Understanding the Behavior of Pthread Applications on Non-Uniform Cache Architectures

THE UNIVERSITY OF UTAH®

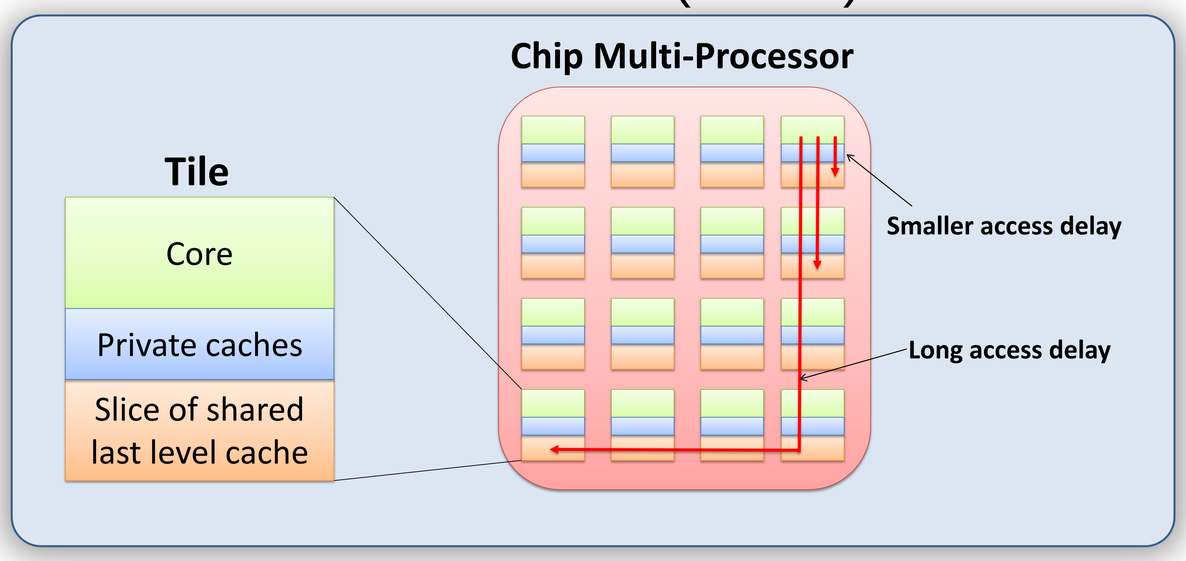
Gagandeep S. Sachdev, Kshitij Sudan, Mary W. Hall & Rajeev Balasubramonian

School of Computing, University of Utah

Why is it important?

- As number of cores in a processor scale up, caches would become banked
  - Keeps individual look-up time small.
  - Allows parallel accesses by different cores.
- Present shared programming model assumes a flat memory.
- Unaware application can have sub-optimal performance.

Non-Uniform Cache Access(NUCA) architecture



Solution Approaches

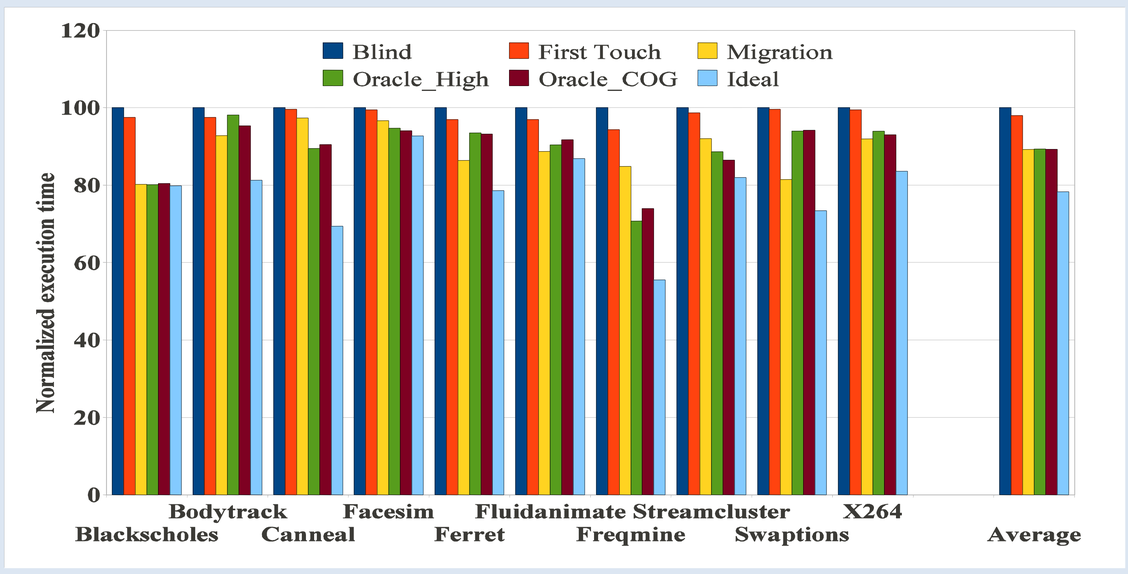
- Dynamic NUCA is proven to be complex
  - Bandwidth and power hungry.
- Static NUCA
  - Physical address determines what bank data resides.
  - Virtual to physical address mapping has to be done smartly.
- Page coloring
  - First touch based coloring is proven to be not accurate.
- Coupling with migration
  - Again power and bandwidth hungry.
  - High overheads.
- We need simple and effective solutions
  - Use of programmer or compiler hints.
  - Get initial placement correct.

Analysis of PARSEC Benchmarks

- Oracle and Migration perform close to Ideal (Blackscholes)
  - Good static mapping perform as good as Migration!
    - Threads work on independent partitions of a large data structure.
  - OS can allocate partitions to optimal banks.
  - First Touch fails
    - Main thread might have initialized .

Experiments

- First Touch
  - Page mapped to the core making first access.
- Migration
  - First touch followed by migrating few pages.
  - Highly accessed pages are chosen.
- Oracle-Two pass simulation
  - First run measures accesses, Second run measures performance.
  - Oracle_High – Page placed in the bank yielding most local hits.
  - Oracle_COG – Page placed at the center of gravity of sharers.
- Ideal
  - Ignores placement and assumes local bank access time for every LLC access.



- Oracle better than Migration (Canneal, Freqmine, Streamcluster)
  - Migration is not ideal or there is a better static mapping.
  - Canneal has uniform sharing on a large data structure.
  - In Streamcluster, threads work primarily on their partition
    - Makes accesses to other partitions too.
  - Migration will migrate pages unnecessarily
    - Based on activity within an epoch.
  - OS can map data partitions to dominant accessor.

- Migration better than Oracle (Bodytrack, Ferret, Fluidanimate, Swaptions, X264)
  - Ferret and Bodytrack
    - Dynamic scheduling of threads.
    - Use global queue to pass work to different pipeline stage.
    - Run-time allocation without data locality
      - Static mapping becomes unsuitable.
    - Don't have a first-in, first-out queue.
      - Data affinity in scheduling tasks needed.
  - Swaptions
    - Pages allocated dynamically from malloc calls
      - Same page allocated to different threads.
      - Static mapping unsuitable.
    - Thread-Private heaps should be used.

Solutions

- Using thread-private storage of stack and local, dynamically allocated data.
- Partitioning global shared objects into separate pages.
- Explicitly marking migratory data.
- Affinity scheduling of parallel tasks.
- Programmer guided pragmas or compiler hints
  - Categorize data structures

Conclusion

- Programming model needs to change
  - For any heterogeneous memory hierarchy.
- Architecture, OS, compiler and application developer should work together
  - Significant performance gains can be achieved.
  - Without increasing system complexity.
- As complexity of memory hierarchy grows, optimizations like these will be critical.