Performance Evaluation of Recently Proposed Cache Replacement Policies

Prepared by :Asma Abdelkarim

Outline

- Motivation
- Simulated Replacement Policies
- Simulation Methodology
- Simulation Results
- Conclusion
- References

Motivation

- Many replacement policies have been recently proposed to reduce the miss-rates/miss-penalty of lower level caches.

- Provide a unified simulation environment for the recently proposed replacement policies

Simulated Replacement Policies

1. Dynamic Insertion Policy (DIP) [4]

- DIP adaptively chooses the appropriate policy to be used from either: LRU or BIP (Bimodal Insertion Policy).
- Normally BIP inserts all new blocks in the least-recently-used position.
- BIP inserts blocks in the most-recently-used position with a low probability.
- BIP can prevent thrashing for memory-intensive workloads.

Simulated Replacement Policies

2. MLP-Aware Replacement Policy [5]

- **Memory Level Parallelism is defined as: *the number of useful long-latency off-chip accesses outstanding when there is at least one such access outstanding.* [5]**
- **Making the replacement policy aware of MLP means:**

  - Blocks with isolated misses are favored over blocks with parallel misses.

  - Thus, reducing the miss penalty.
- **In [5], the linear (LIN) policy is proposed where the victim block is chosen depending on its MLP-cost and recency.**
- **Moreover, and adaptive policy is proposed to choose the appropriate policy from either LIN or LRU.**

Simulated Replacement Policies

3. Adaptive Insertion Policy of LRU-LFU [6]

- **The proposed adaptive policy in [6] dynamically chooses one of two policies from the well-known policies (LRU,LFU,FIFO and random) to be applied.**

- **In this simulation experiment the adaptive LRU-LFU will be implemented.**

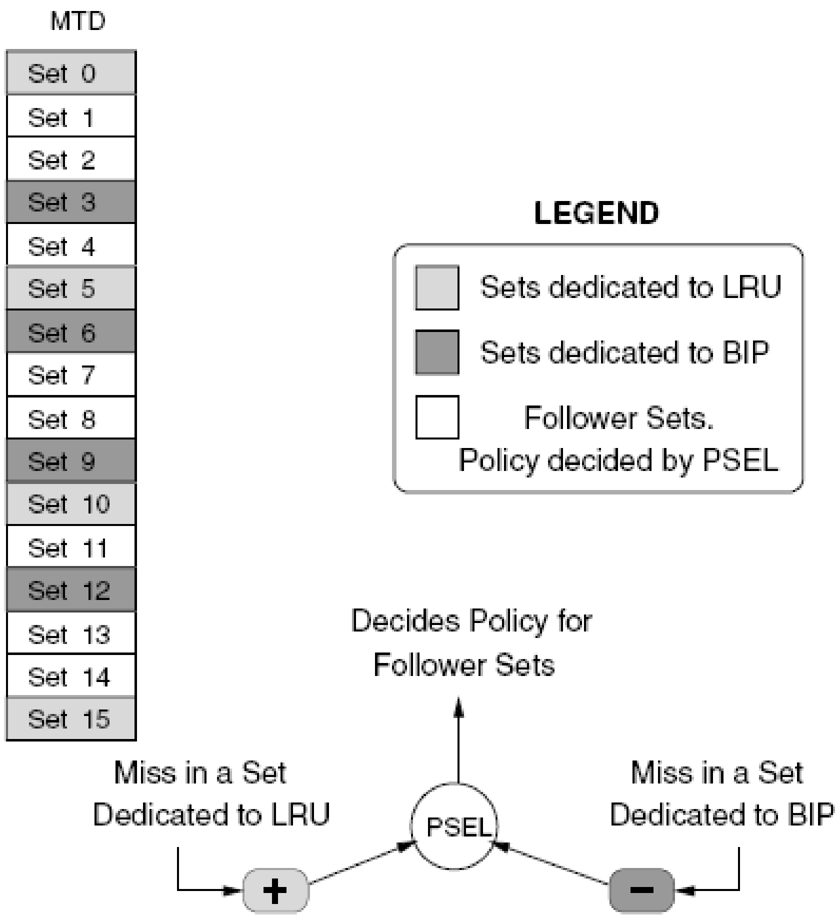
Simulated Replacement Policies

Adaptive Selection: *Set-Dueling [4]*

- Adaptive selection of the three policies will be implemented using Set-Dueling (proposed in [4]).

- Set-Dueling dedicates some sets in the cache (32-64 sets) for each policy.

- Misses occurring in the dedicated sets will be used to decide the selected policy for the rest of the cache (Follower sets) through a counter (PSEL).



Simulation Methodology

The Simulator

- The SimpleScalar toolset: sim-outorder simulator.
  - Sim-outorder models a superscalar processor with speculative execution support and two-level memory hierarchy.
  - Sim-outorder is the most detailed processor among the SimpleScalar toolset.

- Extensions to the SimpleScalar toolset provided by the SimFlex Project [2] to support MLP are used, those include:
  - Implementation of MSHRs (Miss Status Holding Registers).
  - A split-transactional bus that allows misses-under-misses.

- Execution-driven simulation.

Simulation Methodology

Simulated Benchmarks

- 5 SPEC SPU2000 benchmarks are used: ammp, art, bzip2, equake and parser.

- The PISA precompiled binaries are fed to the execution-driven simulator with their inputs.

Benchmark Name	Type	Compulsory Misses	Category
Ammp	FP	5.1%	Computational Chemistry
Art	FP	0.5%	Image Recognition/ Neural Networks
Bzip2	INT	15.5%	Compression
Equake	FP	14.2%	Seismic Wave Propagation Simulation
Parser	INT	20.0%	Word Processing

Simulation Methodology

Processor Specifications

Level-1 Instruction Cache	64KB; 64B line-size; 2-way with LRU replacement Policy. 1 cycle latency.
Level-1 Data Cache	64 KB; 64B line-size; 2-way with LRU replacement Policy. 1 cycle latency.
Level-2 Unified Cache	1 MB; 64B line-size; 16-way set associative 12 cycle latency 8-entry MSHR
Branch Predictor	Tournament predictor 7-cycle branch mis-prediction latency
Window Size	128
Instruction Fetch Queue Size	16
Decode/Issue/Commit Width	8 inst/cycle
Execution Units	4 Integer ALUs, 2 Integer Multiplier/Divider 2 floating point ALUs, 1 floating point Multiplier/Divider
Memory Latency	100 cycles

Simulation Methodology

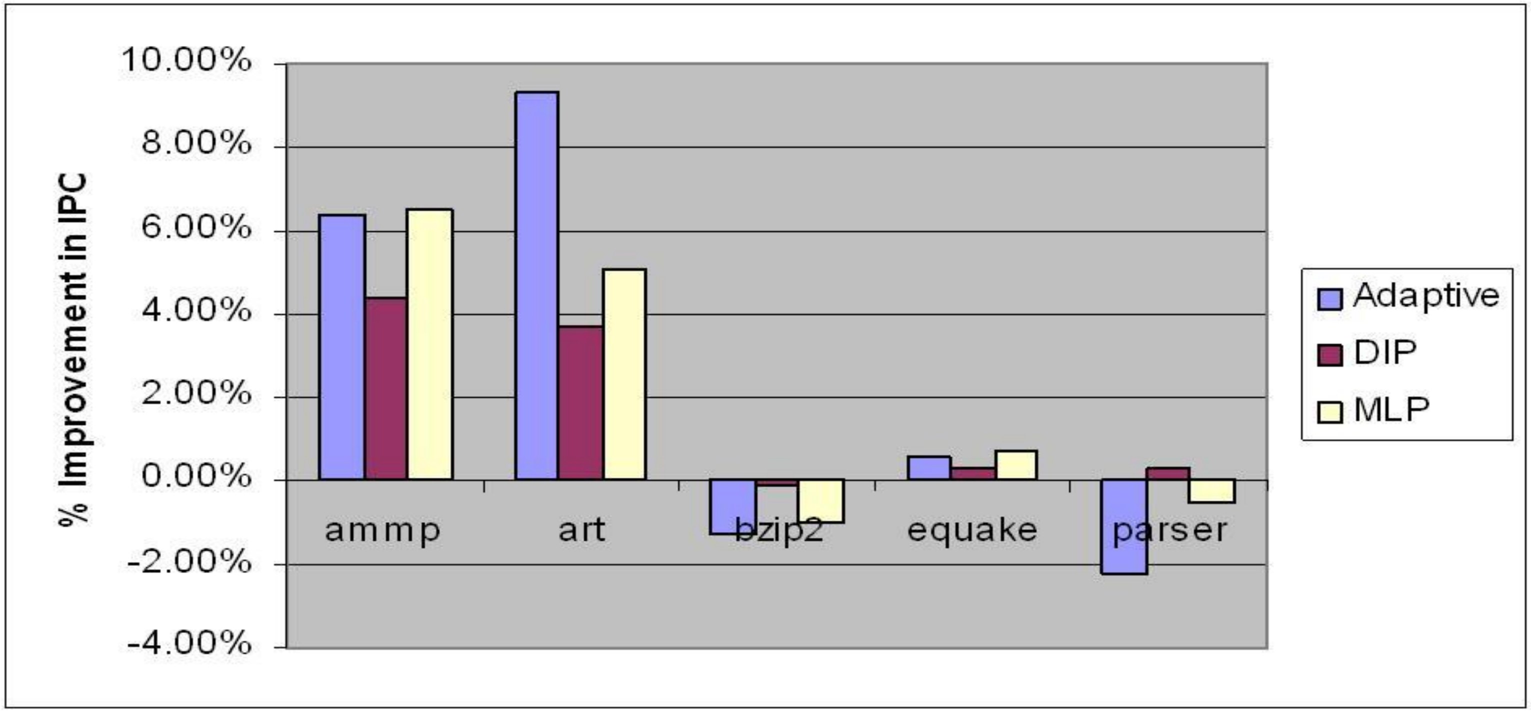
Simulation Run

- **Simulation of the three replacement policies on the 5 benchmarks was as follows:**

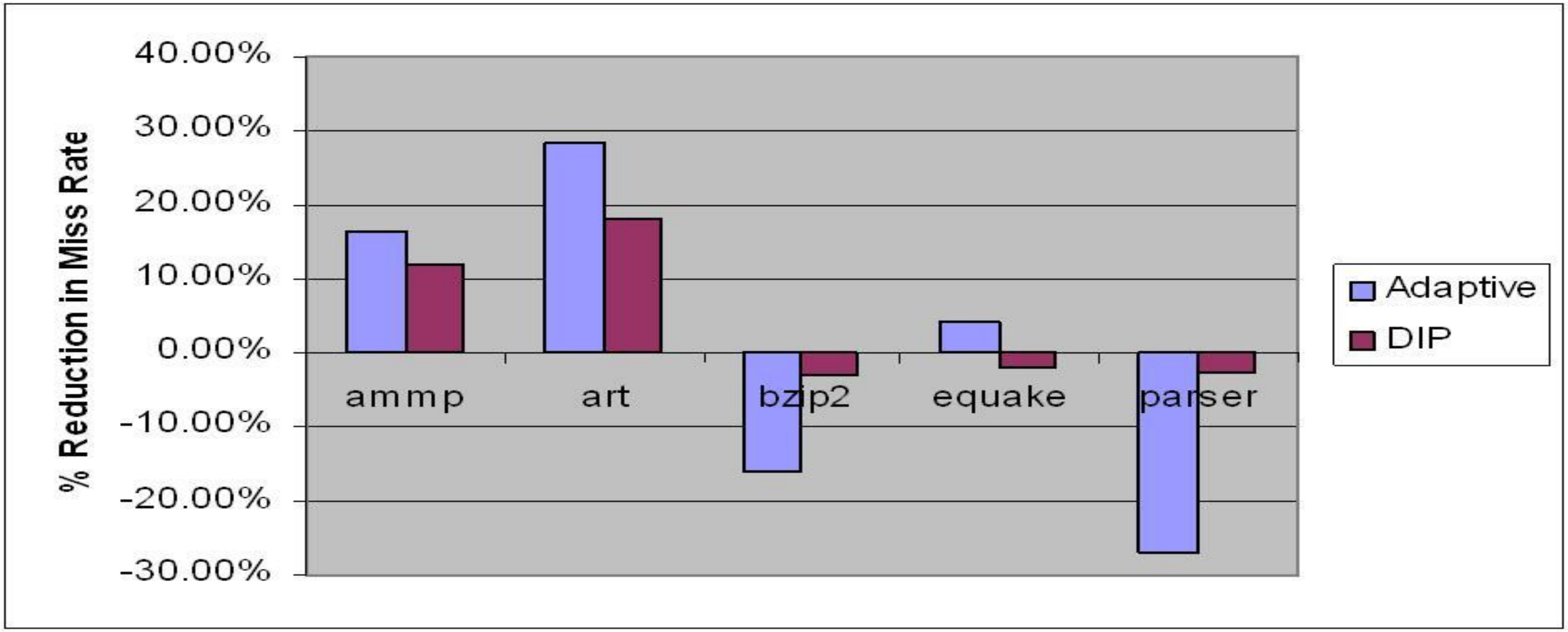
- ***Number of simulated instructions: 250 million.***
- ***Fast Forward interval: 50 million instructions.***

- **Since MLP improves cache performance via improving miss penalty (not miss rates), miss-rates results for MLP are not included.**

Simulation Results



Simulation Results



Discussion

DIP's Results

- DIP achieved better performance for *ammp* and *art*:
  - Memory-intensive workloads
  - DIP will be choosing BIP for these workloads most of the time.

- DIP maintains almost the same performance for LRU-friendly workloads: *bzip2*, *equake* and *parser*.
  - DIP will be choosing LRU for these workloads.
  - Slightly reduced performance for these workloads is due to moments where DIP is mistakenly choosing BIP.

Discussion

MLP's Results

- **MLP achieved better performance for *ammp* and *art*:**
  - Workloads having large number of parallel misses and close MLP-costs for successive misses.
  - MLP will be choosing LIN for these workloads.
  - Achieved improvement is not as much as that in Qureshi's et al paper [5] since values of delta are obtained dynamically during execution.

- **DIP maintains almost the same performance for LRU-friendly workloads: *bzip2*, *equake* and *parser*.**
  - DIP will be choosing LRU for these workloads.
  - Slightly reduced performance for LRU-friendly workloads is due to moments where MLP is mistakenly choosing LIN.

Discussion

Adaptive LRU-LFU Results

- Adaptive LRU-LFU policy achieved better performance for *ammp* and *art*:
  - These workloads have bad performance with the LRU policy.
  - LFU will be chosen for these workloads

- Results showed that the adaptive LRU-LFU policy achieved worse performance for LRU-friendly workloads: *bzip2* and *parser* .
  - Unexpected results: the adaptive policy should at least maintain approximately equivalent performance to LRU.
  - Results need to be revised.

Conclusion

- Adaptive replacement can dynamically choose the appropriate policy depending on the type of the workload:
- LRU-friendly workloads: LRU policy is used, thus maintaining almost the same performance as LRU.
- Other workloads that are not LRU-friendly such as *memory-intensive workloads* and *workloads with low temporal locality*: by choosing other replacement policies such as: *BIP[4]*, *LIN[5]* or *LFU[6]*. Thus, improving the performance over LRU.

References

[1] Austin, T., Larson E. and Ernst, D. (2002) *SimpleScalar: an infrastructure for computer system modeling*. IEEE Computer, pp 59-67.

[2] Falsafi B., Hoe J., Wenisch T. and Wunderlich R. (2004) *SimFlex: Fast, Accurate and Flexible Simulation of Computer Systems*. ACM SIGMETRICS Performance Evaluation Review (PER), Vol. 31, No. 4.

[3] KleinOsowski AJ., Flynn J., Meares N. and Lilja D. (2001) *Adapting the SPEC 2000 Benchmark Suite for Simulation-based Computer Architecture Research*. Workload Characterization of Emerging Computer Applications, pp. 83-100.

[4] Qureshi M., Jaleel A., Patt Y., Jr. S. & Emer J. (2007). *Adaptive Insertion Policies for High Performance Caching*. Proceedings of the 34th annual international symposium on Computer architecture (ISCA'07), pp. 381-391.

[5] Qureshi M., Lynch D., Mutlu O. & Patt Y. (2006). *A Case for MLP-Aware Cache Replacement*. Proceedings of the 33th annual international symposium on Computer architecture (ISCA'06). pp. 167-178.

[6] Subramanian R., Smaragdakis Y. & Loh G. (2006). *Adaptive Caches: Effective Shaping of Cache Behavior to Workloads.* Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture (Micro'06), pp. 385-396.