University of Jordan
Computer Engineering Department

Performance Evaluation of Recently Proposed Cache Replacement Policies

CPE 731: Advanced Computer Architecture
Dr. Gheith Abandah

Asma Abdelkarim

January 19, 2010

Abstract

Recently proposed cache replacement policies tries to reduce the miss rates for level-2 caches in order to reduce long stalls due to accesses to the lower levels in the memory hierarchy. Three of the most important recently proposed replacement policies are: the Dynamic Insertion Policy (DIP), Memory-Level-Parallelism (MLP) Aware Replacement Policies and the Adaptive Replacement Policy combining two of the original replacement policies (LRU, LFU). In this simulation experiment, these policies are simulated for 5 of the SPEC CPU 2000 benchmarks. In general, adaptive replacement policies show the ability of improving the performance of L2 caches for workloads that have bad LRU-performance while maintaining approximately equivalent performance for LRU-friendly workloads.

1. Introduction

The need for better miss rates at the lower-level caches in the memory hierarchy led to the search for new optimized replacement policies. Many of the recently proposed policies depend on tracking the behavior of the workload being executed and provide the policy that best suites it from two of specified policies, these are called adaptive replacement policies. However, the lack of unified simulation environment for the recently proposed policies prevents accurate performance evaluation and comparison. This simulation experiment provides a unified simulation for three of these policies: DIP (Dynamic Insertion Policy), MLP (Memory Level Parallelism)-aware replacement policies and the Adaptive (LRU-LFU) replacement policies.

The rest of this report is organized as follows: section 2 provides an overview of the simulated replacement policies. Section 3 describes the simulation methodology: the used simulator, workloads, and processor specifications. Section 4 represents the simulation results provided both as tables and bar charts for ease of comparison. Section 5 provides discussion and analysis of the obtained results. Finally, a conclusion for the simulation experiment is provided.

2. Simulated Uniprocessor Replacement Policies

Three of the recently proposed replacement policies for the L2 cache are simulated. The adaptive selection for all policies is implemented using the Set-dueling mechanism proposed in [4]. These policies are:

2.1 Dynamic Insertion Policy (DIP) [4]

In [4], Qureshi et al. proposed their DIP replacement policy which adaptively chooses the appropriate policy to be applied to the cache from either LRU or BIP (Bimodal Insertion Policy). BIP prevents thrashing in case of memory-intensive workloads, while LRU has excellent performance for workloads with high temporal locality and workloads whose working sets fit in the cache size.

In order to choose the appropriate policy, DIP reserves portion of the sets (32 sets) as dedicated sets for each policy (LRU-BIP) in order to keep track of the policy that is performing better so far, this mechanism is called set-dueling. Set-dueling uses a saturating counter that indicates the policy that is incurring higher miss rates in the dedicated sets. Thus DIP is expected to achieve better performance than LRU for memory-intensive workloads while maintaining similar performance for LRU-friendly workloads.

2.2 Memory-Level-Parallelism (MLP) Aware Replacement Policies [5]

In [5], Qureshi et al. proposed exploiting Memory-Level-Parallelism (MLP) to reduce the miss penalty to the memory, not the miss rate, by producing the notion of the MLP-aware replacement policy. Their proposal was based on the fact that cache misses do not occur uniformly across the workload; which means that some misses occur in parallel and others occur in isolation. This means that different misses to the blocks of the cache will differ in their exploitation of MLP.

Making the replacement policy aware of MLP means that misses that occur in isolation are favored over misses that occur in parallel. This is done by assigning MLP costs to the individual blocks and depending on these costs along with the recency of the block to decide the victim block on the next miss. Qureshi et al. called this policy the linear (LIN) policy. This policy provides performance improvements for workloads that have close MLP costs for successive misses. However, this is not the case for all workloads. For that, Qureshi et al. proposed adaptive selection between LIN and LRU to maintain at least equivalent performance for workloads that cannot benefit from MLP.

2.3 Adaptive Insertion Policy of LRU and LFU [6]

In [6], Subramanian et al. proposed an adaptive policy that dynamically chooses one of two policies from the well-known policies (LRU, LFU, FIFO, Random) to be applied. In this simulation project, the adaptive policy is implemented for LRU and LFU. In their proposal, Subramanian et al. used the Sampling Based Adaptive Replacement which uses auxiliary tag directories for one of the policies and dedicates sets from the cache for the other policy. In our simulation project, Set-dueling is used where for both policies dedicated sets are used.

3. Simulation Methodology

3.1 Simulator

The replacement policies mentioned in the previous subsection are simulated using the execution-driven SimpleScalar toolset. SimpleScalar is a set of simulators that vary in the level of details that they provide. The most detailed simulator among the SimpleScalar simulators, which is the one used for this simulation experiment, is sim-outorder. Sim-outorder models a superscalar processor with speculative execution support and two-level memory hierarchy. It provides the ability of tuning several detailed design parameters and observing their impacts on the performance, represented in IPC, miss ratios, latency of individual operations… Sim-outorder provides this detailed simulation at the expense of longer simulation time. [1]

In the execution-driven simulation, the workload to be simulated is provided along with the inputs on which it must be executed. SimpleScalar supports the following instruction sets: Alpha, PISA, ARM and x86. The PISA instruction set (the Portable Instruction Set Architecture) is a simple MIPS-like instruction set which is developed for the SimpleScalar toolset. [1]

In order to simulate the MLP-aware replacement policy, extensions provided by the SimFlex Project [2] are used. The SimFlex project includes several extensions to the original SimpleScalar simulator. Among these extensions is the support for memory-level parallelism through MSHRs and a split-transactional bus which allow misses-under-misses to occur and provide the possibility for serving misses in parallel as long as the MSHR registers are not full.

3.2 Benchmarks

In this simulation project the PISA precompiled binaries for 5 SPEC CPU2000 benchmarks are simulated along with their inputs. The 5 benchmarks are selected so that their compulsory misses do not form more than 50% of the total number of misses, in order to make sure that they will make use of optimizations in the replacement policy [4][5]. Table 1 shows the selected benchmarks and the percentage of compulsory misses and category for each benchmark.

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

Table-1: Simulated Benchmarks
(Category column [3], Compulsory misses column [4][5])

3.3 Configuration

The SimpleScalar toolset is extended to include the additional three replacement policies: DIP, MLP-aware and the LRU-LFU adaptive replacement policies. To achieve that, the following files in the SimpleScalar toolset are modified: cache.c, cache.h and sim-outorder.c.

Table 2 shows the specifications of the simulated processor.

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**

Table-2: Simulated Processor's Specifications

3.4 Simulation Run

Running the SPEC SPU2000 benchmarks with their reference input takes several days to weeks to complete. Because of that, the number of simulated instructions in each benchmark is limited to 250 M instruction.

Moreover, a fast forward interval of 50 M instructions is included to make sure that the caches are stable and correct results will be obtained.

The command used to run the sim-outorder simulator for the above processor configuration is as follows:

/path/*Sim-outorder –fastfwd 500000000 –max:inst 250000000 –redir:*output_file.txt* –cache:il1 il1:512:64:2:l –cache:dl1 dl1:512:64:2:l –cache:il2 dl2 –cache:dl2 dl2:2048:64:8:*Tested_Rep_Policy /path/Benchmark_Binary* < */path/input_file

4. Simulation Results

Tables 3 and 4 show the simulation results for the five benchmarks in terms of miss rates and IPCs. Figures 1 and 2 show the results represented in bar charts.

For the MLP-aware policy only the IPC (Instructions per Clocks) is measured, since the MLP-aware policy aims to improve the performance by reducing the miss penalty not the miss rate.

Benchmark	LRU miss rate	DIP miss rate	Adaptive (LRU-LFU) miss rate
ammp	0.9910	0.8713	0.8181
art	0.4281	0.3503	0.3062
bzip2	0.1546	0.1552	0.1798
equake	0.1302	0.1329	0.1247
parser	0.1528	0.1569	0.1946

Table-3: Miss Rates results for the five benchmarks for LRU, DIP and the (LRU-LFU) Adaptive replacement policy

Benchmark	LRU IPC	DIP IPC	Adaptive (LRU-LFU) IPC	MLP IPC
ammp	0.2040	0.2129	0.2171	0.2171
art	0.4890	0.5070	0.5347	0.5144
bzip2	0.9801	0.9796	0.9677	0.9700
equake	2.7371	2.7440	2.7538	2.7558
parser	1.7266	1.7317	1.6883	1.7182

Table-4: IPC results for the five benchmarks for LRU, DIP, MLP and the (LRU-LFU) Adaptive replacement policy

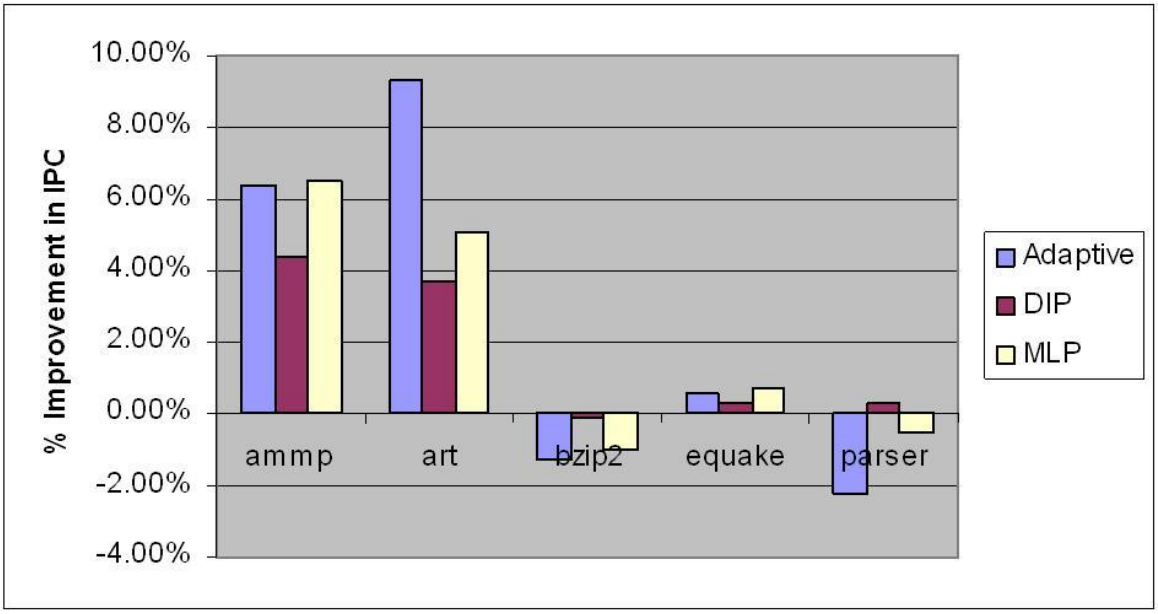

Figure-1: Bar-chart of the IPCs for the five benchmarks for MLP, DIP and (LRU-LFU) Adaptive replacement policy

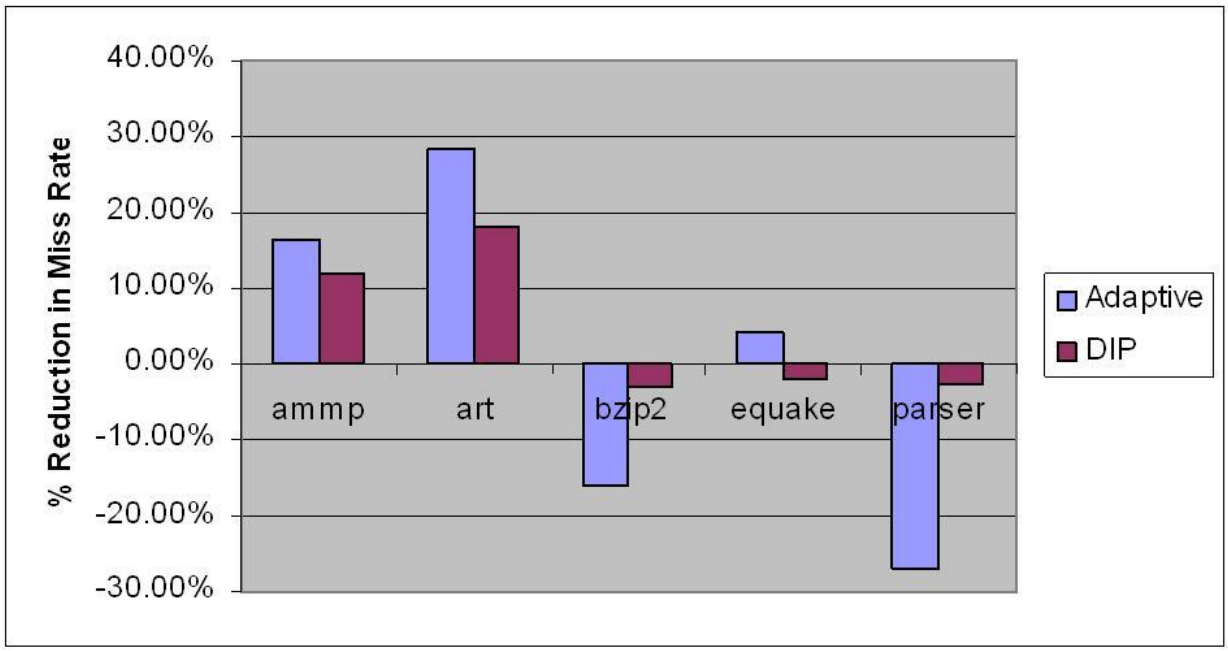

Figure-2: Bar-chart of the miss rates for the five benchmarks for DIP and the (LRU-LFU) Adaptive replacement policy

5. Discussion

For the MLP-aware replacement policy, the results are as expected. The benchmarks ammp and art, has a lot of misses that occur in parallel that can make use of making the replacement policy aware of MLP. However, the amount of improvement is not as much as that in Qureshi et al.'s paper [5], since in their proposal MLP costs are estimated based on delta values that are obtained from static runs of the workloads. In this simulation experiment, delta values are computed and averaged dynamically as misses occur in the workload which produces less accurate MLP-costs.

Other replacement policies (bzip2, equake and parser) do not make use of MLP either because most of their misses are isolated or because they have significantly varying MLP costs among the successive misses. However, their performance is only slightly degraded since the adaptive selection between LIN and LRU will select LRU for most of the time which guarantees almost identical performance to LRU. This slight degradation in the performance is caused by the time intervals where LIN is mistakenly used over LRU.

For both ammp and art, DIP has better performance than LRU. ammp is a memory intensive workload in some phases of its operation. For these phases, DIP will select BIP to be used while keeping on LRU for the LRU-friendly phases, thus improving the performance. art is a memory intensive workload in all phases of its operation, DIP will be using BIP all the time. By keeping fraction of the working set in the cache, BIP prevents thrashing for art, thus improving the performance over LRU. bzip2, equake and parser are all LRU-friendly workloads, DIP maintains almost equivalent performance for these workloads as DIP will be selecting LRU to be applied since it has the better performance.

Similarly, LRU-LFU adaptive replacement policy achieves performance improvements for both ammp and art which have bad performance for LRU. However, it is expected that the adaptive policy must at least maintain equivalent performance for LRU-friendly benchmarks (bzip2, equake and parser). This is not the case in these simulation results, which indicates that some error is occurring when selecting the replacement policy (LRU-LFU) that must be revised.

6. Conclusion

In this simulation experiment five SPEC SPU2000 benchmarks were simulated for three of the recently proposed replacement policies. The benchmarks are: ammp, art, bzip2, equake and parser. The replacement policies are: MLP-aware, DIP and Adaptive (LRU-LFU) insertion policy.

The results showed that adaptive policies can significantly improve the performance of the L2 cache for memory intensive workloads for which LRU has bad performance. Each of the simulated replacement policies has its own way in improving performance for these workloads. What makes adaptive policies appealing is that they maintain approximately equivalent performance for LRU-friendly workloads while achieving this improvement.

The MLP-aware replacement policy and DIP use distinct approaches in improving the performance of the caches; the MLP-aware replacement policy improves miss penalty by exploiting memory level parallelism while DIP improves the miss rate by preventing thrashing of the cache. Combining these two ideas may combine the improvements of these two replacement policies to achieve even more and more performance improvement. Exploring the effect of a combining MLP and DIP is part of my future work on this topic.

7. 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.**