Rethinking Remote Memory Placement on Large-Memory Systems with Path Diversity

Wonkyo Choe, Sang-Hoon Kim, Jeongseob Ahn

Ajou University

Emerging multi-chip systems

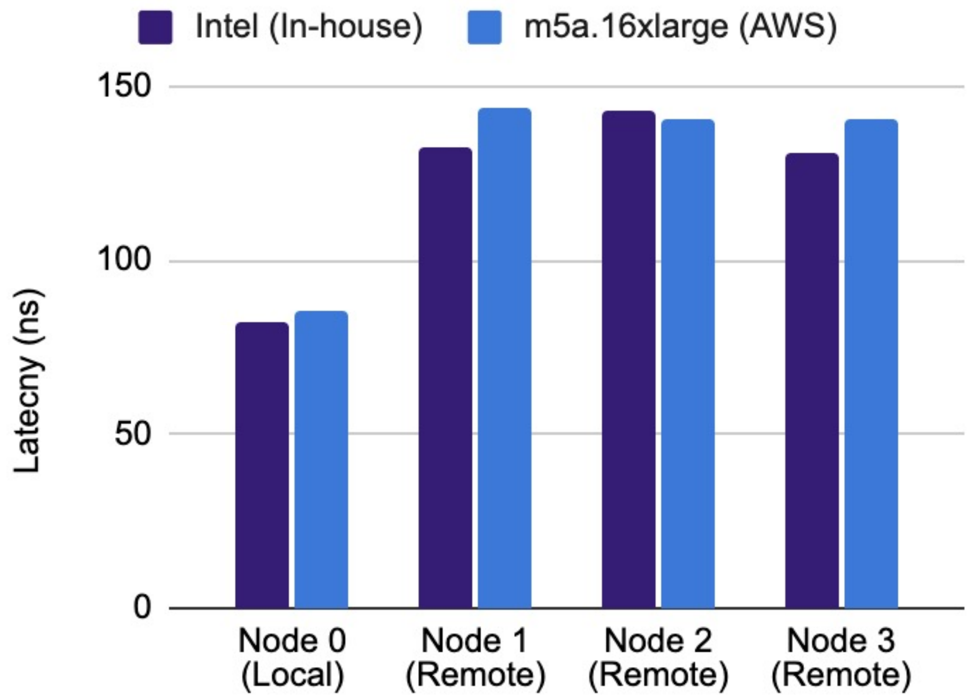
- Data center utilizes multi-chip systems to build scale-up servers
  - Memory controllers for each processor die

- Chip manufacturers is developing such systems
  - AMD EPYC
  - Intel Xeon

- Advanced point-to-point interconnect
  - AMD Infinity Fabric (IF)
  - Intel Ultra Path Interconnect (UPI)

A distinct feature of multi-chip systems

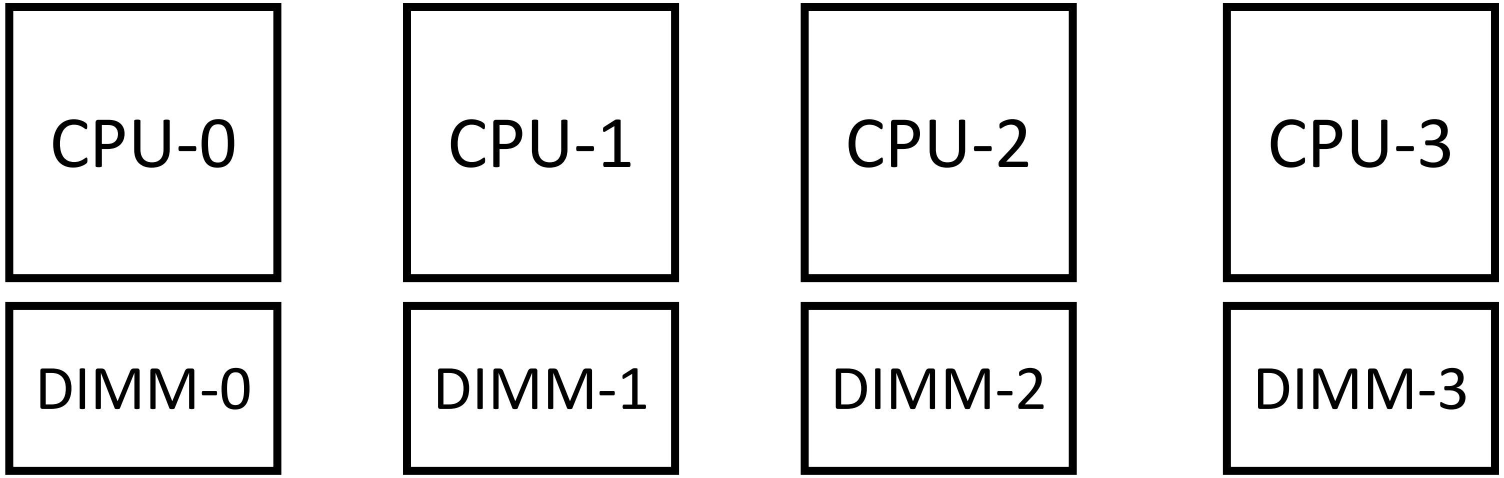
- All remote memory access latencies are similar
  - Local latency ~= 85 ns
  - Remote latecny ~= 140 ns



If the latencies are not that different,
doesn't it matter to allocate memory on any nodes?

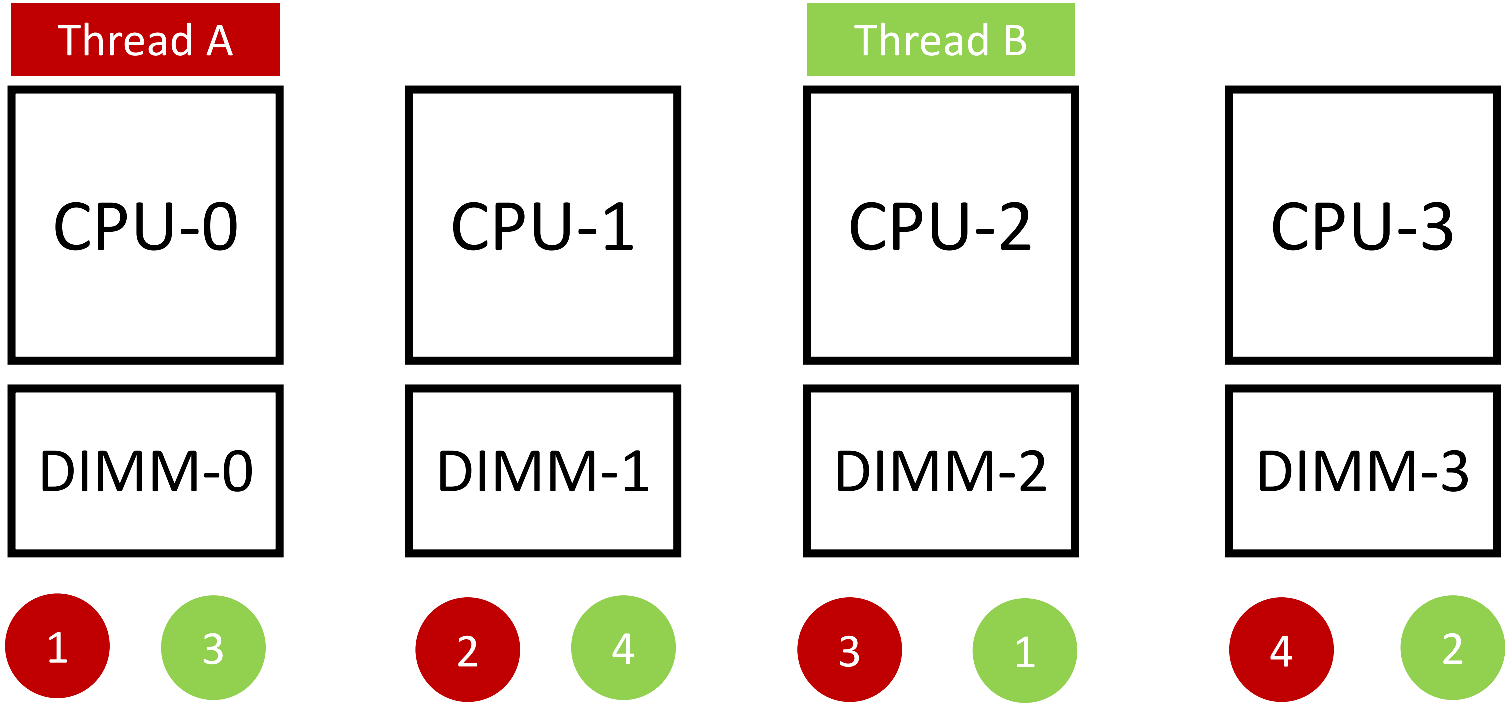
Traditional NUMA systems

- Linux configures multi-chip systems as a cpu node
- Each node has own memory node



Default memory placement (first-touch)

- local memory is not enough
  - Linux requires memory from other memory nodes
  - Fallback node list determined when system boots



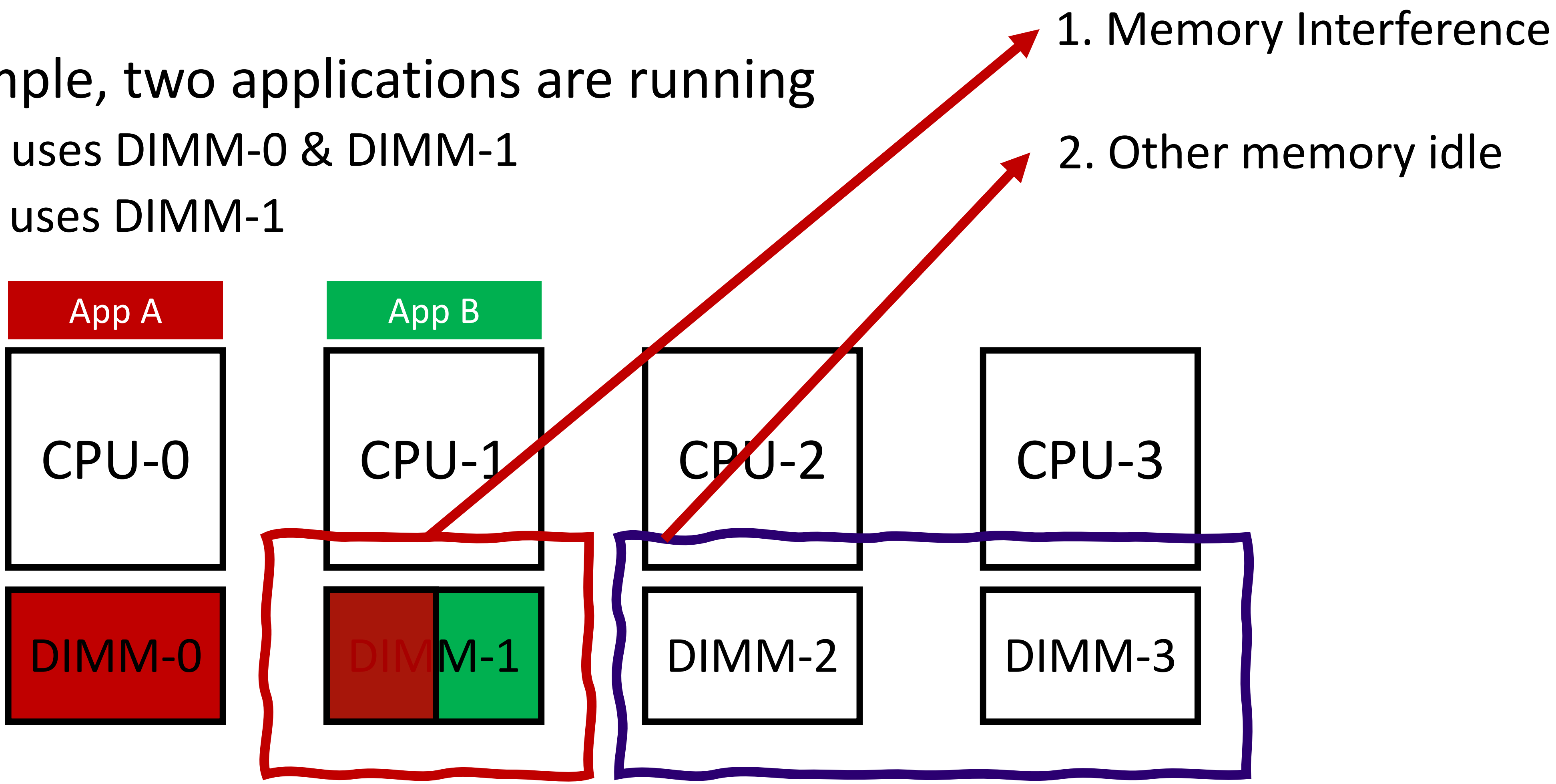
Our insight

1. Existing systems do not exploit diverse memory path (path diversity)
   - All remote latency is almost the same
   - Static Linux’s fallback node list

2. Existing memory placement causes unintended interference
   - Multiple applications would use the same memory node

Memory interference

- For example, two applications are running
  - App A uses DIMM-0 & DIMM-1
  - App B uses DIMM-1



Our insight

1. Existing system not exploit diverse memory path
    - Static Linux's fallback node list
    - All remote latency is almost the same

2. Existing memory placement causes unintended interference
    - Multiple applications would use the same memory node

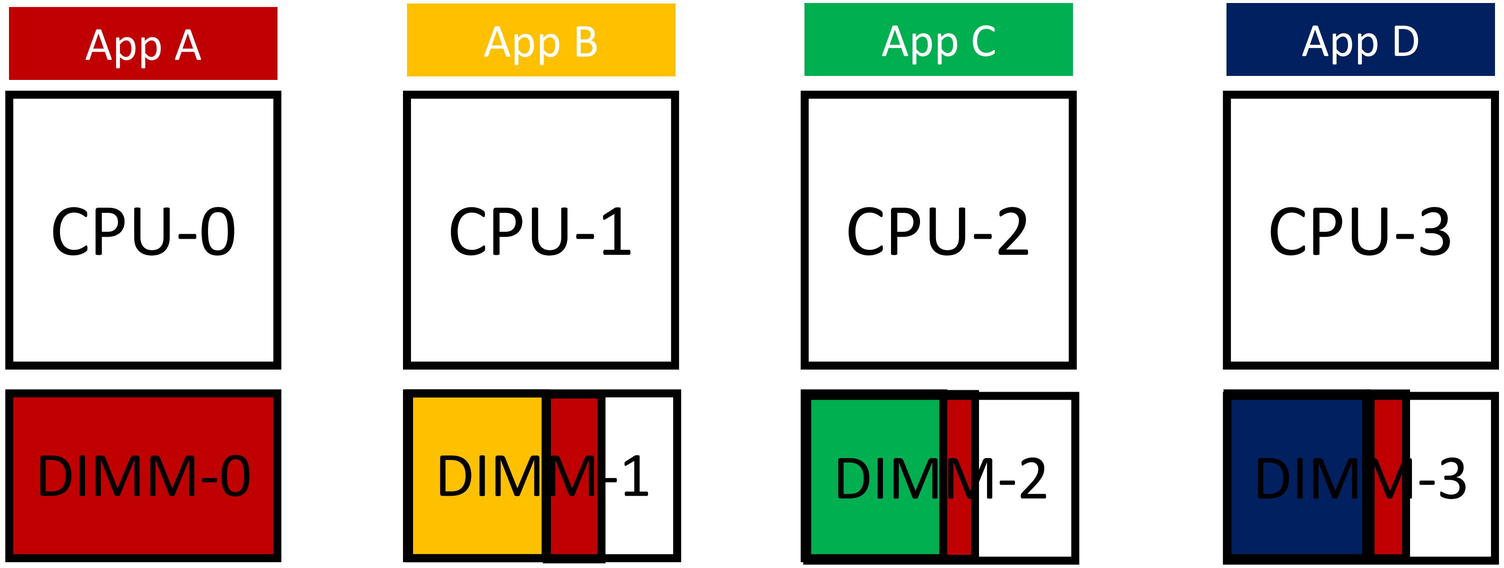
Hybrid & Usage-aware memory placement

Rest of the talk

- Hybrid & Usage-aware memory placement
- Performance evaluation
- Discussion
- Conclusion

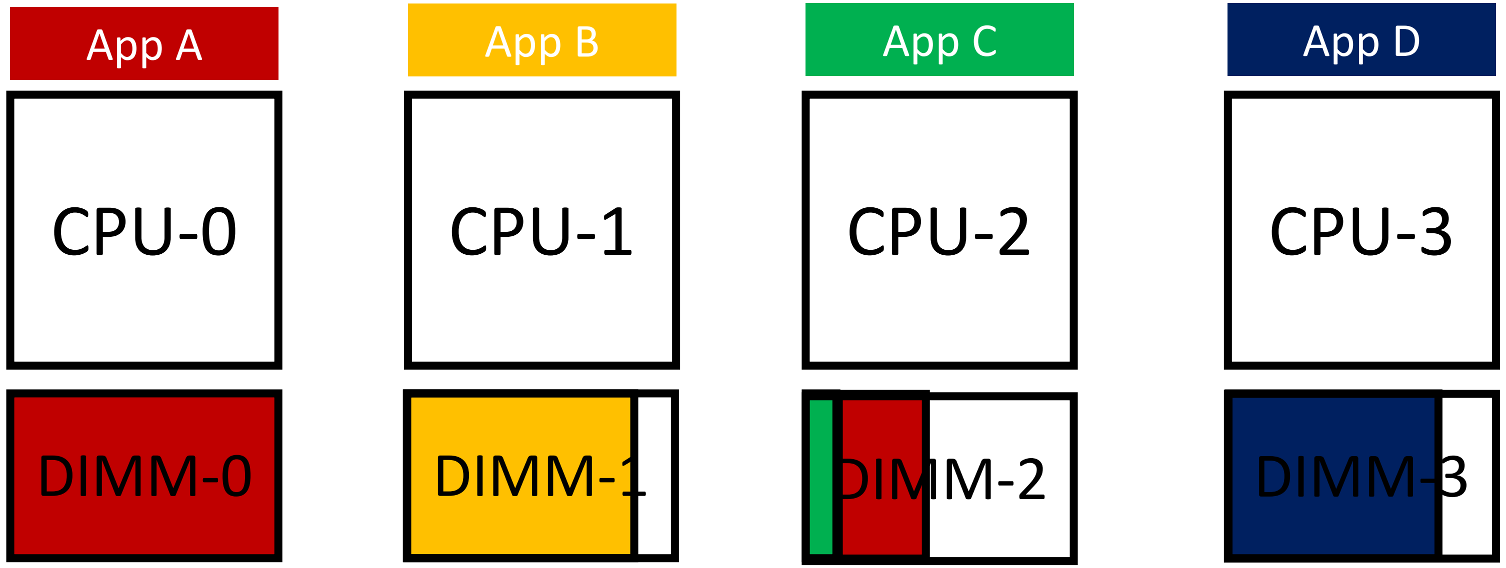
Hybrid placement

- First-touch (Linux default) + page-interleave (round-robin allocation)
  - Use first-touch when allocating local memory
  - Use page-interleave when allocating remote memory



Usage-aware placement

- First-touch (Linux default) + usage-aware
  - Use first-touch when allocating local memory
  - Allocate memory based on memory usage (allocation on the least usage)



How we set our environment

- 4 sockets machine
  - Intel Xeon Gold 6242: A single chip (16 physical cores) on each socket
  - 16GB * 4 socket = total 64 GB memory capacity
- Linux kernel v5.3
  - AutoNUMA enabled
- Benchmark
  - Mcf / fotonik3d / cam4 from SPECCPU 2017
  - MG from NAS parallel benchamrk
  - GUPS from HPC Challenge benchmark
  - Liblinear

Other consideration for evaluation

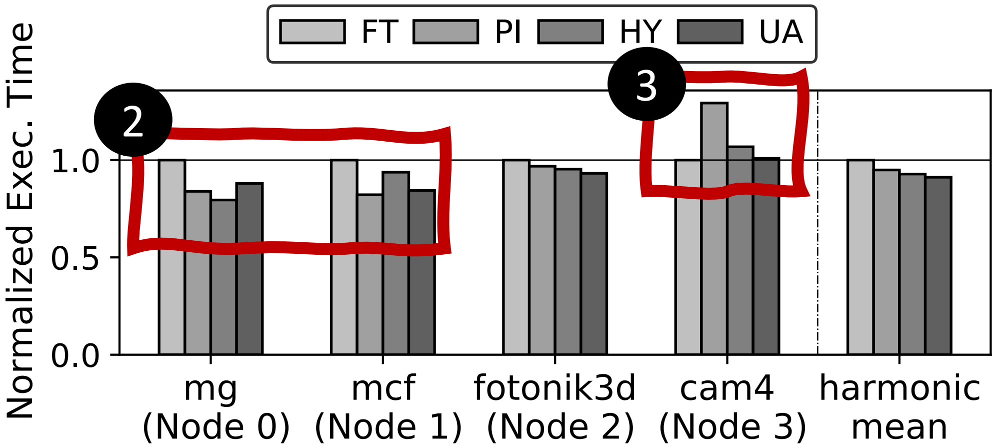
- Memory intensive workload on CPU-0 spills memory
  - Eventually uses the remote memory
  - MG / GUPS / Liblinear
- The rest of workloads on each CPU-X except CPU-0
  - Use only the local memory
- Various mixed sets are experimented
  - Few results are included in the paper

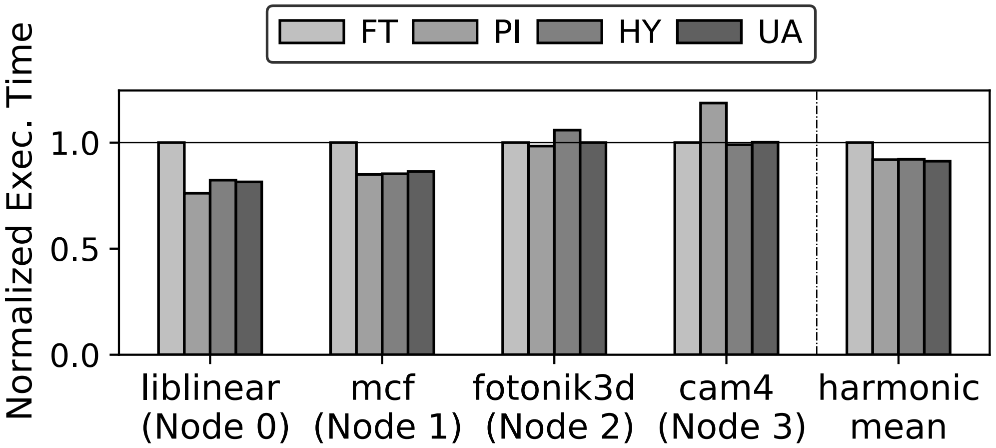
Performance comparison

- First-touch (FT)
- Page-interleave (PI)
  - Allocate a page one by one on each node
  - `numactl`
- Hybrid (HY)
- Usage-aware (UA)

Performance of Proposed polices

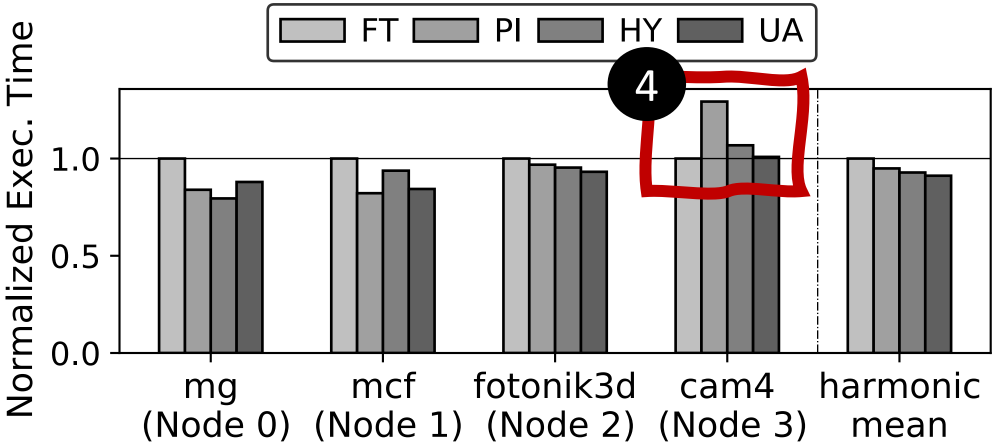
1. All proposed policies are improved over first-touch (FT)
2. Memory intensive workload (mg, liblinear, mcf) perf. Bounded memory bandwidth
3. Page-interleave(PI) impairs other workloads, such as cam4 due to memory interference

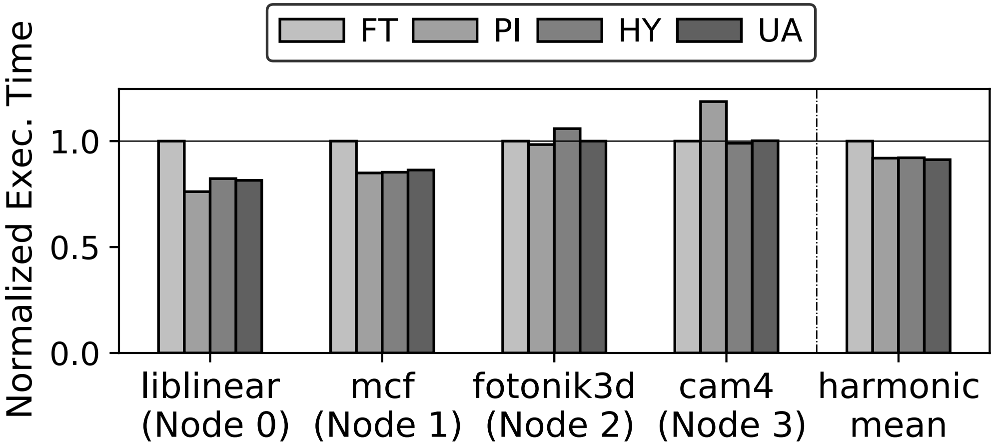




Performance of Proposed polices

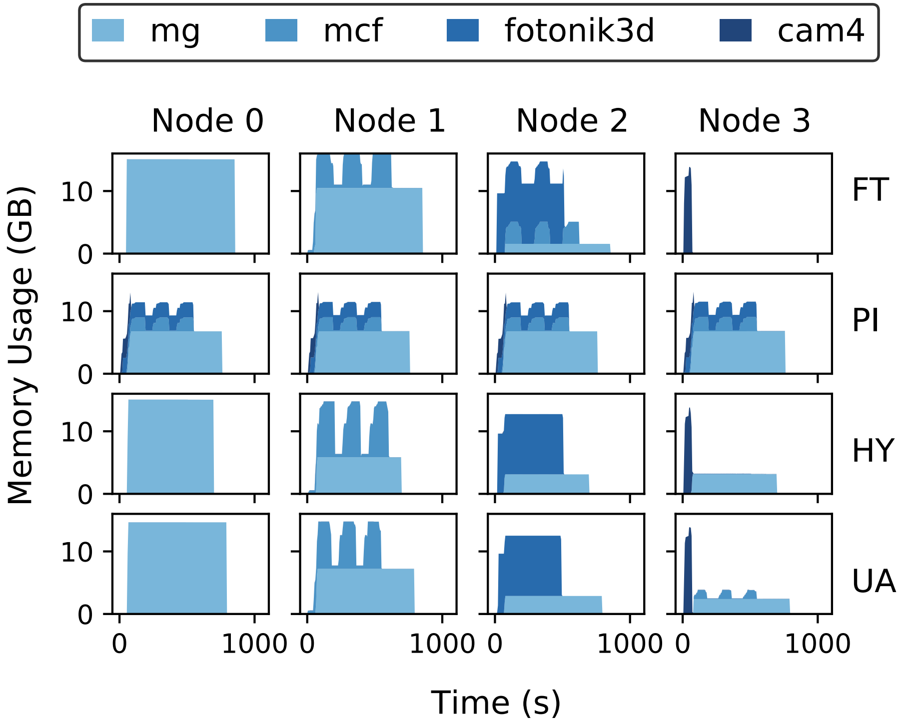
4. Not like PI, our proposed policies, Hybrid(HY) & Usage-aware(UA) not impair cam4
    - Overall, harmonic mean has improved on our policies





Memory allocation graph

- For first-touch, mg significantly interferes with mcf
- For page-interleave, all workloads interferes with each other
- For Hybrid & Usage-aware, memory interference little on mcf



Discussion

- Localizing data
  - Increased remote accesses
  - Hard for scheduler to minimize them

- Applying policies to different typologies
  - Policy on different NUMA group & NUMA distance
  - Allocate closest neighbor group and then faraway group

Conclusion

- Exploiting path diversity on multi-chip systems
- Simple memory placement for multiple applications
  - Hybrid & Usage-aware
- Minimizing hot-spot or interference and show better performance.