MCFQ: Leveraging Memory-level Parallelism and Application's Cache Friendliness for Efficient Management of Quasi-partitioned Caches

D. Kaseridis, M. F. Iqbal, J. Stuecheli and L. John

Department of Electrical & Computer Engineering
The University of Texas at Austin

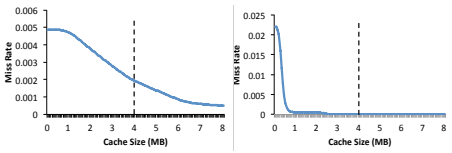
Introduction

- Previous work to address the cache contention problem:
  - **Cache Partitioning Scheme** *[UCP MICRO '06]*
    - Inefficient use of capacity, not scaling to large CMP
  - **Cache Replacement policies**
    - Hard to provide control over cache capacity → interference
  - **Cache-block dead time prediction & Cache Pseudo-partitioning**
    - Destructive interference still presents
    - No absolute control of capacity per core
    - Oblivious to applications memory behavior
- **Memory behavior is important in sharing cache capacity**

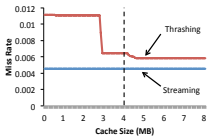
Pseudo-partitions best compromise IF DONE correctly

Motivation

Memory Behavior of Application of Last-level Cache



Cache Friendly (gcc) Cache Fitting (dealII)



Cache Thrashing (lbm, milc)

Requirements of efficient Cache Management scheme

1. Cache space allocated to applications proportionally to the real benefit of using the capacity
2. Thrashing applications have to be isolated
3. Fitting applications have to be guaranteed a minimum capacity
4. Cache capacity allocation priority scheme:
   Friendly > Fitting > Thrashing (few LRU ways)

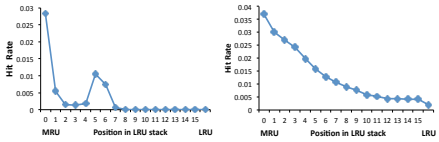
Need a mechanism to allocate priorities within the same category of applications → Interference Sensitivity Factor

Profilers

- **Average MLP**
  - Use of MSHR (Miss Status Hold Register)
  - Two extra counters: aggregated number of misses in structure, overall number of L1 misses added
- **Memory Behavior**
  - Using the Cache miss profilers from MSA circuit
  - Thrashing: $\frac{MPKI_{max.capacity}}{MPKI_{min.capacity}} > Threshold_{Thrashing}$
  - Fitting: MKPI < $Threshold_{Fitting}$
  - Remaining Friendly

Interference Sensitivity Factor

- When more than one applications with same memory behavior
- Based on sensitivity an application to cache contention & how friendly is in sharing capacity
- Estimate: position $Interference\ Sensitivity\ Factor = \sum_{i=0}^{\#ways-1} i * hits(i)$ i=0 MRU



Namd Soplex

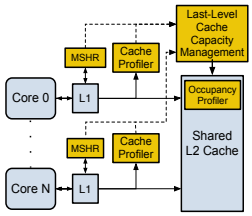
Partition Scaling

- Need to know the actual average occupancy of cache capacity in LLC
- Less than **80%** of ideal → adjust **Insertion Point** (IPs) at next epoch
- Two counters per core:
  - **Occupancy Counter**: Number of actual ways occupied in cache for a core
  - **Cache accesses Counter** per core
- **Important addition to achieve performance targets**

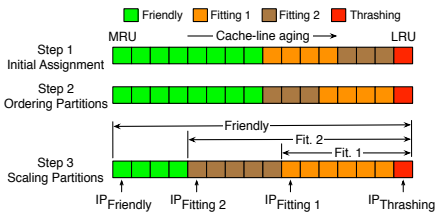
Scheme

MCFQ Policies

- **Insertion Policy (Insertion Points – IPs)**
  - Use UCP to find **ideal** partition sizes
  - Applications Cache friendliness
    - Cache Friendly higher priority
    - Cache Fitting intermediate
    - Cache Thrashing restricted to LRU position
  - Interference sensitivity factor and Partitions Scaling (next slide)
- **Promotion:** Every hit moves line to core's Insertion Point
- **Replacement Policy:** LRU of the whole cache set



Overall Scheme



Example of operation on 4-core CMP 16-way LLC

Evaluation

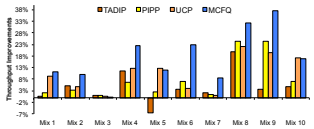
Configuration

Memory Subsystem	L1 D+I	L2	Main Memory	Memory Controller	Prefetcher
	2-way, 64KB, 3 cycles, 64B block	16-way, 4M, 12 cycles, 64B block, Pseudo-LRU	8GB, 16GB/s, DDR3 1066-6-6-6, 16 Req./Core	2 Controllers, 2 Ranks/ Controller, 32 Read/Write entries	H/W stride n, 8 streams/ core
Core processor	Frequency	Pipeline	Reorder Buffer / Scheduler	Branch Predictor	
	4GHz	30 stages / 4 wide fetch-decode	128 / 64 Entries	Direct YAGS / indirect 256 entries	

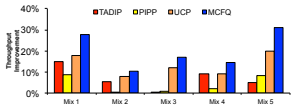
Experiments

4 Cores		8 Cores	
Bench. Group	Benchmarks	Bench. Group	Benchmarks
Mix 1 – All Friendly	soplex, bzip2, h264ref, perlbench	Mix 1 – All Friendly	soplex, omnetpp, perlbench, calculix, gromacs, dealII, calculix, gromacs
Mix 2 – All Fitting	xalancbmk, wrf, tonto, gamess		
Mix 3 – All Thrashing	leslie3d, sjeng, bwaves, zeusmp	Mix 2 – All Fitting	xalancbmk, gobmk, wrf, gobmk, hmmer, astar, gamess, hmmer
Mix 4 – 3 Fr. : 1 Fit	omnetpp, bzip2, calculix, astar		
Mix 5 – 2 Fr. : 2 Fit	bzip2, mcf, gobmk, gamess	Mix3 – 4 Fr.: 2 Fit.: 2 Thr.	omnetpp, bzip2, gobmk, gromacs, povray, h264ref, lbm, libquantum
Mix 6 – 1 Fr. : 3 Fit	omnetpp, xalancbmk, gamess, wrf		
Mix 7 – 3 Fr./Fit. : 1 Thr.	mcf, perlbench, hmmer, bwaves	Mix 4- 2 Fr.: 4 Fit. : 2 Thr.	mcf, gobmk, gromacs, hmmer, gamess, tonto, libquantum, milc
Mix 8 – 3 Fr./Fit. : 2 Thr.	xalancbmk, dealII, milc, zeusmp		
Mix 9 – 2 Fr. : 1 Fit : 1 Thr.	mcf, bzip2, astar, leslie3d	Mix 5 – 2 Fr.: 2 Fit. : 4 Thr.	omnetpp, soplex, gobmk, gamess, libquantum, milc, zeusmp, milc
Mix 10 – 1 Fr. : 2 Fit: 1 Thr.	mcf, gobmk, gamess, libquantum		

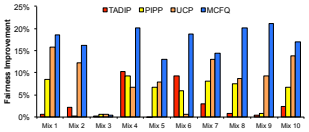
Results



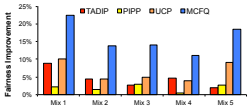
Throughput 4-cores - 19% LRU, 14% TADIP, 13% PIPP, 10% UCP



Throughput 8-cores - 20% LRU, 13% TADIP, 17% PIPP, 8% UCP



Fairness 4-cores - 17% LRU, 12% TADIP, 14% PIPP, 9% UCP



Fairness 8-cores - 15% LRU, 13% TADIP, 12% PIPP, 8% UCP

Conclusions

- Using applications' **Memory behavior** is necessary in sharing cache pseudo-partitions
- Accurate monitors of memory behavior, *interference sensitivity* and *average occupancy* is necessary for **precise control** of partitions
- Careful management can provide most of **benefits of isolated partitions** while leading to **scaling solutions**

Acknowledgements

This work is sponsored by the National Science Foundation under award 0702694 and CRI collaborative awards: 0751112, 0750847, 0750851, 0750852, 0750860, 0750868, 0750884, 0751091.