Refining the Utility Metric for Utility-Based Cache Partitioning

Xing Lin,

Rajeev Balasubramonian,

School of Computing,

University of Utah

Cache partitioning background

- Multi-core architectures have become mainstream
  - Power wall, etc.
- The last-Level cache(LLC) is shared among multiple cores
  - Strict or implicit cache partitions
- Utility-based Cache Partitioning - marginal utility
  - A way is assigned to a core that will benefit most
- Proposed policies
  - UCP (Qureshi and Patt, MICRO '06)
  - PIPP (Xie and Loh, ISCA '09)
  - TADIP (Jaleel, PACT '08)

THE UNIVERSITY OF UTAH

The Problem

- Most utility-based cache partitioning schemes measure the marginal utility in the number of misses.
  - It is easy to get the misses per kilo-instruction(MPKI) curve with a simple shadow tag structure.
  - Optimizing for MPKI is a reasonable approximation for optimizing for IPC

- However, the extent of this approximation has not been quantified.
  - Cache miss has a different impact on performance because of MLP and latency tolerance of the code.
  - This is a well-known phenomenon.

Related work

- **Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches,** M. Qureshi and Y. Patt, MICRO '06.
- **A Cache for MLP-Aware Cache Replacement,** M. Qureshi, D. Lynch, O. Mutlu, and Y. Patt, ISCA '06
- **IPC-Based Cache Partitioning: An IPC-Oriented Dynamic Shared Cache Partitioning Mechanism,** G. Suo, X. Yang, G. Liu, J. Wu, K. Zeng, B. Zhang,

  and Y. Lin, ICHIT '08

Contributions of this work

- The first work to quantify the extent of divergence between cache partitions, optimized for MPKI and IPC.
  - This new data is a useful reference for future cache policy studies.

- Introduce a simple CPI prediction scheme to achieve nearly optimal cache partitioning

Experimental Methodology

- Simulator:
  - CMP$im(provided by the first JILP Workshop on Computer Architecture Competitions, 2010)
  - Out-of-order superscalar processor
- Workloads:
  - 23 SPEC CPU2006 benchmarks
  - Simulated the 2B instructions
- Analytic scripts:
  - Wrote Perl scripts to analyze cache partitions optimized for MPKI or IPC for all possible 2-, 3- and 4-benchmark workloads

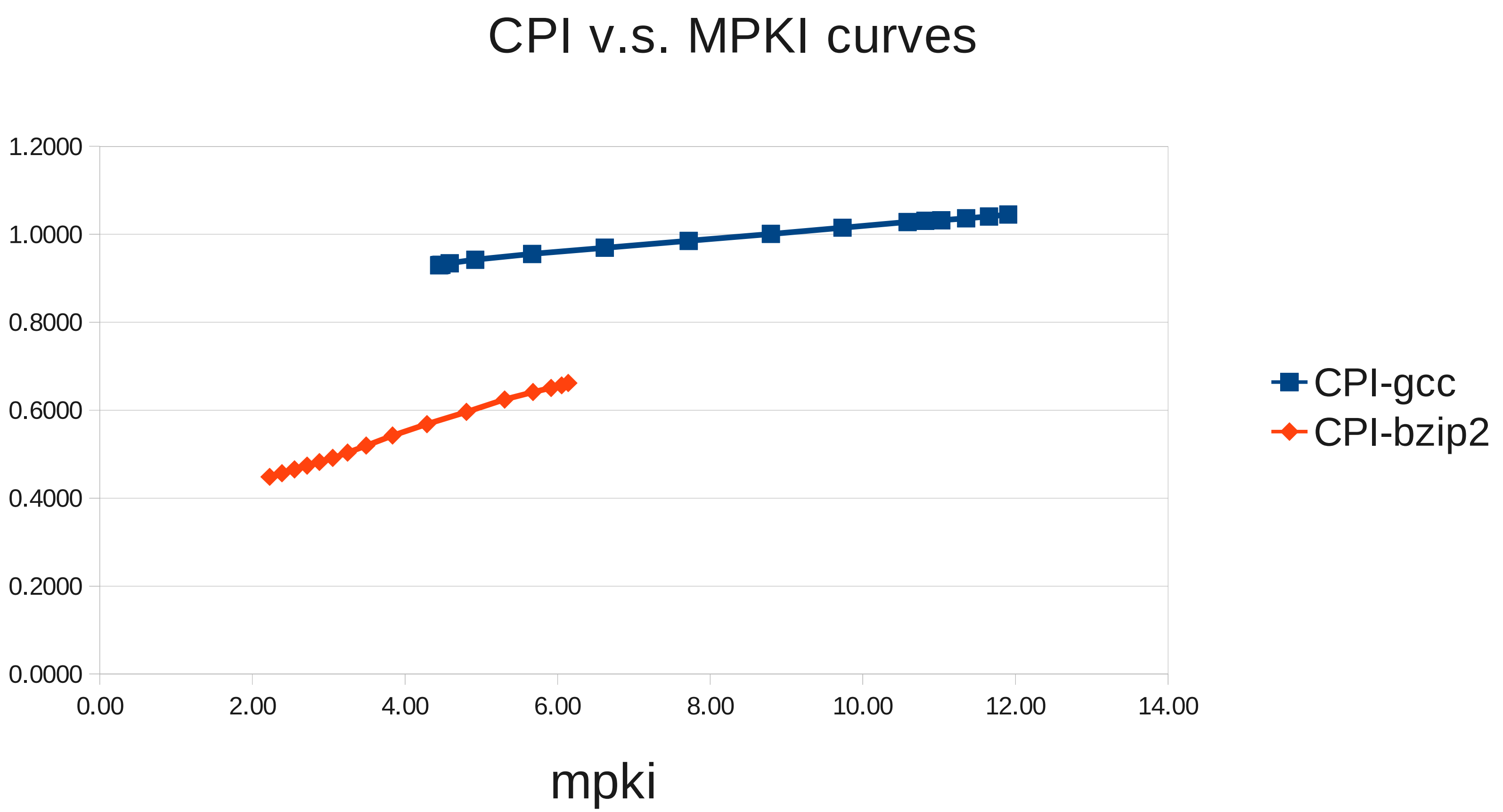
THE UNIVERSITY OF UTAH

MPKIs and CPIs for bzip2

ways	cache size(KB)	LLC misses	MPKI	CPI
1	64	12278744	6.14	0.6617
2	128	12105769	6.05	0.6566
3	192	11828125	5.91	0.6504
4	256	11355935	5.68	0.6411
5	320	10614493	5.31	0.6243
6	384	9611348	4.81	0.5960
7	448	8579246	4.29	0.5682
8	512	7672831	3.84	0.5422
9	576	6984007	3.49	0.5194
10	640	6496669	3.25	0.5034
11	704	6102369	3.05	0.4914
12	768	5758217	2.88	0.4822
13	832	5430133	2.72	0.4738
14	896	5102029	2.55	0.4653
15	960	4773461	2.39	0.4566
16	1024	4451004	2.23	0.4483

THE UNIVERSITY OF UTAH

CPI v.s. MPKI curves for bzip2 and gcc

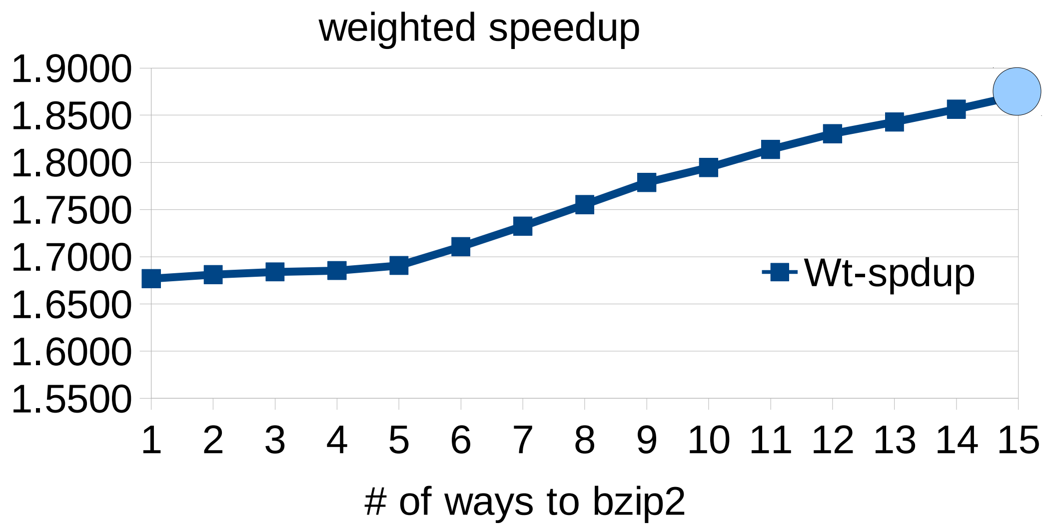


THE UNIVERSITY OF UTAH

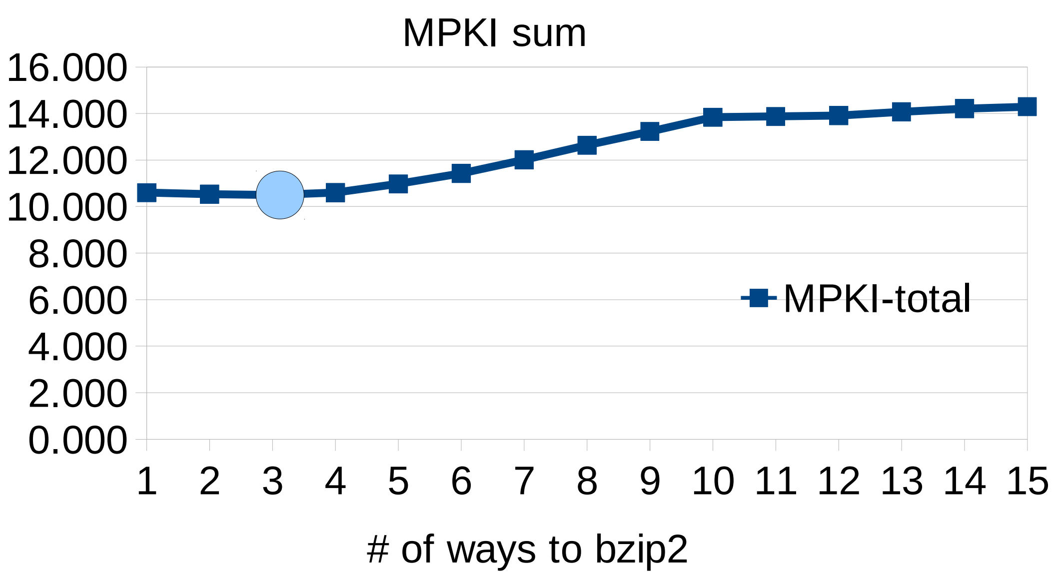
Cache partitions example

ways to bzip2	MPKI-bzip2	MPKI-gcc	CPI-bzip2	CPI-gcc	MPKI-total	Wt-spdup
1	6.139372	4.46	0.661668	0.929918	10.597	1.6768
2	6.052884	4.48	0.656552	0.930904	10.531	1.6810
3	5.914062	4.59	0.650407	0.934170	**10.499**	1.6840
4	5.677967	4.92	0.641133	0.942286	10.598	1.6854
5	5.307246	5.67	0.624320	0.955335	10.974	1.6907
6	4.805674	6.62	0.595972	0.969432	11.424	1.7107
7	4.289623	7.72	0.568210	0.984973	12.009	1.7324
8	3.836415	8.80	0.542185	1.000820	12.632	1.7553
9	3.492003	9.73	0.519377	1.014930	13.227	1.7787
10	3.248334	10.59	0.503391	1.027950	13.836	1.7945
11	3.051184	10.82	0.491443	1.030730	13.870	1.8137
12	2.879108	11.03	0.482165	1.031800	13.906	1.8304
13	2.715066	11.35	0.473807	1.036410	14.068	1.8427
14	2.551014	11.65	0.465347	1.040640	14.205	1.8563
15	2.386730	11.91	0.456604	1.044860	14.292	**1.8711**

Cache partitions example



High is good



Low is good

Minimize MPKI sum does not always lead to better performance.

THE UNIVERSITY OF UTAH

Extent of divergence when optimized for MPKI and weighted speedup

Metric	2 Programs	3 Programs	4 Programs
Divergent cases	84/253 (33.20%)	828/1771 (46.75%)	4827/8855 (54.51%)
Wt-Spdup ≥ 10% Wt-Spdup ≥ 8% Wt-Spdup ≥ 6% Wt-Spdup ≥ 4% Wt-Spdup ≥ 2%	3/84 (3.57%) 4/84 (4.76%) 8/84 (9.52%) 12/84 (14.29%) 24/84 (28.57%)	4/828 (0.48%) 39/828 (4.71%) 99/828 (11.96%) 151/828 (18.24%) 262/828 (31.64%)	0/4827 (0.0%) 1/4827 (0.02%) 312/4827 (6.46%) 1268/4827 (26.27%) 1793/4827 (37.15%)
MPKI ≥ 50% MPKI ≥ 40% MPKI ≥ 30% MPKI ≥ 20% MPKI ≥ 10% MPKI ≥ 5%	4/84 (4.76%) 4/84 (4.76%) 6/84 (7.14%) 8/84 (9.52%) 11/84 (13.10%) 18/84 (21.43%)	27/828 (3.26%) 31/828 (3.74%) 72/828 (8.70%) 92/828 (11.11%) 134/828 (16.18%) 257/828 (31.04%)	154/4827 (3.19%) 189/4827 (3.92%) 290/4827 (6.01%) 574/4827 (11.89%) 1000/4827 (20.72%) 1666/4827 (34.51%)
Wt-Spdup avg (all) Wt-Spdup avg (diff) MPKI avg (all) MPKI avg (diff)	0.59% 1.79% 2.54% 7.66%	0.95% 2.03% 3.89% 8.31%	1.13% 2.08% 4.42% 8.12%

UNIVERSITY OF UTAH

Extent of divergence when optimized for MPKI and IPC-sum

Metric	2 Programs	3 Programs	4 Programs
Divergent cases	110/253 (43.48%)	1088/1771 (61.43%)	6548/8855 (77.50%)
IPC-Sum ≥ 20% IPC-Sum ≥ 15% IPC-Sum ≥ 10% IPC-Sum ≥ 5%	5/110 (4.55%) 10/110 (9.09%) 16/110 (14.55%) 29/110 (26.36%)	26/1088 (2.39%) 77/1088 (7.08%) 187/1088 (17.19%) 352/1088 (32.35%)	8/6548 (0.12%) 140/6548 (2.14%) 959/6548 (14.65%) 2426/6548 (37.05%)
MPKI ≥ 50% MPKI ≥ 40% MPKI ≥ 30% MPKI ≥ 20% MPKI ≥ 10% MPKI ≥ 5%	12/110 (10.91%) 15/110 (13.64%) 18/110 (16.36%) 19/110 (17.27%) 25/110 (22.73%) 42/110 (38.18%)	96/1088 (8.82%) 128/1088 (11.76%) 207/1088 (19.03%) 252/1088 (23.16%) 331/1088 (30.42%) 565/1088 (51.93%)	412/6548 (6.29%) 507/6548 (7.74%) 859/6548 (13.12%) 1454/6548 (22.21%) 2384/6548 (36.41%) 3580/6548 (54.67%)
IPC-Sum avg (all) IPC-Sum avg (diff) MPKI avg (all) MPKI avg (diff)	1.85% 4.26% 6.97% 16.02%	2.90% 4.72% 9.84% 16.02%	**3.40%** 4.60% 10.01% 13.54%

UNIVERSITY OF UTAH

Main observations from our results

- With more programs, the extent of divergence increases and the speedup of optimizing for IPC becomes more significant.
- For 4-benchmark workloads, the average speedup in IPC-sum can be as high as 3.4% by optimizing for IPC-sum.
- For some workloads(~9%), a considerable performance improvement(>= 6%) can be made by optimizing for IPC.
- Performance improvement can be made even if the MPKI sum is increased significantly(>= 30%).
- For each benchmark, the CPI is roughly linear with the MPKI.
  - CPI(w) = c1 + c2 * MPKI(w)
  - c2 actually measures the latency tolerance for each benchmark.

Optimal CPI Prediction

- $CPI(w) = c1 + c2 \times MPKI(w)$
- Steps:
  - Two samples are magically selected to calculate c1 and c2.
  - c1 and c2 are used to convert the MPKI curve into the estimated CPI curve.
  - Error is measured as the maximum difference between estimated CPI curve and the actual CPI curve.
- Results:
  - Average error: 0.38%
  - Max error: 2.04%
  - Error > 1%: 2/23
  - Error > 0.5%: 4/23

Value of c2 from optimal CPI prediction

- C2: the latency tolerance for each benchmark.

gromacs	0.1367	hmmer	0.0812
gamess	0.0630	namd	0.1625
calculix	0.1116	astar	0.1731
mcf	0.0539	cactusADM	0.0627
lbm	0.1630	bwaves	0.0097
h264ref	0.1187	libquantum	0.0366
leslie3d	0.0449	milc	0.1725
soplex	0.0786	zeusmp	0.0520
sphinx3	0.1029	povray	0.1416
sjeng	0.1327	omnetpp	0.0328
bzip2	0.0557	tonto	0.0714
gcc	0.0157		

THE UNIVERSITY OF UTAH

Practical CPI prediction

- Optimal CPI prediction requires a long exploratory stage to get CPIs for all possible ways.
- $CPI(w) = c1 + c2 \times MPKI(w)$
- Steps:
  - two same samples are selected for all benchmarks to calculate c1 and c2.
  - c1 and c2 are used to convert the MPKI curve into estimated CPI curve.
  - Error is measured as the maximum difference between estimated CPI curve and the actual CPI curve.
- Results: way 4 and 15 are the best two samples.
  - Average error: 0.53%
  - Max error: 2.53%
  - Error > 2%: 2/23
  - Error > 1%: 3/23

Cache partitioning based on practical CPI prediction, optimized for IPC-sum

Metric	2 programs		3 programs		4 programs	
Divergent cases	116/253 (45.85%)		1099/1771 (62.06%)		6524/8855 (73.68%)	
IPC-sum ≥ 20% IPC-sum ≥ 15% IPC-sum ≥ 10% IPC-sum ≥ 5%	5/116 (4.31%) 10/116 (8.62%) 16/116 (13.79%) 29/116 (25.00%)		26/1099 (2.37%) 77/1099 (7.01%) 187/1099 (17.02%) 348/1099 (31.67%)		8/6524 (0.12%) 140/6524 (2.15%) 957/6524 (14.67%) 2413/6524 (36.99%)	
MPKI ≥ 50% MPKI ≥ 40% MPKI ≥ 30% MPKI ≥ 20% MPKI ≥ 10% MPKI ≥ 5%	12/116 (10.34%) 15/116 (12.93%) 18/116 (15.52%) 18/116 (15.52%) 24/116 (20.69%) 40/116 (34.48%)		98/1099 (8.92%) 133/1099 (12.10) 206/1099 (18.74%) 252/1099 (22.93%) 330/1099 (30.03%) 556/1099 (50.59%)		400/6524 (6.13%) 494/6524 (7.57%) 838/6524 (12.84%) 1424/6524 (21.83%) 2334/6524 (35.78%) 3568/6524 (54.69%)	
IPC-sum avg (all) IPC-sum avg (diff)	1.84% 4.01%	1.85% 4.26%	2.88% 4.65%	2.90% 4.72%	3.39% 4.61%	3.40% 4.60%
MPKI avg (all) MPKI avg (diff)	6.87% 14.98%	6.97% 16.02%	9.85% 15.87%	9.84% 16.02%	9.88% 13.40%	10.01% 13.54%

Conclusion

- MPKI based cache partitioning can lead to sub-optimal cache partitions for a certain number of workloads.
  - Divergence increases as more programs share the cache.
- CPI prediction based on way 4 and 15 works quite well.
- Cache partitioning based on proposed CPI prediction scheme can achieve nearly optimal performance.

Source code is available for public verification at:
https://github.com/xinglin/mpki-cpi

Questions?

Cache partitioning based on practical CPI prediction, optimized for wt-spdup

Metric	2 programs		3 programs		4 programs	
Divergent cases	89/253 (35.18%)		844/1771 (47.66%)		5038/8855 (56.89%)	
Wt-Spdup ≥ 10%	3/89 (3.37%)		4/844 (0.47%)		0/5038 (0.00%)	
Wt-Spdup ≥ 8%	4/89 (4.49%)		39/844 (4.62%)		1/5038 (0.02%)	
Wt-Spdup ≥ 6%	8/89 (8.99%)		99/844 (11.73%)		303/5038 (6.01%)	
Wt-Spdup ≥ 4%	12/89 (13.48%)		151/844 (17.89%)		1253/5038 (24.87%)	
Wt-Spdup ≥ 2%	23/89 (25.84%)		262/844 (31.04%)		1779/5038 (35.31%)	
MPKI ≥ 50%	4/89 (4.49%)		25/844 (2.96%)		135/5038 (2.68%)	
MPKI ≥ 40%	4/89 (4.49%)		29/844 (3.44%)		155/5038 (3.08%)	
MPKI ≥ 30%	6/89 (6.74%)		70/844 (8.29%)		251/5038 (4.98%)	
MPKI ≥ 20%	7/89 (7.87%)		100/844 (11.85%)		538/5038 (10.68%)	
MPKI ≥ 10%	10/89 (11.24%)		147/844 (17.42%)		981/5038 (19.47%)	
MPKI ≥ 5%	17/89 (19.10%)		262/844 (31.04%)		1703/5038 (33.80%)	
Wt-Spdup avg (all)	0.58%	0.59%	0.93%	0.95%	1.11%	1.13%
Wt-Spdup avg (diff)	1.66%	1.79%	1.96%	2.03%	1.96%	2.08%
MPKI avg (all)	2.42%	2.54%	3.90%	3.89%	4.34%	4.42%
MPKI avg (diff)	6.89%	7.66%	8.19%	8.31%	7.63%	8.12%

UNIVERSITY OF UTAH