



On-chip memories, the OS perspective

Carlos Villavieja¹, Isaac Gelado¹, Alex Ramírez^{1,2}, Nacho Navarro¹
{cvillavi,igelado,aramirez,nacho}@ac.upc.edu

¹UPC / ²BSC

HiPEAC Industrial Workshop
Barcelona, 4th June 2008

WIOSCA 2008 (ISCA WORKSHOP)



- Review of several CMP

2. Motivation for on-chip Memories

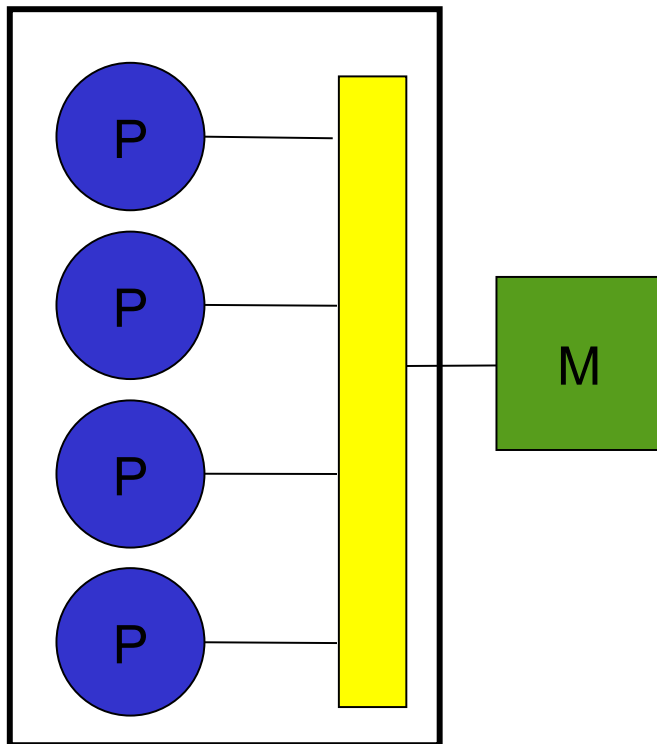
- Example usage cases

3. Operating System / Runtime

- Implications – Service requirements
- Memory management

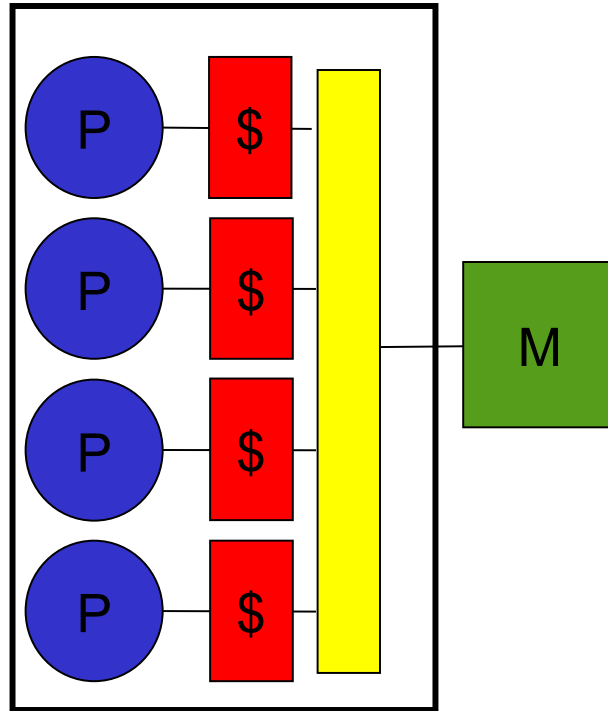
4. Conclusions

Off-chip memory bandwidth is a limiting factor

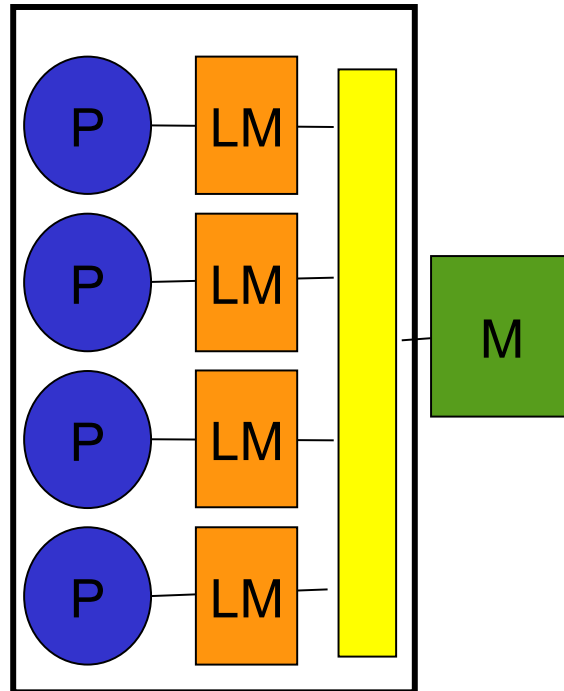


- All processors in a CMP share the memory bandwidth
- Memory bandwidth does not scale with the number of processors
 - Limited number of pins
- Uniform memory access time
 - But ... very high latency
- **MEMORY WALL PROBLEM**

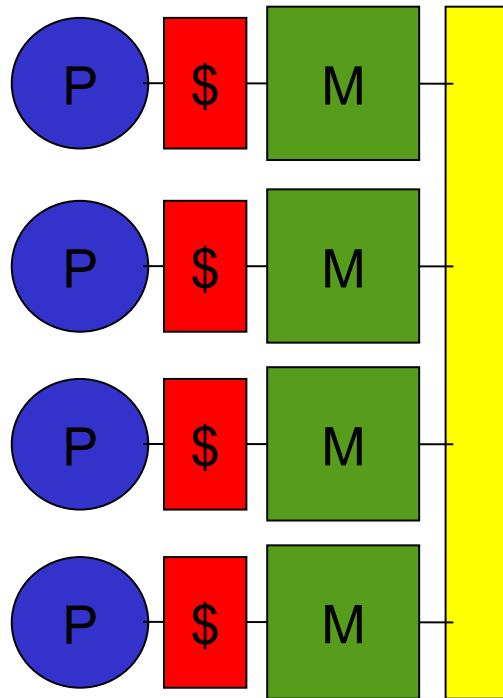
Cache coherency



- Each processor has a private cache memory
- Memory bandwidth scales with the number of processors
 - As long as the working set fits in cache
- Limited scalability of cache coherency protocol
- Unpredictable memory latency
- **SCALABILITY AND UNPREDICTABILITY LATENCY**

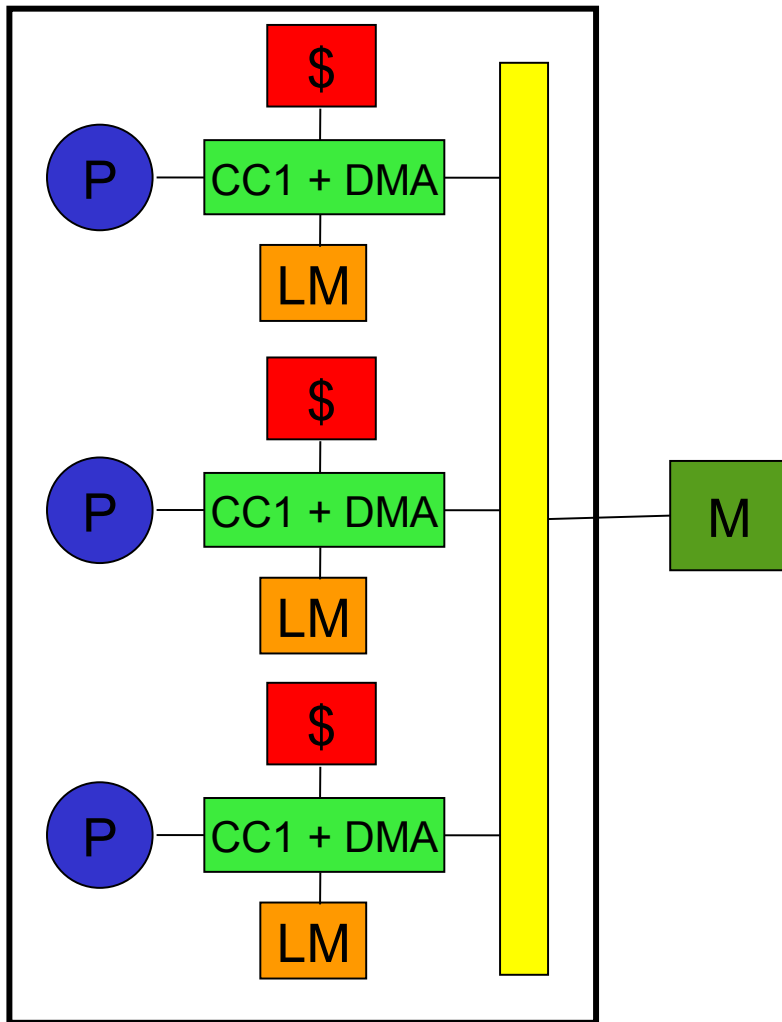


- Each processor has a local on-chip memory
- Memory bandwidth scales with the number of processors
 - Working set must fit in the local memory
- Predictable memory latency
 - Programmer manages LM
 - No possible miss
- **Hard to program for distributed memory**
 - Application partitioning
 - Data Distributed in different address spaces



- CC-NUMA Architecture has the best of both worlds
- Single address space + Cache coherency
 - Ease of programming
- Distributed memory
 - Scalability
- **HIGH SHARING** → **high contention**
 - **Ex: Lock starvation**

Our proposal: On-chip CC-NUMA



- Bring the DSM CC-NUMA architecture to the CMP
- Non-cacheable on-chip local memories
 - Local memory updates do not generate coherency traffic
- Single address space
 - Any processor can access any memory location (ld/st, DMA)
- On-chip memories provide lower latencies (high sharing has lower contention)

Commercial processors with on-chip memories

- Common features:
 - Multiple address spaces
 - Once data is partitioned, it is hard to access remote data
 - Proposed Solution : on-chip CC-NUMA

Examples:

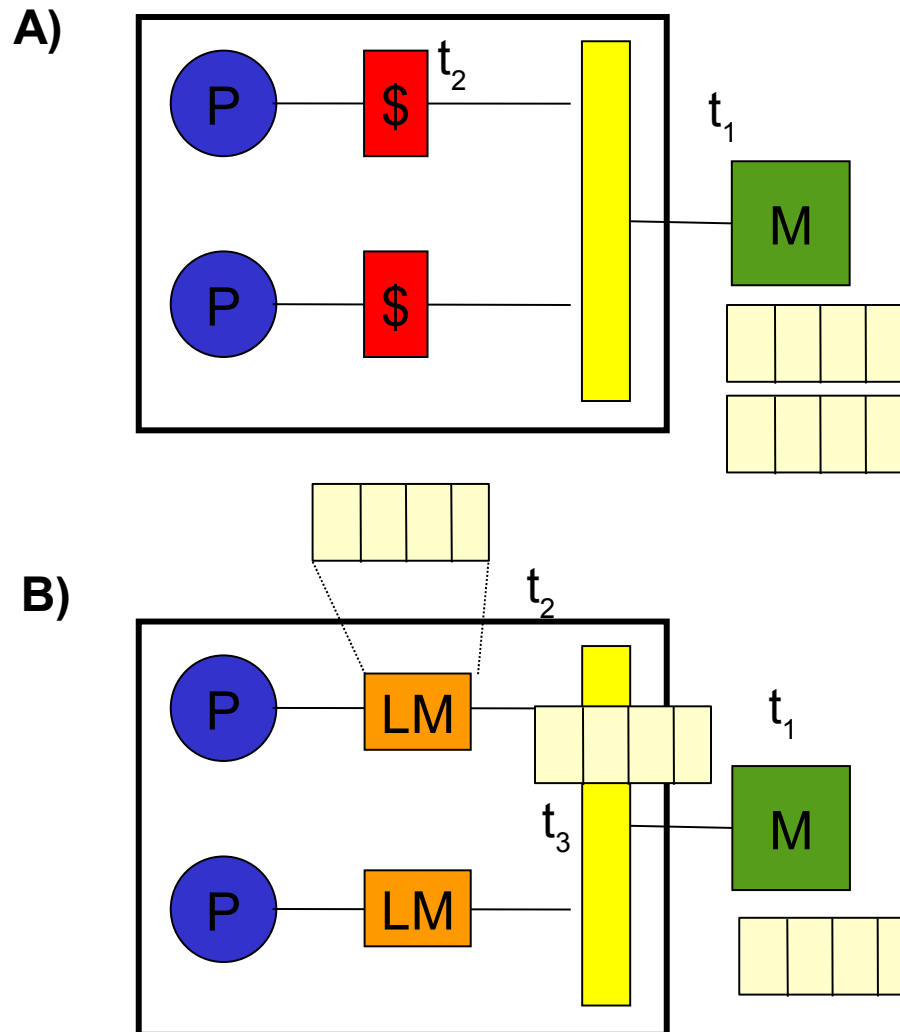
- Cell
 - Distributed Memory
- BlueGene/P
 - On-Chip memory (L3 - hybrid cache)
- Texas
 - DSP On-chip private memory
- Intel Polaris (80-core)

Advantages of on-chip local memories



- Predictable latency
 - Streaming computing
- Not involved in the cache coherency protocol
 - LM Scale well
 - One single copy of data in LM
- Requires application partitioning
- Use cases (cache vs LM)
 - Double buffering
 - Producer-consumer
 - Producer-multiple consumers

Double buffering



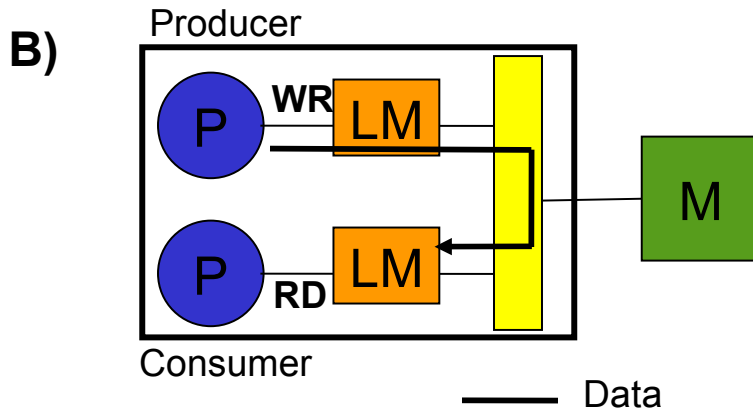
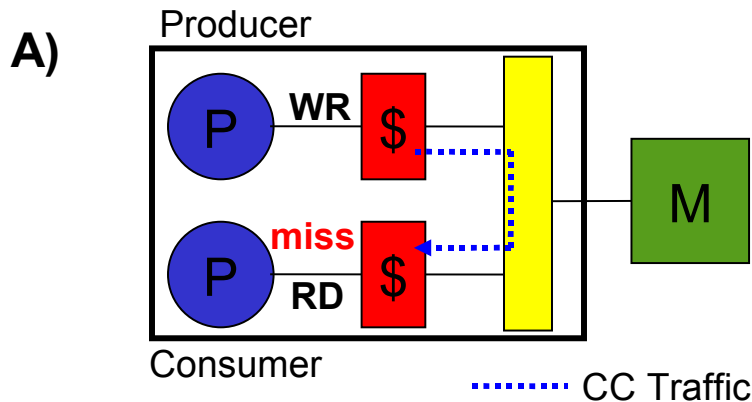
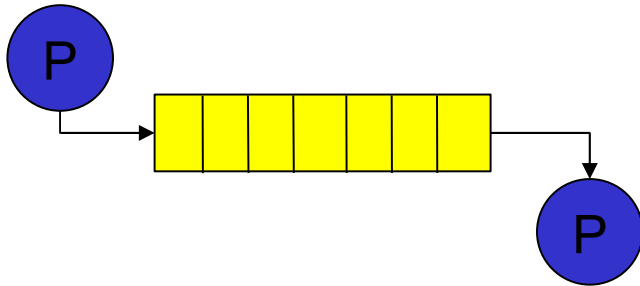
A) Using caches:

- Processor reads data from cache
- Cache misses and requests data to memory
- Processor stalls until data arrives

• Using **DMA** and local memories:

- Processor programs DMA to transfer next data block (t_1)
- Processor computes on the current data block (t_2)
- DMA transfers next data block in parallel with computation (t_3)

Producer-consumer



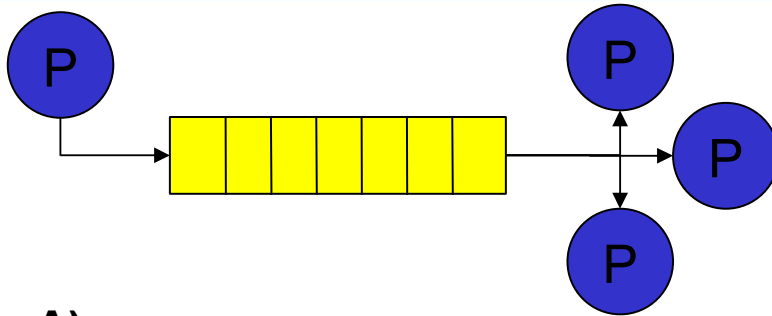
A) Using caches:

- Produces writes its local cache
- Consumer misses on its local cache
 - Coherency protocol transfers cache line from producer to consumer
- LOAD latency is on the critical path

B) Using local memories:

- Consumer allocates buffer on its local memory
- Producer writes directly on the remote on-chip memory
 - STORE latency not on the critical path
- Consumer reads data from its local memory
 - LOAD latency is low, and guaranteed

Producer-multiple consumers



A) Using caches:

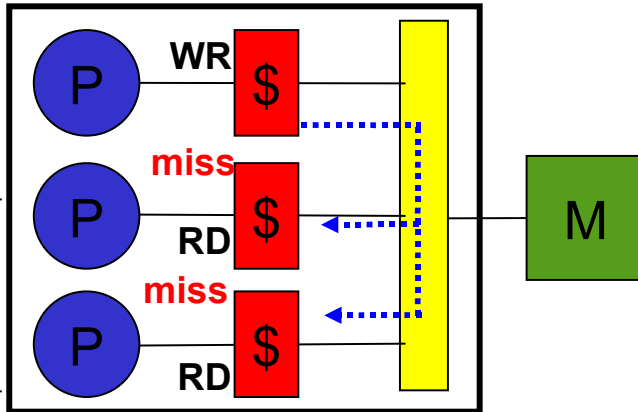
- Producer writes data on buffer
- Buffer is distributed to the multiple consumer caches
- When producer overwrites the buffers, has to invalidate ALL consumer caches

B) Using local memories:

- Producer writes data on buffer
- Consumers COPY data from buffer to local memory
- Producer is free to overwrite buffer without telling the consumers

A)

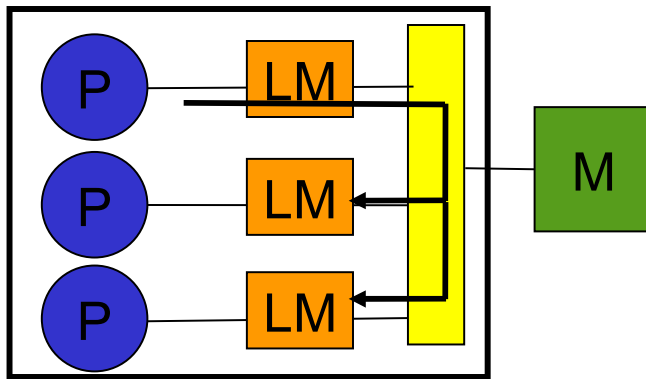
Producer



Consumer

B)

Producer



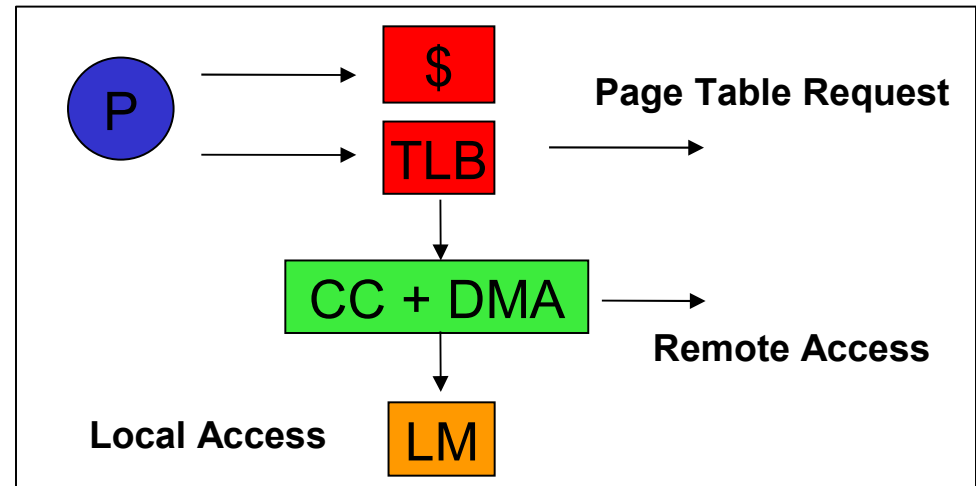
Consumer

..... CC Traffic

— Data

Implementation

- Single address space -> single load/store instruction
- A day in the life of a load instruction
 - TLB: Logical to physical address translation
 - Physical address determines which memory holds the data
 - Local memory access
 - Remote on-chip memory access
 - Off-chip memory access: cacheable memory location
 - Cache hit / miss





- Page allocation
 - Dynamic memory allocation
 - DSM allocates memory on the local DRAM
 - + On-chip DSM does not have enough local memory to always do that ...
 - + Application specifies where to allocate memory
- Page migration
 - Moving a page from remote to local memory (or vice versa)
 - DSM automatically migrates pages based on usage pattern
 - + Cost of an on-chip remote access not so high
 - Application specifies when to migrate pages
- Thread migration
 - Moving a thread and all its local memory to a different processor



- Runtime library calls

- `void* malloc(thread_id id, latency_t latency, size_t size, bool mandatory)`

- `thread_id`: allocate memory local to which thread?
 - `latency`: how local to that thread do we want the memory?
 - Priority class (ex: LM, ANY_LM, OFF_M)
 - `mandatory`: fail with an error if no such memory is available

- `void* realloc(void *ptr, size_t size, thread_id id, latency_t lat, bool mandatory)`

- In addition to resizing a block, can also change its physical location

- Runtime page migration

- The OS may decide to migrate a page to a different physical location based on usage pattern
 - Already done in DSM ... with varying degree of success
 - The OS may "swap out" local memory pages to off-chip memory
 - Same as swapping memory pages to disk



- Allocating a page in local memory
 - Update Page Table (off-chip memory – 2x300 cycles)
 - Update TLB 'on first access' (2x300 cycles)
- First access to a remote local memory
 - TLB miss, update local TLB (2x300 cycles)
- Page migration
 - Invalidate page at all TLBs, update page table
- Thread Migration:
 - Implies migrating all its local memory:
 - Latency requirements Checklist
 - Page allocation has a latency requirements that must be kept when migrating

Approximated Costs

Operation	Cycles
Off-chip Mem	300
LM	20
RLM	40

Conclusions



- This presentation explores
 - Situations where on-chip DSM performs better
 - Required runtime services for on-chip DSM
 - Memory management
 - Thread migration
- On-chip DSM offers an interesting candidate for future CMPs
 - Scalability
 - Predictability
 - Real-time applications
 - Ease of programming
 - Plain shared memory applications will work (only slower)
 - Later on, optimize data distribution and allocation



Thanks & Questions

**On-chip
memories, the
OS perspective**

Carlos Villavieja¹, Isaac Gelado¹, Alex Ramírez^{1,2}, Nacho Navarro¹
{cvillavi,igelado,aramirez,nacho}@ac.upc.edu

¹UPC / ²BSC



On-chip memories, the OS perspective

Carlos Villavieja¹, Isaac Gelado¹, Alex Ramírez^{1,2}, Nacho Navarro¹
{cvillavi, igelado, aramirez, nacho}@ac.upc.edu

¹UPC / ²BSC

HiPEAC Industrial Workshop

Barcelona, 4th June 2008

WIOSCA 2008 (ISCA WORKSHOP)