

## OS Controlled Multi-core L2 Cache Management

Shibo Wang

## Issue of Multi-core Cache

- ▶ Cache performance is important
  - ▶ Ever widening processor-memory speed gap
  - ▶ Severely limited chip bandwidth [Huh 2001]
- ▶ Current multi-core cache performance is not good
  - ▶ Stall time due to L2 cache miss is very significant [Basu 2007]
  - ▶ Low utilization of cache for memory intensive workloads [Soares 2008]
- ▶ ...

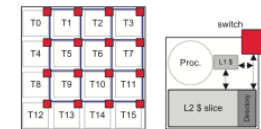
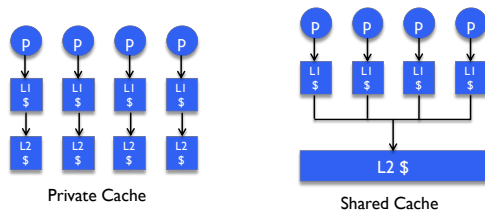


Figure 1. An example 16-core tiled processor chip and its tile (core).

## Multi-core Cache Organization



Hit latency  
Miss rate (Utilization of cache capacity)  
Coherence

## Issues in Multi-core L2 Cache Design

- ▶ Challenges in multi-core L2 caches
  - ▶ Uncontrolled contention in shared L2 caches
  - ▶ (Distributed L2 caches + wire delay) ⇒ NUCA (non-uniform cache access) [Cho 2006]
  - ▶ High on-chip network traffic & related power consumption
- ▶ ...



Program and data location

### Possible Solutions

- ▶ Managing Shared L2 Caches on Multicore Systems in Software [Tam 2007]
  - ▶ Focus on uncontrolled contention on shared L2 caches
- ▶ Managing Distributed, Shared L2 Caches through OS-Level Page Allocation [Cho 2006]
  - ▶ Focus on how to distribute data to minimize overall data access latencies

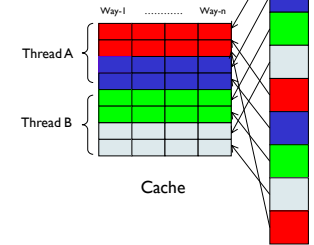
### Page coloring

Cache View



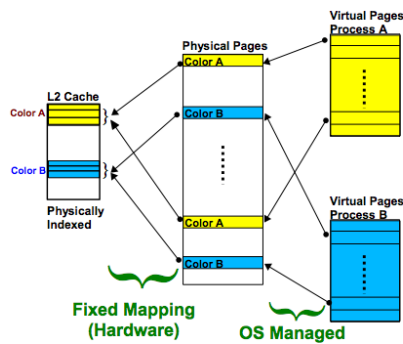
Physical Address

$$\text{Color \#} = \frac{\text{CacheSize}}{\text{PageSize} * \text{CacheAssociativity}}$$

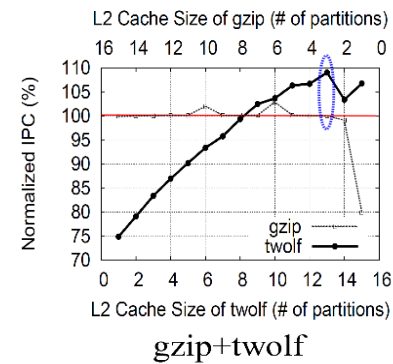


- ▶ Traditional use
  - ▶ To distribute cache accesses evenly across the whole cache

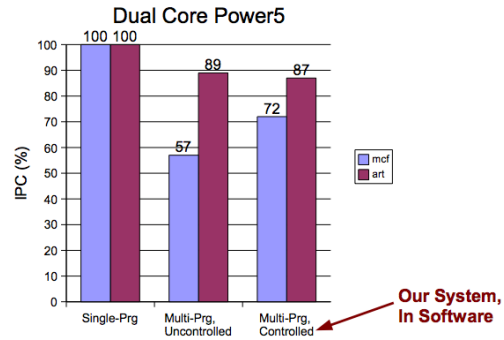
### Software partitioning by page coloring



### Benefits of partitioning



## Solving Uncontrolled contention in Shared L2 Cache



## Data mapping to minimize data access latency

- ▶ Data mapping = deciding data location (i.e., cache slice)
- ▶ Shared caching
  - ▶ Data mapping determined by address
    - ▶  $\text{slice number} = (\text{block address}) \% (N_{\text{slice}})$
    - ▶ No explicit control Mapping granularity = block
- ▶ OS-level page allocation
  - ▶ Coarse granularity
  - ▶ Benefits:
    - ▶ Explicit control
    - ▶ Less on-chip network traffic
    - ▶ Utilize exiting memory optimization technologies, such as OS page coloring

## Data mapping via page allocation

- ▶ Congruence group  $CG(i)$ 
  - ▶  $CG(i) = \{\text{physical page (PPN} = j) \mid \text{pmap}(j) = i\}$
  - ▶ Set of physical pages mapped to slice  $i$
  - ▶ A free list for each  $i \Rightarrow$  multiple free lists, each associated with a CG
- ▶ On each page allocation, consider
  - ▶ Data proximity
  - ▶ Cache pressure
  - ▶ ...
  - ▶ (e.g.) Profitability function

## Profile-driven page mapping

- ▶ Using profiling collect:
  - ▶ Inter-page conflict information
  - ▶ Per-page access count information
- ▶ Page mapping cost function (per slice)
  - ▶ Given program location, page to map, and previously mapped pages
  - ▶  $(\# \text{ conflicts} \times \text{miss penalty}) + \text{weight} \times (\# \text{ accesses} \times \text{latency})$
  - ▶ *weight* as a knob
    - ▶ Larger value  $\Rightarrow$  more weight on proximity (than miss rate)
    - ▶ Optimize both miss rate and data proximity

## OS-based Cache partitioning

- ▶ **Static partitioning**
  - ▶ Profiling & analyzing model
- ▶ **Dynamic partitioning**
  - ▶ Page recoloring is expensive [Lin 2008]
  - ▶ Identify hot pages & selective coloring [Zhang 2009]



## Summary

- ▶ **Managing multi-core caches via software cache partitioning is effective**
  - ▶ Solve uncontrolled contention in shared L2 cache
  - ▶ Minimize overall data access latency
- ▶ **Some other benefits**
  - ▶ Mimicking hardware caching straightforward
  - ▶ Flexible & practical
  - ▶ Employing high-level knowledge about program accessing pattern
- ▶ **Issue**
  - ▶ overhead of page recoloring
  - ▶ Memory pressure



## References

- ▶ Managing Distributed, Shared L2 Caches through OS-Level Page Allocation [Cho, MICRO'06]
- ▶ Managing Shared L2 Caches on Multicore Systems in Software [Tam, WIOSCA'07]
- ▶ Gaining Insights into Multicore Cache Partitioning: Bridging the Gap between Simulation and Real Systems [Lin, HPCA'08]
- ▶ Towards Practical Page Coloring-based Multi-core Cache Management [Zhang, EuroSys'09]
- ▶ Reducing the Harmful Effects of Last-Level Cache Polluters with an OS-Level, Software-only pollute buffer [Soares, MICRO'08]
- ▶ Scavenger: A New Last Level Cache Architecture with Global Block Priority [Basu, MICRO'07]
- ▶ Exploring the Design Space of Future CMPs [Huh, PACT'01]



# Thank you

Disclaimer: Parts of the lecture slides contain original work of Sangyeun Cho, Lei Jin, David Tam, Reza Azimi, Livio Soares, Michael Stumm, Xiao Zhang, Sandhya Dwarkadas and Kai Shen. The slides are intended for the sole purpose of instruction of operating systems at the University of Rochester. All copyrighted materials belong to their original owner(s).

