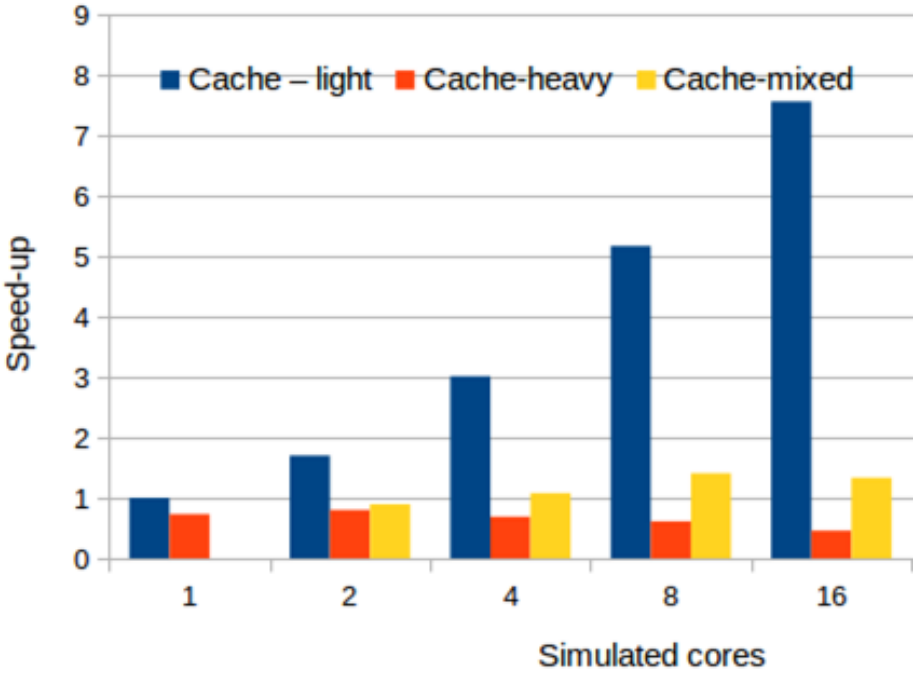


Preliminary Simulator Speedups



- Microbenchmarks on cache-based classic memory
  - Cache-light: computationally intensive
    - small number of accesses to L2/system queue
  - Cache-heavy: memory intensive with very high $ miss rate
    - Many accesses to L2/system queue
    - CPU queues block all the time

Preliminary Simulator Speedups



- Microbenchmarks on cache-based classic memory
  - Cache-light:
  - Many accesses to L2/system queue
  - CPU queues block all the time

Advantageous for coherent-less memory (e.g. SPM)

Feedback

- Comments and suggestions?
- Interested in participating?
- Contact us:
  - Bryan – bdonyana@uci.edu
  - Majid – anamakis@uci.edu
  - Tiago – tmuck@uci.edu

Thank you

Q&A

duttgroup.ics.uci.edu