MULTIPLE CORES & MEMORY CONTROLLERS ON CHIP

- Moore’s Law continues: more transistors on a chip
- Need more MCs to maintain memory bw/core
- How do we position memory controllers?
- Latency considerations:
  - NUMA: dominated by local vs. remote latency
  - ANUMA: gradual continuum of access latencies

MOTIVATING EXPERIMENTS

- Observed 14% ping-pong latency differential
- MC access latency increases with Manhattan distance

FUTURE WORK

- More experimentation to understand benefits
  - As a function of application properties
  - ANUMA: considering cache affinity
  - Optimizing for application-level goals
  - SLAs, performance goals
  - Multi-threaded applications

PROBLEM STATEMENT

- MC access asymmetries result in bad placement
- How do we detect it?
- How do we fix it?

THREAD PLACEMENT SOLUTION

- MC access tracking
  - Page fault on memory access
  - Each fault is attributed to specific MCs
  - Adjustable sampling frequency (20Hz)
    - Trades off overhead vs. accuracy
- Thread placement
  - Place threads nearest to MCs they use
    - Greedy order w.r.t. memory intensity
  - Weighted geometric placement algorithm

EXPERIMENTAL SETUP & RESULTS

- TilePro64, 8x8 board, standard linux host
- Placement policies compared:
  - Base: linux scheduler
  - Overhead: base + stats collection
  - Weighted: our approach
  - Brute: brute force approach

Result Takeaways

- Asymmetry-aware placement helps
- Stats collection overhead is high
  - Calls for per-core, per-MC stats counters

Reference