International Journal of Networking and Computing – www.ijnc.org, ISSN 2185-2847
Volume 10, Number 2, pages 200–212, July 2020

A Cache Replacement Policy with Considering Fluctuation Patterns of Total Priority Value

Jubee Tada
Graduate School of Science and Engineering, Yamagata University
Yonezawa, Yamagata, 992-8510, Japan

and

Ryosuke Higashi
Faculty of Engineering, Yamagata University
Yonezawa, Yamagata, 992-8510, Japan

Received: February 13, 2020
Revised: May 3, 2020
Accepted: May 25, 2020
Communicated by Hideharu Amano and Minoru Uehara

Abstract

In recent computer systems, a high-performance memory system is required for improving the computing performance. To improve the performance of the memory system, the capacity and the associativity of cache memories tend to be enlarged in all the memory hierarchy. Therefore, a high-performance cache replacement policy is required, which can be implemented with a small overhead and can adapt the behavior of various applications.

Under such a situation, the adaptive demotion policy with considering global fluctuations of priority values (ADP-G) is proposed. ADP-G is based on the adaptive demotion policy (ADP) which can adapt the behavior of various applications by changing its insertion, promotion, demotion and selection (IPDS) policies. ADP-G selects the appropriate IPDS policies from two alternatives, one is suitable for a temporal locality and the other has a scan/thrashing tolerant. For this selection, ADP-G uses the global fluctuations of priority values of all the blocks in the cache. Although ADP-G is a high-performance cache replacement policy as compared with conventional ones, it causes a problem when the performance of the scan/thrashing tolerant IPDS policies is higher than that of the temporal locality IPDS policies.

This paper discusses a novel cache replacement policy to solve this problem. At first, the fluctuation patterns of total priority value in several benchmarks are analyzed. As a result, two characteristics are observed which are useful to detect that the running application the suitable for the temporal locality IPDS policies or the scan/thrashing tolerant IPDS policies. Based on this analysis, a cache replacement policy with considering fluctuation patterns of total priority value is proposed, and it is called ADP-P. ADP-P has two states which suitable for the temporal locality and the scan/thrashing tolerant. To detect the suitable state, ADP-P uses the total of the priority values of all the blocks in the cache, and the value is stored in the global total register (GTR). When the state is for the temporal locality, the state is changed to the scan/thrashing tolerant state if the reduction of GTR is larger than the threshold value. When the state is for the scan/thrashing tolerant, the state is changed to the temporal locality state if the fluctuation of GTR which is larger than the threshold value is not caused for a fixed number of cache accesses.

Experimental results show the proposed cache replacement policy achieves a reduction of cache misses compared to LRU policy and ADP-G. When the L3 cache size is 4MB, in the

geometric mean of all the benchmarks, the proposed cache replacement policy achieves a 6.0% and 11% MPKI reduction compared to ADP-G and LRU policy, respectively.

Keywords: Memory Hierarchy, Cache Memory, Cache Replacement Policy

1 Introduction

In recent multi-core processors, there are a large number of cores and these cores share the last-level cache (LLC). As increase the number of cores, it is needed that the cache size and the associativity of LLC is enlarged to reduce the number of memory accesses. Also, to reduce the accesses to LLC, the cache size and associativity of the local caches in the core tend to be enlarged. Therefore, a high-performance cache replacement policy is required, which can be implemented with a small overhead and adapt to a behavior of various applications.

Under such a situation, the adaptive demotion policy with considering global fluctuations of priority values (ADP-G) has been proposed [1]. ADP-G is based on the adaptive demotion policy (ADP) [2] which can adapt a behavior of various applications by changing its insertion, promotion, demotion and selection (IPDS) policies. ADP-G selects the appropriate IPDS policies from two alternatives, one is suitable for temporal locality and the other has scan/thrashing tolerant. ADP-G uses the global fluctuations of priority values in the cache for this selection. Although ADP-G achieves a high-performance as compared with conventional cache replacement policies, it has a problem when the performance of the scan/thrashing tolerant policies is higher than that of the temporal locality policies [1].

To solve this problem, this paper proposes a cache replacement policy with considering fluctuation patterns of total priority value. At first, the fluctuation patterns of total priority value are analyzed. Based on these analyses, a novel scheme which can select the appropriate IPDS policies is proposed. The proposed scheme selects the appropriate IPDS policies based on the fluctuation patterns of total priority value, and the cache replacement policy which adopts this scheme is called ADP-P. The performance of ADP-P is evaluated by the processor simulator.

This paper is organized as follows. Section 2 outlines ADP and ADP-G. In Section 3, fluctuation patterns of total priority value are analyzed, and the outline and behavior of ADP-P are described. Section 4 evaluates the performance of ADP-P. Section 5 concludes this paper.

2 Related work

Adaptive Demotion Policy (ADP) is a cache replacement policy which focuses on the demotion of the priority values at a cache miss [2]. ADP prepares 2-bit priority values for each cache block. The priority value of a block is promoted when cache hits occur in that block. The behavior at a cache miss is as follows. First, the block that has the smallest priority in the accessed set is selected as the victim. Next, the priority values of all the blocks in the accessed set are decreased by the subtract value. Finally, a new block coming from the lower-level memory hierarchy is stored.

The behavior of the ADP is decided by the insertion, promotion, demotion and promotion (IPDS) policies. The insertion policy decides the initial priority of the incoming block. A large initial priority value suits the temporal locality. In contrast, a small initial priority value is appropriate for a scan access pattern. The promotion policy decides how to promote the priority of the hit block. Mainly the policy is selected from two alternatives: Hit Priority (HP) and Frequently Priority (FP). The HP policy promotes the priority to the maximum as like the LRU policy. It is suitable for the temporal locality. The FP policy increases the priority by one as like the Least Frequently Used (LFU) policy. Because the block which is not re-referenced tends to have a lower priority, the FP policy is suitable for a scan access pattern. The demotion policy decides how to decrease the priority values of the blocks in the accessed set at a cache access. To solve the problems which caused by the exist cache replacement policies, ADP dynamically calculates the subtraction value with based on the average of the priority values of the accessed set [2]. The selection policy selects a victim from the blocks

[0]This is an abstract footnote

which has the smallest priority value. Two selection policies are considered: The Left-side Selection (LS) policy and the Random Selection (RS) policy. The LS policy selects the block which has the lowest way number as a victim, and the RS policy selects the block as a victim at random. It is considered that the LS policy is suitable to the temporal locality, and the RS policy is suitable for the scan/thrashing tolerant because it prevents evicting of the useful block by scan/thrashing access pattern with a certain probability.

At every cache access, the priority values of the blocks in the accessed set are updated by the ADP controller [2]. The detail of the ADP controller is as follows. At a cache hit, the ADP controller increases the priority value of the hit way by using the priority value of the block and the hit way information. This increase is done based on the promotion policy, and the value is not increased above the maximum value. The increased value is outputted as a new priority value of the hit block. At a cache miss, the ADP controller calculates the subtraction value by the priority values of the blocks in the accessed set, and the priority values are reduced by the subtract value. this calculation is done based on the demotion policy. At this time, the values are not reduced below the minimum value. Then, these values are outputted as a new priority values of the accessed blocks excluding the victim block. In addition, the ADP controller selects one block from the blocks which have the smallest priority as a victim based on the selection policy. The incoming block is stored instead of the victim block, with the initial priority value which is set by the insertion policy. Also, the ADP controller can output the difference of the previous priory values and the new priority values. This difference is useful to calculate the total priority value of the all the blocks in the cache.

ADP can suit for any characteristics of applications by selecting appropriate IPDS policies. As mentioned above, a small initial priority value, the FP policy, and the RS policy is suitable for the scan/thrashing tolerant. On the other hand, a large initial priority value, the HP policy, and the LS policy is suitable for the temporal locality. Hence, if it is possible to detect the characteristics of the running application and select an appropriate IPDS policies, the performance of ADP will be increased. In order to dynamically change IPDS policies, D-ADP (Dynamic-ADP) is proposed [2]. D-ADP uses the set-dueling method [3] to select the appropriate policies from two alternatives. In D-ADP, aggressive policies are hardly to adopt as these two alternatives because the extreme policies sometime cause significantly increase of the cache misses. Under such a situation, ADP with considering Global Fluctuations of priority values (ADP-G) [1] is proposed.

Figure 1 shows the outline of ADP-G. ADP-G selects appropriate IPDS policies from two alternatives, one is suitable for temporal locality (TL) and the other has scan/thrashing tolerant (ST). The global total register (GTR) is prepared to store the total of the priority values of all blocks in the cache. Also, several partial total registers (PTRs) (in Figure 1, the number of PTRs is four) are prepared to store the total of the priority values of partitions which consists of contiguous sets in the cache. The values of all the registers are updated at every cache access. In addition, the previous value registers (preGTR and prePTRs) are prepared, and the values of GTR and PTRs are saved to preGTR and prePTRs at regular intervals, respectively. At regular intervals, the behavior of the running application is detected by comparing the values of recent GTR and PTRs to preGTR and prePTRs.

The values of GTR and PTRs are updated at every cache access. As mentioned above, the ADP controller can output how much to increase or decrease these registers. By using the value, GTR and PTRs can store the total of the priority values of all blocks in the cache and the total of the priority values of partitions, respectively.

The detail of the method is as follows. There are four states to select the IPDS policies. State 1 and 2 (TL states) output the policies suitable for the temporal locality, and State 3 and 4 (ST states) output the policies suitable for scan/thrashing access. The behavior at State 1 is as follows. At regular intervals, GTR and preGTR are compared. If the decrease of GTR is larger than the threshold value, PTRs and prePTRs are compared. If all the PTRs are decreased, the state is change to State 2. The behavior of State 2 is same as that of State 1 except the next state is State 3. The behavior of State 3 is as follows. At regular intervals, GTR and preGTR are compared. If the increase is larger than the threshold value, the state is change to State 4. At State 4, the next state is State 1 and the other behavior is same as that of State 3.

ADP-G can achieve high-performance and be implemented with small hardware resources. In

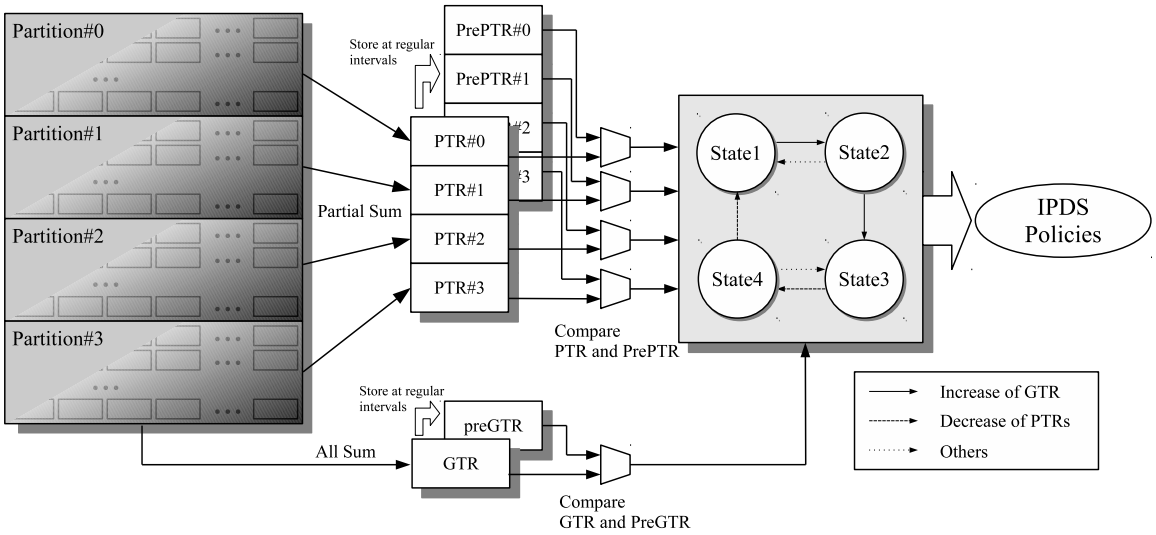


Figure 1: Outline of ADP-G

Table 1: the number of misses of ADP-G (2MB L3 cache)

Benchmarks	$M_{1/32}$	$M_{1/64}$	Ratio
bzip2_chicken	290312	436313	1.50
bzip2_combined	503921	454368	0.902
bzip2_source	517044	475176	0.919
perlbench_splitmail	649849	818502	1.26
zeusmp	3766401	4247381	1.13

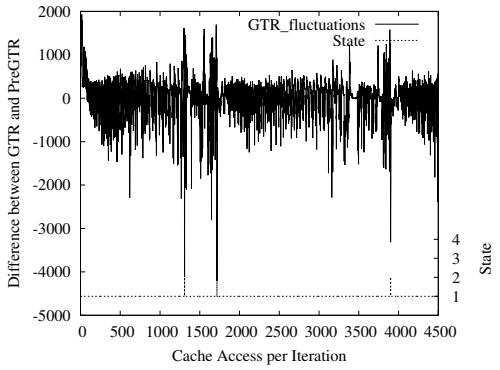
addition, ADP-G does not cause a significantly performance drop at a specific application. However, ADP-G does not select the appropriate IPDS policies in several applications, especially when the performance of the scan/thrashing tolerant policies is higher than that of the temporal locality policies. Therefore, the performance of ADP-G is lower than that of the D-ADP when the cache size is large. To solve this problem, this paper proposes a novel IPDS policies selection scheme which focuses on the fluctuation patterns of total priority value.

3 A cache replacement policy with considering fluctuation patterns of total priority value

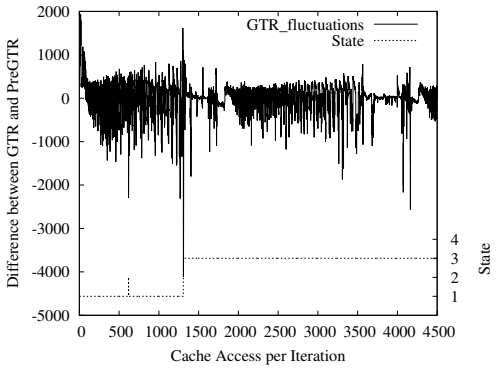
3.1 Analyses on fluctuation patterns of total priority value

ADP-G selects appropriate IPDS policies from two alternatives. For this selection, ADP-G uses the global fluctuations of priority values of all the blocks in the cache. ADP-G has two threshold values; *TH* and *TL*. *TH* is the threshold value for State 1 and State 2. *TL* is that for State 3 and State 4.

Table 1 shows the number of cache misses of ADP-G with two *TH*, and the ratio of these. In this experiment, ADP-G is adapted to 2MB L3 cache. The other experimental environments are the same as described in Section IV. When *TH* is small, ADP-G will easily change its state from State 1 or 2 to State 2 or 3. $M_{1/32}$ illustrates the number of cache misses when the *TH* is the quotient of the maximum value of GTR divided by 32. $M_{1/64}$ illustrates the number of cache misses when the *TH* is the quotient of the maximum value of GTR divided by 64. As shown in Table 1, when *TH* is small, cache misses are reduced in several benchmarks, but several benchmarks cause a cache miss

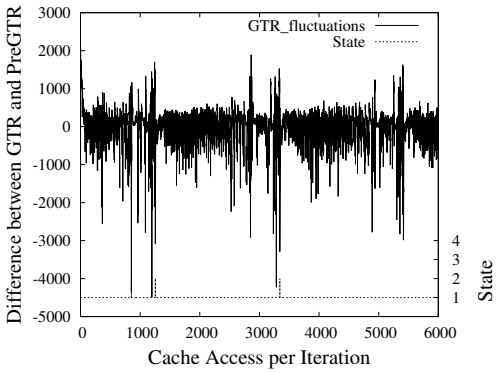


(a) TH=1/32

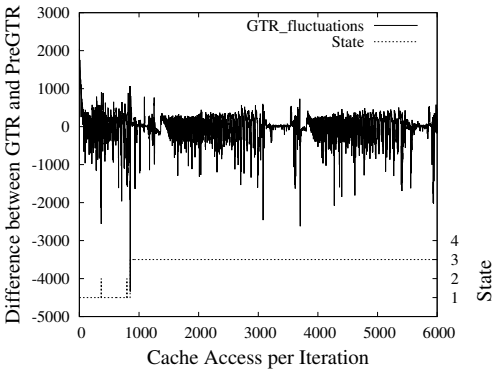


(b) TH=1/64

Figure 2: Fluctuation of GTR (bzip2_combined)

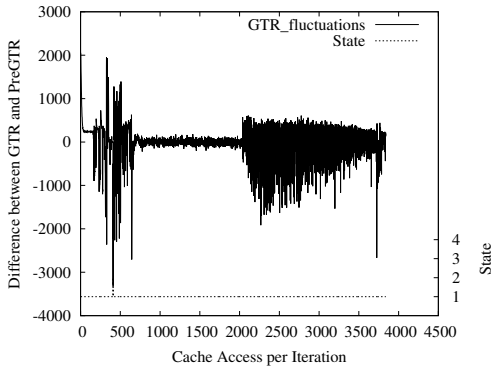


(a) TH=1/32

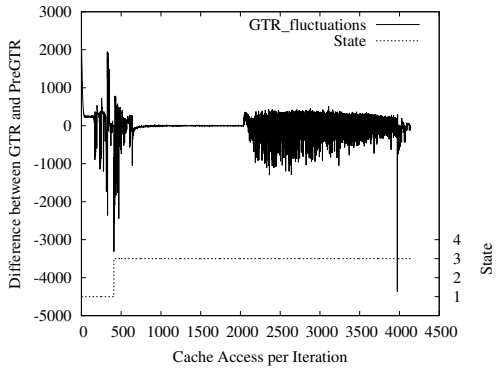


(b) TH=1/64

Figure 3: Fluctuation of GTR (bzip2_source)

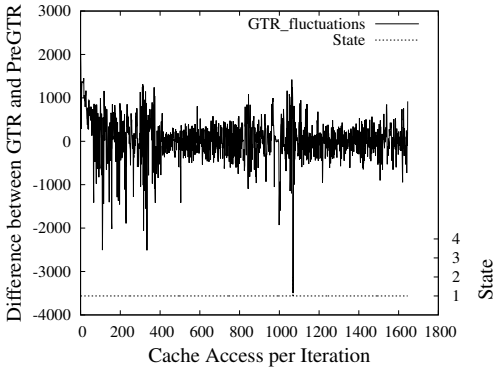


(a) TH=1/32

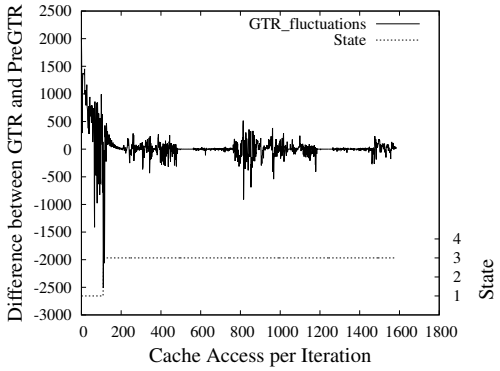


(b) TH=1/64

Figure 4: Fluctuation of GTR (bzip2_chicken)

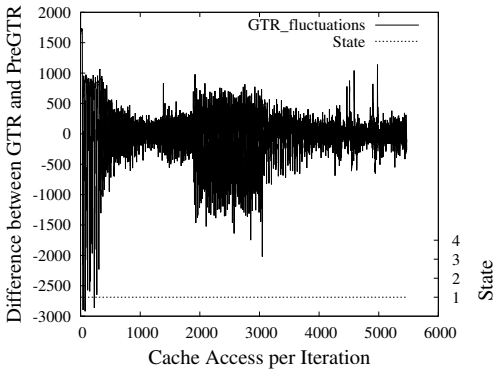


(a) TH=1/32

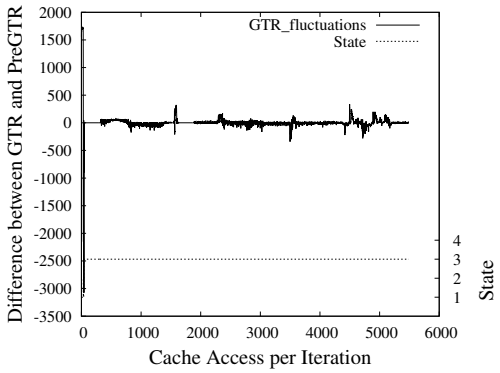


(b) TH=1/64

Figure 5: Fluctuation of GTR (perlbench_splitmail)



(a) TH=1/32



(b) TH=1/64

Figure 6: Fluctuation of GTR (zeusmp)

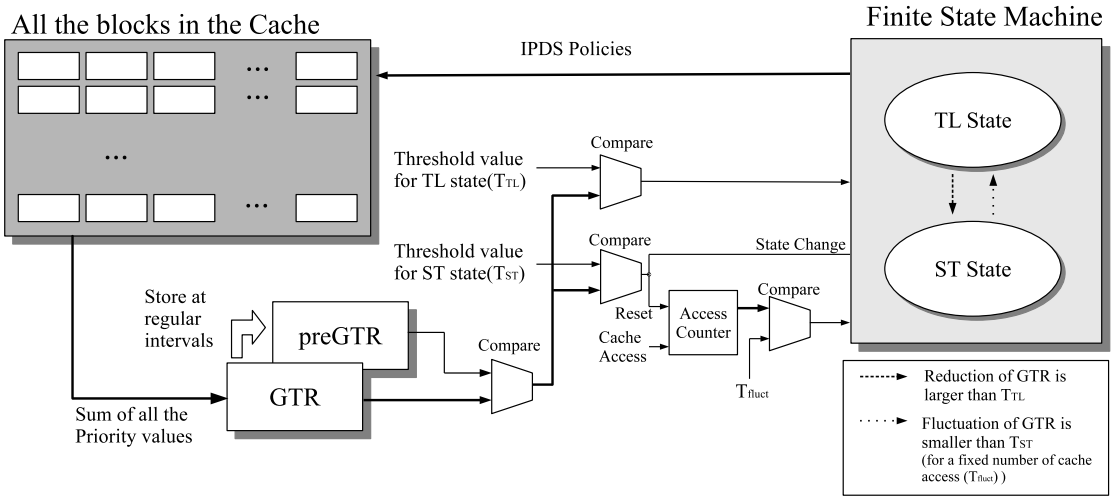


Figure 7: Outline of ADP-P

increase.

Figures 2 to 6 shows the fluctuation of GTR and State on each benchmark. In these figures, the horizontal axis illustrates the quotient of the number of cache accesses by the interval. Figure 2 shows the results of bzip2_combined. In this benchmark, $M_{1/64}$ is smaller than $M_{1/32}$, so it is thought that ST states is suitable for this benchmark. However, as shown in Fig. 2, the state is almost TL states during the simulation when *TH* is 1/32. This is because that the value of GTR is significantly fluctuated, and it causes the transition to TL states even if the transition to the ST states is occurred. Although this problem can be solved if a small *TH* is adopted, it will impact the performance of the other applications. Therefore, a novel scheme to change the state from TL states to ST states is required. As shown in this figure, the value of GTR fluctuates significantly when the *TH* is 1/32. As shown in Figure 3, this tendency is also seemed in bzip2_source which is suitable for ST states. Therefore, it is assumed that the change TL states to ST states will reduce cache misses when the fluctuation of GTR is large.

Figure 4 shows the results of bzip2_chicken. These benchmarks suit for TL states because $M_{1/32}$ is smaller than $M_{1/64}$. However, as shown in Fig. 4, the state is almost ST states during the simulation when *TH* is 1/64. this is because that the fluctuation of GTR is slightly, and it does not cause the transition to TL states. Therefore, a novel scheme to change the state from ST states to TL states is required. In this figure, there is the area where the fluctuation of GTR is slightly. As shown in Figures 5 and 6, this area is also seemed in the other benchmarks which are suitable for TL states such as perlbench_splitmail. Such areas do not appear in Figures 2 and 3. If these areas are appeared, it is thought that the change ST states to TL states will cause a reduction of cache misses.

3.2 ADP with considering fluctuation patterns of total priority value

Figure 7 shows the proposed scheme which is based on ADP. The basically behavior of the ADP-P is the same as ADP. Based on the IPDS policies, the initial priority value of the incoming block is set, the priority value of the hit block is promoted at cache hit, the priority values of the accessed set is demoted at a cache miss, and one block which has the smallest priority value is selected as a victim. The same as ADP-G, GTR and preGTR are prepared, and the behavior of the running application is detected by comparing the values of recent GTR to preGTR at regular intervals. There are two states; one is TL (temporal locality) state and the other is ST (scan/thrashing tolerant) state. Each state outputs appropriate IPDS policies for that state. Considering the analyses on the fluctuation patterns of total priority value, the conditions to change state is decided. The detail of the conditions are as follows. When the state is TL State, the state is changed to ST state if the reduction of GTR

Table 2: Cache Parameters

cache	size	assoc.	latency
Level1Instruction (L1I)	32KB	8	1
Level1Data (LID)	32KB	8	1
Level2 (L2)	256KB	8	8
Level3 (L3)	2MB/4MB	16	20
Main Memory	4GB	-	150

Table 3: IPDS policies for ADP-P

	TL states	ST States
Insertion	I2 (10)	I0 (00)
Promotion	HP (maximize)	FP (increase by 1)
Demotion	HOA (half of average)	HOA (half of average)
Selection	LS (Leftside)	RS (Random)

is larger than the threshold value. When the state is ST state, the state is changed to TL state if the fluctuation of GTR which is larger than the threshold value is not caused for a fixed term. The threshold value for TL state is illustrated as T_{TL}, and the threshold value for ST state is illustrated as T_{ST}. The fixed term is expressed by the number of cache accesses, and illustrated as T_{fluct}. The cache replacement policy which adopts this scheme is called ADP-P, which means ADP with considering fluctuation patterns of total priority value.

4 Evaluation of ADP-P

The simulation experiments are performed to show the performance of ADP-P. For these experiments, the simulator including ADP-P is developed based on the gem5 simulator system with Alpha 21264 instruction [4]. The benchmarks included in the SPEC CPU2006 [5] are examined as workloads for the experiments. For each benchmark, a representative phase of one billion instructions is extracted from the full execution by SimPoints [6], and the extracted phase is used for the evaluations. In each simulation, the first 50M instructions are used for warming up, and the following 950M instructions are for obtaining the simulation results. Table 2 shows cache parameters used in the simulation. In all the experiments, the L3 cache adopts the experimental replacement policy. The other caches adopt LRU policy. In all the caches, the block size is 64 bytes.

Table 3 shows two IPDS policies for ADP-P, one is for TL states and the other is for ST states. In all the experiments, Half of Average (HOA) is selected as the demotion policy because of HOA achieves a high performance in any cache hierarchy excluding L1I cache [2]. As shown in Section 2, A large initial priority value, the HP policy, and the LS policy is suitable for the temporal locality. A small initial priority value, the FP policy, and the RS policy is suitable for the scan/thrashing tolerant. Therefore, in the experiments of this paper, I2/HP/HOA/LS are selected for TL state, and I0/FP/HOA/RS are selected for ST state as the IPDS policies, respectively.

In the evaluations, the performances of LRU policy, D-ADP and ADP-G are also evaluated. D-ADP selects the IPDS policies from two alternatives which are same as ADP-P by the set-dueling. In ADP-G, the IPDS policies is the same as ADP-P, and TH and TL are set to the quotient of the maximum value of GTR divided by 32 and 64, respectively. The number of partitions is set to four. The interval is set to the half of the number of sets.

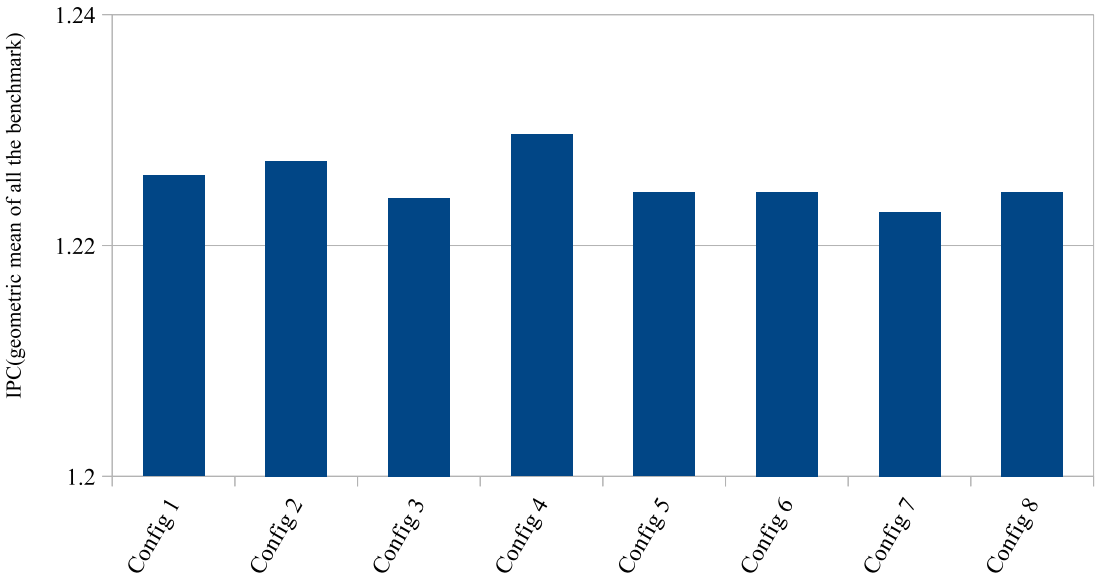


Figure 8: Effects of the Parameters of ADP-P (4MB L3).

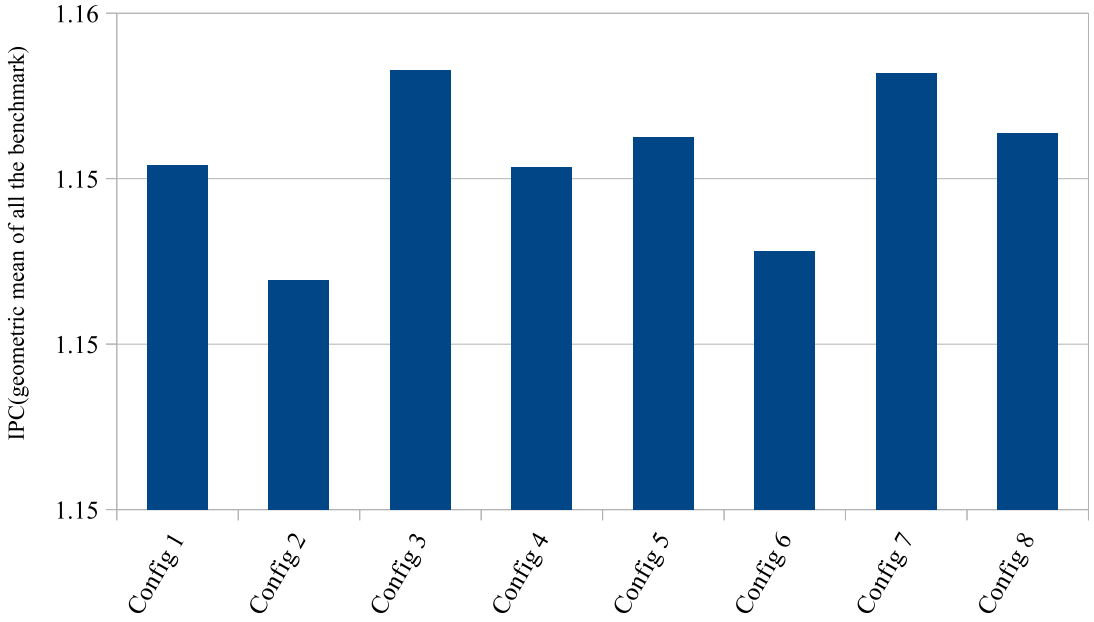


Figure 9: Effects of the Parameters of ADP-P (256KB L2).

Table 4: Experimented Parameters for ADP-P

Configuration	T_{TL}	T_{ST}	T_{fluct}
Config 1	7/256	32	32
Config 2	7/256	32	64
Config 3	7/256	64	32
Config 4	7/256	64	64
Config 5	8/256	32	32
Config 6	8/256	32	64
Config 7	8/256	64	32
Config 8	8/256	64	64

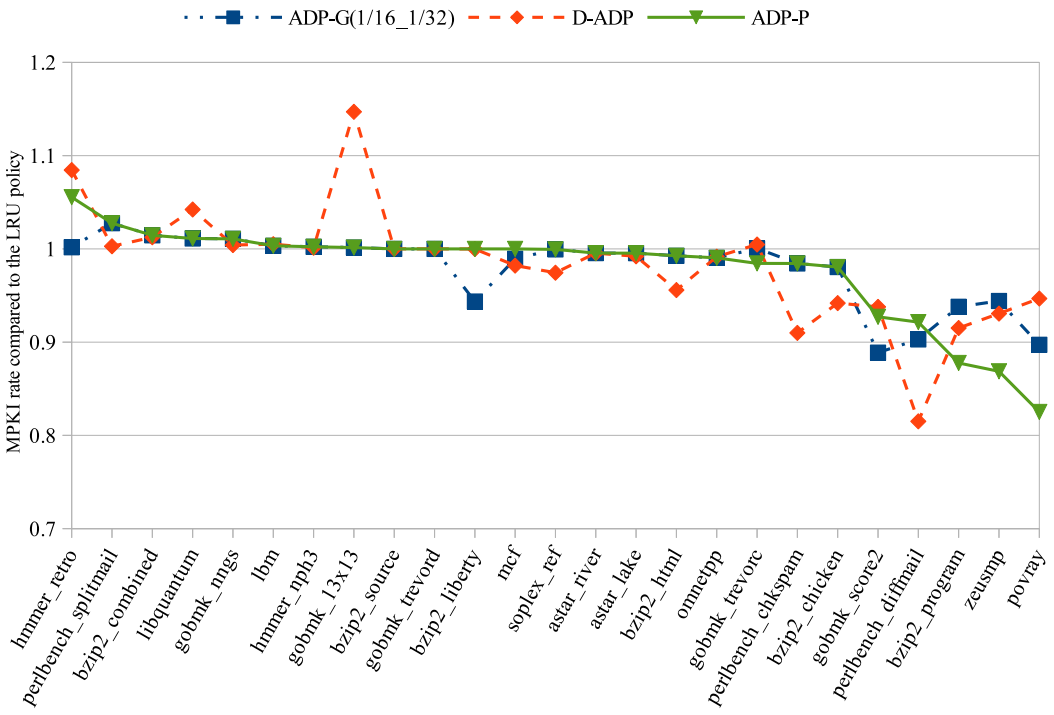


Figure 10: MPKI rate compared to LRU policy (2MB L3).

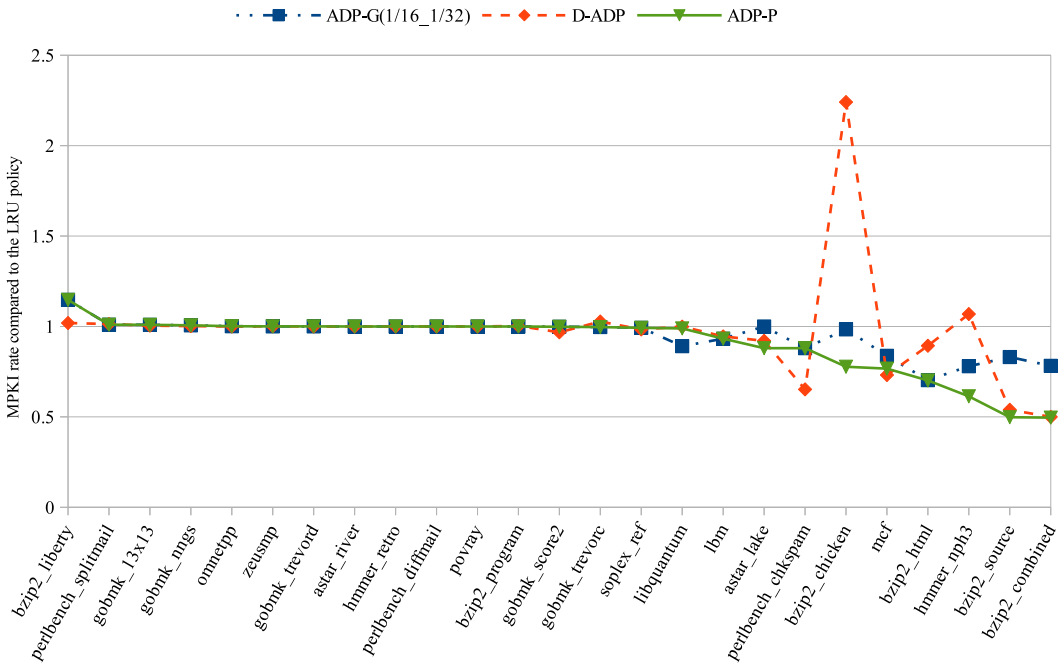


Figure 11: MPKI rate compared to LRU policy (4MB L3).

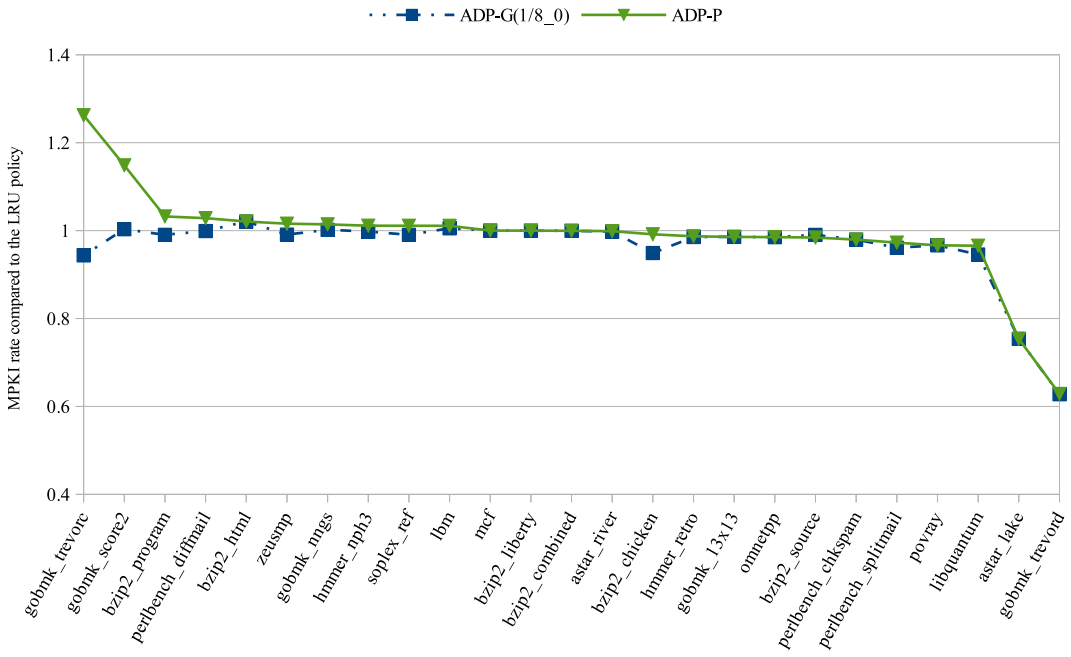


Figure 12: MPKI rate compared to LRU policy (256KB L2).

To decide the parameters of ADP-P at L3 cache and L2 cache, preliminary experiments are conducted. Experimented parameters are shown in Table 4. In Table 4, T_{TL} shows the value to be multiplied by the maximum value of GTR. Other parameters are used directly. Figure 8 and 9 show the geometric mean of IPC for all the benchmarks for L3 cache and L2 cache, respectively. Figure 8 shows the result of L3 cache when the cache size is 4MB. As shown in Figure 8, ADP-P achieves the highest performance with Config 4. In this configuration, T_{TL} is set to the product of the maximum value of GTR and 7/256, and T_{ST} is set to 64. T_{fluct} is set to 64. Figure 9 shows the result of L2 cache when the cache size is 256KB. As shown in Figure 9, the performance of ADP-P is highest with Config 3. In this configuration, T_{TL} is set to the product of the maximum value of GTR and 7/256, and T_{ST} is set to 64. T_{fluct} is set to 32. In L2 cache, the effective T_{fluct} tends to be lower than that of L3 cache. It is considered that facilitating of the transition from ST state to TL state will cause the performance improvement because of the temporal locality is suitable for many applications in L2 cache. Based on this experiment, the parameters of ADP-P are decided. For L3 cache, T_{TL} is set to the product of the maximum value of GTR and 7/256, T_{ST} is set to 64, and T_{fluct} is set to 64. For L2 cache, T_{fluct} is set to 32, and the other parameters are set to the same as L3 cache.

Figures 10 and 11 show the Misses Per Kilo-Instructions (MPKI) rate of all the benchmarks compared to LRU policy when the cache sizes are 2MB and 4MB, respectively. As shown in Figures 10 and 11, D-ADP and causes significantly increase of the cache misses in several applications. As compared to D-ADP, ADP-G does not cause significantly increase of the cache misses. However, as shown in Figure 11, ADP-G cannot select appropriate IPDS policies in several benchmarks when the cache size is 4MB. When the L3 cache size is 2MB, in the geometric mean of all the benchmarks, ADP-P achieves a 0.5%, 0.3% and 2.3% MPKI reduction compared to D-ADP, ADP-G and LRU policy, respectively. When L3 the cache size is 4MB, in the geometric mean of all the benchmarks, ADP-P achieves a 5.9%, 6.0% and 11% MPKI reduction compared to D-ADP, ADP-G and LRU policy, respectively. Although there is no significant difference in the performance of ADP-G and ADP-P when the cache size is 2MB, ADP-P can select the appropriate IPDS policies in several benchmarks when the cache size is 4MB.

Previous work does not show the performance of the ADP-G at L2 cache [1]. To illustrate the performance of the ADP-G and ADP-P at L2 cache, the simulation experiments are performed. As compared to L3 cache, it is considered that the cache replacement policy which can suit the temporal locality will work well for L2 cache.

Figure 12 shows MPKI rate of all the benchmarks compared to LRU policy when L2 cache adopts ADP-G and ADP-P. *TH* of ADP-G is set to the quotient of the maximum value of GTR divided by 8. *TL* is set to 0. Although ADP-G and ADP-P achieves a MPKI reduction, the performance of ADP-P is lower than that of ADP-G. In the geometric mean of all the benchmarks, ADP-G and ADP-P achieves 1.1% and 0.72% MPKI reduction compared to LRU policy, respectively. As shown in Figures 10 to 12, ADP-P is suitable when the applications cause scan accesses or thrashing, and ADP-G is suitable when the application have enough temporal locality. To realize the cache replacement policy which can select appropriate IPDS policies correctly, analyses of global fluctuation patterns of priority values and the novel scheme based on these analyses will be required.

As shown in the experimental results, ADP-P is useful when ADP-G cannot detect the characteristics of the running application. Especially in the detection of the application which is suitable for ST state, the performance of the ADP-P is better than that of the ADP-G. On the other hand, the ADP-G is suitable for detecting the application which is suitable for TL state. Since the schemes of ADP-G and ADP-P can be coexist, a novel cache replacement policy which consists of these schemes will be realized. The performance evaluation of this policy is future work.

5 Conclusions

This paper proposes a high-performance cache replacement policy called ADP-P. ADP-P focuses on the fluctuation patterns of the total priority value to detect the behavior of the running application and select the suitable parameters for that. Evaluation results show ADP-P achieves the number

of misses reduction compared to LRU policy, D-ADP and ADP-G. In the geometric mean of all the benchmarks, ADP-P achieves a 11% the number of misses reduction compared to LRU policy, when the cache size is 4MB. In future work, the cache replacement policy not only considering the fluctuation patterns of total priority value, but also considering global fluctuation patterns of priority values will be discussed.

References

[1] J. Tada. A cache replacement policy with considering global fluctuations of priority values. *International Journal of Networking and Computing*, 9(2):161–170, 2019.

[2] J. Tada et al. An adaptive demotion policy for high-associativity caches. In *Proceedings of international symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART2017)*, 2017.

[3] M. K. Qureshi et al. Adaptive insertion policies for high performance caching. In *Proceedings of International Symposium on Computer Architecture (ISCA 2007)*, 2007.

[4] N. Binkert et al. The gem5 simulator. *ACM SIGARCH Computer Architecture News*, 39(2):1–7, 2011.

[5] J. L. Henning. Spec cpu2006 benchmark descriptions. *ACM SIGARCH Computer Architecture News*, 34(4):1–17, 2006.

[6] G. Harmerly et al. Simpoint 3.0: Faster and more flexible program phase analysis. *Journal of Instruction Level Parallelism*, 7(4):1–28, 2005.