Memory System Performance in a NUMA Multicore Multiprocessor

Zoltan Majo and Thomas R. Gross

Department of Computer Science
ETH Zurich

Summary

- NUMA multicore systems are **unfair** to local memory accesses

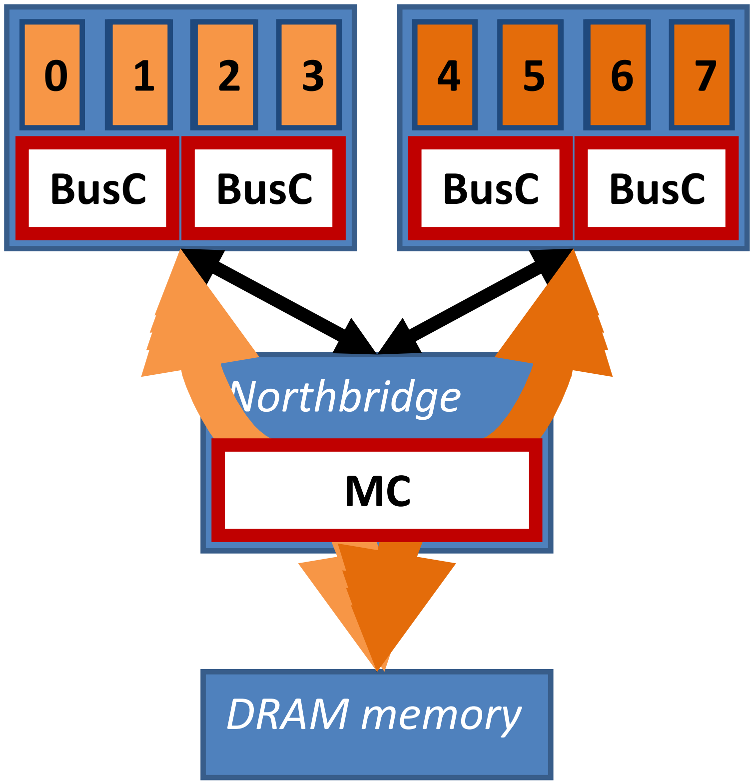
- **Local execution** sometimes **suboptimal**

Outline

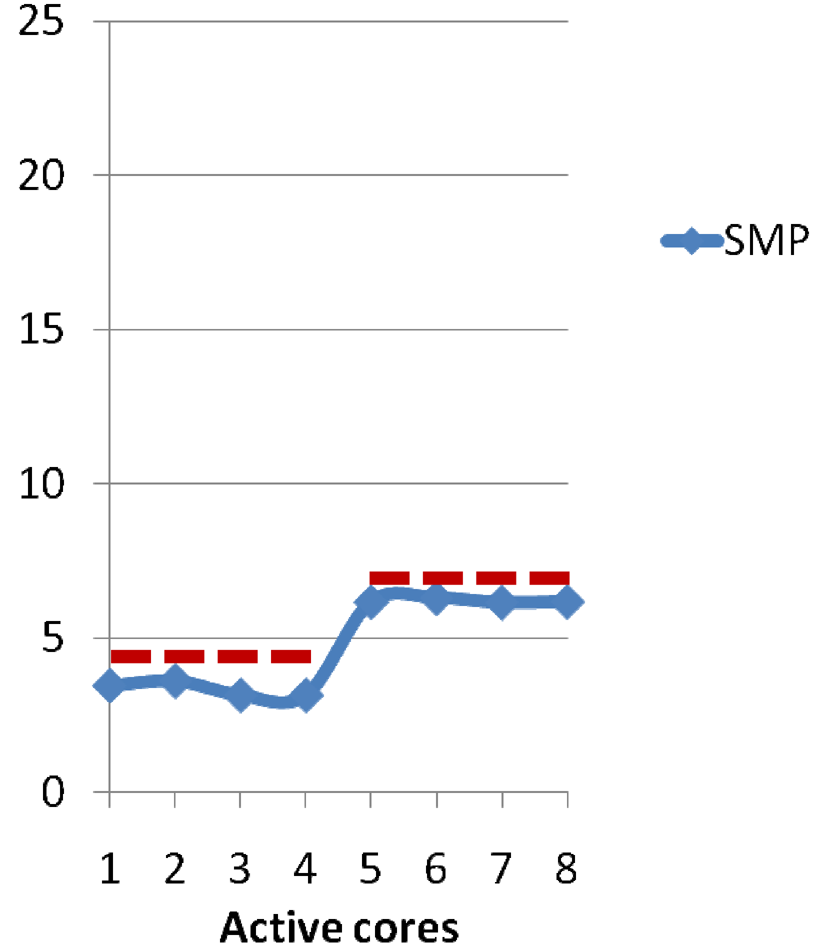
- NUMA multicores: how it happened
- Experimental evaluation: Intel Nehalem
- Bandwidth sharing model
- The next generation: Intel Westmere

NUMA multicores: how it happened

First generation: SMP

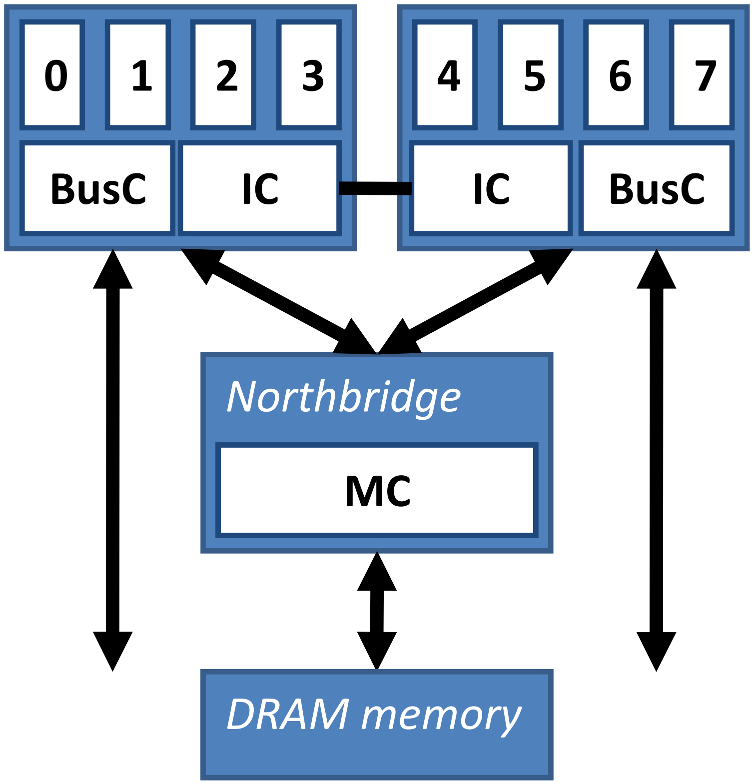


Total bandwidth [GB/s]

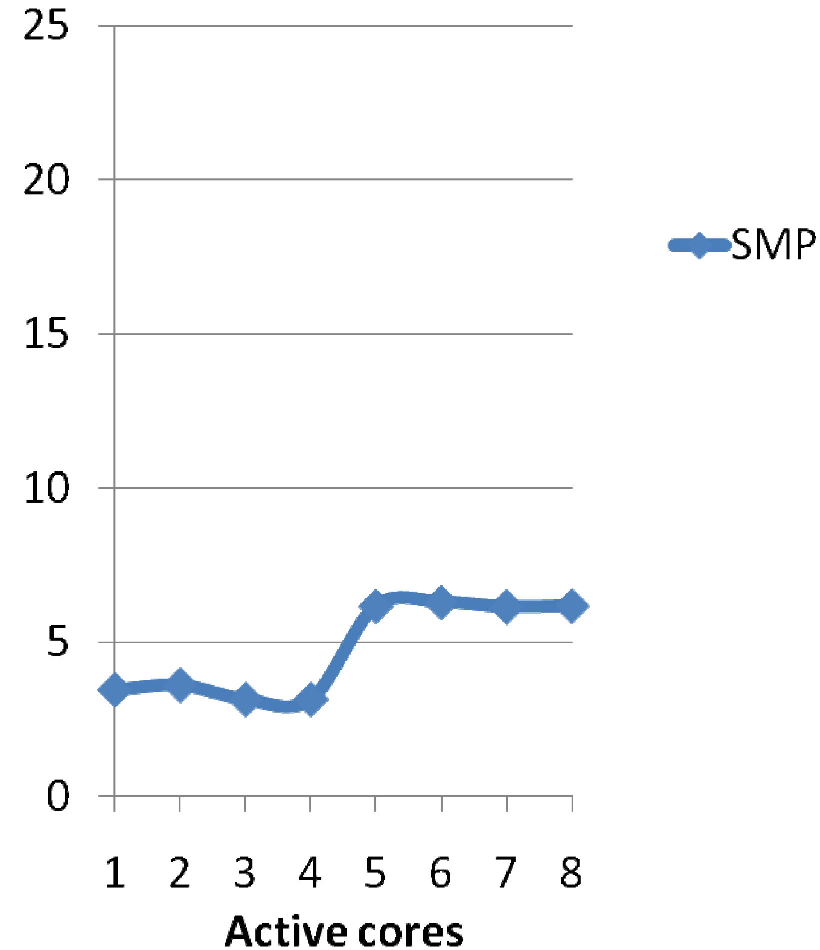


NUMA multicores: how it happened

Next generation: NUMA

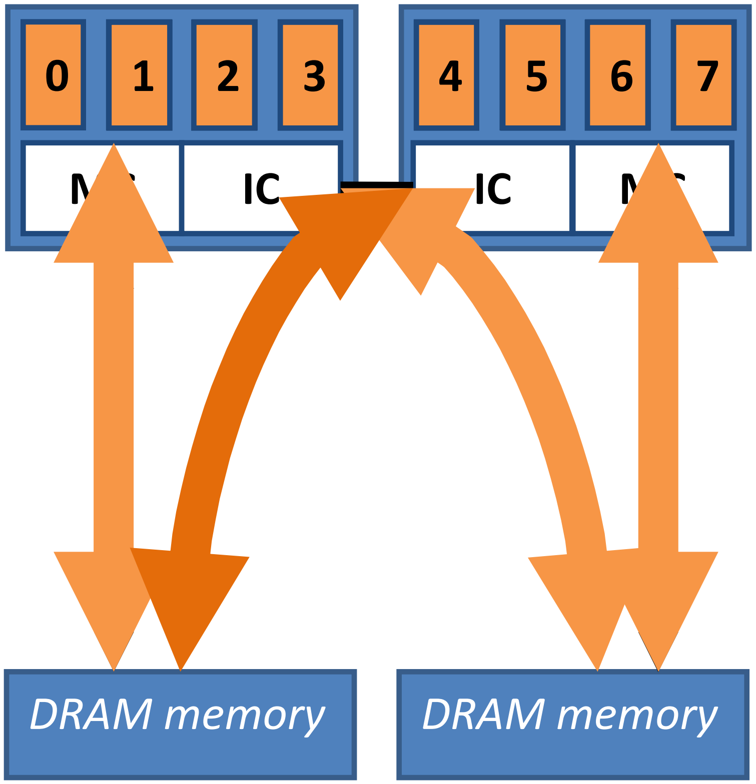


Total bandwidth [GB/s]

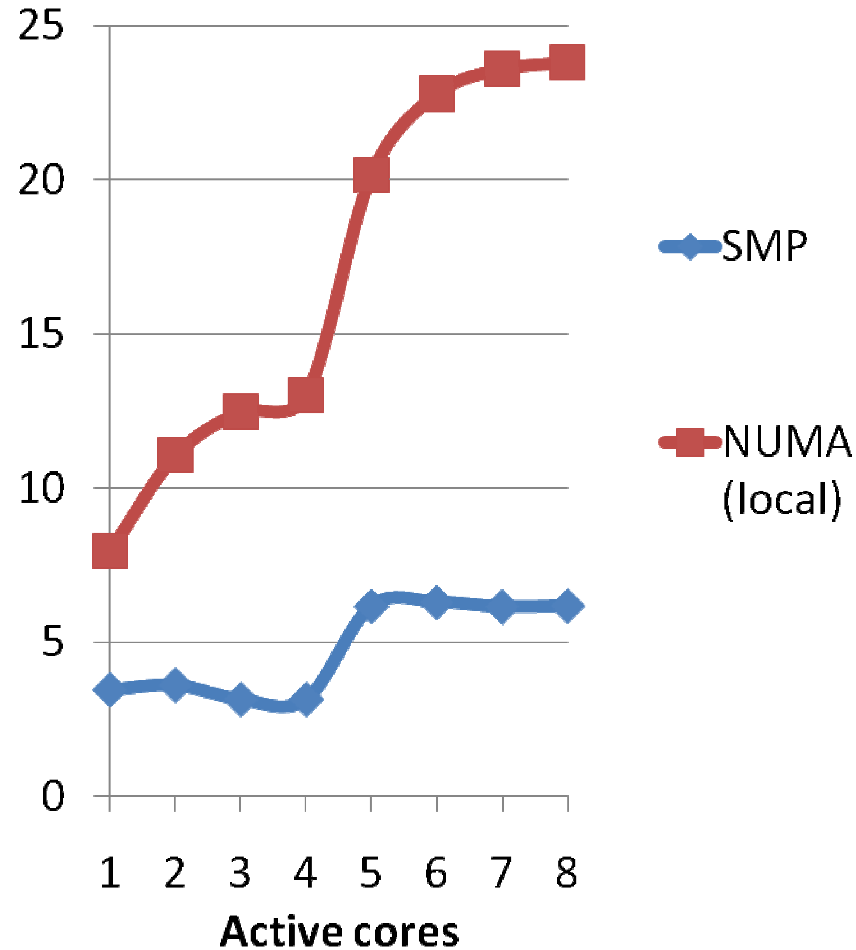


NUMA multicores: how it happened

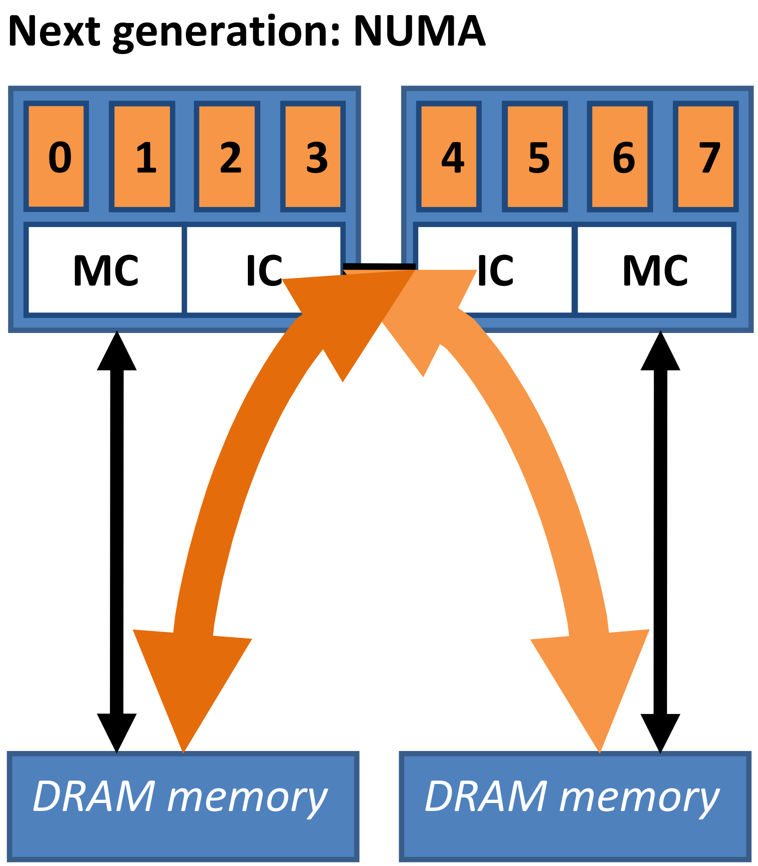
Next generation: NUMA

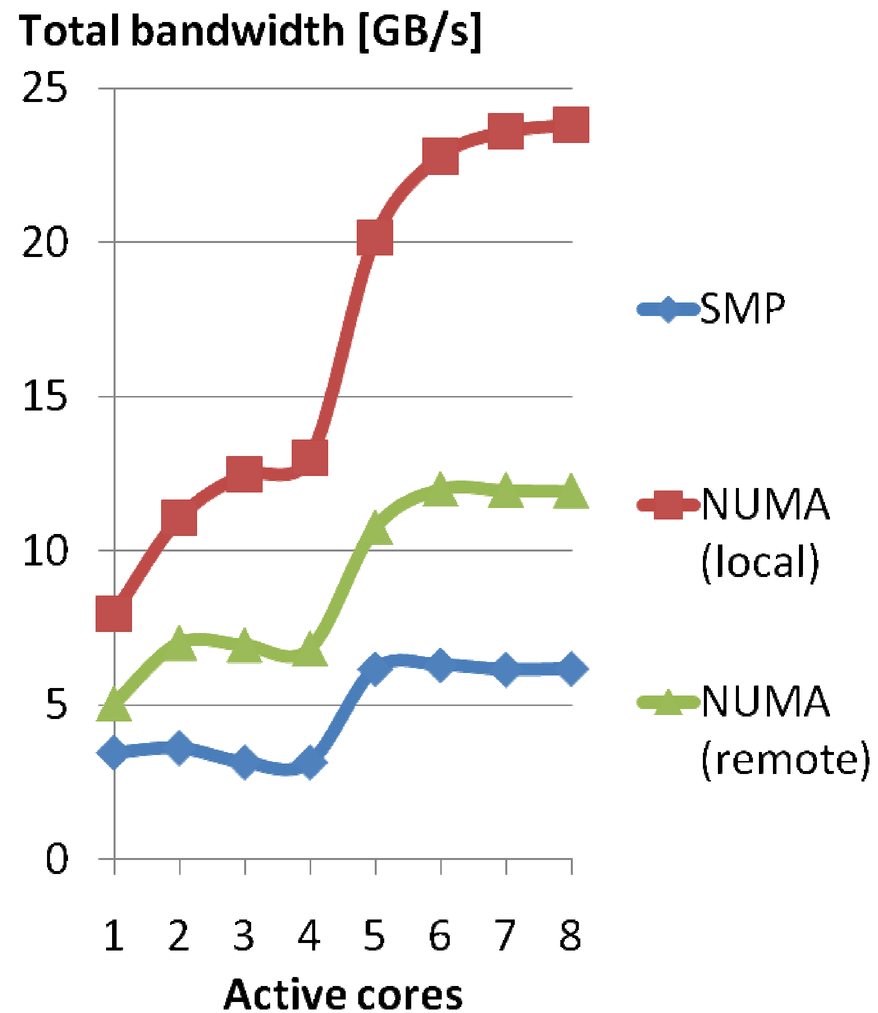


Total bandwidth [GB/s]



NUMA multicores: how it happened



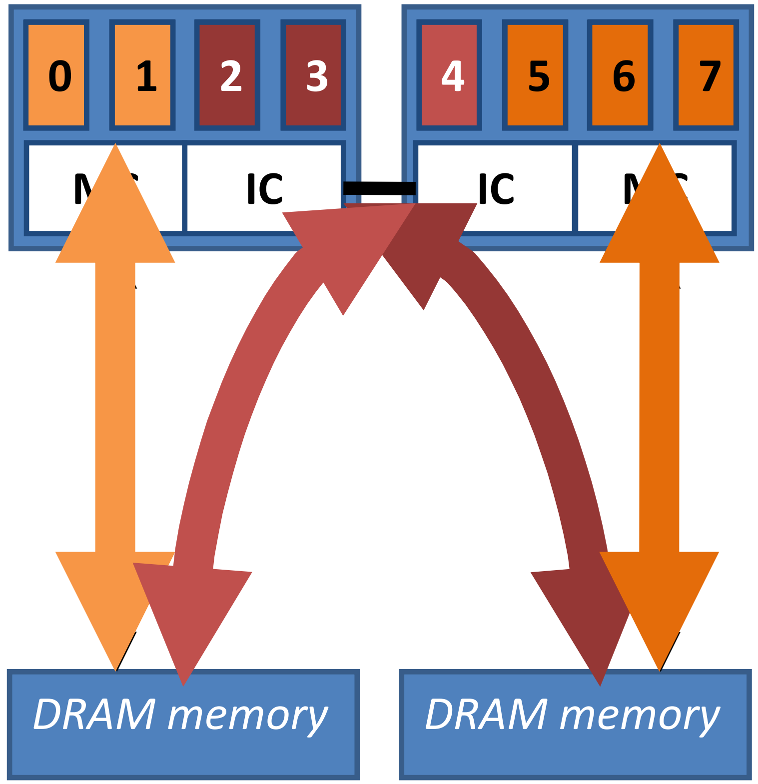


Bandwidth sharing

- Frequent scenario:

  bandwidth shared between cores

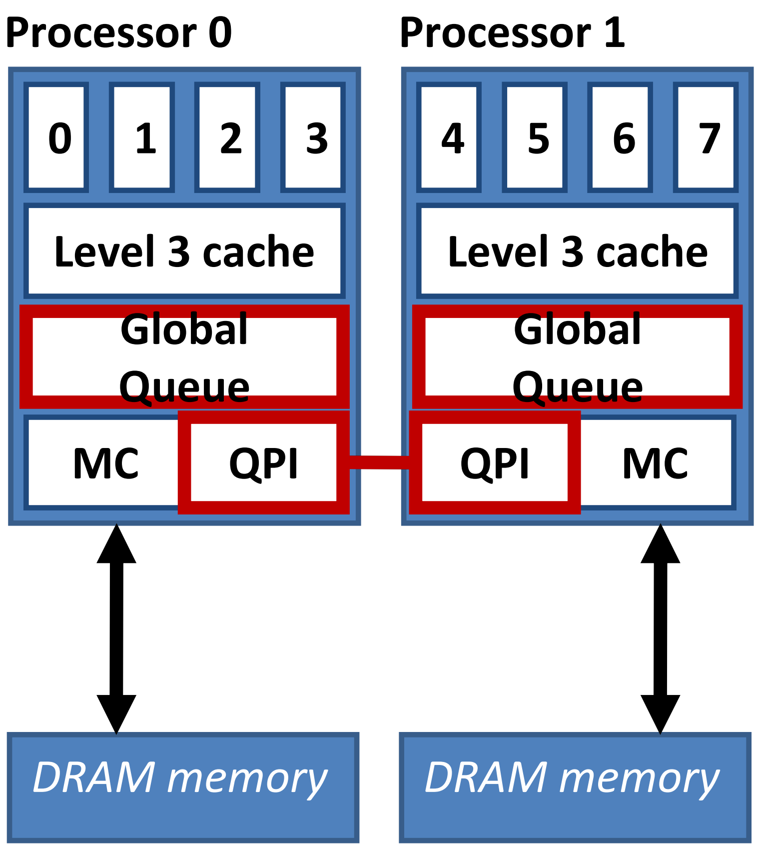
- **Sharing model** for the Intel Nehalem



Outline

- NUMA multicores: how it happened
- Experimental evaluation: Intel Nehalem
- Bandwidth sharing model
- The next generation: Intel Westmere

Evaluation system



Intel Nehalem E5520

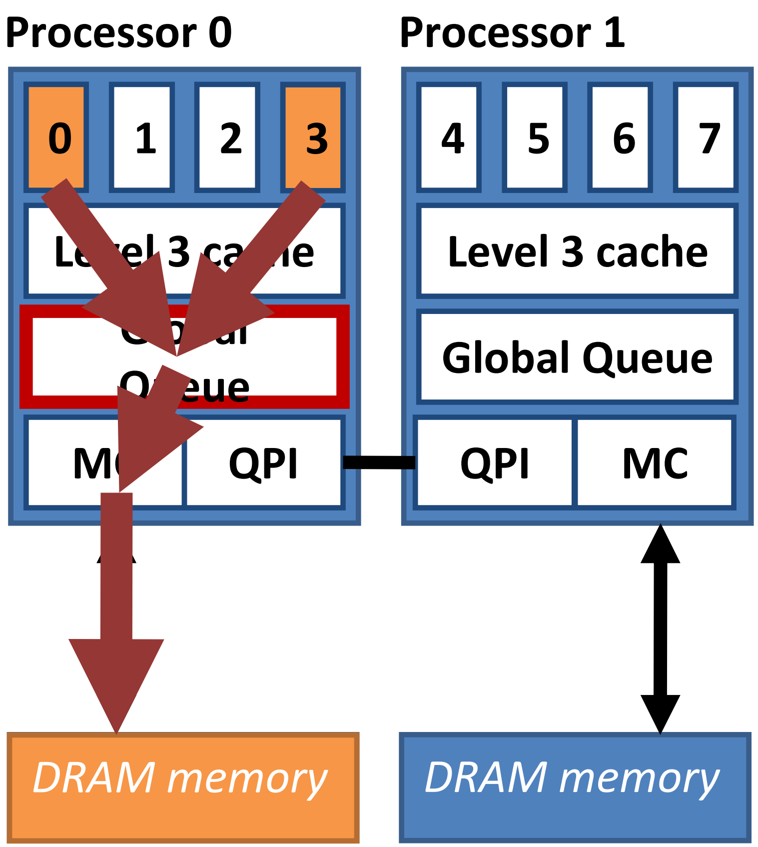
2 x 4 cores

8 MB level 3 cache

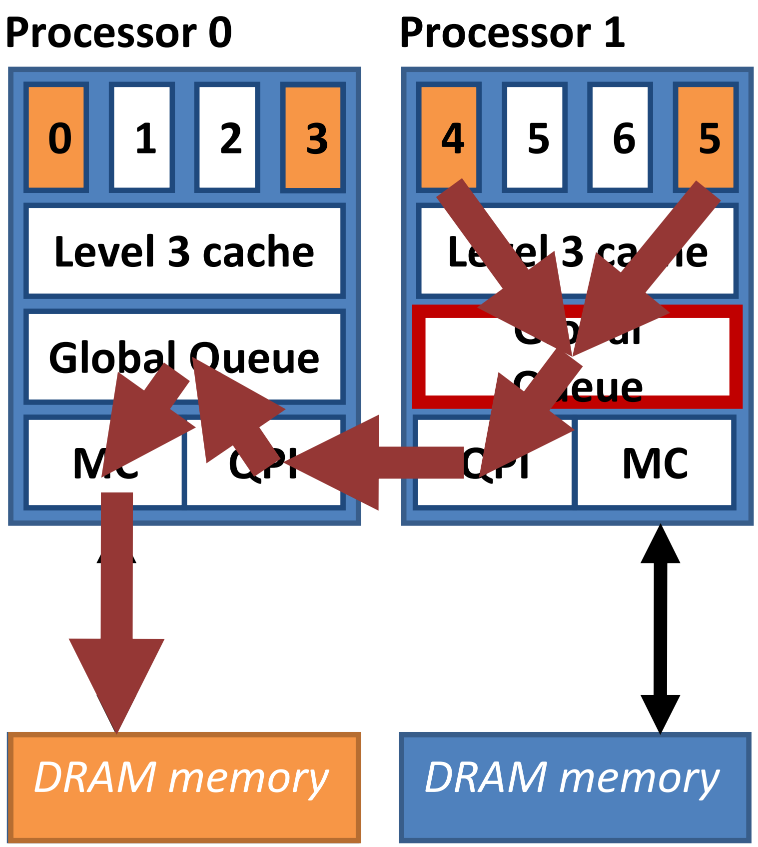
12 GB DDR3 RAM

5.86 GT/s QPI

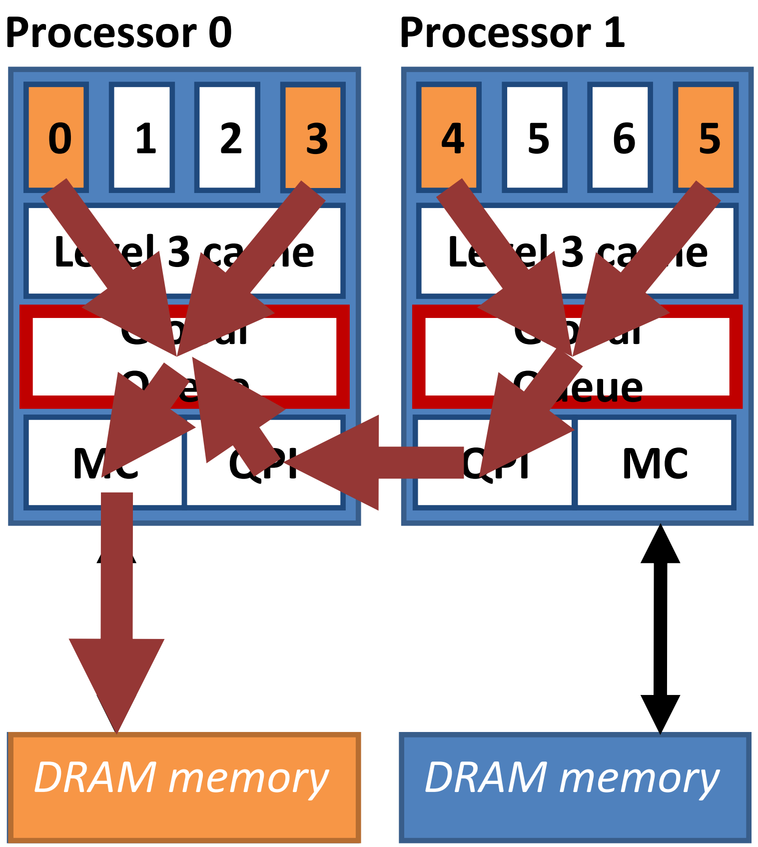
Bandwidth sharing: local accesses



Bandwidth sharing: remote accesses



Bandwidth sharing: combined accesses



Global Queue

- Mechanism to **arbitrate** between different types of memory accesses

- We look at **fairness** of the Global Queue:
  - **local** memory accesses
  - **remote** memory accesses
  - **combined** memory accesses

Benchmark program

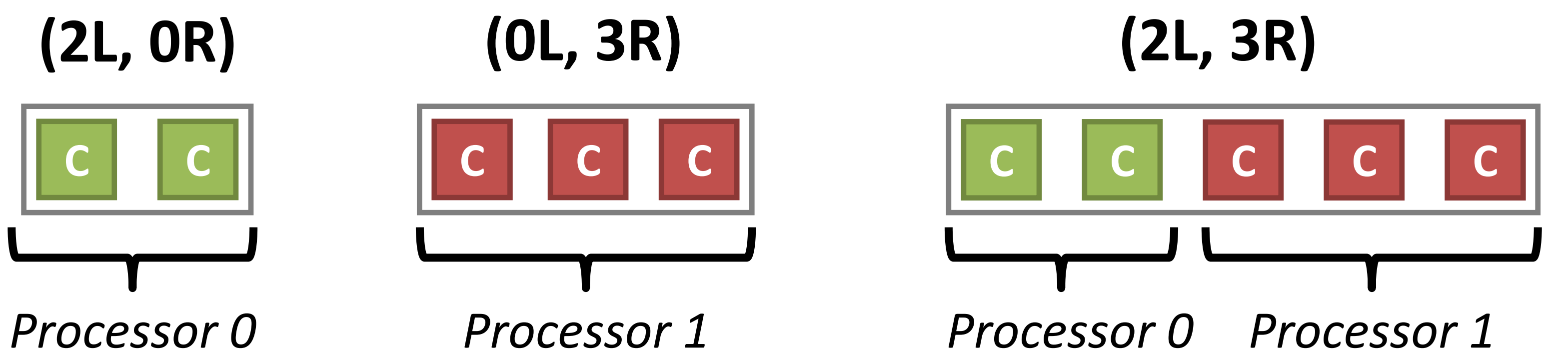
- STREAM **triad**

```
for (i=0; i<SIZE; i++)
{
  a[i]=b[i]+SCALAR*c[i];
}
```

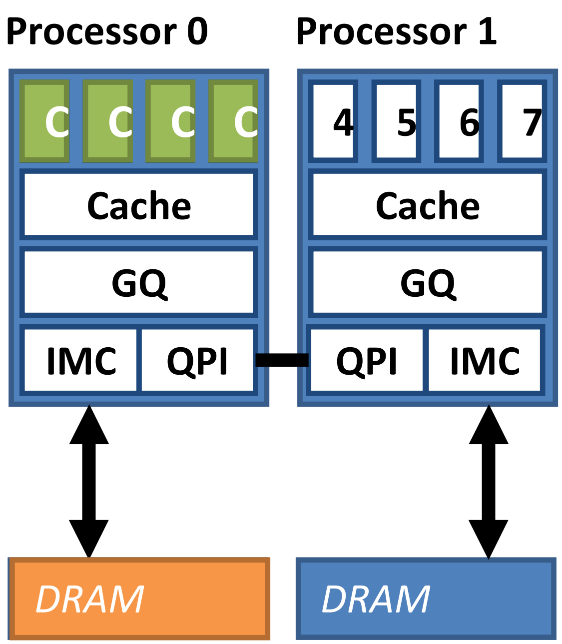
- Multiple co-executing triad **clones**

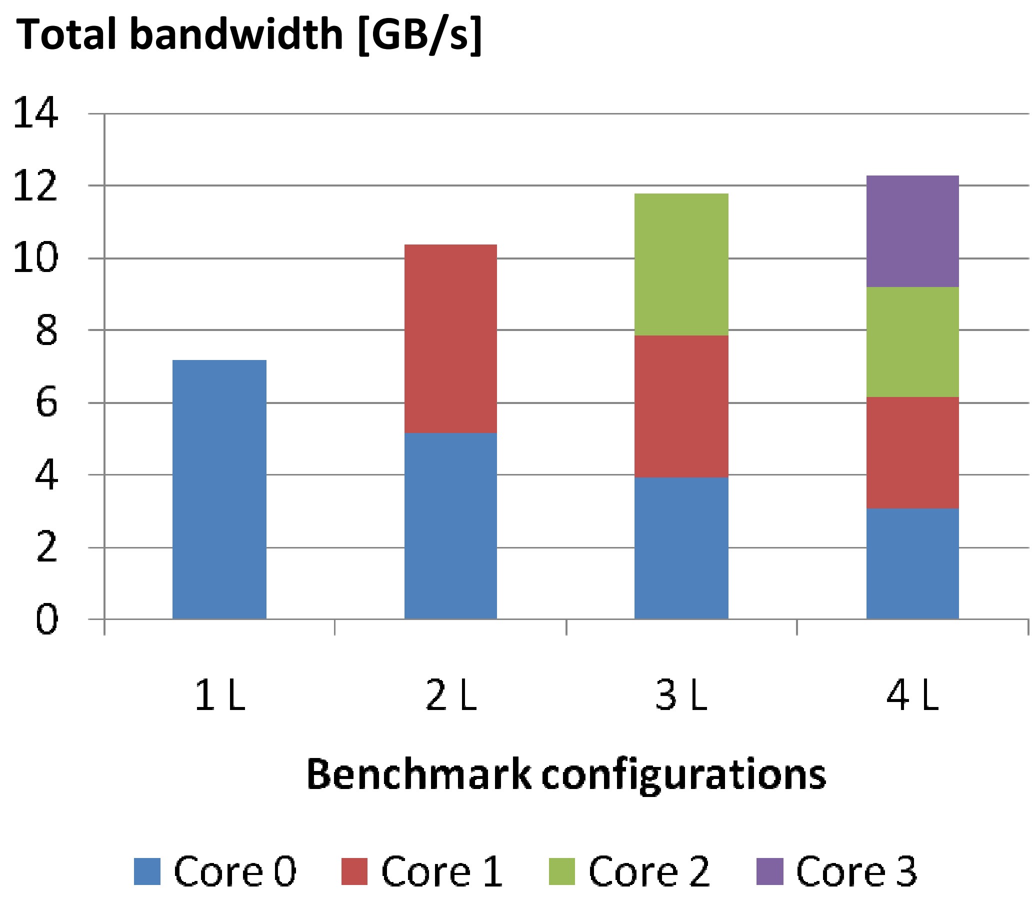
Multi-clone experiments

- All memory allocated on Processor 0
- Local clones: C Remote clones: C
- Example benchmark configurations:

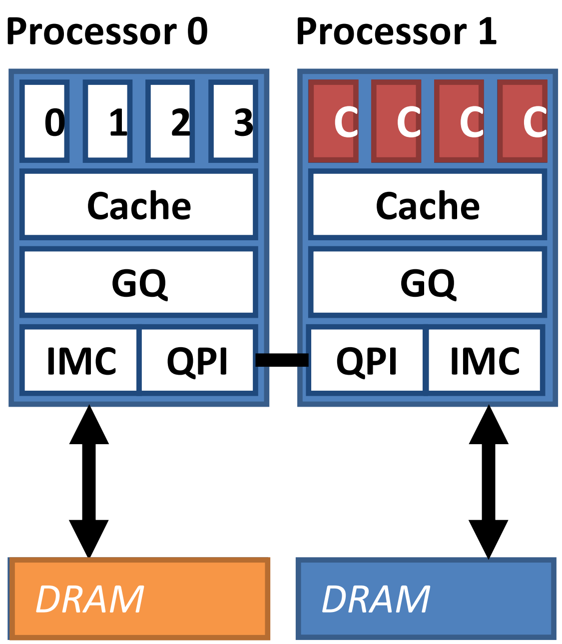


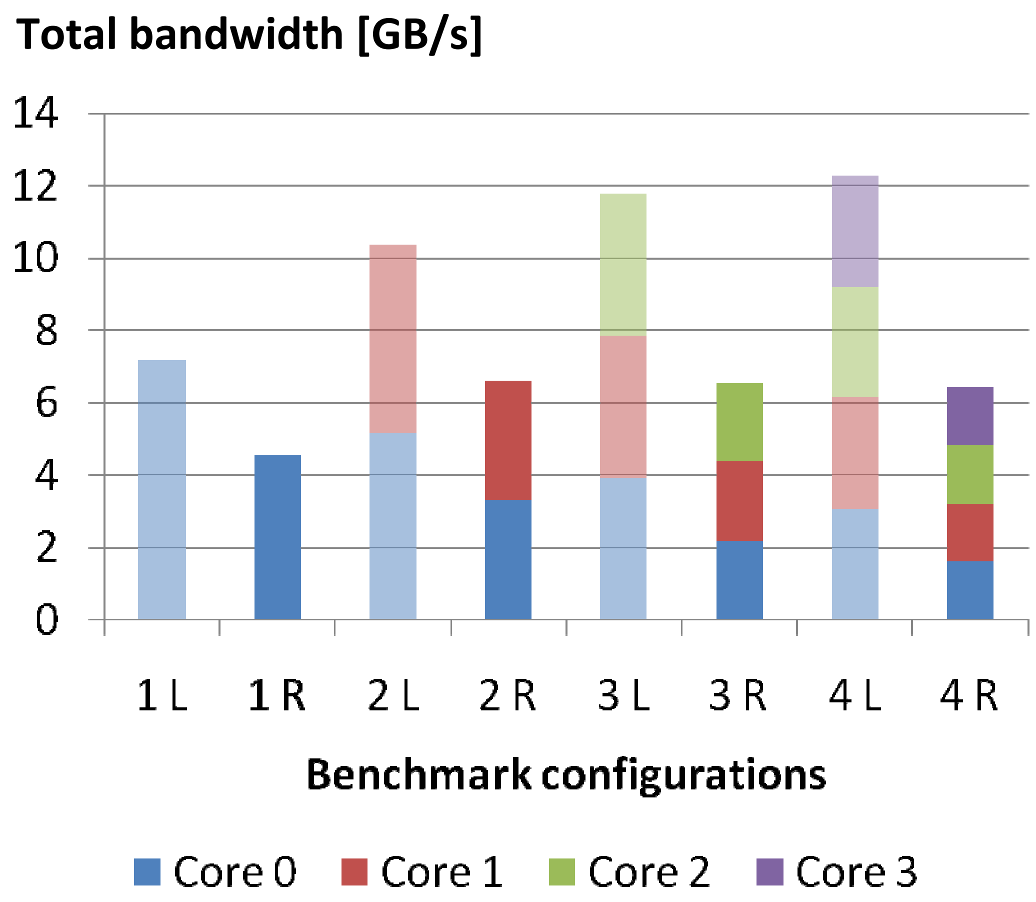
GQ fairness: local accesses





GQ fairness: remote accesses





Global Queue fairness

- Global Queue **fair** when there are **only local/remote** accesses in the system

- What about **combined accesses**?

GQ fairness: combined accesses

Execute clones in **all possible** configurations

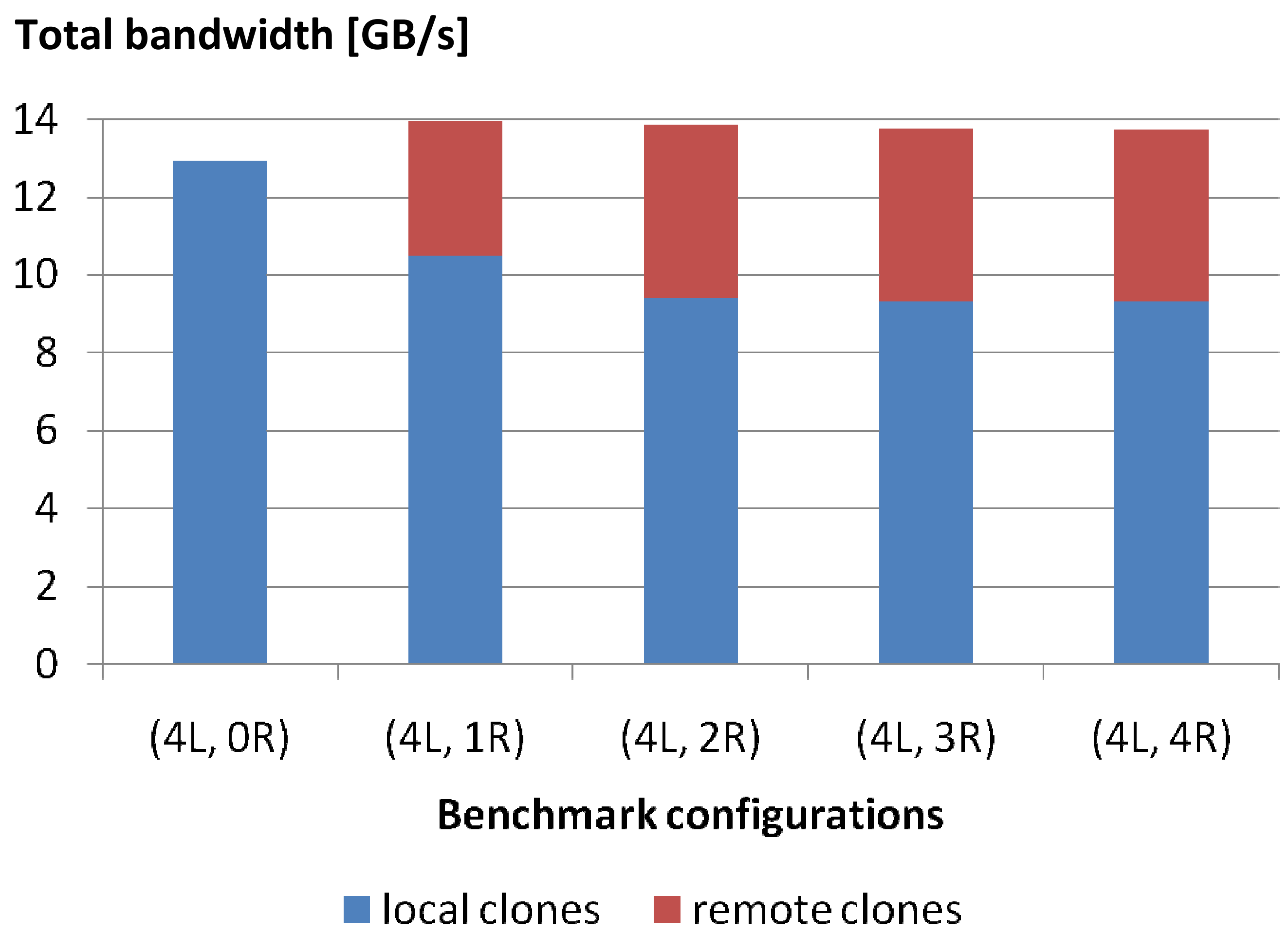
	# local clones				
	0	1	2	3	4
# remote clones					
0					
1					
2					
3			**(2L, 3R)**		
4					

GQ fairness: combined accesses

Execute clones in **all possible** configurations

	# local clones				
	0	1	2	3	4
# remote clones					
0					
1					
2					
3					
4					

GQ fairness: combined accesses

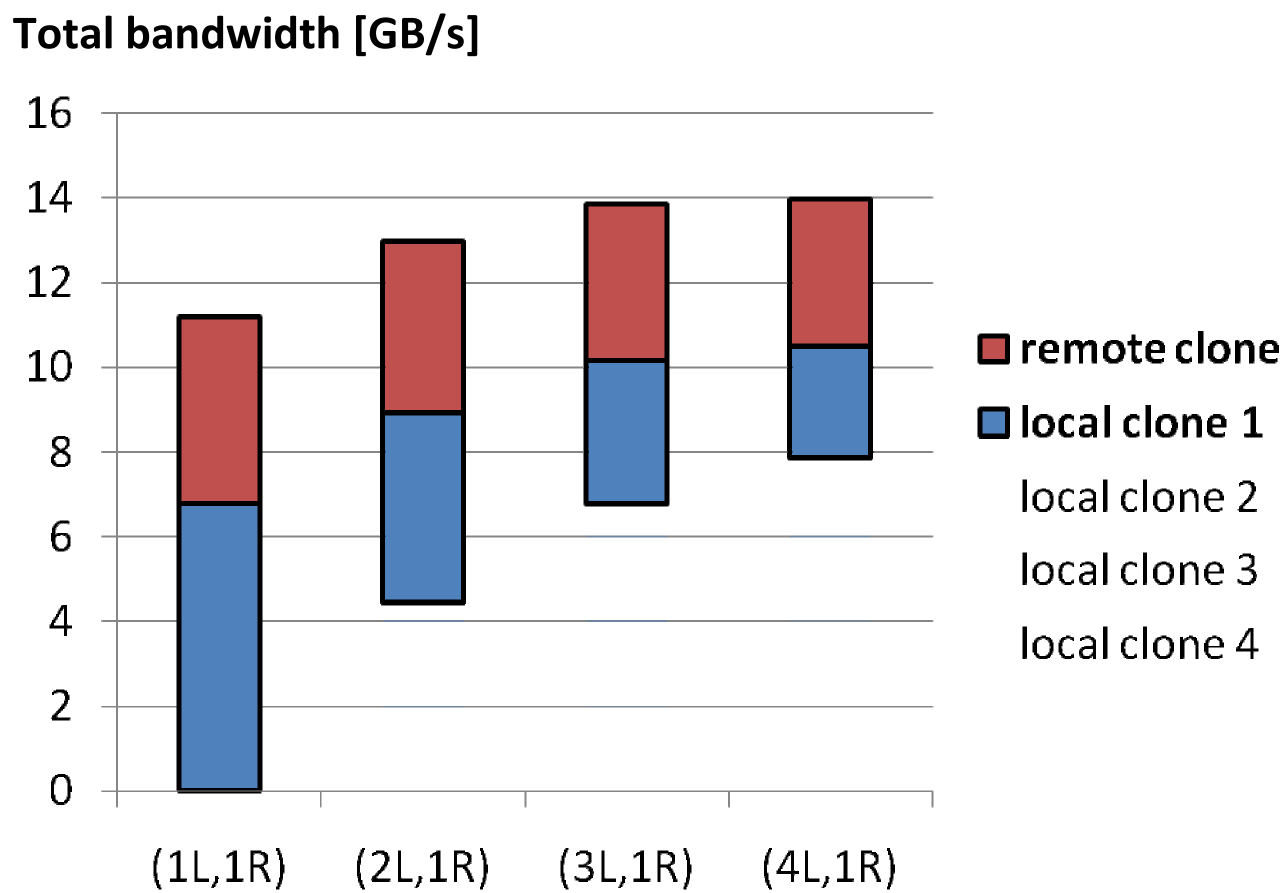


GQ fairness: combined accesses

Execute clones in **all possible** configurations

	# local clones				
	0	1	2	3	4
# remote clones					
0					
1					
2					
3					
4					

Combined accesses



Combined accesses

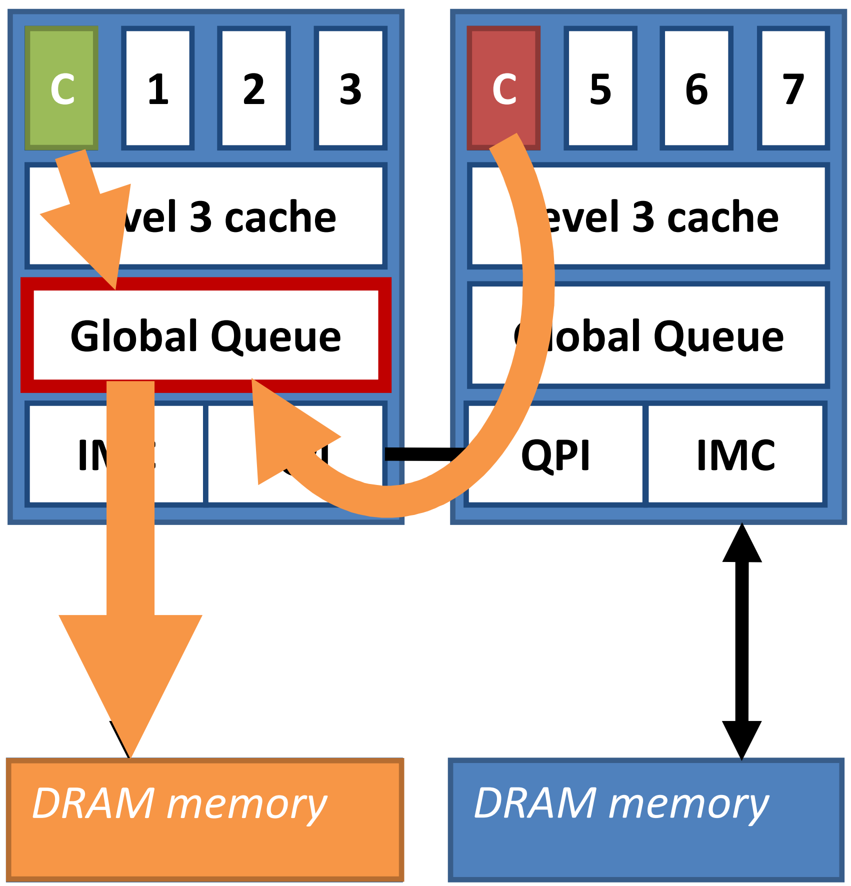
- In configuration **(4L, 1R)** remote clone gets **30% more bandwidth** than a local clone

- **Remote execution can be better** than local

Outline

- NUMA multicores: how it happened
- Experimental evaluation: Intel Nehalem
- Bandwidth sharing model
- The next generation: Intel Westmere

Bandwidth sharing model

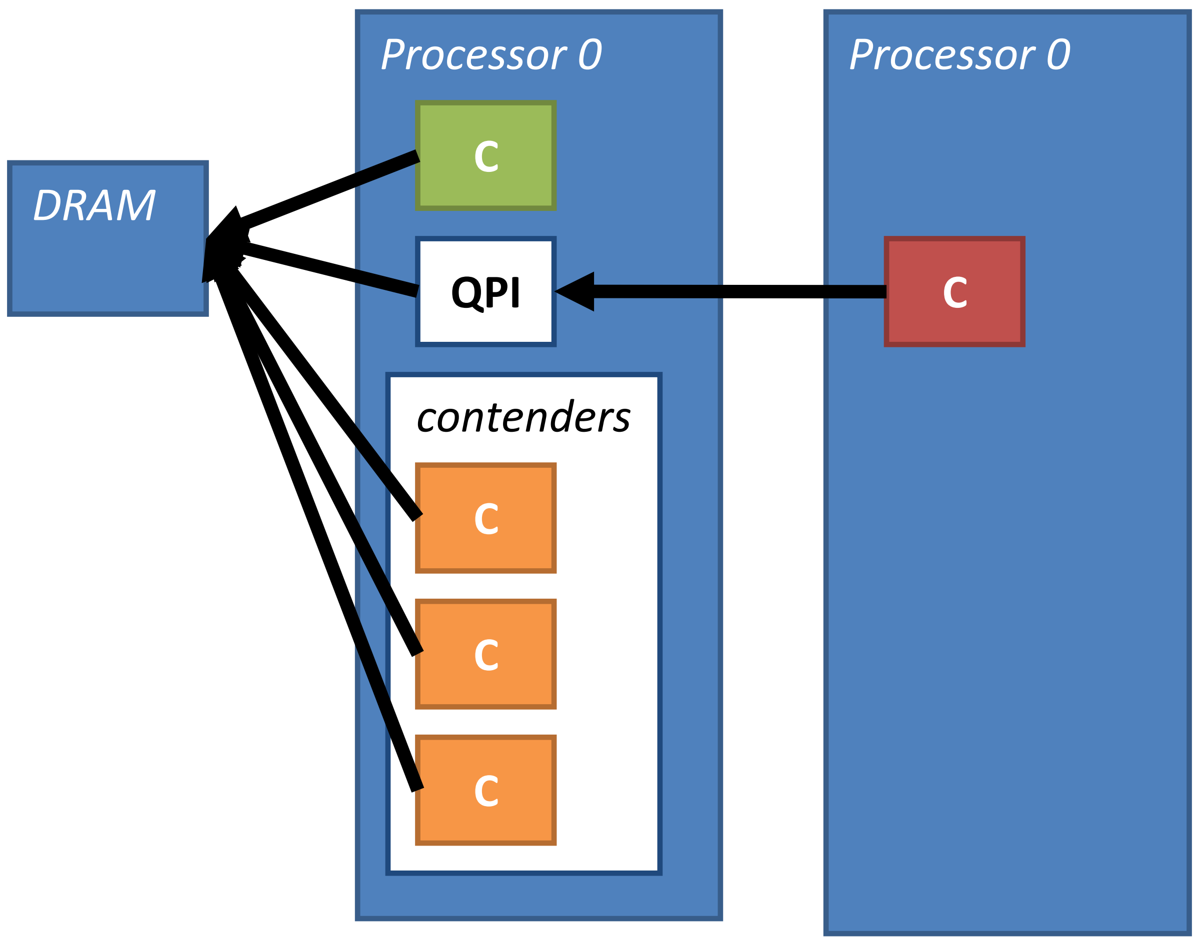


Sharing factor (β)

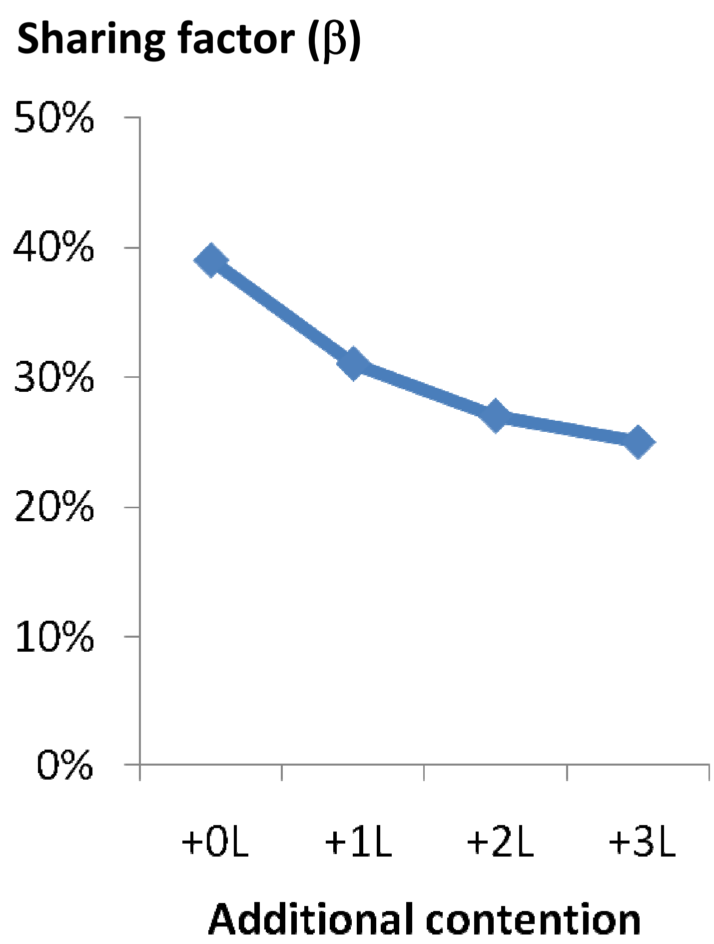
- Characterizes the **fairness** of the Global Queue

- Dependence of sharing factor on **contention**?

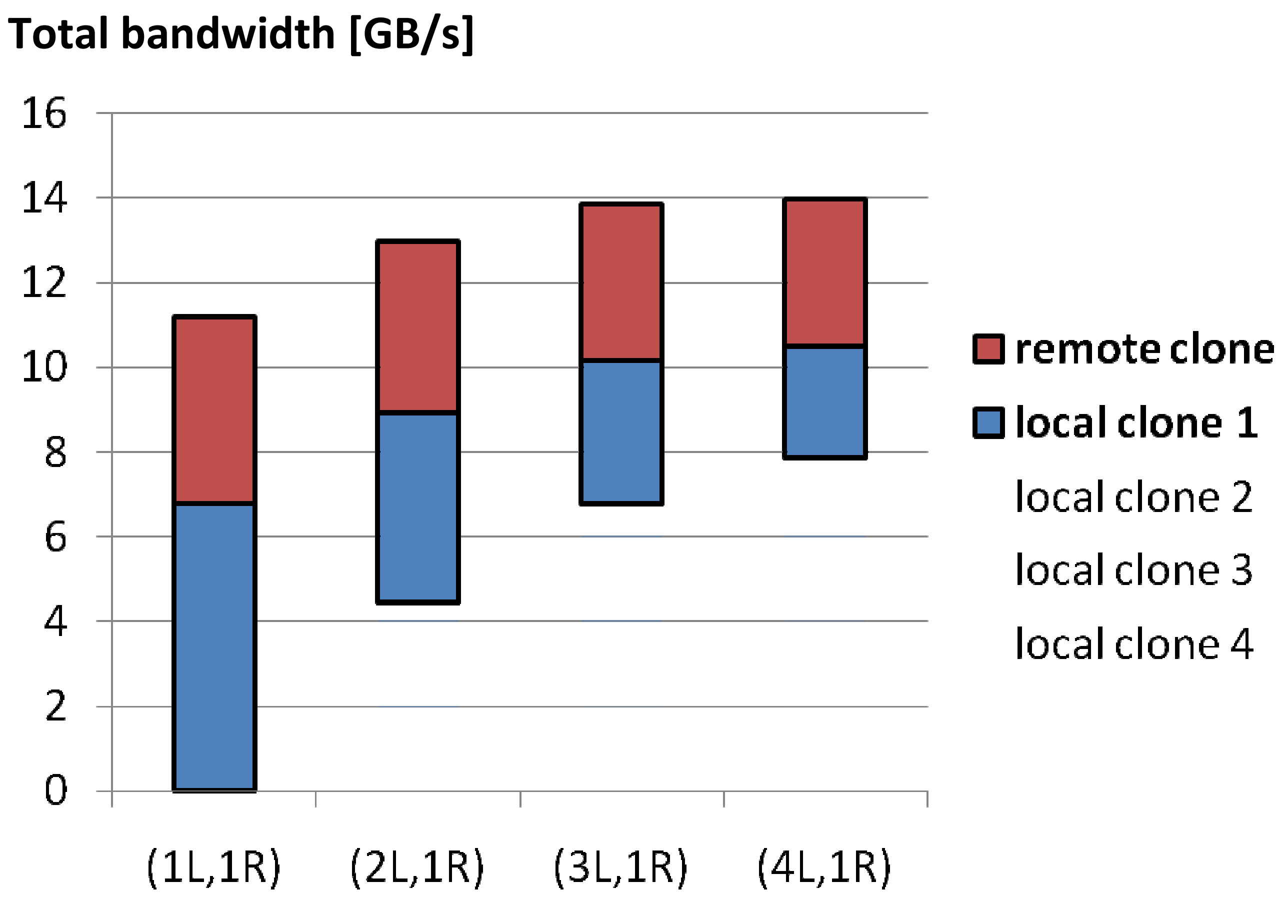
Contention affects sharing factor



Contention affects sharing factor



Combined accesses



Contention affects sharing factor

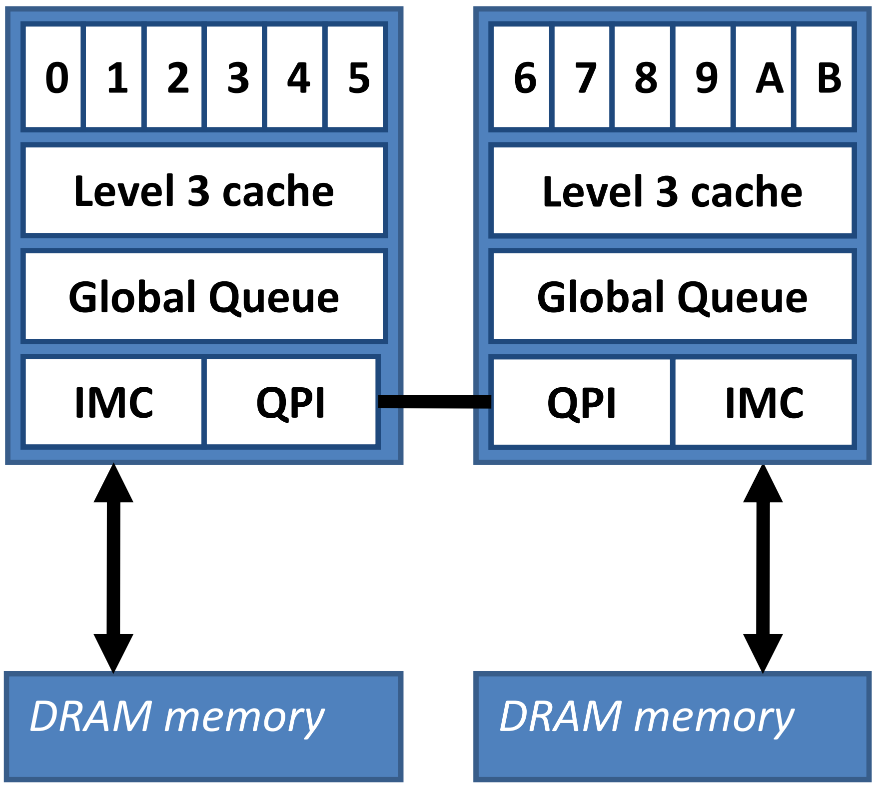
- Sharing factor decreases with contention

- With local contention remote execution becomes more favorable

Outline

- NUMA multicores: how it happened
- Experimental evaluation: Intel Nehalem
- Bandwidth sharing model
- The next generation: Intel Westmere

The next generation



Intel Westmere X5680

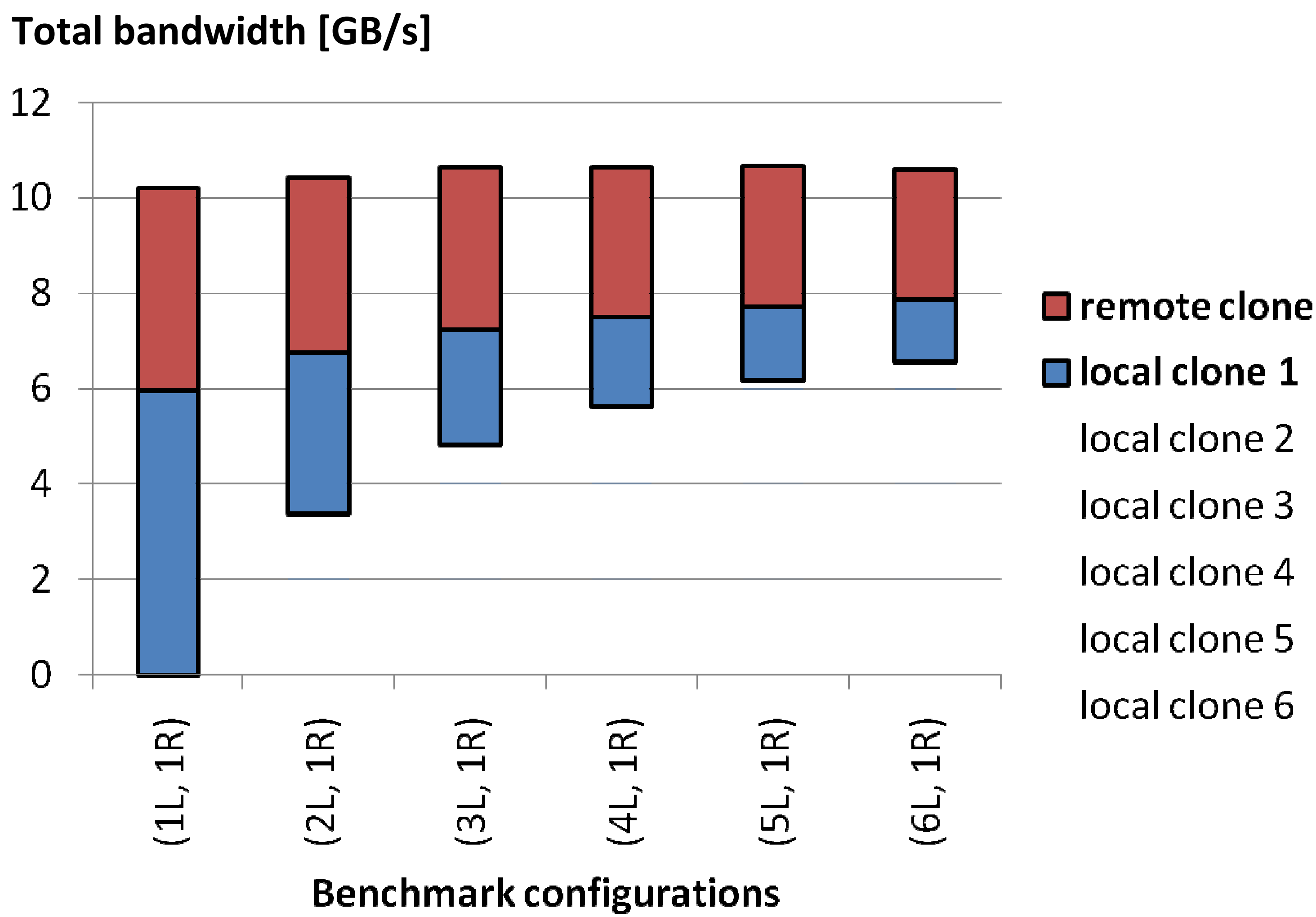
2 x 6 cores

12 MB level 3 cache

144 GB DDR3 RAM

6.4 GT/s QPI

The next generation



Conclusions

- Optimizing for **data locality** can be **suboptimal**

- Applications:
  - OS scheduling (see ISMM'11 paper)
  - data placement and computation scheduling

Thank you! Questions?