Hardware Execution Throttling for Multicore Resource Management

Xiao Zhang

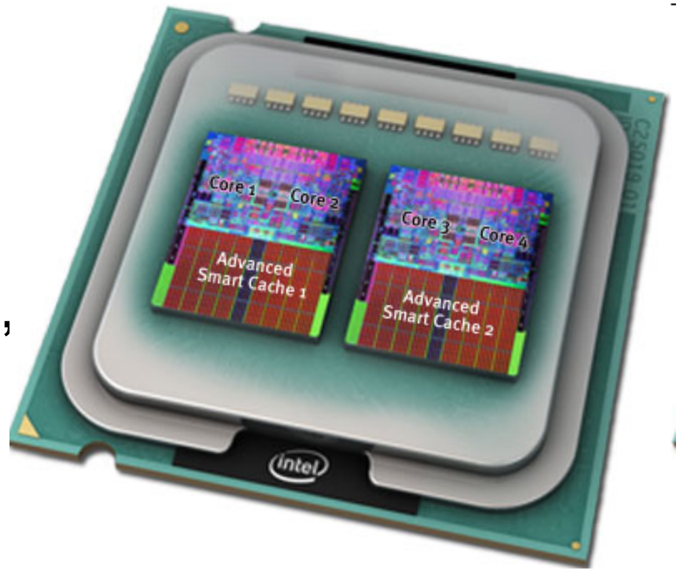
Sandhya Dwarkadas

Kai Shen

UNIVERSITY of ROCHESTER

The Multi-Core Challenge

- Multi-core chip
  - Dominant on market
  - Last level on-chip cache is commonly shared by sibling cores, however sharing is not well controlled



source: http://www.intel.com

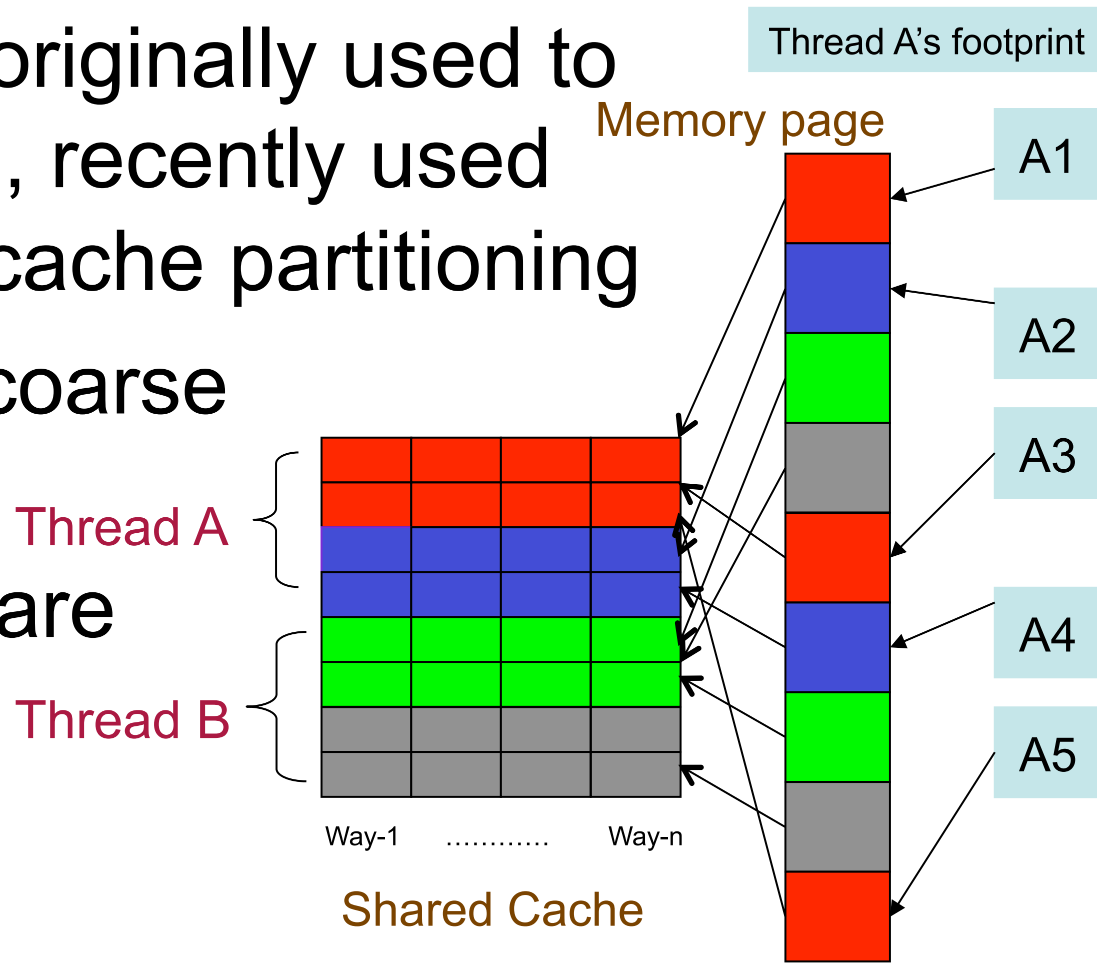
- Challenge: Performance Isolation
  - Poor & unpredictable performance
  - Denial of service attacks

A Full Solution Includes …

- Good mechanism
  - Should be both efficient and practical to deploy
  - Main focus of this talk
- Good policy to govern mechanism
  - as important as mechanism, and not easy
  - Omitted in this talk

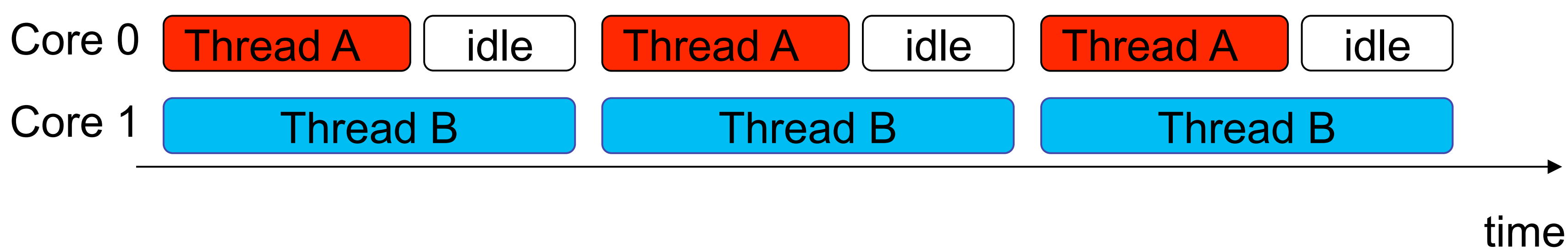
Existing Mechanism(I): Software based Page Coloring

- Classic technique originally used to reduce cache miss, recently used by OS to manage cache partitioning
- Partition cache at coarse granularity
- No need for hardware support



Existing Mechanism(II): Scheduling Quantum Adjustment

- Shorten the time quantum of app that overuses cache
- May let core idle if there is no other active thread available

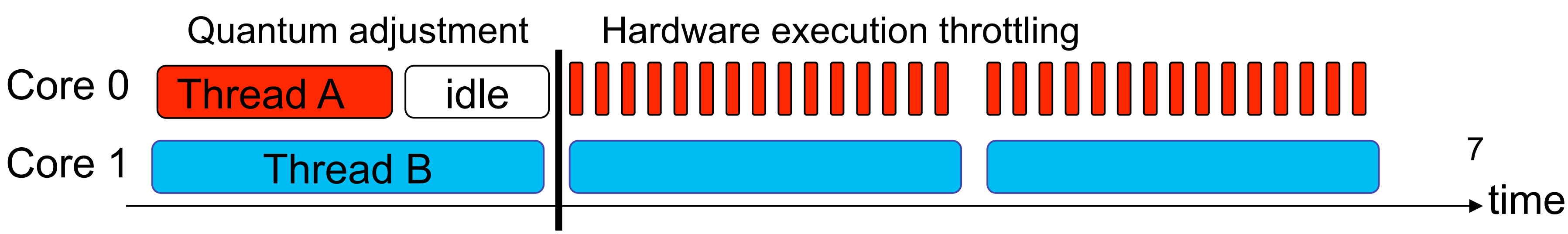


New Mechanism: Hardware Execution Throttling

- Throttle the execution speed of app that overuses cache
  - Duty cycle modulation
    - CPU works only in duty cycles and stalls in non-duty cycles
    - Allow per-core control (vs. per-processor control for existing Dynamic Voltage Frequency Scaling)
  - Enable/disable cache prefetchers
    - L1 prefetchers
      - IP: keeps per-instruction load history to detect stride pattern
      - DCU: prefetches next line when it detects multiple loads from the same line within a time limit
    - L2 prefetchers
      - Adjacent line: Prefetches the adjacent line of required data
      - Stream: looks at streams of data for regular patterns

Brief View of Hardware Execution Throttling

- Comparison to page coloring
  - Little complexity to kernel
    - Code length: 40 lines in a single file (as a reference our page coloring implementation takes 700+ lines of code crossing 10+ files)
  - Lightweight to configure
    - Read plus write register: duty-cycle 265 + 350 cycles, prefetcher 298 + 2065 cycles, which is less than 1 microsecond on a 3Ghz CPU (as a reference re-coloring a page takes 3 microseconds on the same CPU)
- Comparison to scheduling quantum adjustment
  - More fine-grained controlling



Evaluation

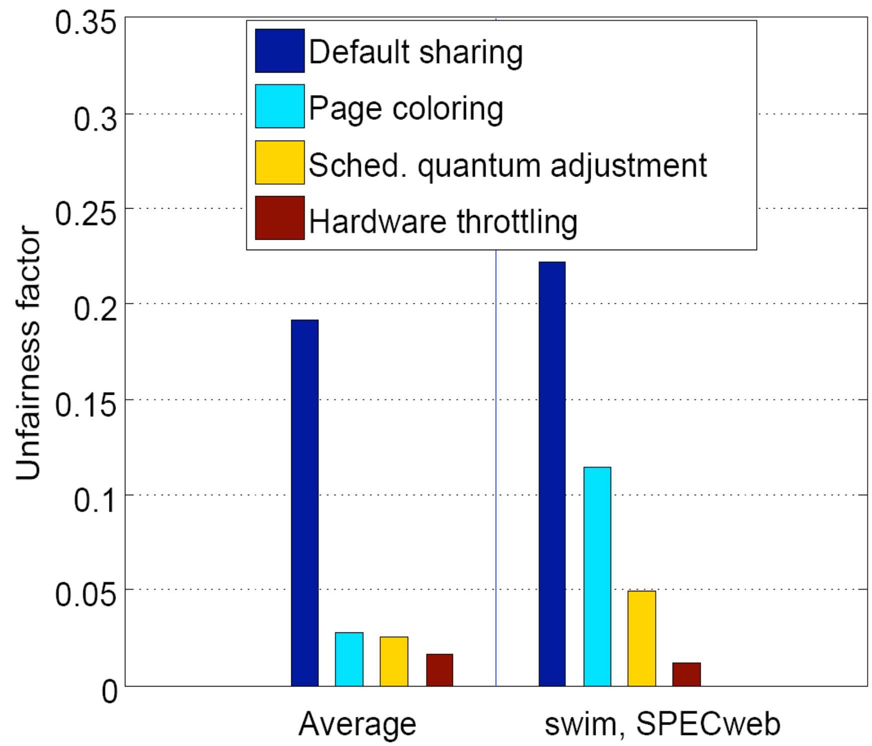
- Candidate mechanisms
  - Page coloring
  - Scheduling quantum adjustment
  - Hardware execution throttling
- Experiment setup
  - Conducted on a 3.0 Ghz Intel dual-core processor
  - 3 SPECCPU-2000 apps (swim, mcf, & equake) and 2 server-style apps (SPECjbb2005 & SPECweb99), running all possible pair-wise co-schedule
- Goal: evaluate their effectiveness in providing performance fairness
  - For each mechanism, tune its configuration offline to achieve best fairness

Fairness Comparison

- Unfairness factor: coefficient of variation (deviation-to-mean ratio, σ / μ) of co-running apps' normalized performances

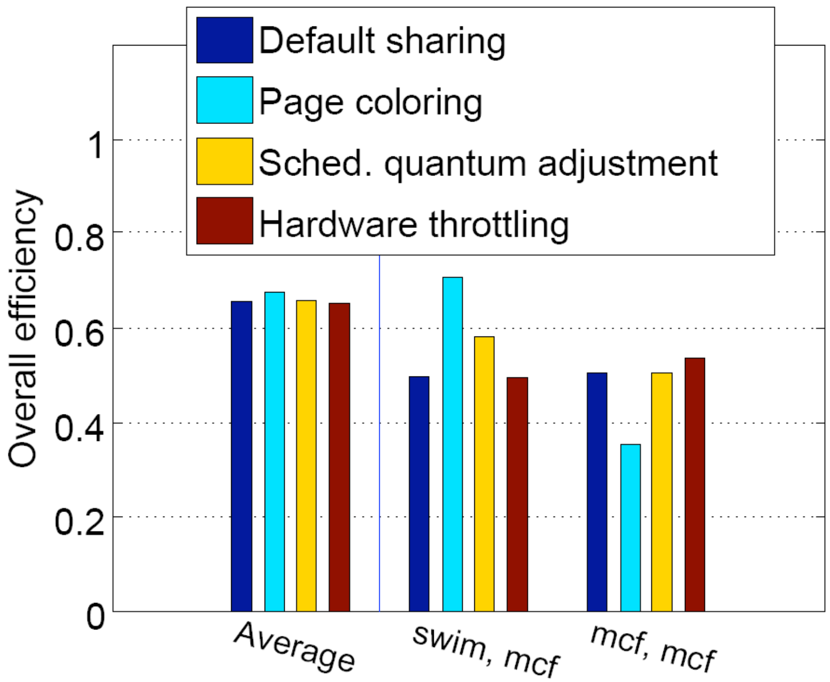
- On average all three mechanisms are effective in improving fairness

- Case {swim, SPECweb} illustrates limitation of page coloring



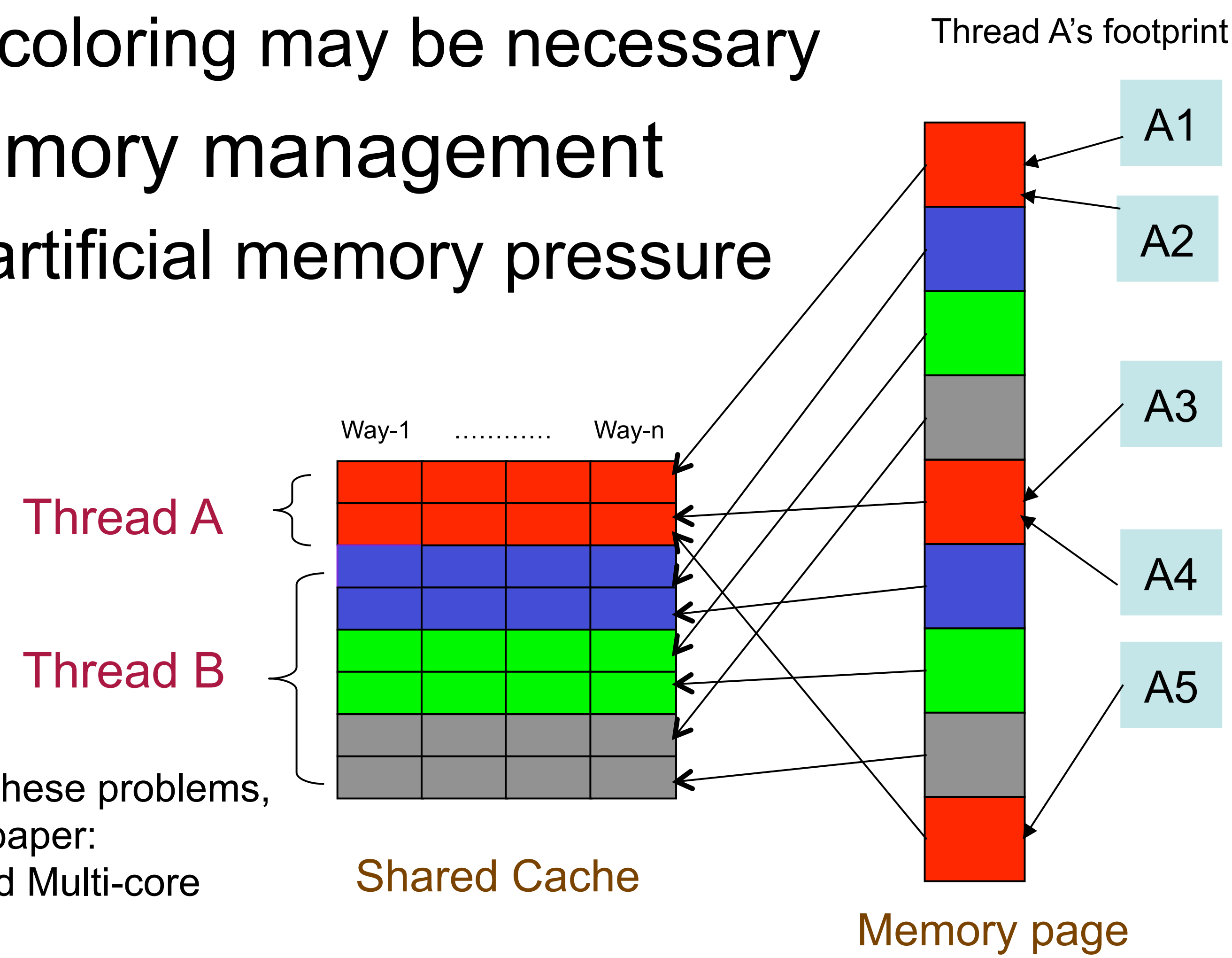
Performance Comparison

- System efficiency: geometric mean of co-running apps' normalized performances
- On average all three mechanisms achieve system efficiency comparable to default sharing
- Case where severe inter-thread cache conflicts exist favors segregation, e.g. {swim, mcf}
- Case where well-interleaved cache accesses exist favors sharing, e.g. {mcf, mcf}



Drawbacks of Page Coloring

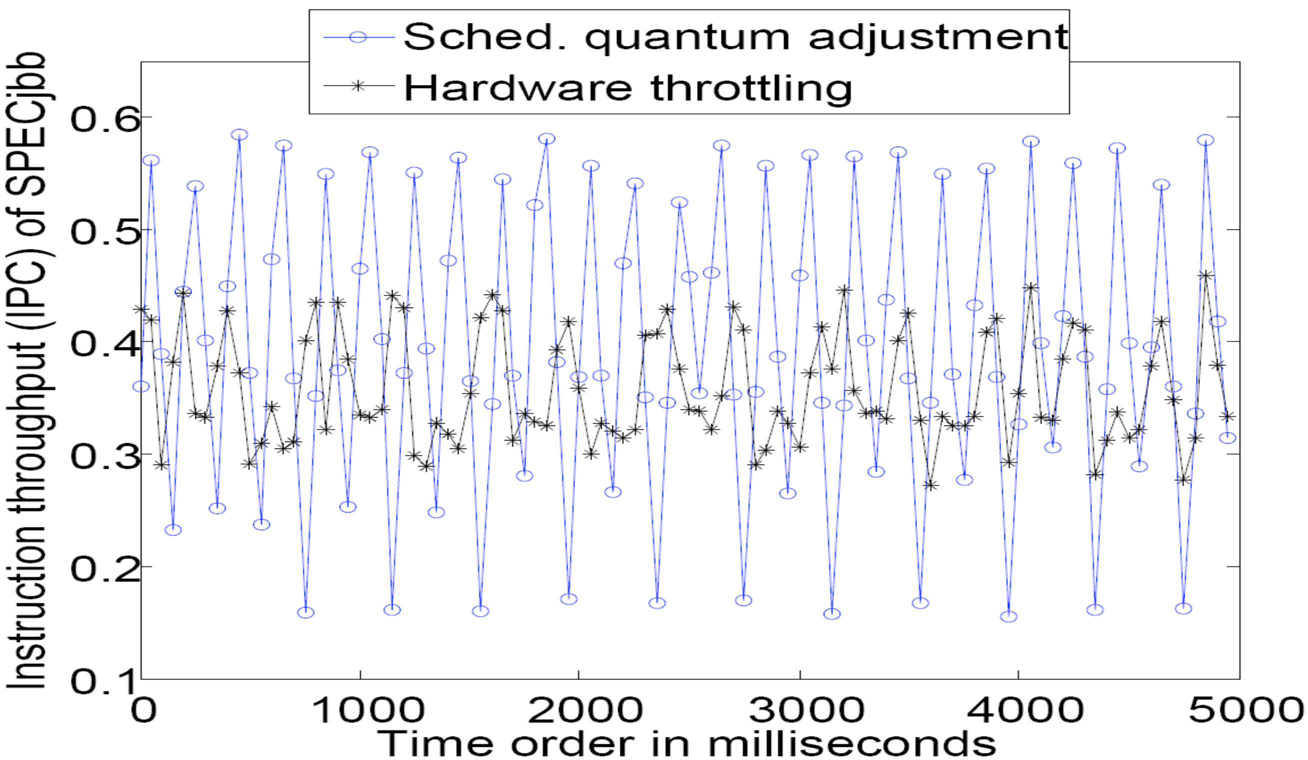
- Expensive re-coloring cost
  - Prohibitive in a dynamic environment where frequent re-coloring may be necessary
- Complex memory management
  - Introduces artificial memory pressure



For more details on tackling these problems, please read our Eurosys'09 paper: Practical Page coloring based Multi-core Cache Management

Drawback of Scheduling Quantum Adjustment

- Coarse-grained control at scheduling quantum granularity may result in fluctuating service delays for individual transactions



Summary

- Hardware execution throttling mechanism for multi-core cache management
  - Fine-grained control
  - Lightweight solution that cleverly reuses existing hardware features
  - System efficiency is competitive to default sharing, largely comparable to scheduling quantum adjustment, but inferior to ideal page coloring

- Future work
  - Investigate policy for online configuration