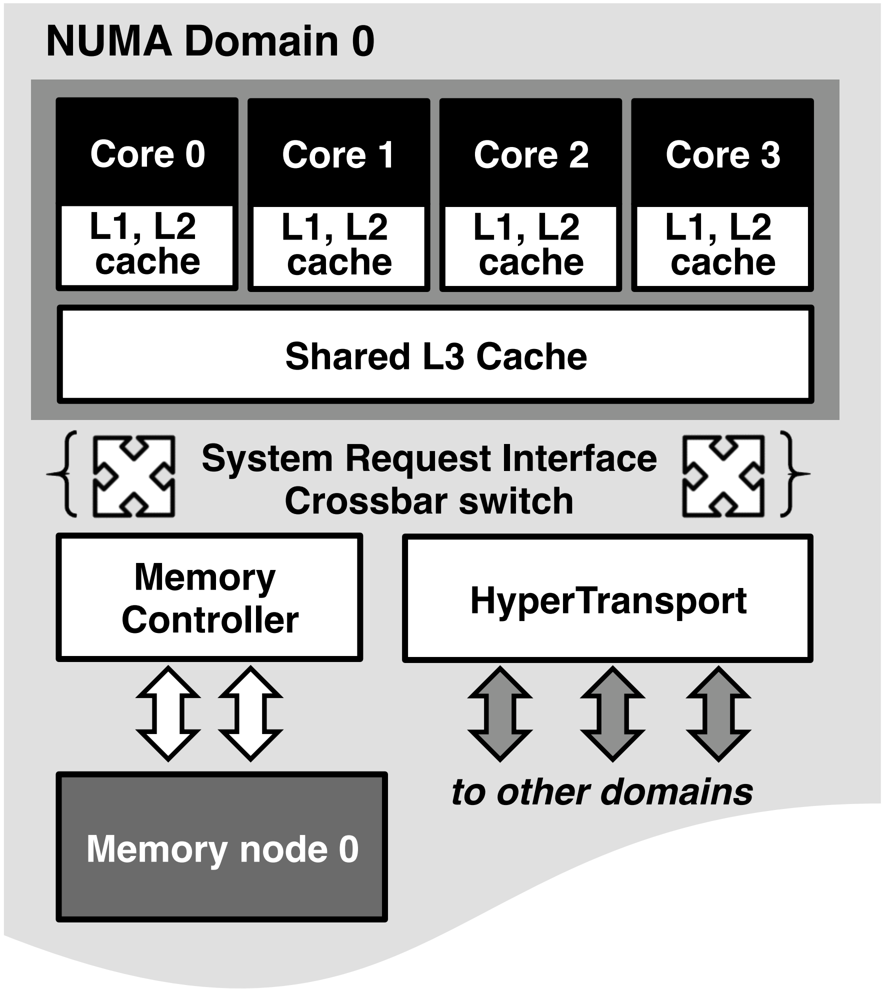
SFU SIMON FRASER UNIVERSITY
THINKING OF THE WORLD

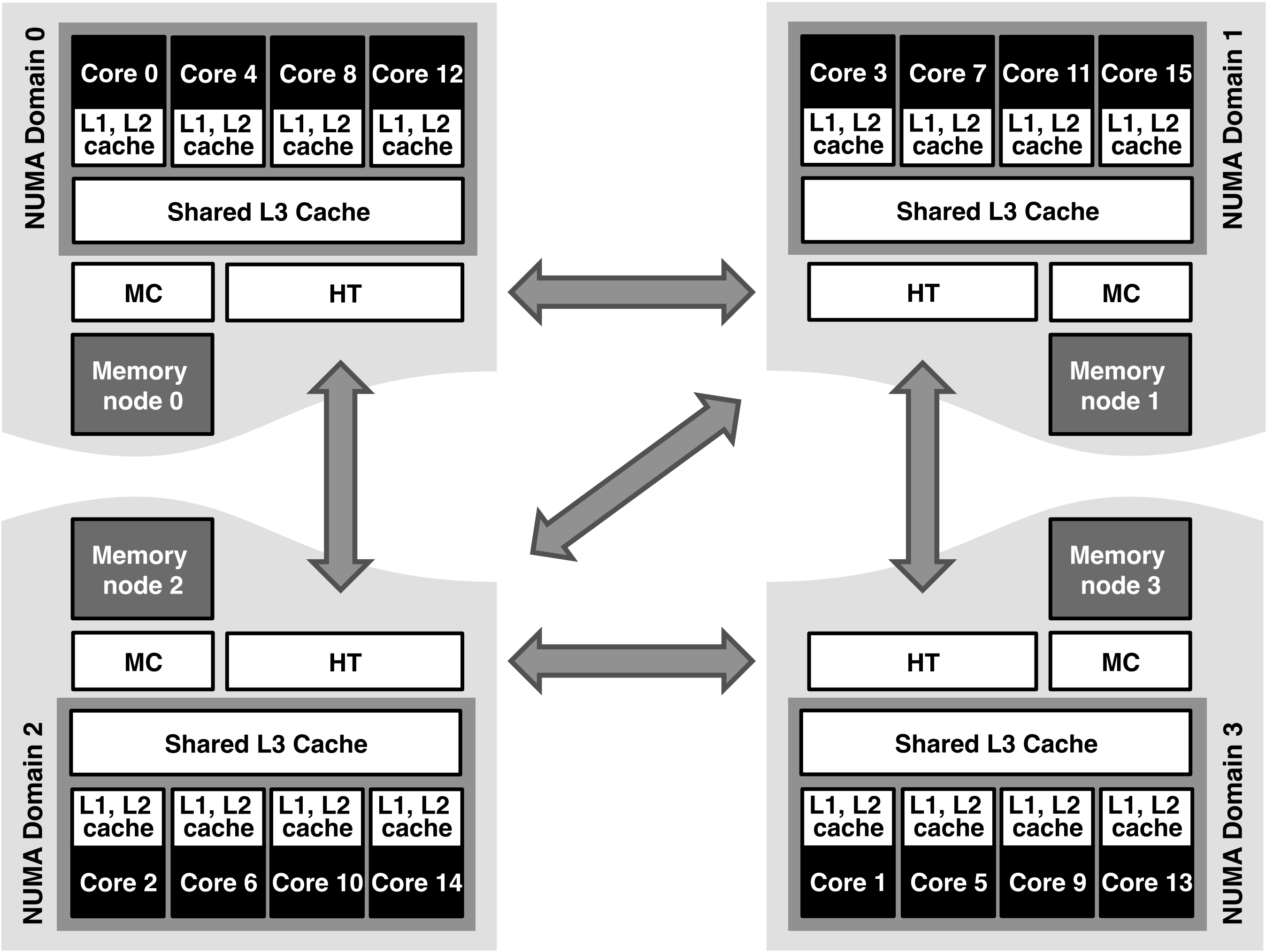
A Case for NUMA-aware Contention Management on Multicore Systems

Sergey Blagodurov sergey_blagodurov@sfu.ca
Sergey Zhuravlev sergey_zhuravlev@sfu.ca
Mohammad Dashti mohammad_dashti@sfu.ca
Alexandra Fedorova alexandra_fedorova@sfu.ca

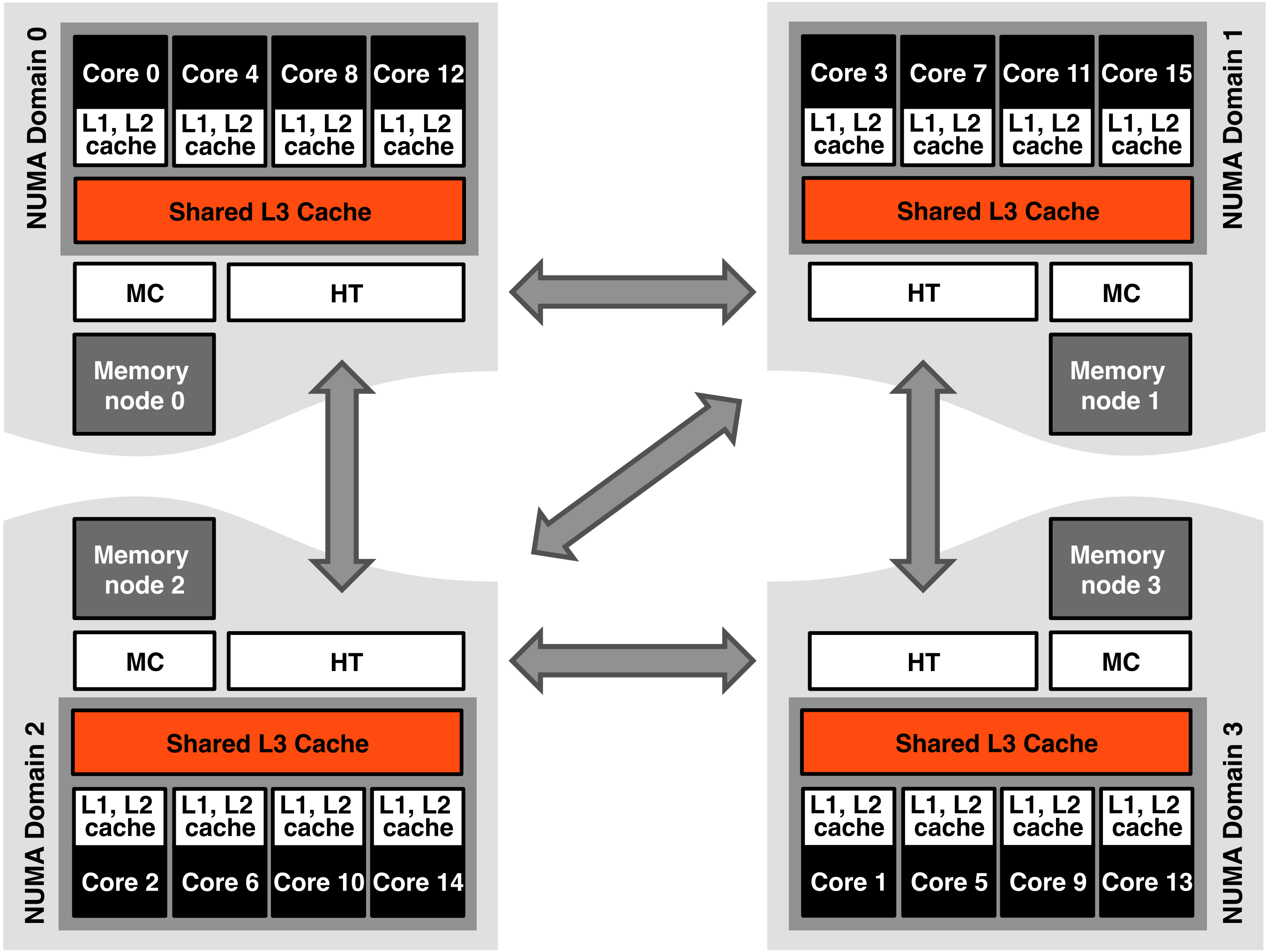
USENIX ATC'11 / Scheduling session
15th of June



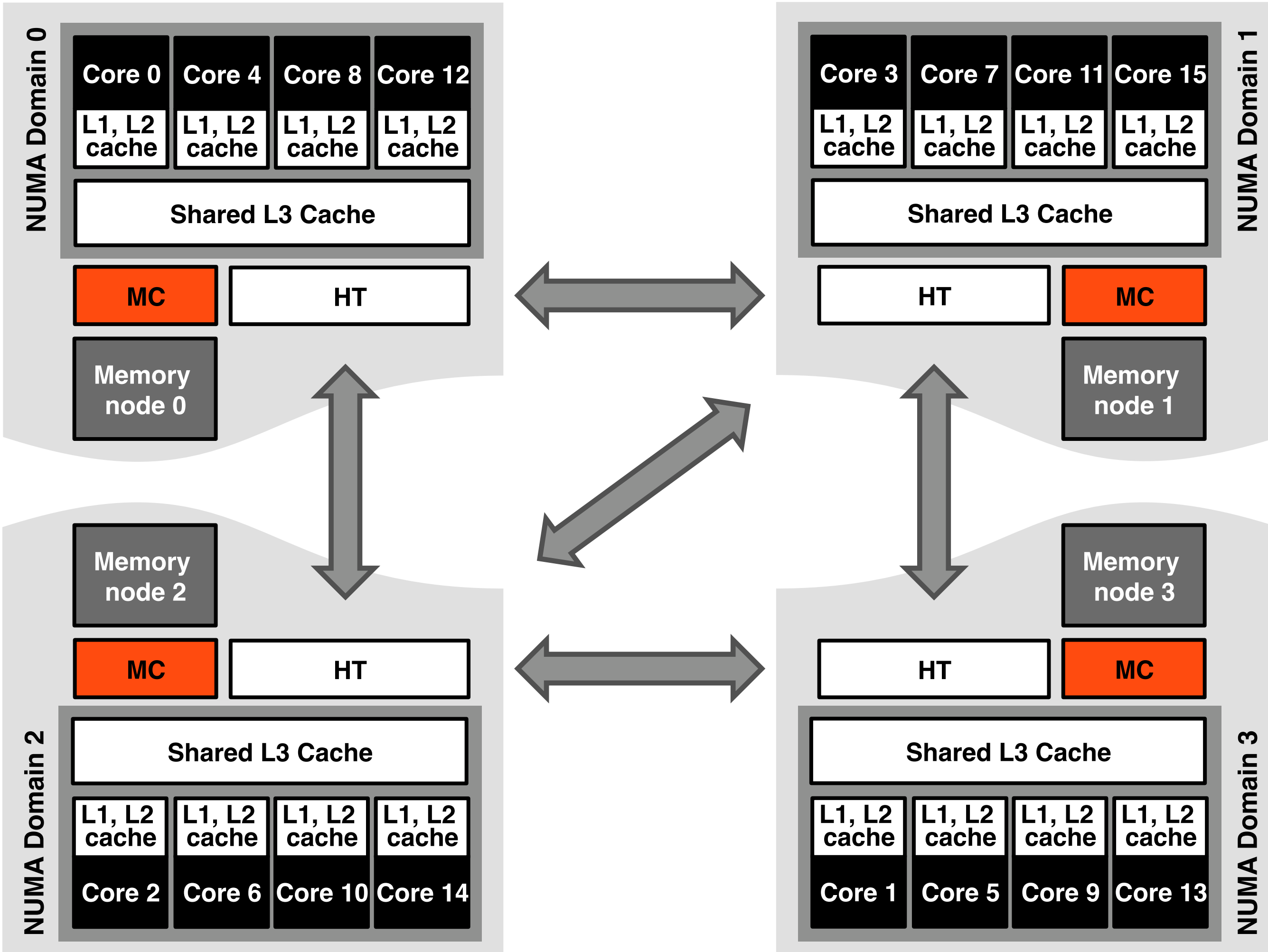
An AMD Opteron 8356 Barcelona domain



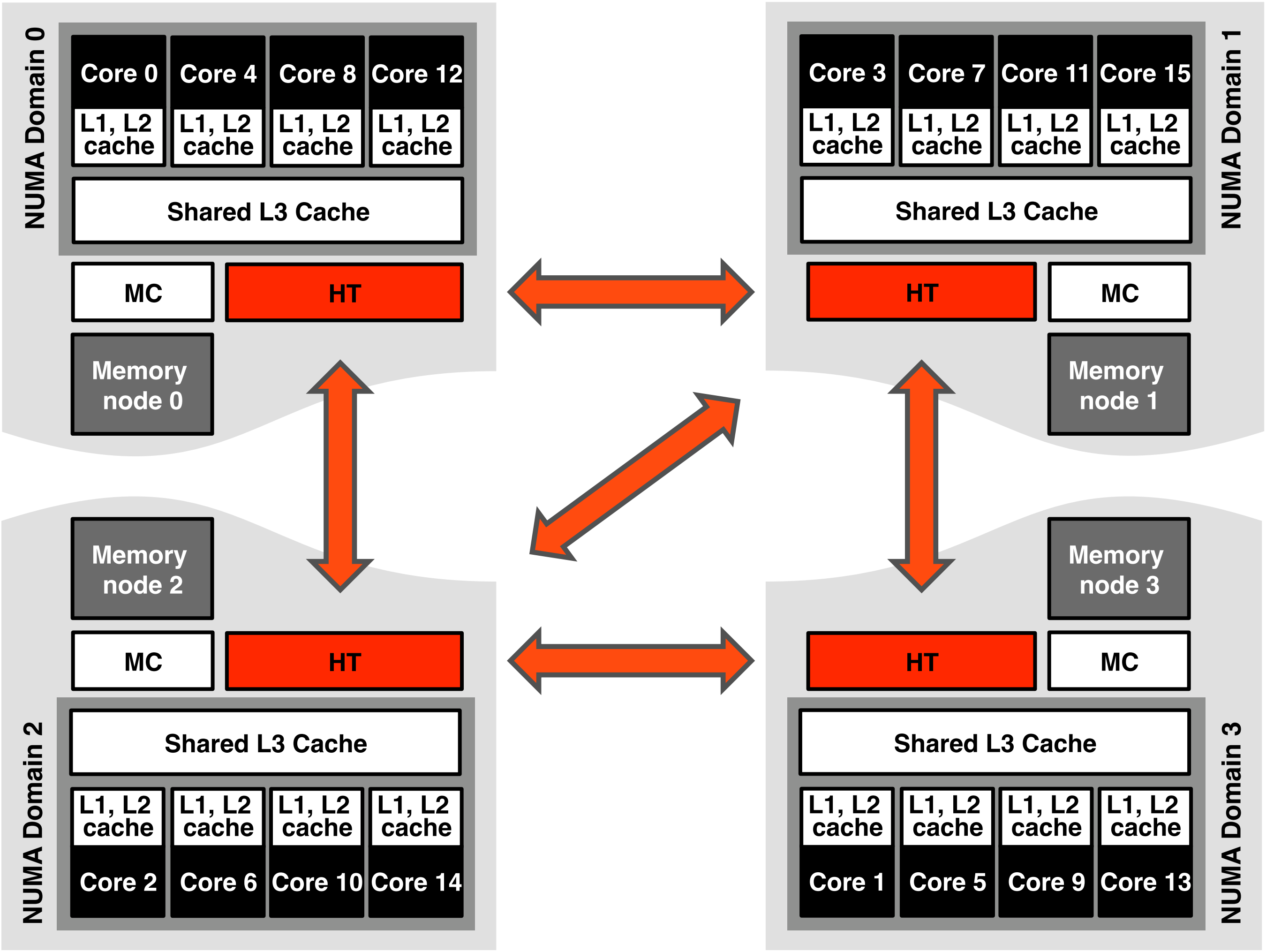
An AMD Opteron system with 4 domains



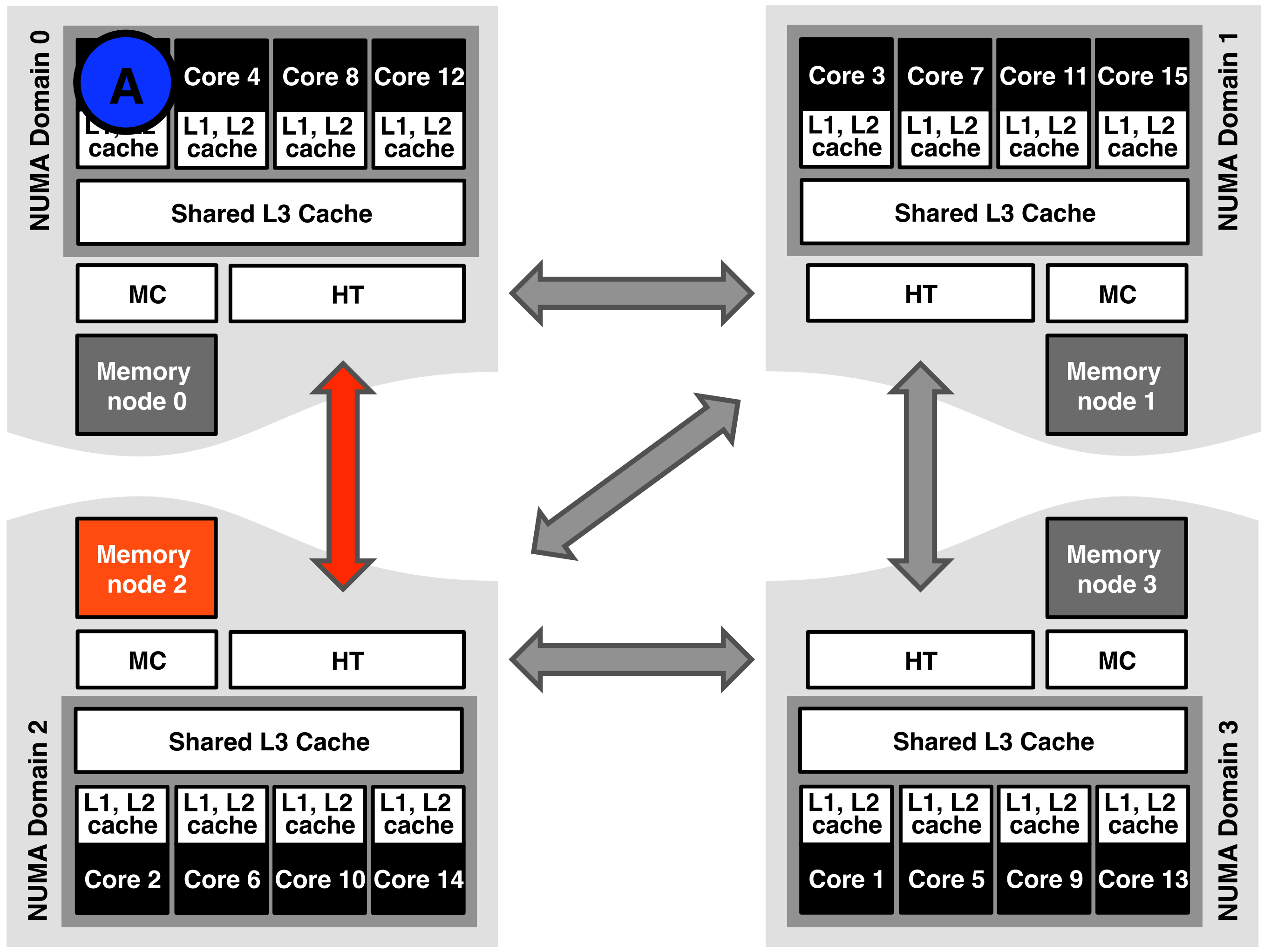
Contention for the shared last-level cache (CA)



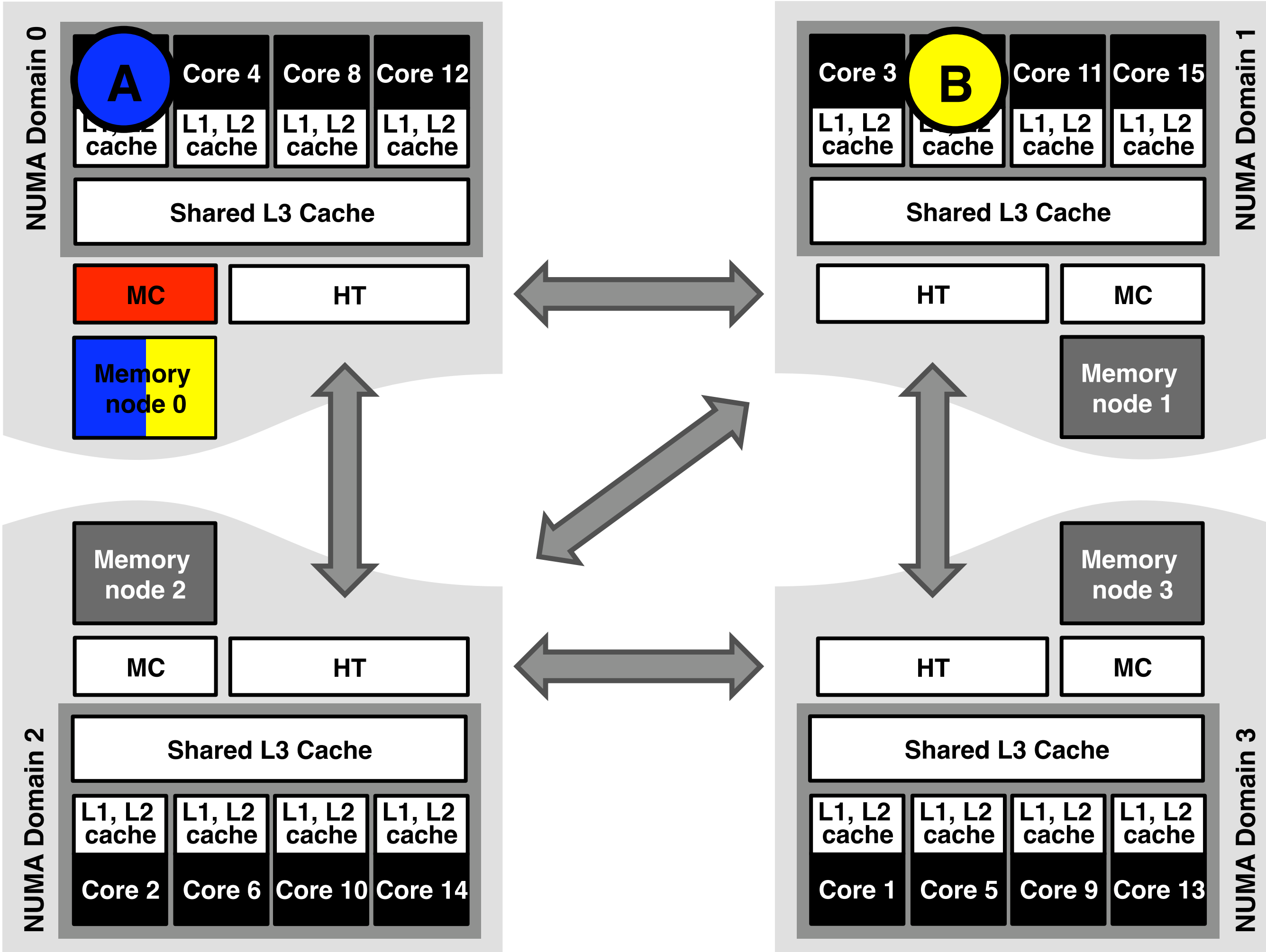
Contention for the memory controller (MC)



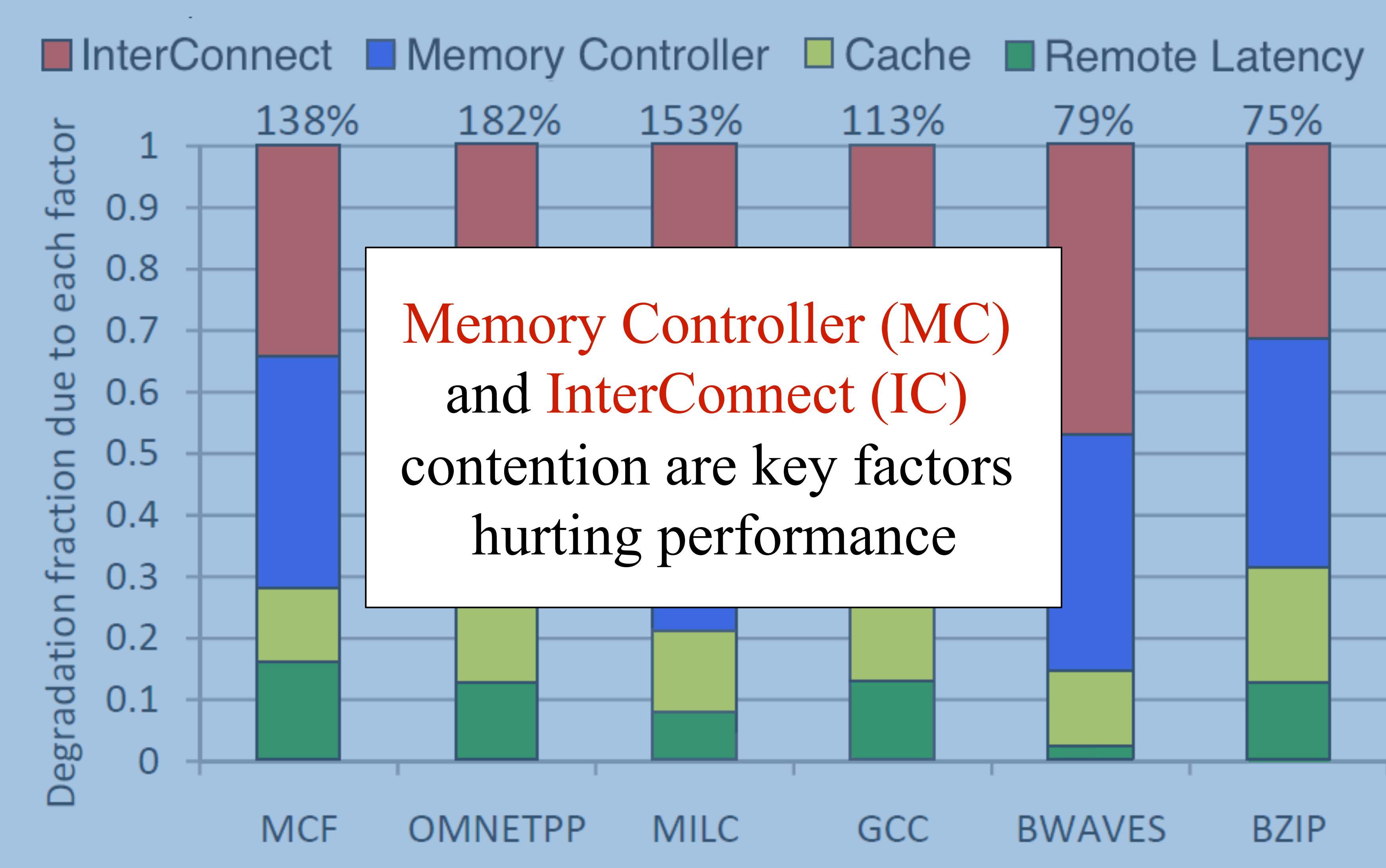
Contention for the inter-domain interconnect (IC)



Remote access latency (RL)



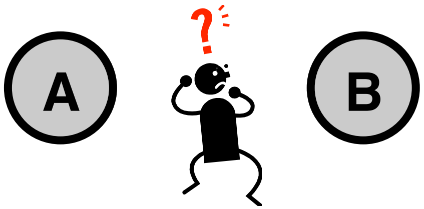
Isolating Memory controller contention (MC)



Dominant degradation factors

Characterization method

- Given two threads, decide if they will hurt each other's performance if co-scheduled



Scheduling algorithm

- Separate threads that are expected to interfere

Contention-Aware Scheduling

Limited observability

- We do not know for sure if threads compete and how severely
- Hardware does not tell us

Trial and error infeasible on large systems

- Can’t try all possible combinations
- Even sampling becomes difficult

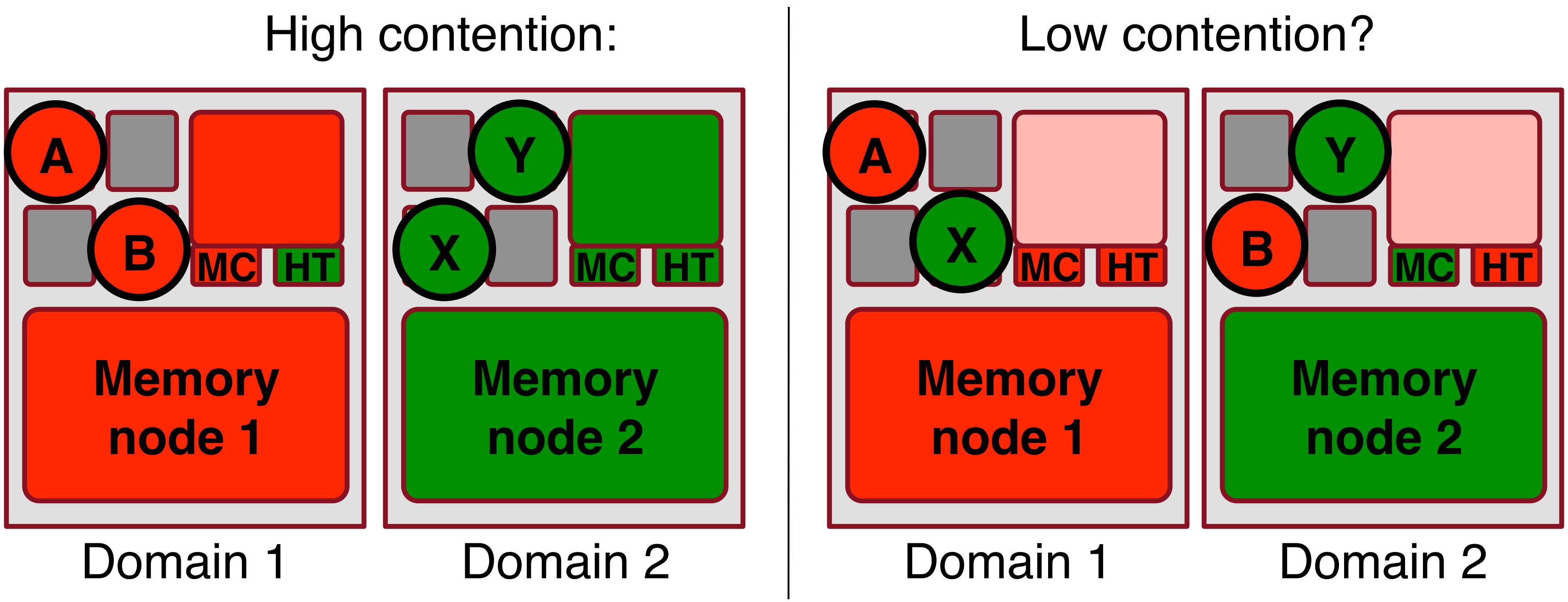
A good trade-off: measure LLC Miss rate!

- Assumes that threads interfere if they have high miss rates
- No account for cache contention impact
- Works well because cache contention is not dominant

Characterization Method

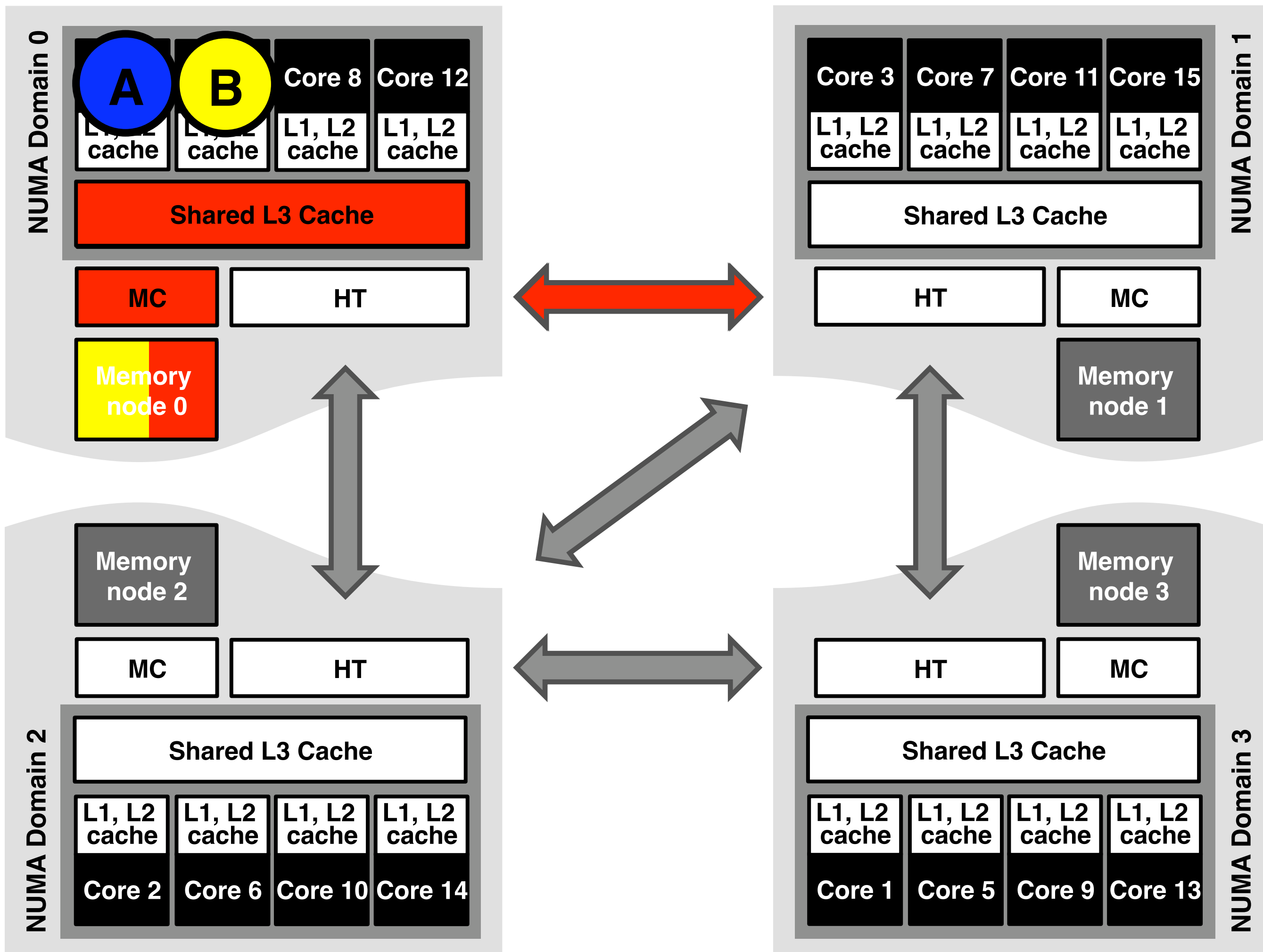
Sort threads by LLC missrate: A B X Y

Goal: isolate threads that compete for shared resources

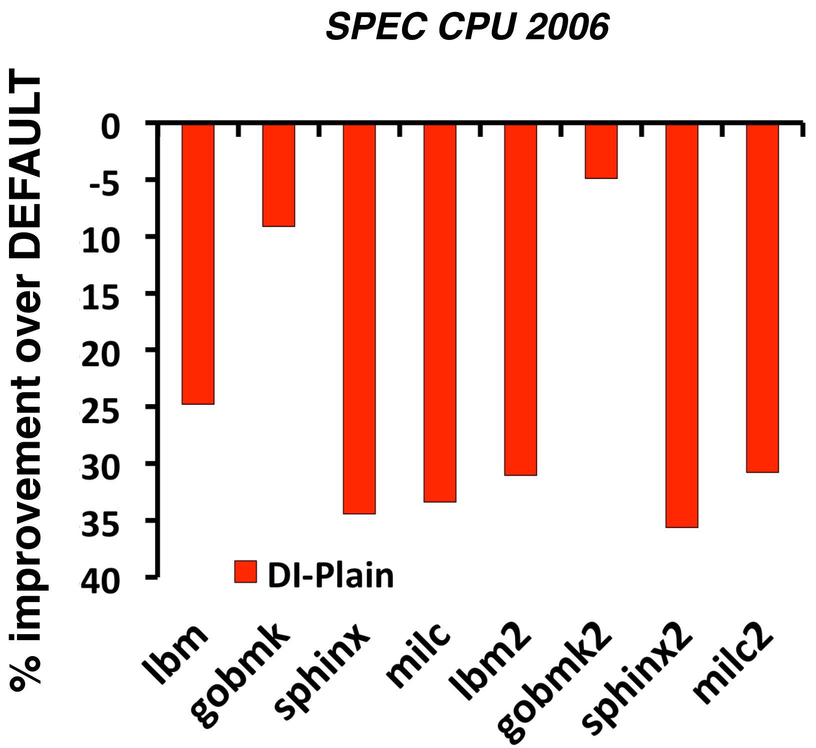


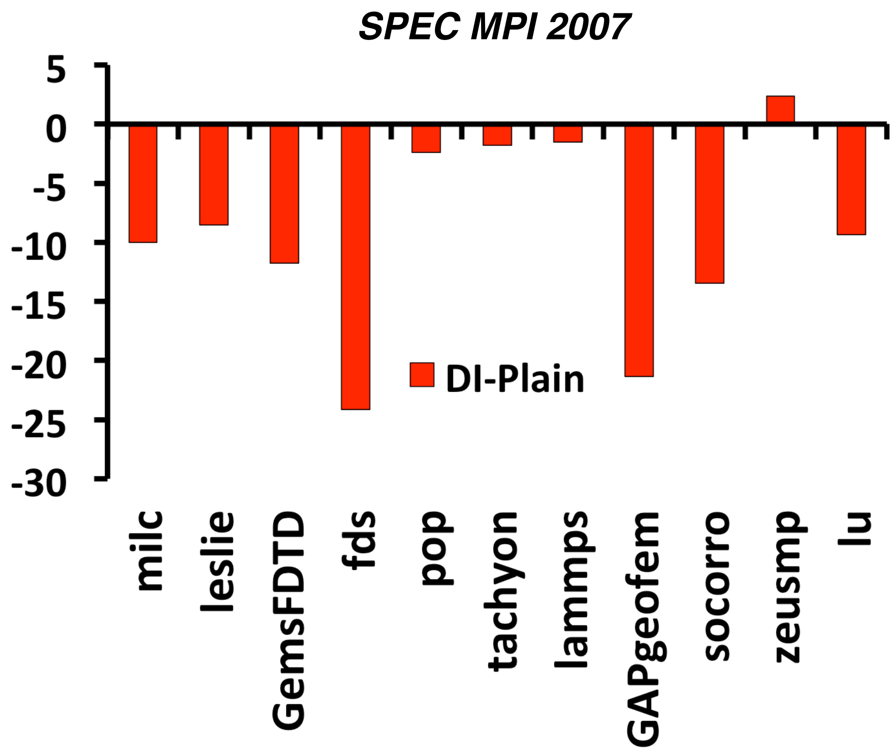
Migrate competing threads to different domains

Our previous work: an algorithm for UMA systems Distributed Intensity (DI-Plain)



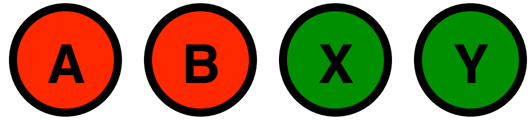
Failing to migrate memory leaves MC and introduces RL



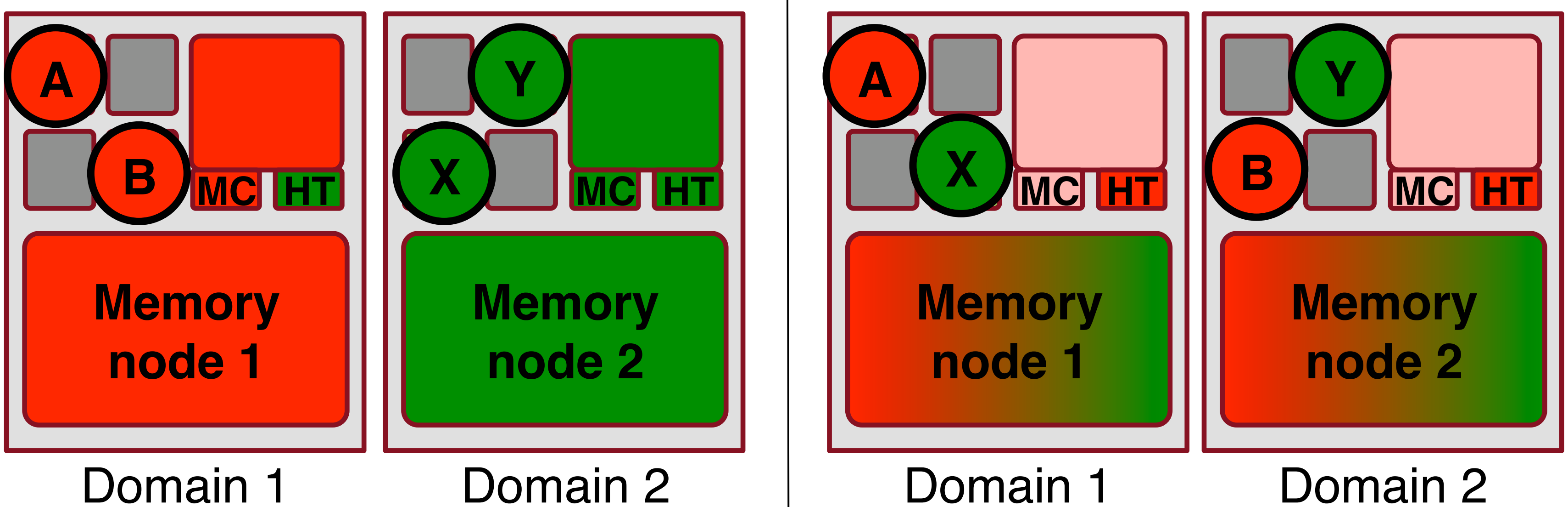


DI-Plain hurts performance on NUMA systems because it does not migrate memory!

Sort threads by LLC missrate:

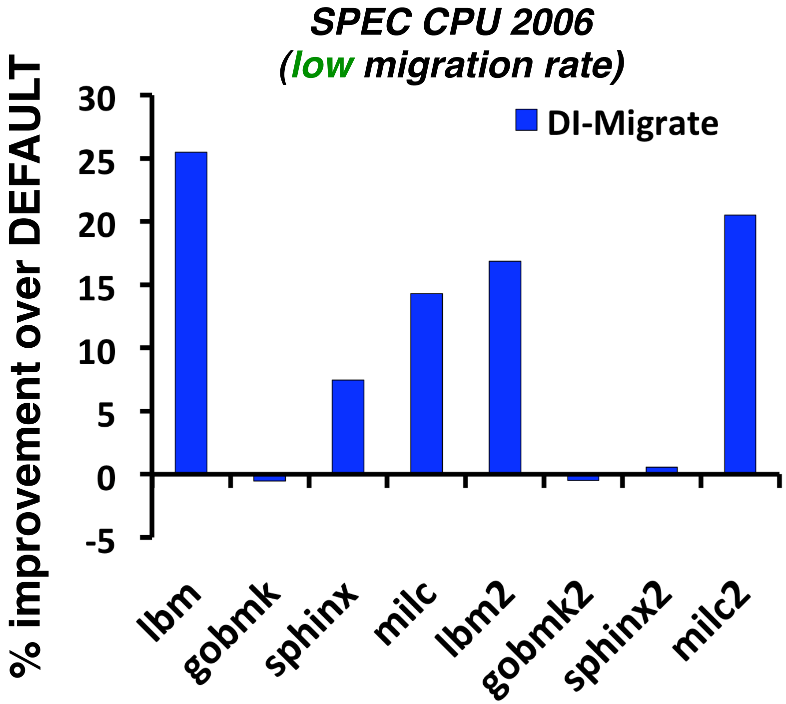


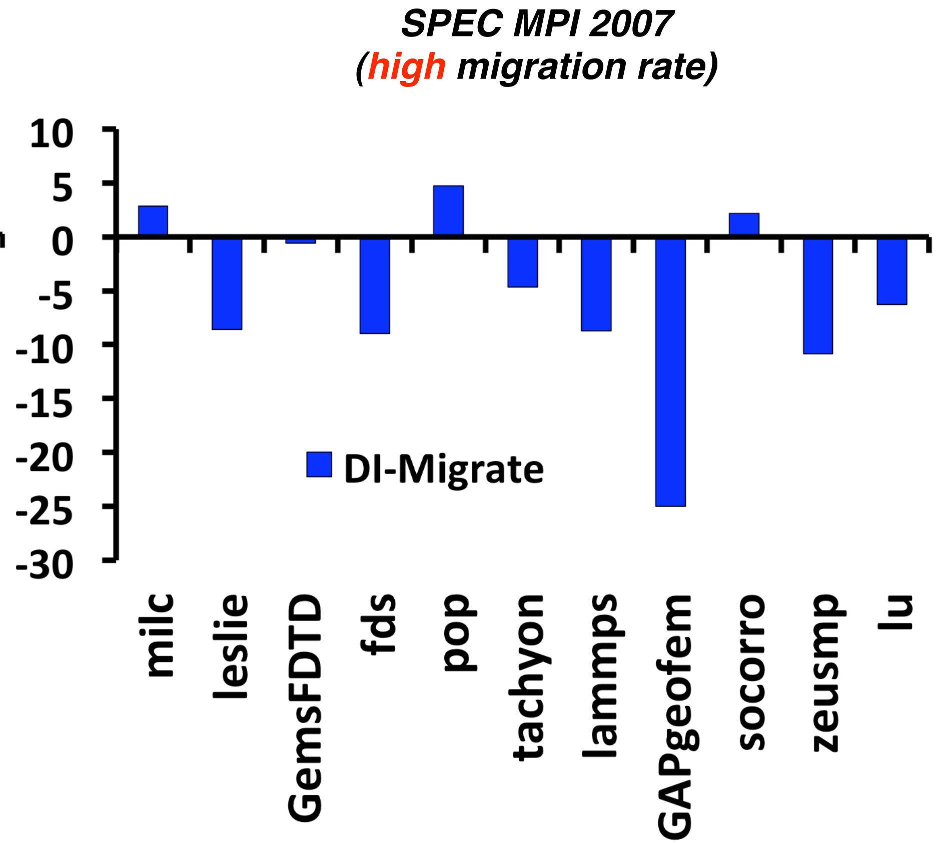
Goal: isolate threads that compete for shared resources *and pull the memory to the local node upon migration*



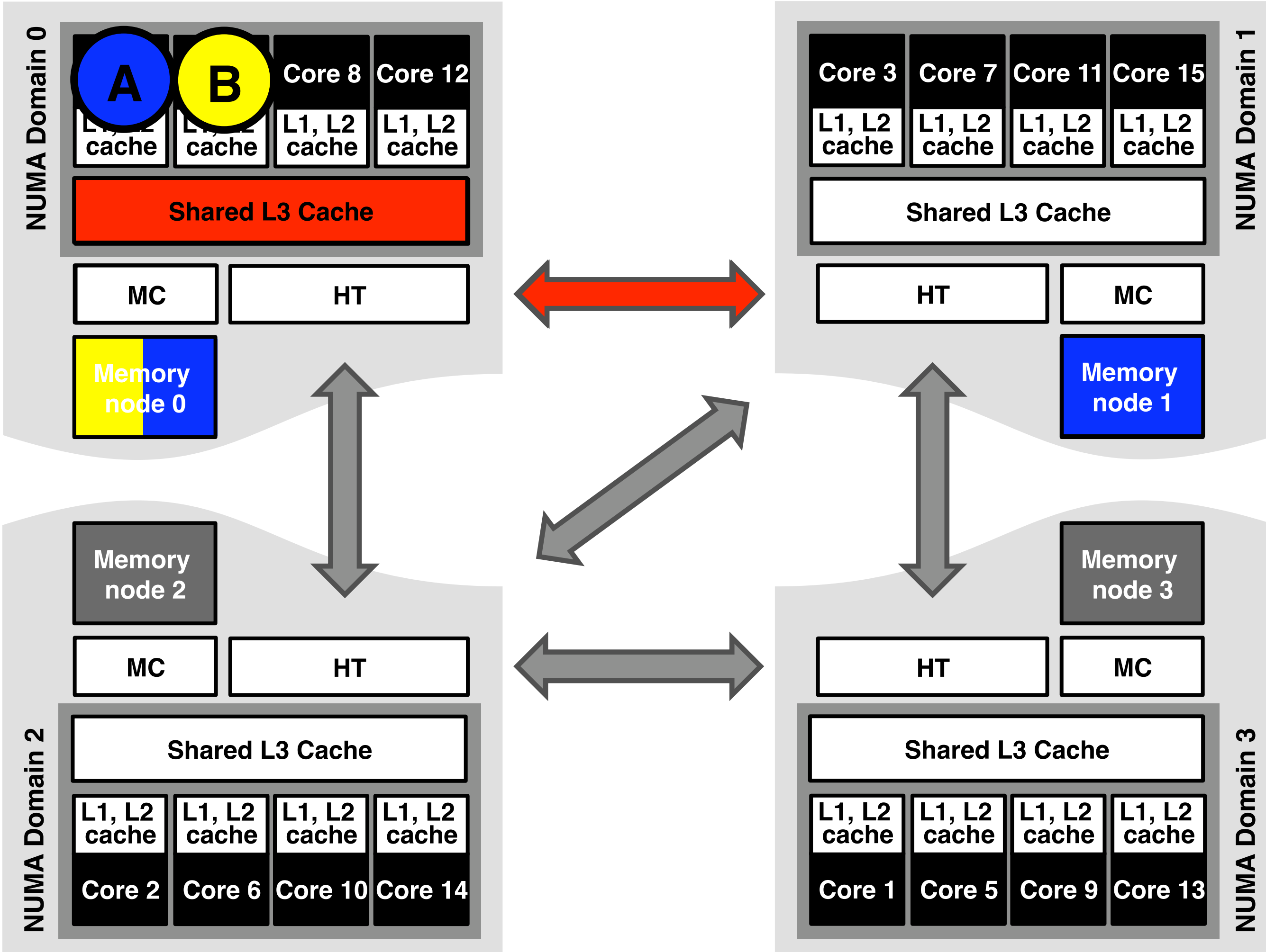
Migrate competing threads along with memory to different domains

Solution #1: Distributed Intensity with memory migration (DI-Migrate)





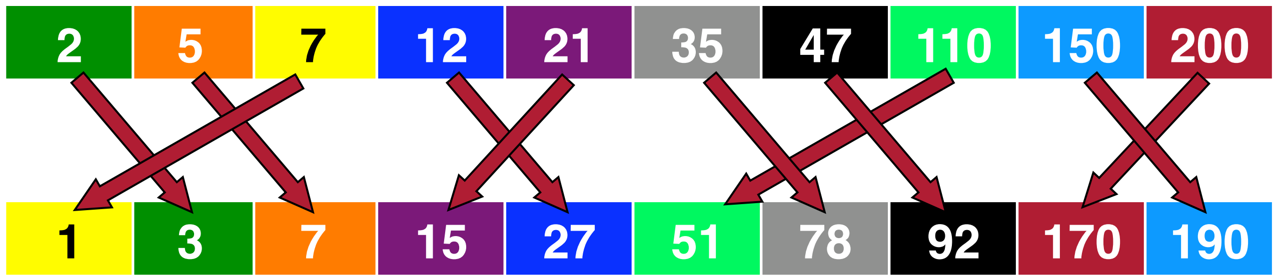
DI-Migrate performs too many migrations for MPI. Migrations are expensive on NUMA systems.



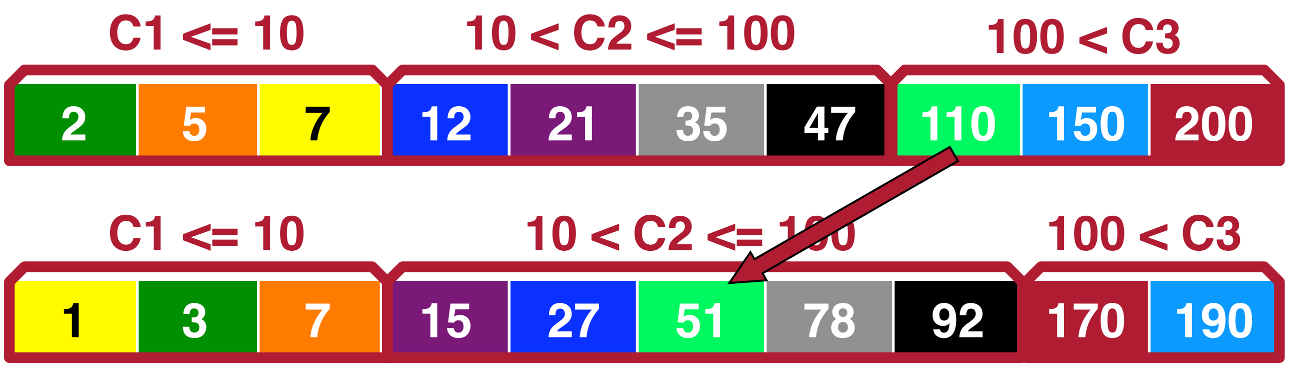
Migrating too frequently causes IC

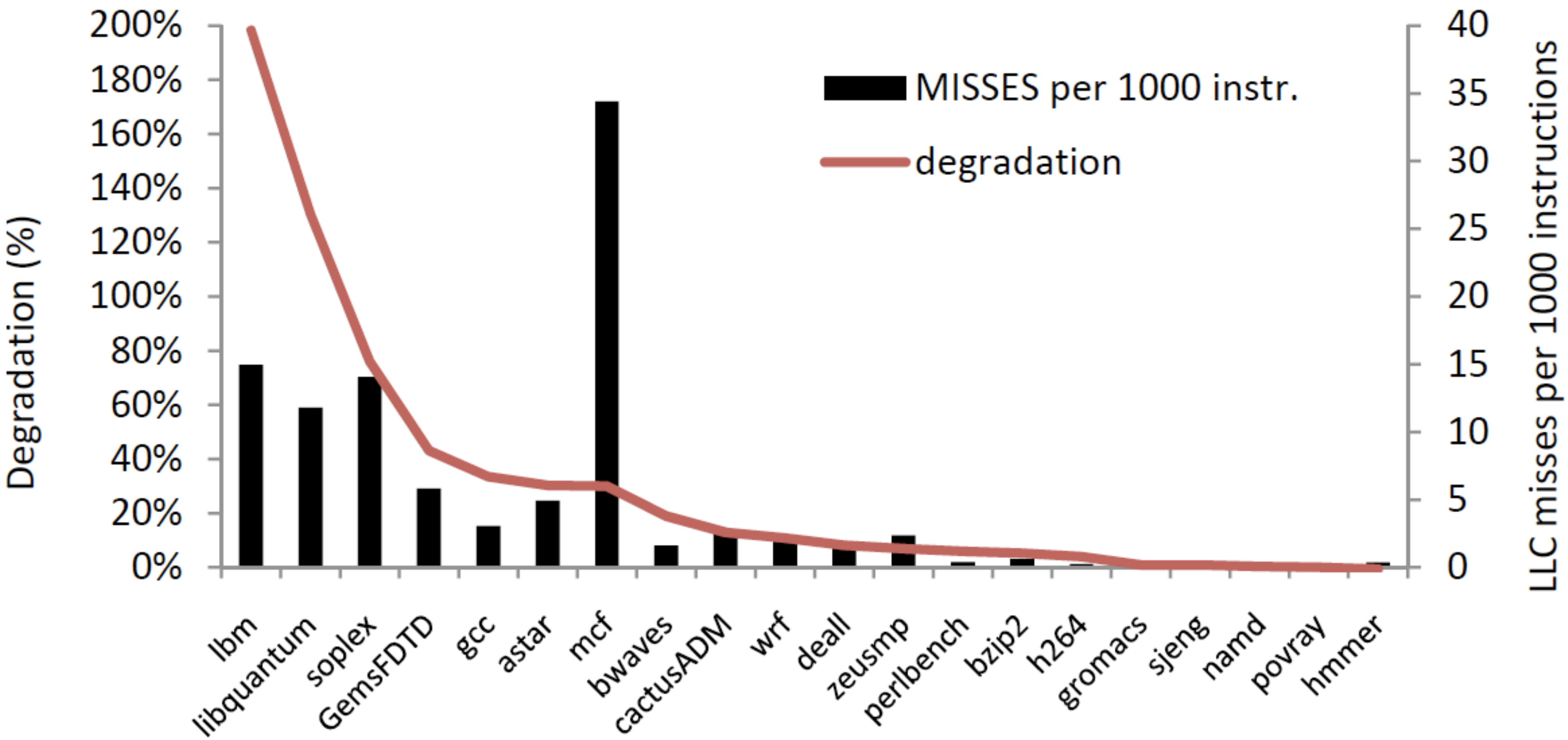
Solution #2: Distributed Intensity NUMA Online (DINO)

DI-Migrate:
- threads sorted by miss rate
- if array positions change, we migrate thread and memory



DINO:
- threads sorted by class
- only migrate if we jump from one class to another





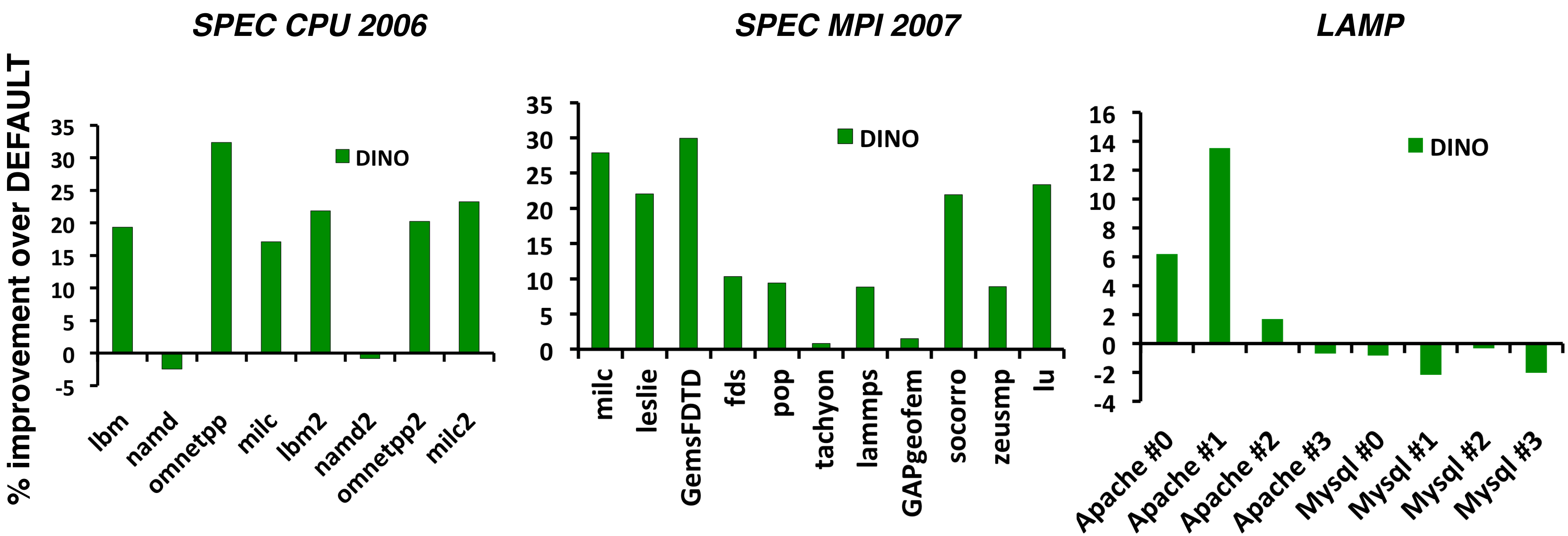
Loose correlation between miss rate and degradation, so most migrations will not payoff

	SPEC CPU 2006				
	soplex	*milc*	*lbm*	*gamess*	*namd*
DI-Migrate	**36**	**22**	**11**	**47**	**41**
DINO	**8**	**6**	**5**	**7**	**6**

	SPEC MPI 2007				
	leslie	*lamps*	*GAPgeofem*	*socorro*	*lu*
DI-Migrate	**381**	**135**	**237**	**340**	**256**
DINO	**2**	**1**	**3**	**2**	**1**

Average number of memory migrations per hour of execution (DI-Migrate and DINO)

DINO significantly reduces the number of migrations



DINO results

On NUMA systems we need to schedule threads *and memory*

- Memory Controller contention when memory is not migrated
- Interconnect Contention when memory is migrated too frequently

DINO is the contention-aware scheduling algorithm for NUMA systems that

- migrates the memory along with the application
- eliminates excessive migrations by trying to keep the workload on their old nodes, if possible
- utilizes Instruction Based Sampling to perform partial memory migration of “hot” pages

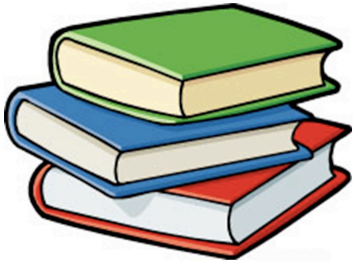
Summary

- Read our Linux Symposium 2011 paper:

"User-level scheduling on NUMA multicore systems under Linux"

- Source code is available at:

http://clavis.sourceforge.net



For further information

Any [time for] questions?