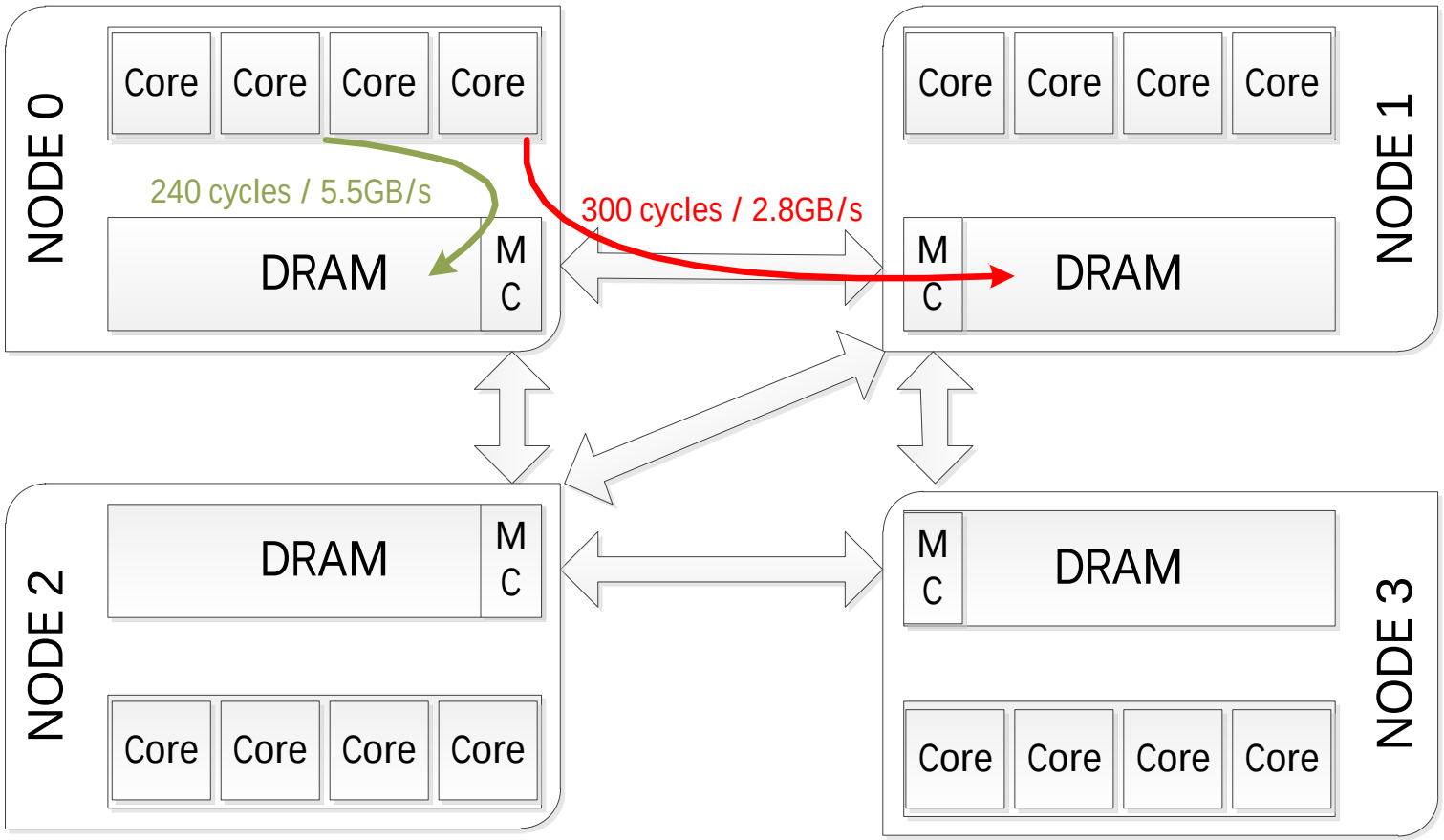
Traffic Management: A Holistic Approach to Memory Placement on NUMA Systems

Mohammad Dashti[1], Alexandra Fedorova[1], Justin Funston[1], Fabien Gaud[1], Renaud Lachaize[2], Baptiste Lepers[3], Vivien Quéma[4], Mark Roth[1]

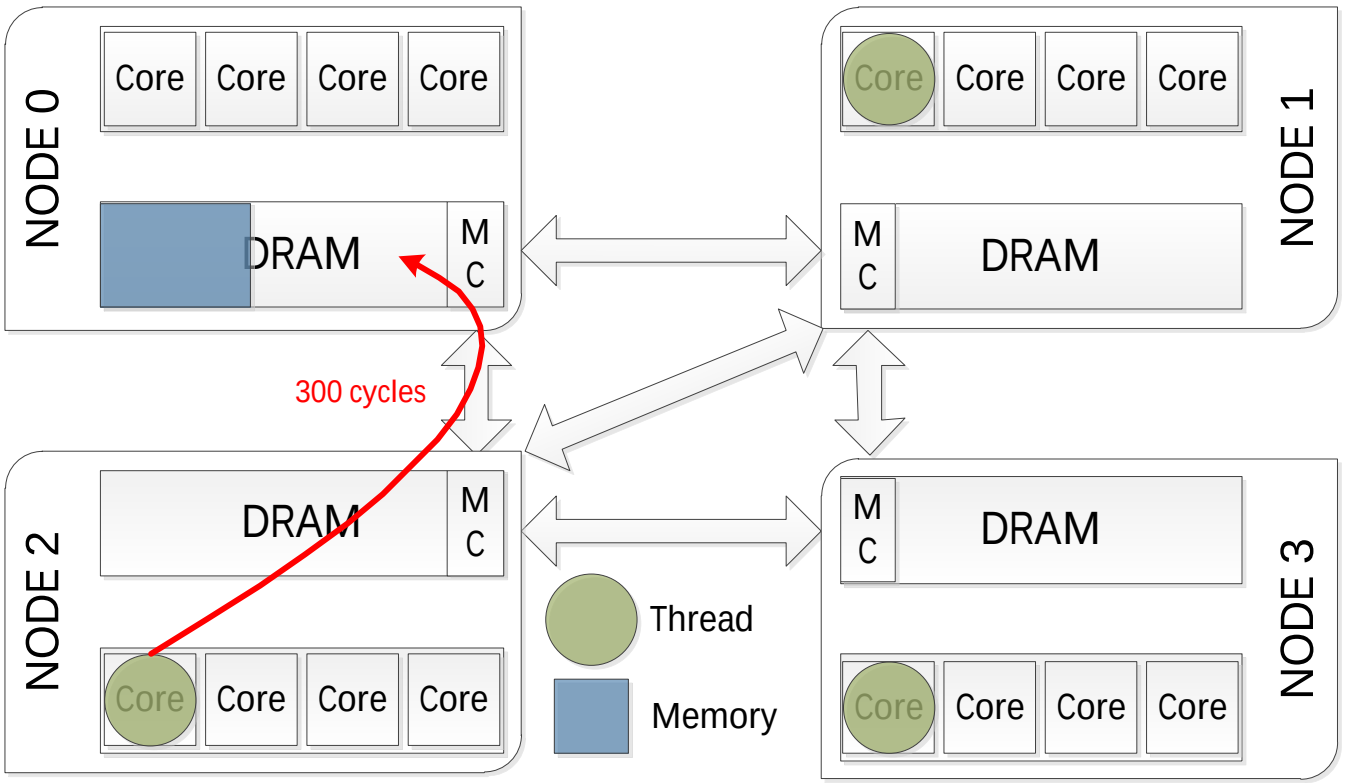
[1]Simon Fraser University
[2]Université Joseph Fourier
[3]CNRS
[4]Grenoble INP

March 19, 2013

New multicore machines are NUMA

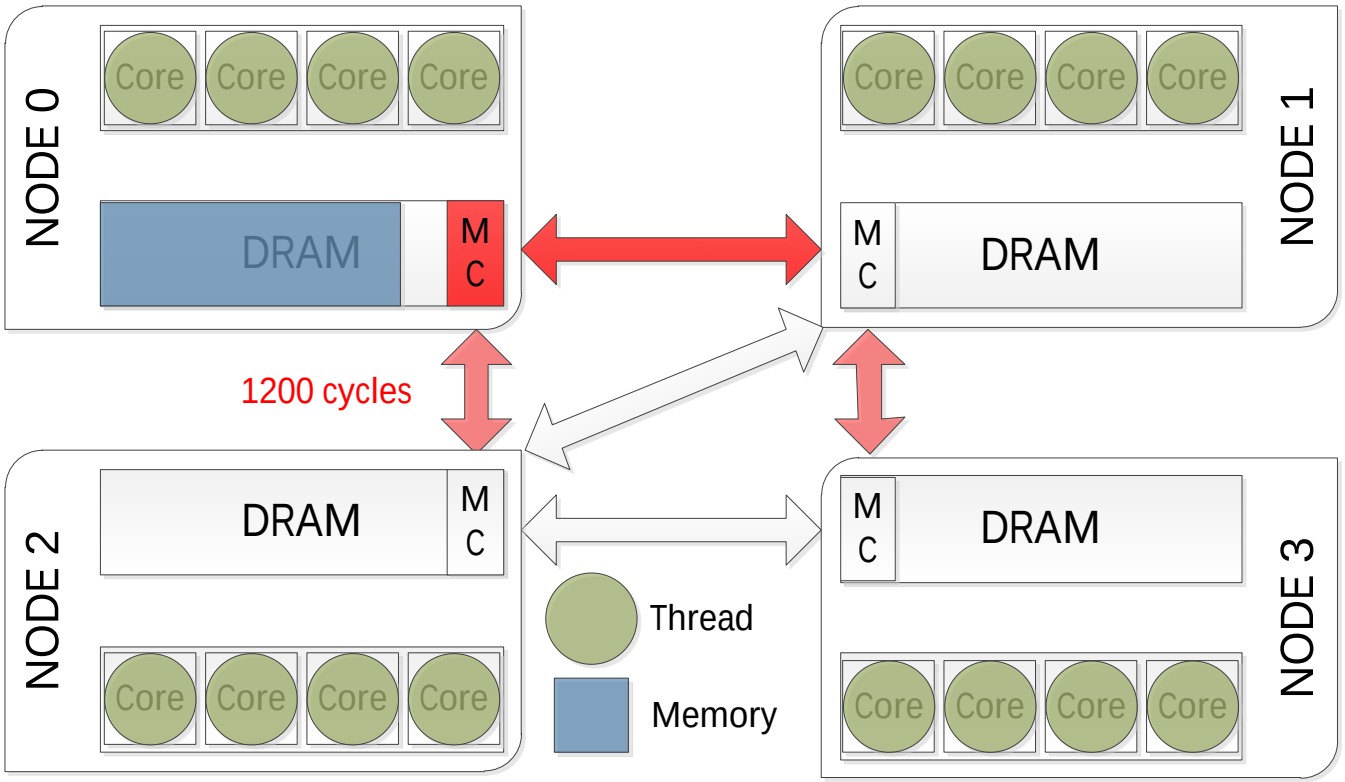


Well-know issue: remote access latency overhead



- Impacts performance by at most 30%

New issue: Memory controller and interconnect congestion



Current solutions

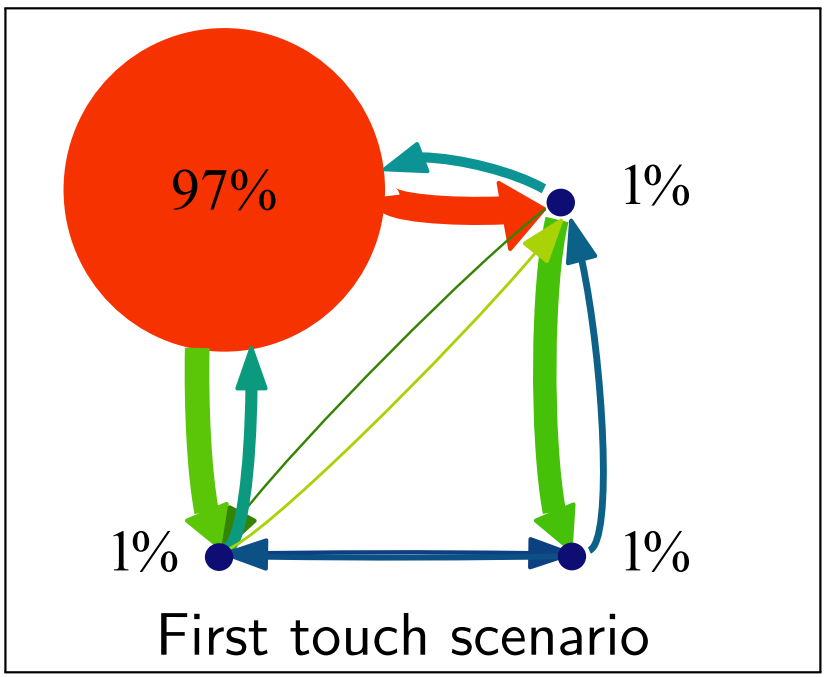
- Try to improve locality
  - Thread scheduling and page migration (USENIX ATC'11)
  - Thread Clustering (EuroSys'07)
  - Page replication (ASPLOS'96)
  - Etc.

- **But the main problem is MC/interconnect congestion**

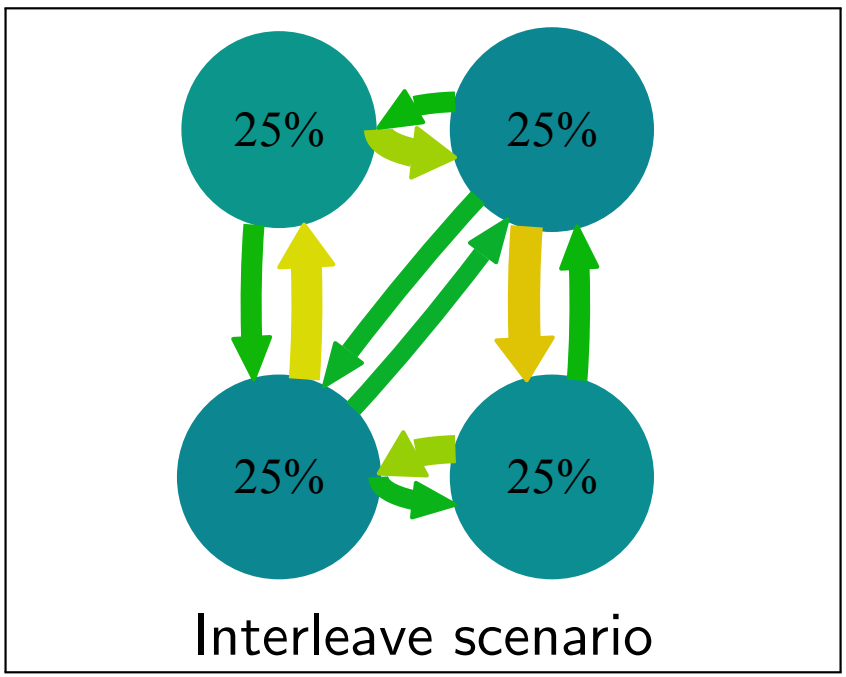
MC/Interconnect congestion impact on performance

- 16 threads, one per core
- Memory either allocated on first touch or interleaved

Example: Streamcluster

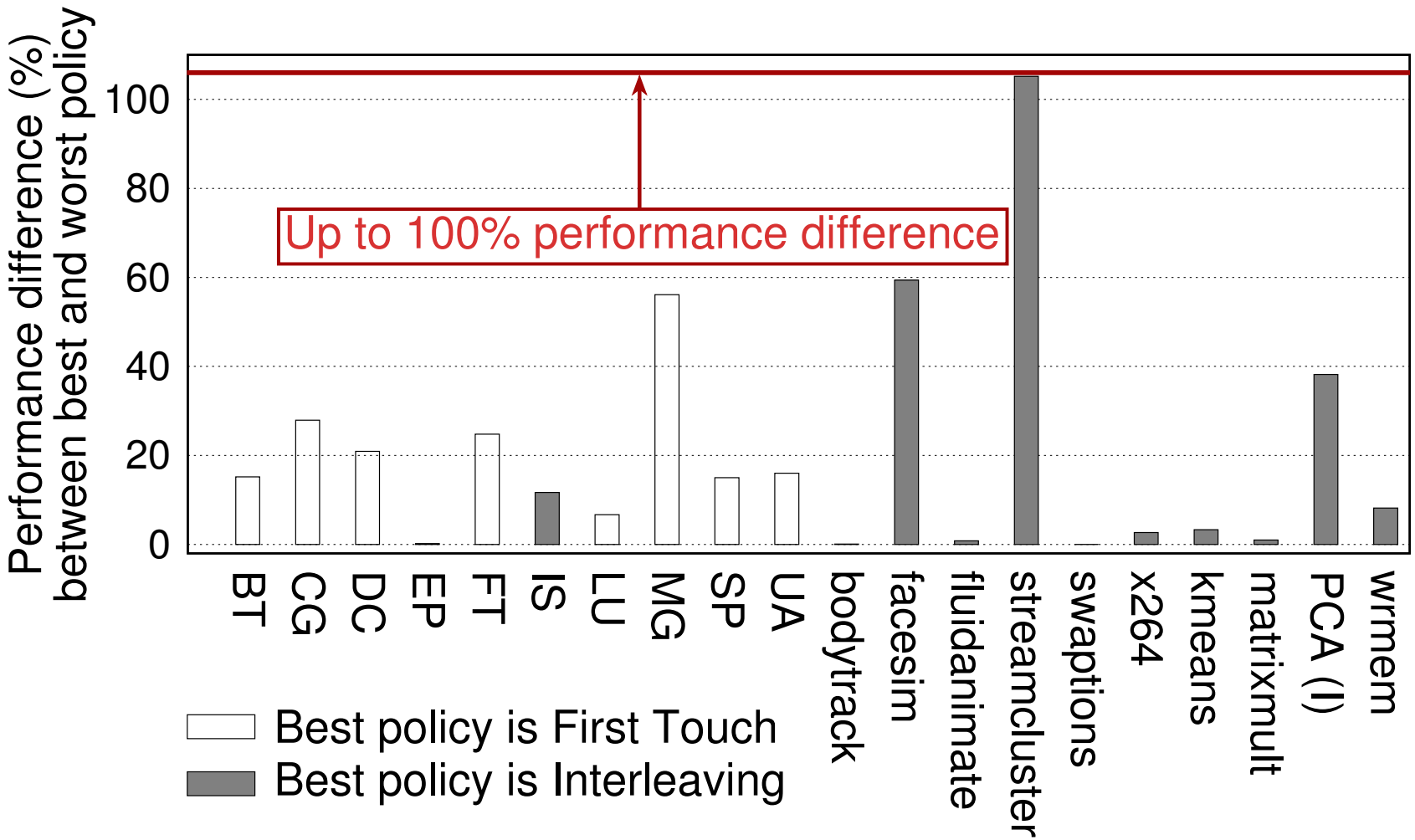


First touch scenario



Interleave scenario

MC/Interconnect congestion impact on performance (2)



Why do applications benefit from interleaving? (1)

	Streamcluster	
	Interleaving	*First touch*
Local access ratio	25%	25%
Memory latency (cycles)	471	**1169**
Memory controller imbalance	7%	**200%**
Interconnect imbalance	21%	**86%**
Perf. improvement / first touch	105%	-

$\Rightarrow$ Interconnect and memory controller congestion drive up memory access latency

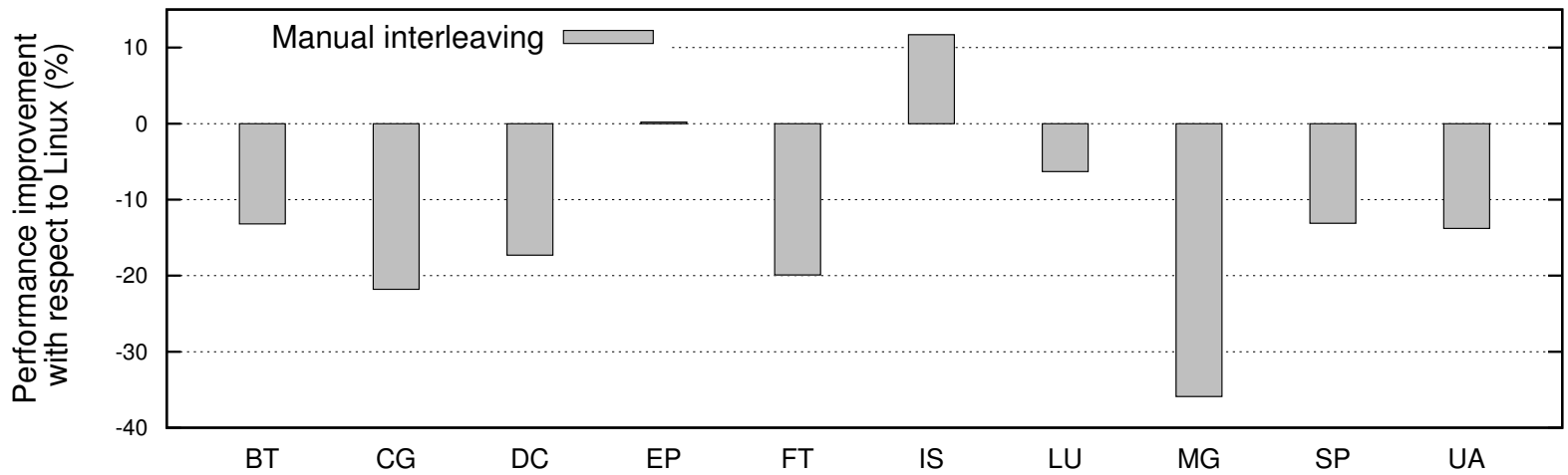
Why do applications benefit from interleaving? (2)

	PCA	
	Interleaving	*First touch*
Local access ratio	**25%**	33%
Memory latency (cycles)	480	**665**
Memory controller imbalance	4%	**154%**
Interconnect imbalance	19%	**64%**
Perf. improvement / first touch	38%	-

$\Rightarrow$ Balancing load on memory controllers is more important than improve locality

Conclusions

- **Balance is more important than locality**
  - Memory controller and interconnect congestion can drive up access latency
- Always manually interleaving memory is **NOT** the way to go

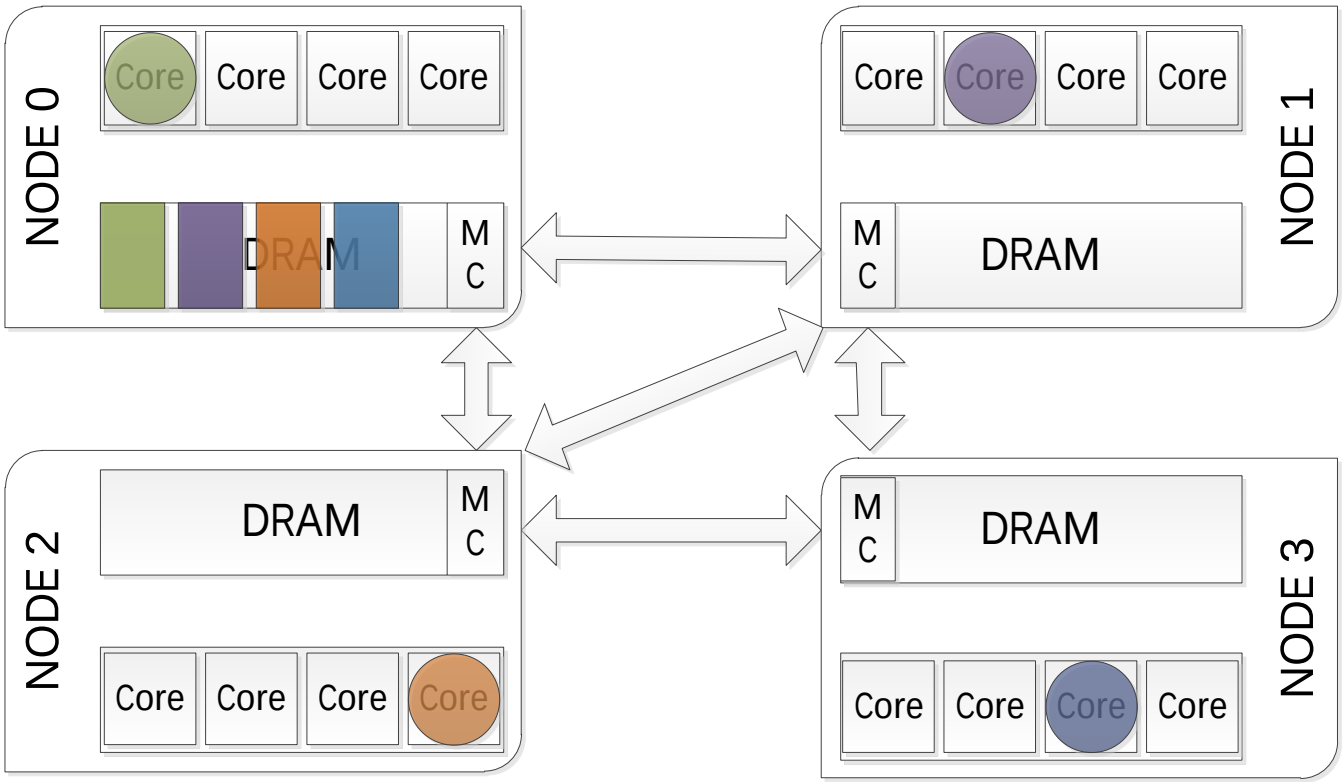


$\Rightarrow$ Need a new solution

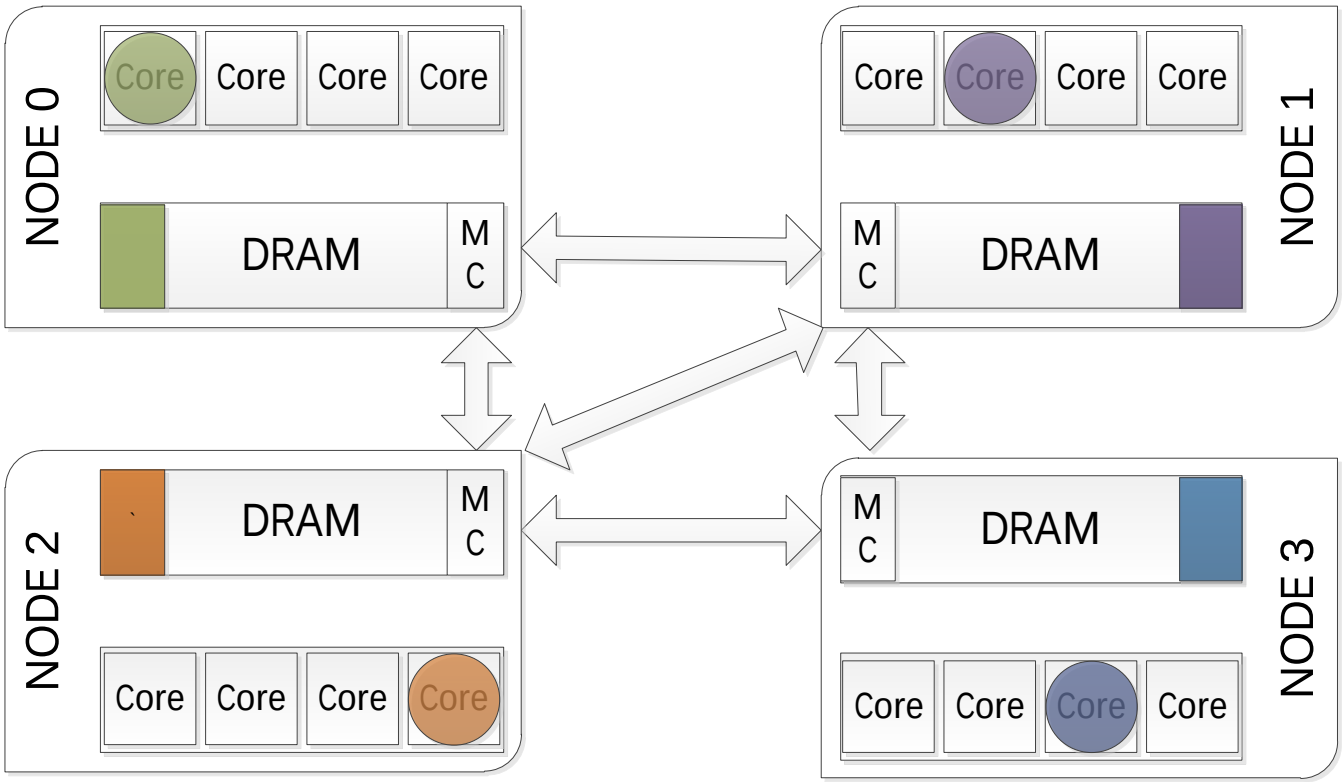
Carrefour: A new memory traffic management algorithm

- First goal: balance memory pressure on interconnect and MC
- Second goal: improve locality

Mechanism #1: Page relocation



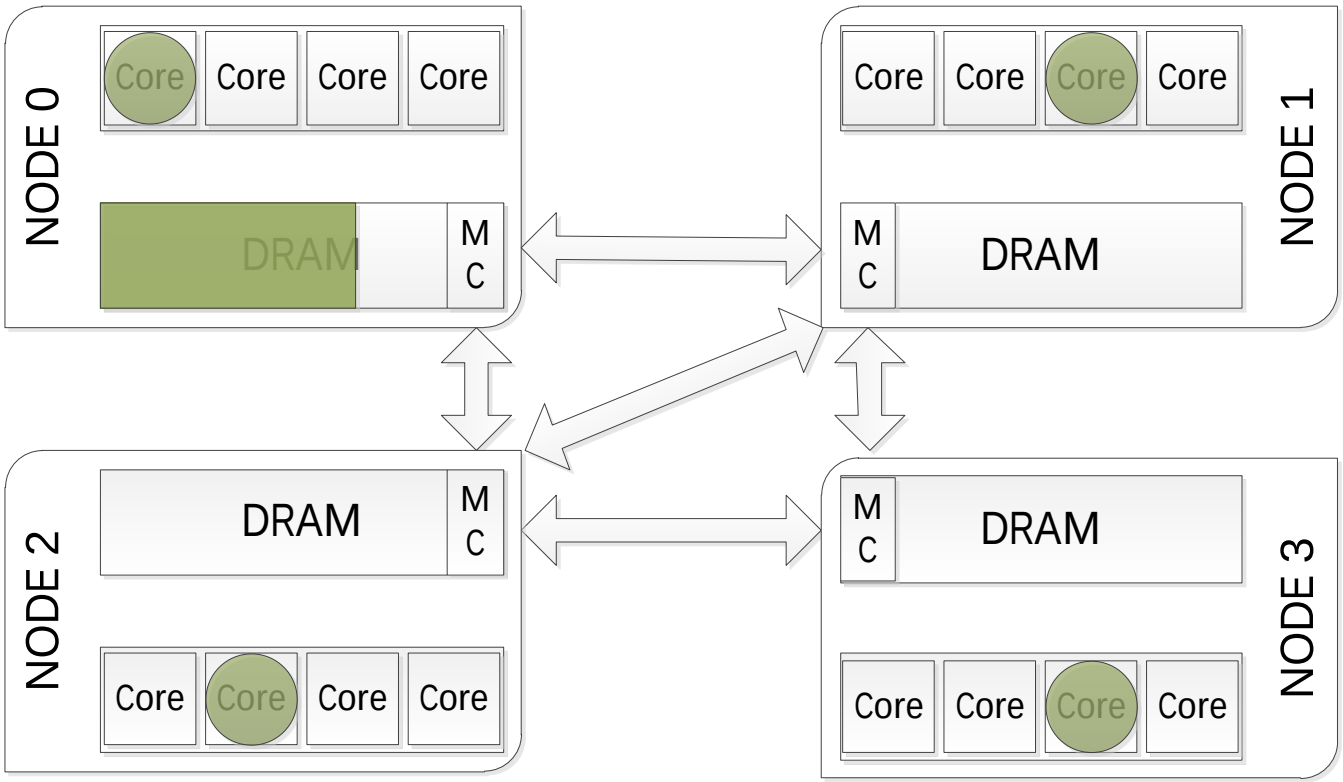
Mechanism #1: Page relocation



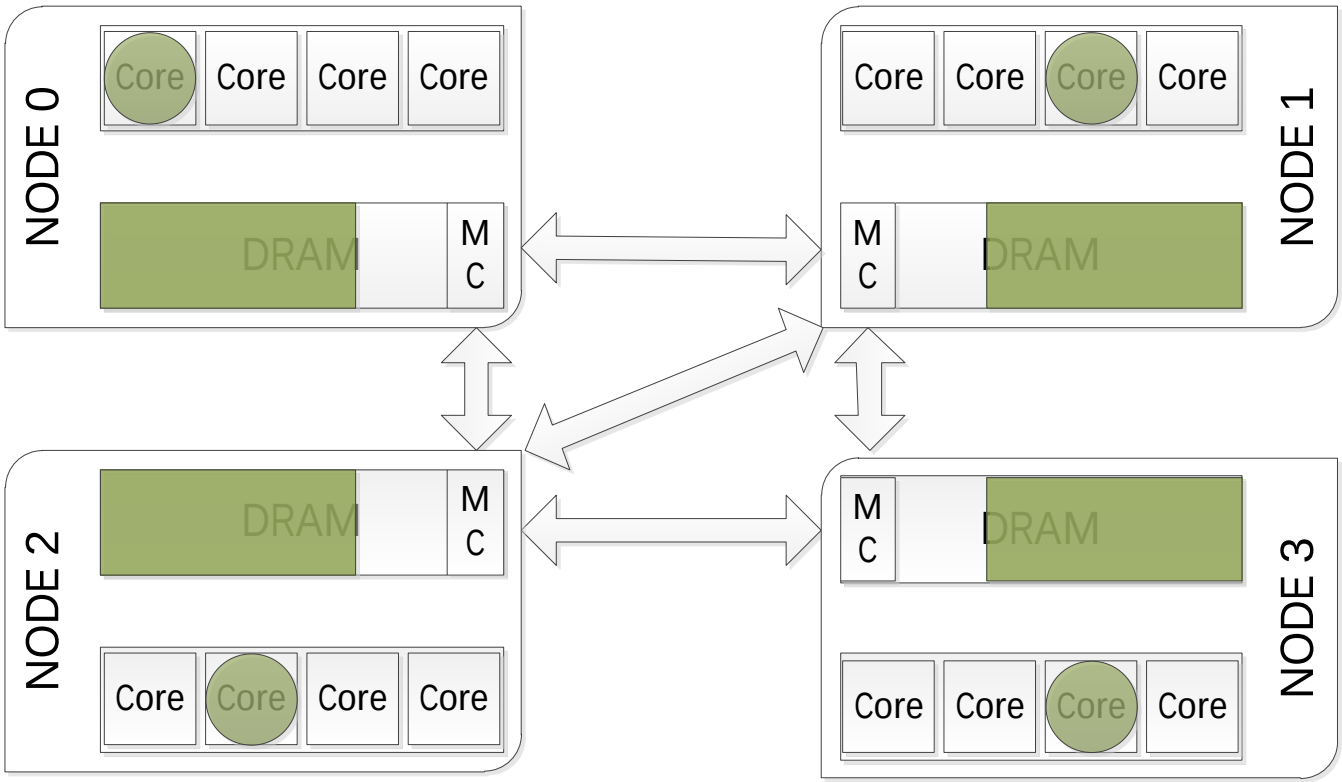
- ☺ Better locality
- ☺ Lower interconnect load
- ☺ Balanced load on MC

- ☹ Cannot be applied if region is shared by multiple threads

Mechanism #2: Page replication



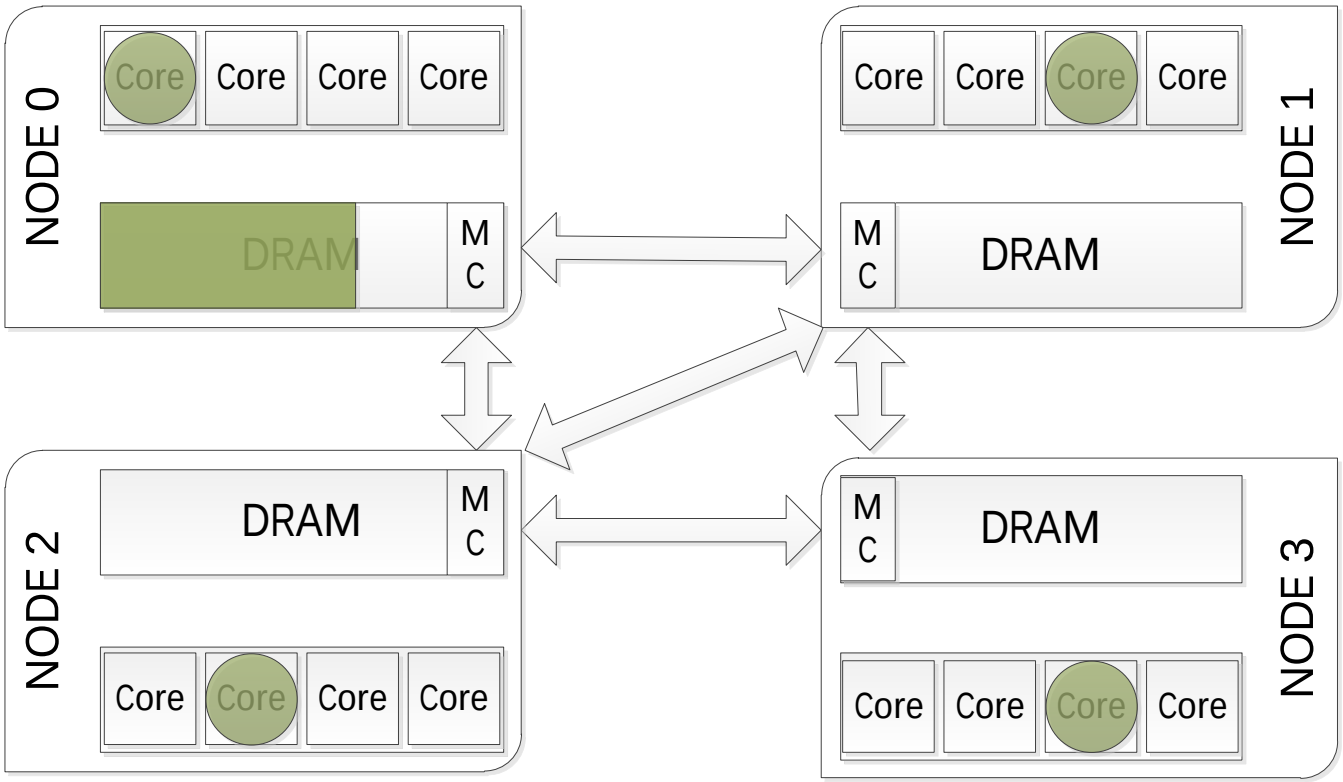
Mechanism #2: Page replication



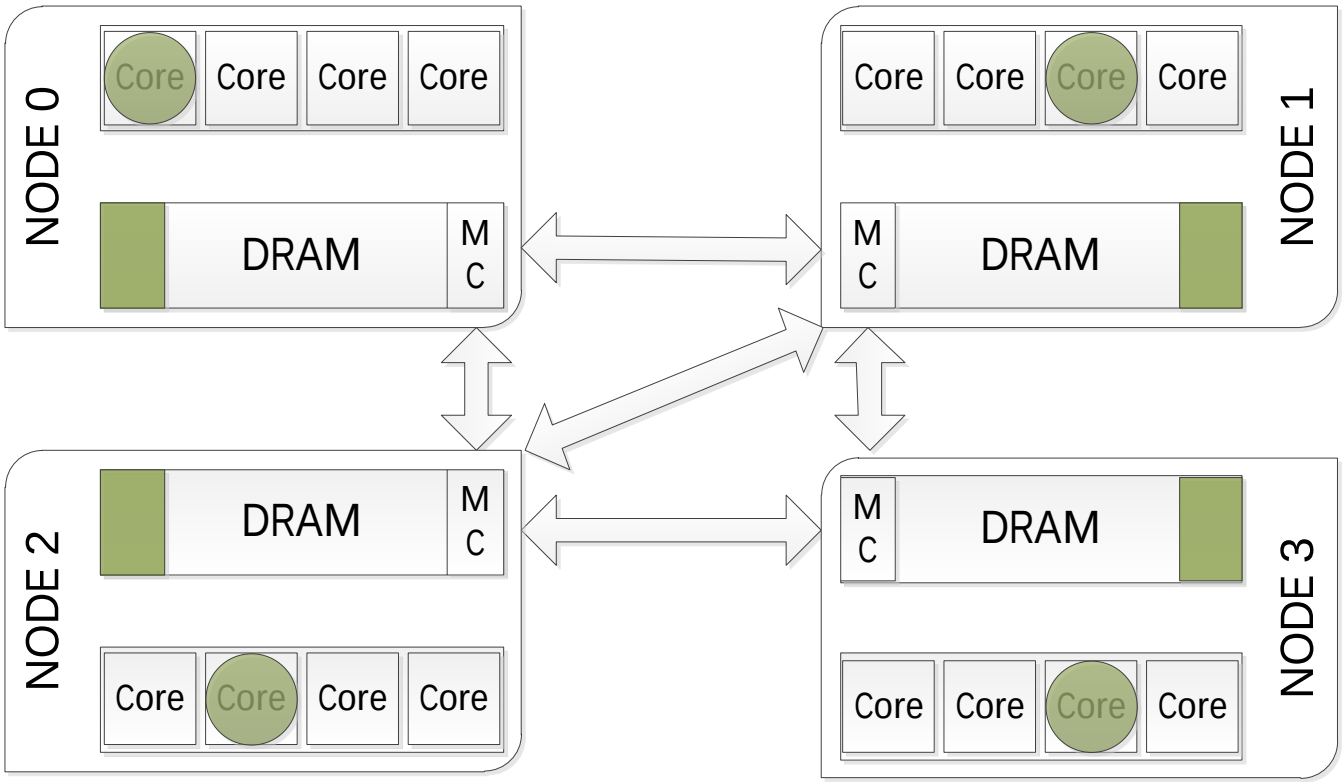
- ☺ Better locality
- ☺ Lower interconnect load
- ☺ Balanced load on MC

- ☹ Higher memory consumption
- ☹ Expensive synchronization

Mechanism #3: Page interleaving



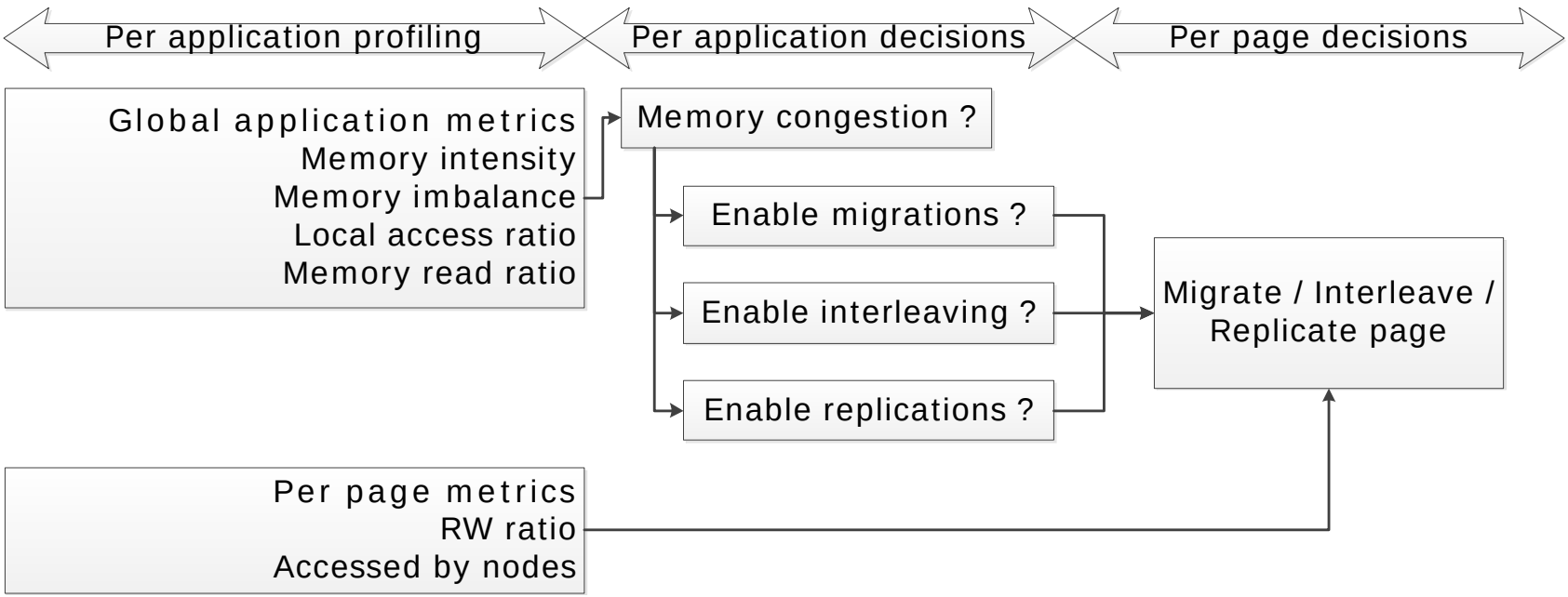
Mechanism #3: Page interleaving



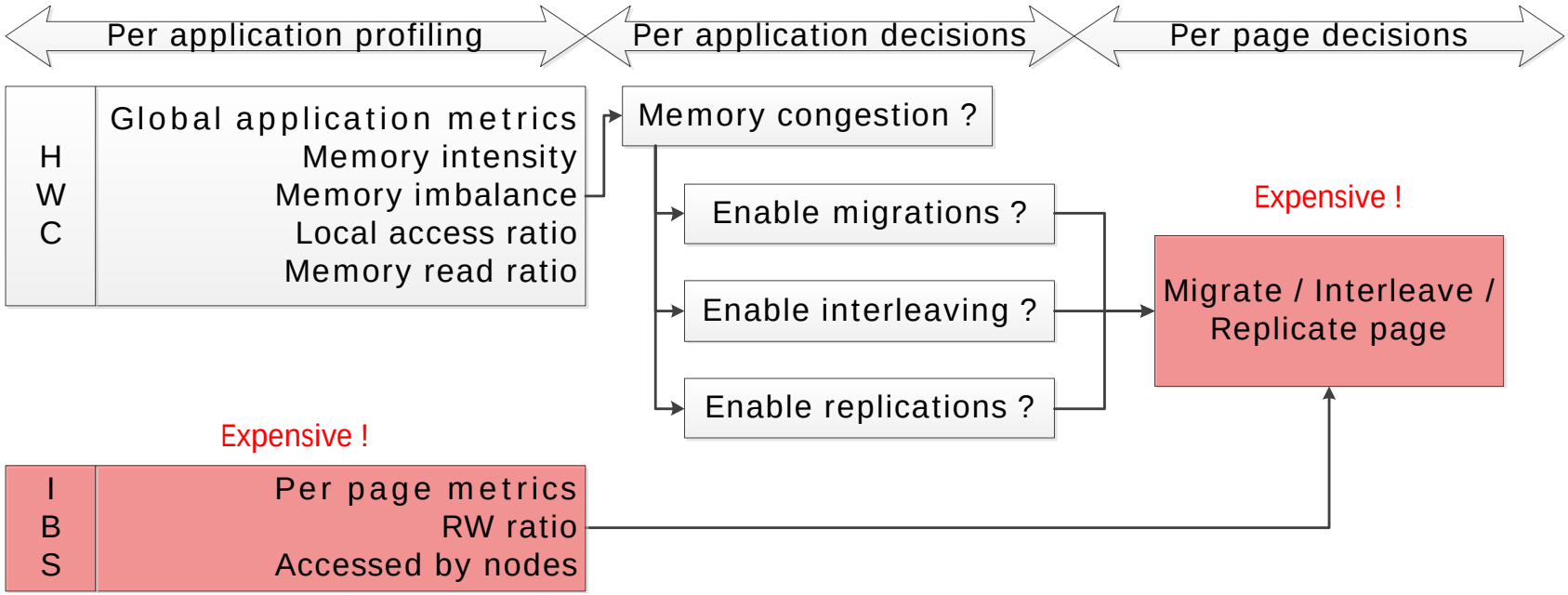
- ☺ Balanced load on interconnect
- ☺ Balanced load on MC
- ☹ Can decrease locality

Carrefour in details

- Goal: Combine these techniques to:
  1. Balance memory pressure
  2. Increase locality

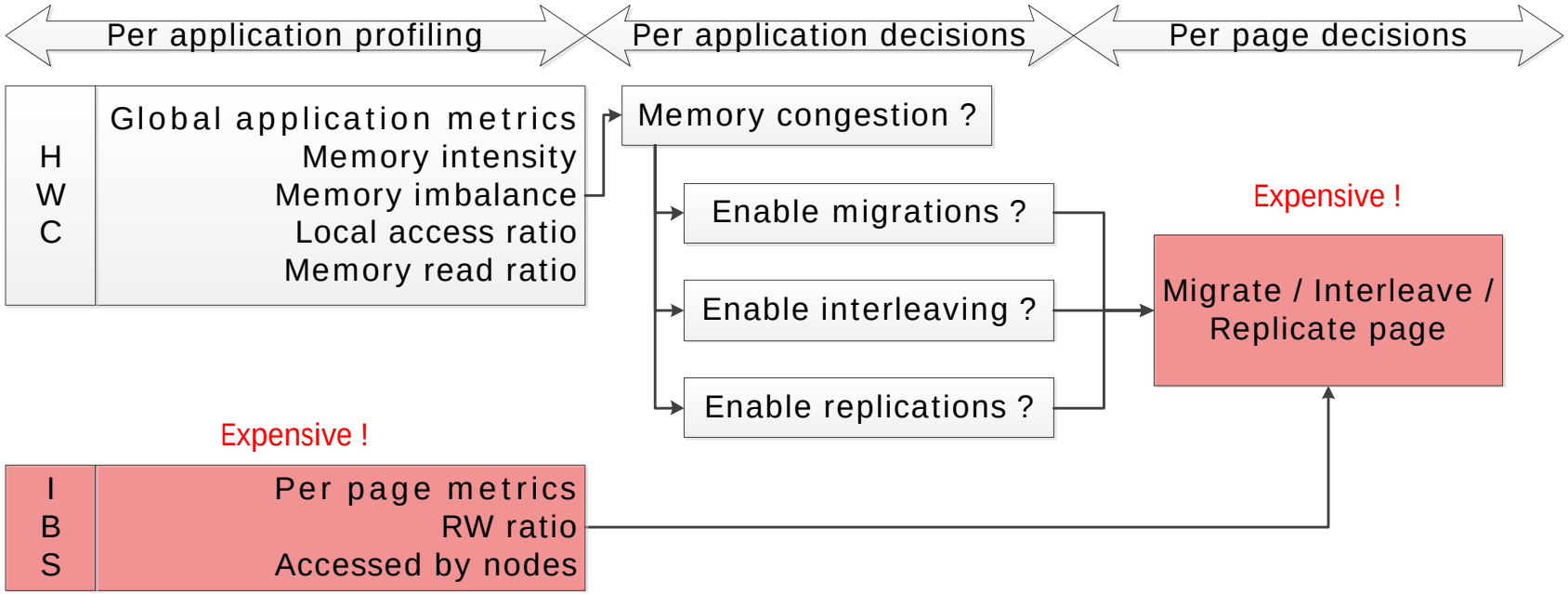


Carrefour in details



- Accurate and low-overhead page access statistics
  - Adaptive IBS sampling
  - Include cache accesses
  - Use hardware counter feedback

Carrefour in details

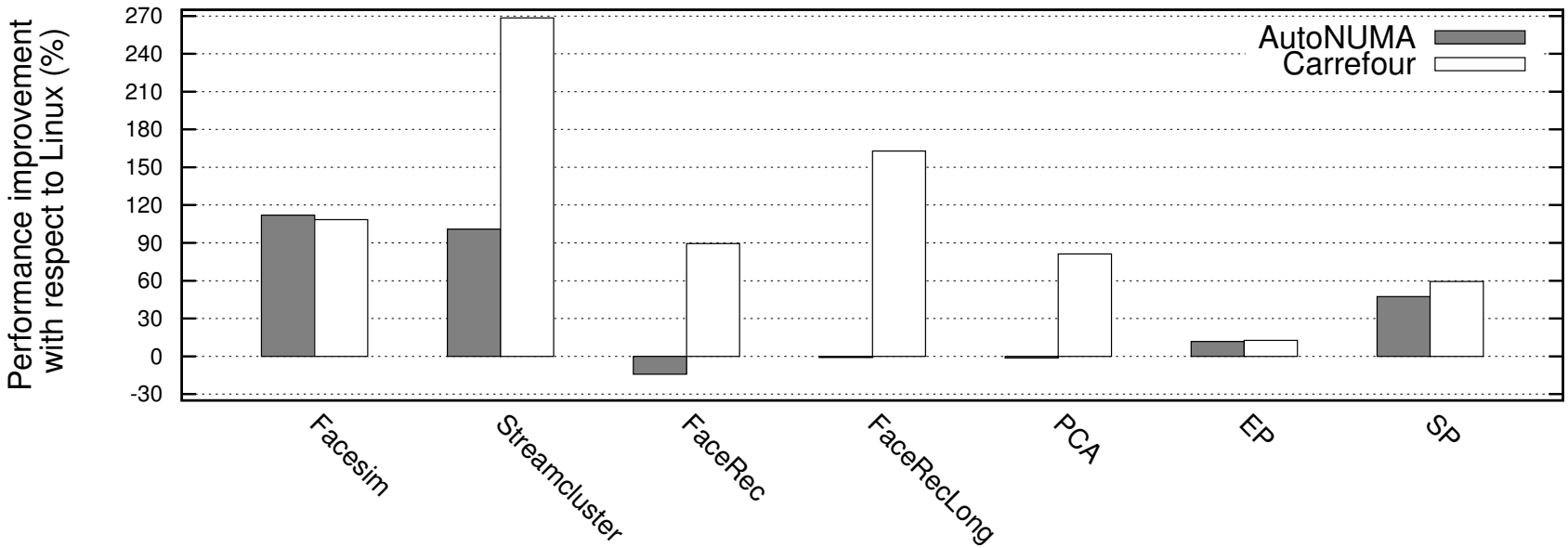


- Efficient page replication
  - Use a careful implementation (fine grain locks)
  - Prevent data synchronization

Evaluation

- Carrefour is implemented in Linux 3.6
- Machines
  - 16 cores, 4 nodes, 64 GB of RAM
  - **24 cores, 4 nodes, 64 GB of RAM**
- Benchmarks (23 applications)
  - Parsec
  - FaceRec
  - Metis (Map/Reduce)
  - NAS
- Compare Carrefour to
  - Linux (default)
  - Linux Autonuma
  - ~~Manual Interleaving~~

Performance



⇒ **Carrefour significantly improves performance !**

Carrefour overhead

Configuration	**Maximum overhead / default**
Autonuma	25%
Carrefour	4%

- Carrefour average overhead when no decision are taken: 2%

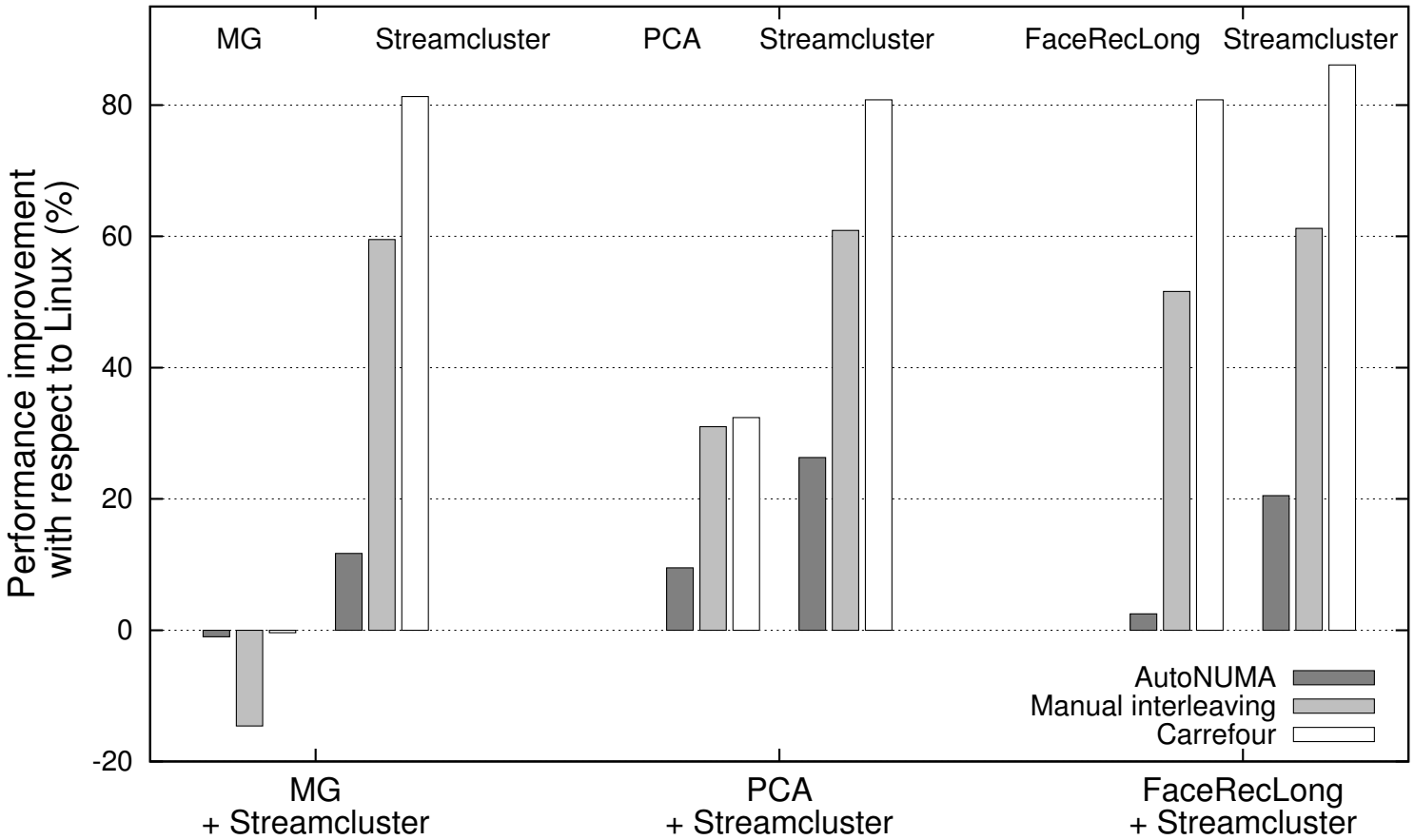
Conclusion

- In modern NUMA systems:
  - Remote latency overhead is not the main bottleneck
  - MC and interconnect congestion can drive up memory latency
- **Carrefour:** a memory traffic management algorithm
  - First goal: balance memory pressure on interconnect and MC
  - Second goal: improve locality
- **Performance:**
  - Improves performance significantly (up to 270%)
  - Outperforms others solutions

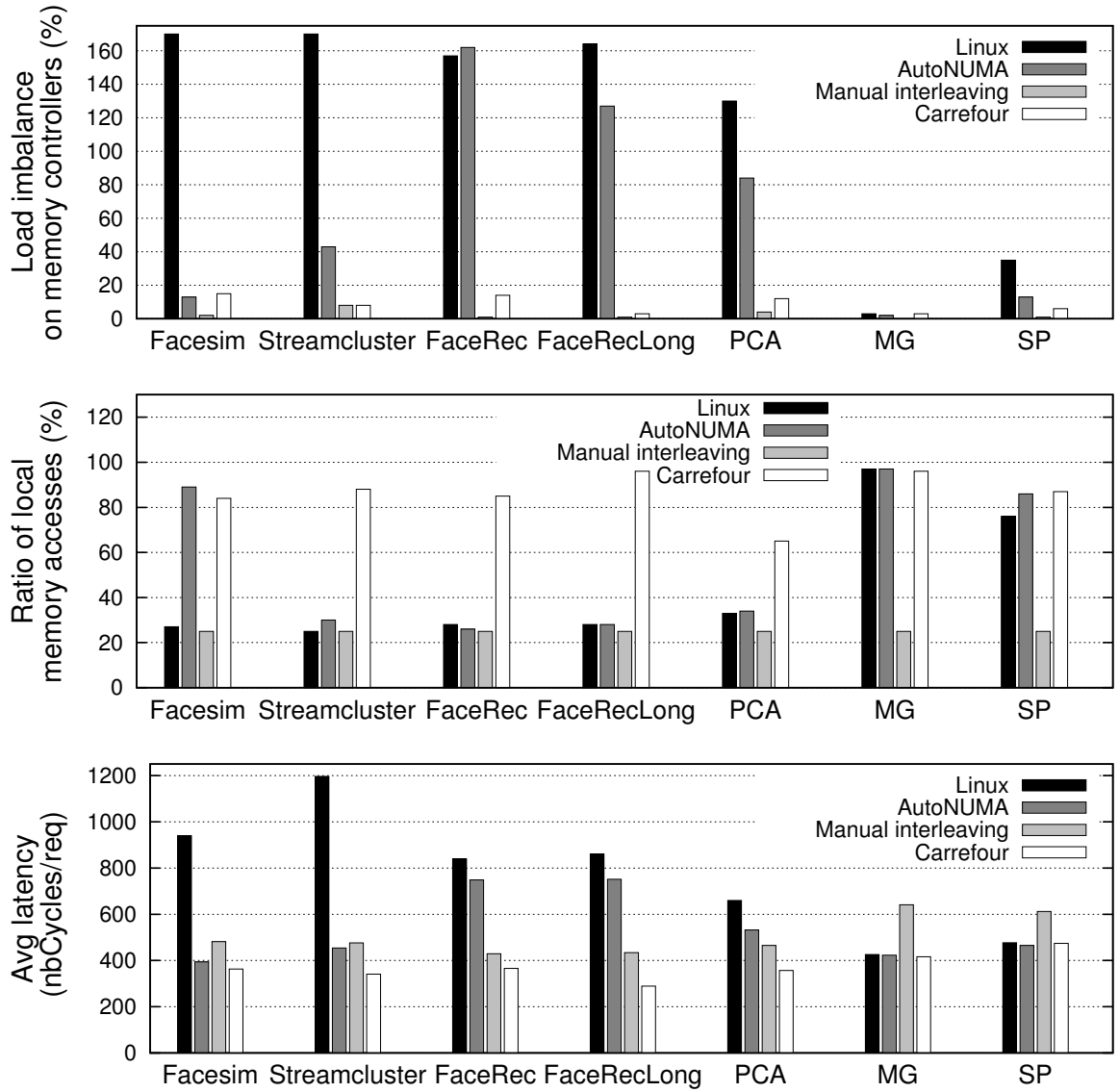
Questions?

https://github.com/Carrefour

Carrefour supports multi-applications workloads



Detailed profiling



Energy consumption