Early Experiences with Separate Caches for Private and Shared Data

Juan M. Cebrián, Alberto Ros,
Ricardo Fernández-Pascual and Manuel E. Acacio

{jcebrian, aros, rfernandez,
meacacio}@ditec.um.es

Depart. of Computer Engineering
University of Murcia

Sep 3, 2015

Outline

Outline

Current Trends
Motivation

- Shared-memory architectures are predominant in multi-core microprocessors from all market segments
  - Correctness is ensured by means of coherence protocols and consistency models
  - But performance and scalability are limited by the amount and size of the messages used to maintain coherence

Current Trends
Motivation

- Shared-memory architectures are predominant in multi-core microprocessors from all market segments
  - Correctness is ensured by means of coherence protocols and consistency models
  - But performance and scalability are limited by the amount and size of the messages used to maintain coherence
- Blindly keeping coherence for all memory accesses translates into an unnecessary overhead for data that will remain coherent after the access
  - Read-only shared data
  - Private data

INTRODUCTION
IDEA

- Why treat all accesses the same regardless of the nature of the data being accessed?

INTRODUCTION
IDEA

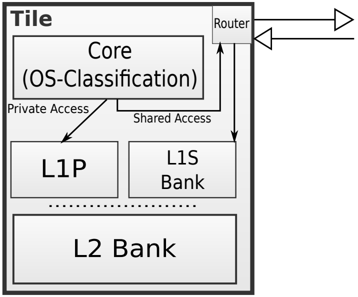
- Why treat all accesses the same regardless of the nature of the data being accessed?
- Idea use of dedicated caches for **private** (+shared read-only) and **shared data**
    - Eliminates the need for a coherence directory, improving scalability of the multi-core architecture
    - Reduces the pressure on the L1P and overall combined L1D misses
    - Reduces the amount of duplicated data in L1 caches

INTRODUCTION
IDEA

- Why treat all accesses the same regardless of the nature of the data being accessed?
- Idea use of dedicated caches for **private** (+shared read-only) and **shared data**
    - Eliminates the need for a coherence directory, improving scalability of the multi-core architecture
    - Reduces the pressure on the L1P and overall combined L1D misses
    - Reduces the amount of duplicated data in L1 caches
- However
    - Increases latency when accessing the shared data (uses the network and may require high bandwidth)
    - Requires a classification mechanism to detect the nature of the data

Outline

PROPOSAL
DEDICATED CACHE DESIGN



- Access the classification mechanism
- Check the local cache for private data or the router for shared data
- Update the classification mechanism based on the nature of the data and move data between the private and shared caches

DEDICATED CACHE DESIGN

- Private $ (L1P) independent for each core
- Shared $ (L1S) logically shared but physically distributed
- Coherence messages only required when nature changes

Proposal
Classification Mechanism

Classification Mechanism

Our proposal requires a classification mechanism that detects private and shared data

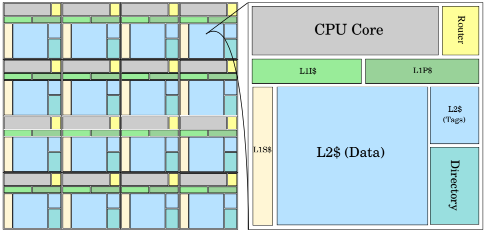
- Many mechanisms in the literature that work at different levels: compiler, OS, coherence protocol, etc

We chose a OS-level (at a page granularity) mechanism because of its simplicity

- When a core accesses a page it is marked private in the page table
- A second access to a page (marked as private) by a different core will force an update on the page table state to shared

Outline

Evaluation Environment



Evaluation Environment

- The simulated architecture corresponds to a single chip multiprocessor (*tiled*-CMP) featuring 16 cores
- Real system data accesses captured by PIN are used as input for the GEMS 2.1 simulation environment
- The interconnection network uses Garnet (for tiled and custom layouts) and a “simple” idealized network model without contention for our idealized evaluation
- We compare against a traditional MESI protocol
- We evaluate our proposal using the applications from the SPLASH-2 benchmark suite with the recommended input sizes

Evaluation Environment

Simulation Parameters

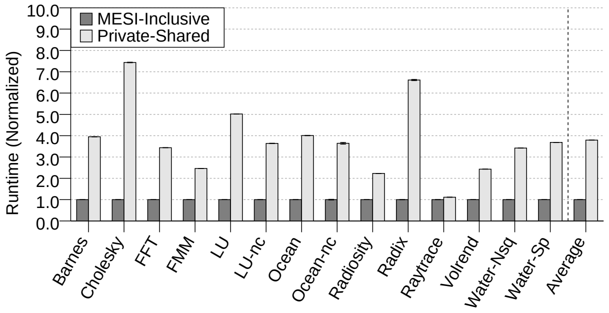
Memory parameters	
Block size	64 bytes
L1 cache (data & private & instr.)	32 KB, 4 ways
L1 access latency (data & private & instr.)	4 cycle
L1S (shared)	(32 KB, 4 ways) per tile
L1S access latency	4 + Network
L2 cache (shared)	512 KB/tile, 16 ways
L2 access latency	12 cycle
Cache organization	Inclusive
Directory information	Included in L2
Memory access time	160 cycles

Network parameters	
Topology	Base: 2-D mesh (4$\times$4) Other: Custom/Simple Network
Routing method	X-Y determinist
Message size	5 flits (data), 1 flit (control)
Link time	1 cycle
Bandwidth	1 flit per cycle

Outline

Results

Normalized Runtime (to unified/MESI L1D)



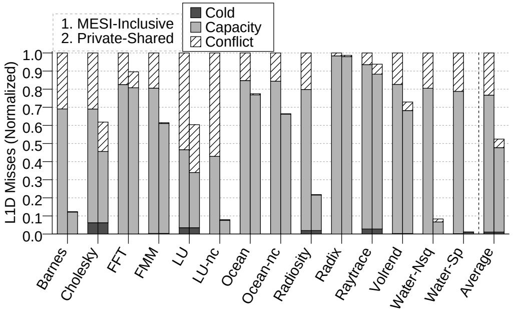
Tiled design, 2D mesh Garnet network. CLH locks

- Realistic layout design, but huge overall slowdown (3.7x)
- Great variability in the results (some applications hide latency better)

Results
Cache Misses

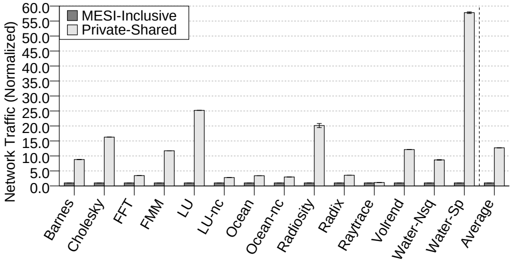
- Now we must discover why we experienced this behavior while others did not, and how we could solve this issue

Results
Cache Misses



- Now we must discover why we experienced this behavior while others did not, and how we could solve this issue
- The extra combined capacity of the L1D (L1P+L1S) causes a significant reduction on the number of cache misses (4-Way 128-entry L1S)
- But L1S accesses cost as much as misses on the private cache, so performance benefits are minimal

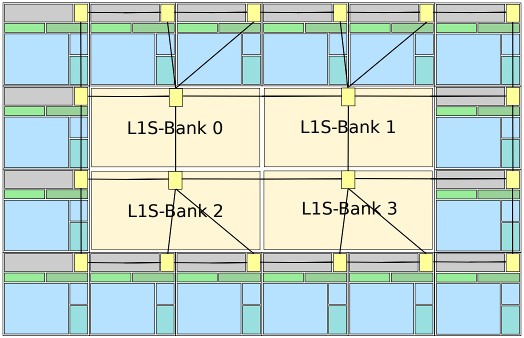
Results
Network Traffic



- Huge increment in traffic due to the shared cache (around 13× on average)
- High variability that **but** similar performance degradation
- Performance penalty only critical for those applications that have shared accesses in their critical path of execution

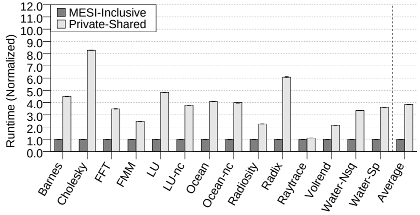
Results

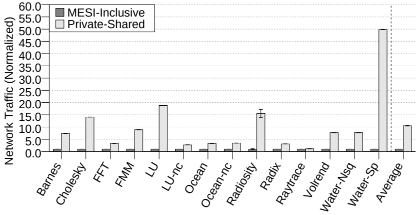
Other Approaches: Centralized Design



- Custom Layout. Place all L1S in the centre of the layout

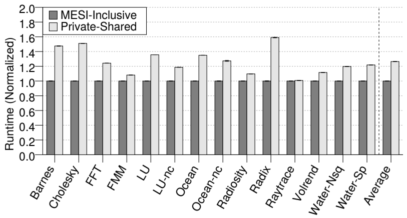
Results
Other Approaches: Centralized Design II

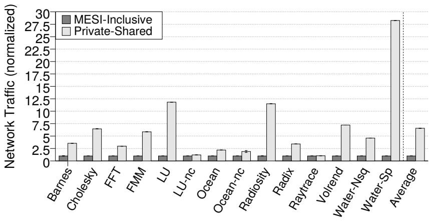




- Runtime increased (3.9x vs 3.4x)
- Network traffic slightly decreased (10x vs 13x)
- Less hops but more contention on L1S controllers

Results
Other Approaches: Idealized Design

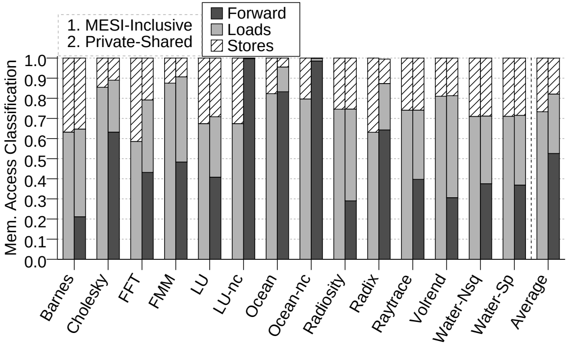




- Idealized network (P2P, without contention)
- Runtime increased by 25%, network traffic increases by 7x
- Even in this optimistic design the L1S accesses outweigh the gains from removing the coherence protocol

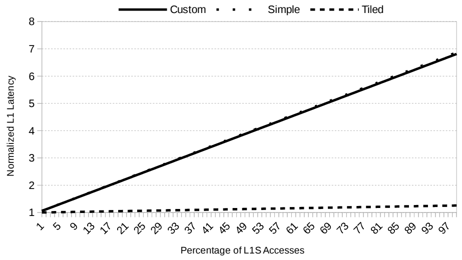
Outline

DISCUSSION
PRIVATE AND SHARED DATA



- 52% shared accesses does not match the literature (10%)
- Build a model based on empirical data to estimate performance based on the amount of private and shared accesses

Discussion
Performance Model



- Based on access and miss latencies to both private and shared caches
- Even for 10% ratio we still have a 60% performance degradation
- But we are modeling a latency-oriented architecture
  - NVIDIA architectures separate private and shared data (programmer)

Discussion
Conclusions

Conclusions

- We analyze a dedicated cache design that takes into consideration the nature of the data being accessed. Our results show two main drawbacks that limit the usability of our implementation:
  - Low accuracy on the classification mechanism (can be improved to 90%)
  - Huge increase of the latency of shared accesses (due to the interconnection network. 6.75 ideal to 23 cycles realistic latency).
- We believe there is also room for improvement by using a specialized network since our design only requires 8 bytes to be sent as control/request plus 16 bytes to be returned (control + data word).
- With improved accuracy for access clasisfication and reduced network latency, we believe our approach can become useful as the number of cores increases and coherence protocols face more scalability issues.
- Additional research is needed to discover the number of cores required to make the dedicated design feasible in terms of performance.
- Throughput oriented cores (GPUs or accelerators like the Xeon Phi) can also benefit from our design, since additional memory latency is usually hidden in those systems by swapping execution threads.

Early Experiences with Separate Caches for Private and Shared Data

Juan M. Cebrián, Alberto Ros,
Ricardo Fernández-Pascual and Manuel E. Acacio

{jcebrian, aros, rfernandez, meacacio}@ditec.um.es

Depart. of Computer Engineering
University of Murcia

Sep 3, 2015