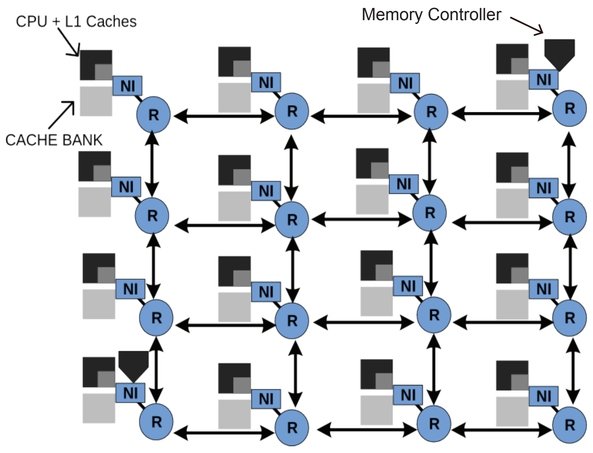
APPLICATION-TO-CORE MAPPING POLICIES TO REDUCE MEMORY INTERFERENCE IN MULTI-CORE SYSTEMS

Reetuparna Das§, Rachata Ausavarungnirun, Onur Mutlu, Akhilesh Kumar‡, Mani Azimi‡
(Carnegie Mellon University, §University of Michigan, ‡Intel Labs)

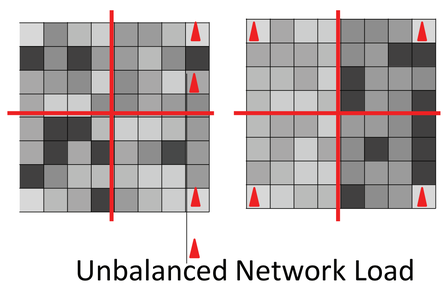
BACKGROUND & PROBLEMS

NETWORK ON-CHIP

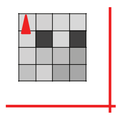


PROBLEMS

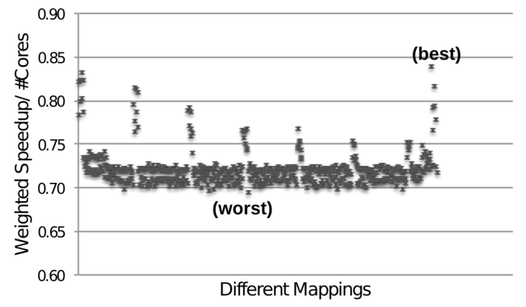
- **Current operating systems are unaware of:**
  - **On-chip interconnect topology**
  - **Application interference characteristics**



Unbalanced Network Load



Unaware of the location of the memory controller



System performance varies with different mappings

OUR SOLUTION

KEY INSIGHTS

1. **Network & memory load not balanced across the network**
2. **Overall performance degrades when applications that interfere significantly with each other get mapped to closeby cores**
3. **Some applications benefit significantly from being mapped close to a shared resource**

IDENTIFYING SENSITIVE APPLICATIONS

- **Stall Time per Miss (STPM): average number of cycles a core is stalled because of a cache miss**

→ **Applications with high STPM are interference-sensitive**

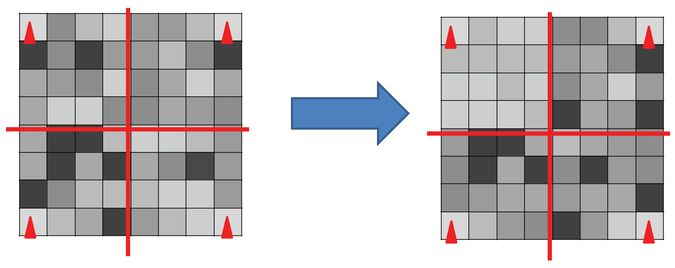
- **L1 Misses per Thousand Instruction (MPKI)**

→ **Applications with high MPKI are network-intensive**

- **Sensitive applications are applications with high STPM and high MPKI**

APPLICATION-TO-CORE MAPPING POLICY

1. **Clustering: A sub-network where applications mapped to a cluster typically access resources within that cluster**
2. **Mapping policy across clusters:**
   - **Equally divides the network load among clusters**
   - **Protects interference-sensitive applications from others by assigning them their own cluster**
3. **Mapping policy within cluster: Maps network-intensive and interference-sensitive applications close to the memory controller**
4. **Dynamically migrate applications between cores**



Balanced Mapping with Reduced Interference

Radial Inter-cluster Mapping

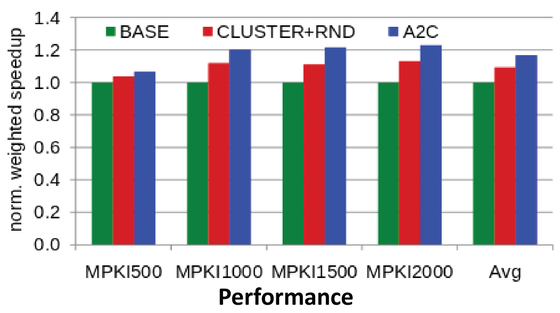
KEY RESULTS

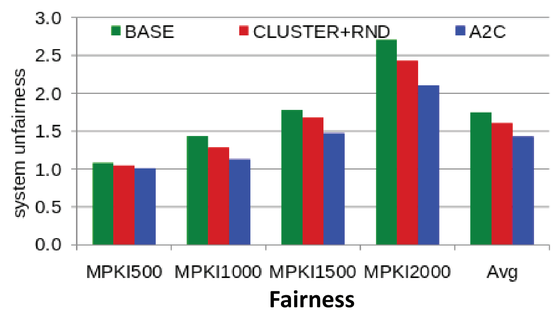
METHODOLOGY – 3 SYSTEMS

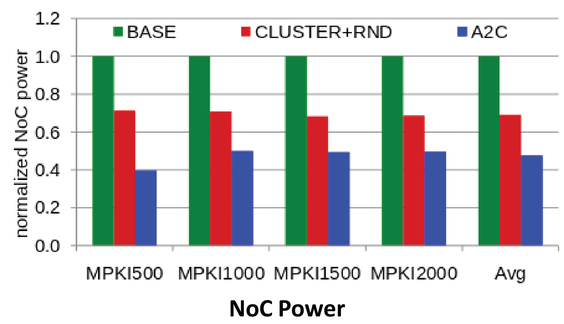
- **Baseline with random mapping (BASE),**
- **Random mapping of applications to cores (CLUSTER+RND)**
- **Our final system with application-to-core (A2C)**

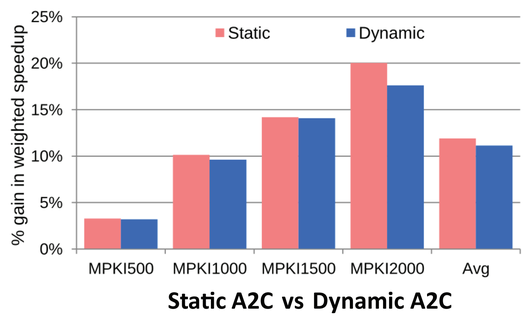
Number of Cores	60
L1 Cache	32KB per core. 4 ways, 2-cycle latency
L2 Cache	256KB per core, 16 ways, 6-cycle latency
MSHR	32 entries
Main Memory	4GB. 160-cycle latency 4 channels at 16GB/s
Network Router	4 VCs per port, 4 flits per VC 2-stage wormhole
Network Topology	8x8 mesh, 128 bit bi-directional links
Memory Management	4KB physical and virtual page 512 entries TLB CLOCK page allocation and replacement

RESULTS









Intel Science & Technology Center for Cloud Computing

Carnegie Mellon University · Georgia Tech · intel · PRINCETON UNIVERSITY · UC Berkeley