Thread Clustering: Sharing-Aware Scheduling on SMP-CMP-SMT Multiprocessor

David Tam, Reza Azimi, and Michael Stumm

presented by
Fernando Gutierrez
Introduction: Problem

- High popularity of systems that are multi-processor, that can run multithreading.

- Shared memory multiprocessors overhead.
  - Communication between threads that are in different unit
Introduction: Problem
Introduction: Solution

• Hardware for monitoring real time performance
  – Performance monitoring units (PMU) and hardware performance counter (HPC)

• Find out: cache miss was local or remote
Solution: Phases

1) Monitoring Stall Breakdown

2) Detecting Sharing Patterns

3) Thread Clustering

4) Thread Migration
Monitoring Stall Breakdown

- Threshold activations

- If 20% of cycles are used for remote cache access, then do sharing detection
Detecting Sharing Patterns

- shMaps

- Sampling
  - Temporal
  - Spacial
Thread Clustering

- Considers only non-zero vectors
- Intensity of sharing
- Clusters formation
  - Similarity threshold
  - Symmetry of sharing

\[ \text{similarity}(T_1, T_2) = \sum_{i=0}^{N} T_1[i] \times T_2[i] \]

where \(i\) is the \(i\)th entry of the vector \(T_x[\ ]\).
Thread Migration

- Sorting of clusters
- Assign large cluster to chip with low number of threads
- Non clustered threads are move to keep balance
Experiment Setting

- **Platform**
  - 2x2x2 configuration in Power5 machine

- **Workloads**
  - Synthetic micro benchmark: simple multi-thread program
  - VolonoMark: chat server, connection per threat
  - SPECjbb2000: warehouse access
  - RUBiS: auction site, focus on database
Results: Performance

- Reduction in Remote Cache Stalls (%)
- Speedup over Default Linux (%)

Graphs showing performance metrics for SPEC JBB2000, RUBIS, and Volano.
Discussion

- Local Cache Contention
  - Clustering threads can create cache contention
- Migration Costs
  - Threat migration has low cost
- PMU Requirements
  - PMU basic capabilities
- Hardware Properties
  - Disparity between local and remote cache latencies