Journal of Signal and Information Processing, 2017, 8, 152-160
http://www.scirp.org/journal/jsip
ISSN Online: 2159-4481
ISSN Print: 2159-4465

Adaptive Cache Allocation with Prefetching Policy over End-to-End Data Processing

Hang Qin[1], Li Zhu[2*]

[1]Computer School, Yangtze University, Jingzhou, China
[2]Oujiang College, Wenzhou University, Chashan University, Wenzhou, China
Email: 68681700@qq.com, *yeah_1397118@hotmail.com

How to cite this paper: Qin, H. and Zhu, L. (2017) Adaptive Cache Allocation with Prefetching Policy over End-to-End Data Processing. *Journal of Signal and Information Processing*, **8**, 152-160.
https://doi.org/10.4236/jsip.2017.83010

Received: July 1, 2017
Accepted: July 14, 2017
Published: July 31, 2017

Abstract

With the speed gap between storage system access and processor computing, end-to-end data processing has become a bottleneck to improve the total performance of computer systems over the Internet. Based on the analysis of data processing behavior, an adaptive cache organization scheme is proposed with fast address calculation. This scheme can make full use of the characteristics of stack space data access, adopt fast address calculation strategy, and reduce the hit time of stack access. Adaptively, the stack cache can be turned off from beginning to end, when a stack overflow occurs to avoid the effect of stack switching on processor performance. Also, through the instruction cache and the failure behavior for the data cache, a prefetching policy is developed, which is combined with the data capture of the failover queue state. Finally, the proposed method can maintain the order of instruction and data access, which facilitates the extraction of prefetching in the end-to-end data processing.

Keywords

End-to-End, Data Processing, Storage System, Cache, Prefetching

1. Introduction

End-to-end processing refers to a system, which performs a business process from beginning to end, including all intermediate steps, such as data capture, data processing, analysis, and the generation of outputs. As for the memory and processor performance gap with a number of access optimization technologies, including non-blocking cache, prefetching, access instruction related prediction and so on, these technologies are concerned with how to reduce or tolerate the delay of access [1] [2]. Consequently, bandwidth optimization is the key to future processor performance improvement [3] [4]. Therefore, the optimization of

processor performance from the delay cannot satisfy the current situation, we must also consider the delay and bandwidth optimization [5] [6].

There are two main types of bandwidth optimization technology: One is to *increase the processor's transmission bandwidth*, including improving the processor interface frequency and data path, using on-chip memory controller and other technologies. The other is *to reduce processing unnecessary data transfer*. Therefore, when the cache instruction misses, it is effective to improve the processor bandwidth utilization by reducing the unnecessary data transmission of the processor. Currently, there are many studies on the implementation of the cache write failure strategy, including write allocate and non-write allocation, in terms of the strategies to improve the strategy [7] [8]. The advantage of the write allocation is to save access bandwidth, while the advantage of the non-write allocation strategy can reduce the blocking frequency of the storage management queue, and finally reduce the cache port occupancy [9].

In this paper, the performance of the storage system is optimized by analyzing the memory behaviors, in terms of on the improvement of the value for the processor and the access delay and bandwidth of the optimized processor [10] [11]. Then, we develop a series of performance optimization techniques for storage systems, and give performance evaluation and analysis on the proposed optimization techniques.

The main contribution of this paper lies in the analysis of the failure behavior of cache, and a new cache write failure processing strategy, *i.e.*, adaptive output allocation. This strategy combines the advantages of write allocation and non-write allocation strategies, thereby improving the processor's bandwidth utilization significantly. Compared with the traditional cache write failure processing strategy, hardware cost of adaptive cache write allocation strategy is small to avoid unnecessary data transmission, reduce cache pollution, and decrease the storage management queue blocking frequency.

2. Cache Processing Policy

Cache read failure processing strategy is concerned with how to reduce the different access delay, where it is to improve the processor bandwidth utilization, and reduce unnecessary data transmission. Currently, the cache processing policy is usually divided into write allocation and non-write allocation, according to whether or not a failure block is allocated in the cache when the write fails. *Write allocation strategy*, refers to the write failure occurred in the cache allocation of the corresponding cache block, can write directly to the cache. *Non-write allocation strategy*, refers to the write failure occurs, can write low-level storage system in the corresponding block, and not to the cache. The goal of writing failure management strategy is to improve the bandwidth of the processor.

The improved non-write allocation strategy is to set the write buffer to check if the write buffer is full. This method can be some of the same cache row operation combined to reduce the number of write operations. Setting write buffer and write-validate cache is similar because not every cache line is filled, write

low-level storage system, the need to be divided into multiple write operations or with masked write back, still cannot solve the non-write allocation strategy to waste storage bandwidth.

It is clear that because of the spatial nature of the program execution, some cache blocks will soon be filled with consecutive instructions, such as the cache block called the modified one, otherwise known as non-modified one. Because the entire cache block to amend the entire cache line are filled with modification, using non-write allocation strategy, write low-level storage system does not need to split into multiple write or need low-level storage system to provide masked writing. The write-allocating strategy is used for the full-modified cache block.

The current method of optimizing the write failure strategy is still a write allocation strategy, and if it is not necessary to split into multiple write operations to write the lower storage system, do not give full play to the advantages of non-write distribution strategy.

3. End-to-End Cache Write Allocation in the Cloud Environment

3.1. Adaptive Cache Write Allocation Strategy

According to the analysis of the failure behavior, the cache block is not necessary to fill the cache. The whole modified cache block and non-modified cache block is different, writing low-level storage system does not need to split into multiple write or masked write, will not waste storage bandwidth, suitable for non-write allocation strategy.

Adaptive cache write allocation policy, together with the invalid memory command sent to the memory failure queue to exit the storage management queue, in the memory queue to complete the stored operation. Since there is already a cache row size data field in the failover queue to hold the cache access to the data returned from the lower storage system, it is not necessary to add additional data field overhead to write the data of the stored instruction in the queue, the full-modified cache block is collected in the failover queue. The modified cache block can use non-write allocation strategy, which writes directly to the lower storage system. If there is the same cache block fetching instruction to enter, the corresponding item is set to fetch instruction, and the switch is used to write the allocation strategy.

The end-to-end principle is a design framework in computer networking. The processor delays the critical path. As the processor selected to send out the request to make a record path delay is relatively long, so does the use of random strategy to choose. When the processor executes the synchronization instruction and the cache instruction, it is necessary to clear the access failure queue. All the items are switched to the write cache, and the corresponding cache block is retrieved from the lower storage system.

Figure 1 shows the adaptive cache allocation in the cloud environment, in terms of VM (virtual machine) and SSD (Solid State Drives) cache over the end-to-end processing. Consequently, the processing flow of the adaptive cache write

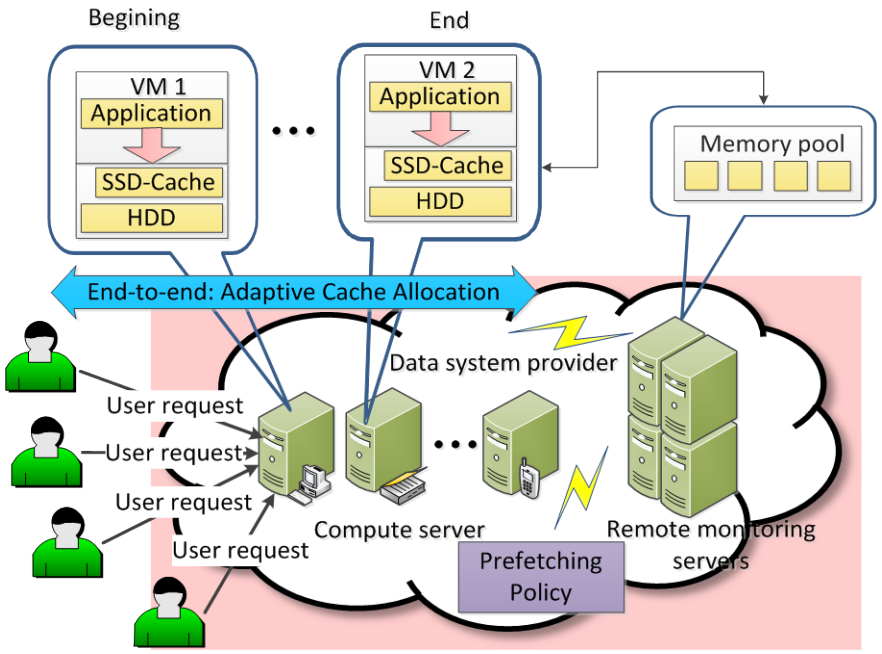

Figure 1. Adaptive cache allocation in the end-to-end cloud environment.

allocation strategy in the end-to-end cloud environment includes the following steps:

Step 1) The storage management queue sends out the invalid access request to the access failure queue. Then, the stored instruction writes the data to the data field of the access failure queue item and exits the storage management queue in the cloud environment;

Step 2) Determine whether the failure access instruction corresponds to whether the cache block is hit in the write queue. If yes, the data returned from the write queue is written with the data field of the corresponding failover queue item cache block, perform *Step* 5); otherwise, perform *Step* 3);

Step 3) Determine whether the fetch instruction or the number of instructions in the cloud environment:

- If it is fetch instruction, go to Step 4);
- If the collection for the full modification of the cache block, the implementation of Step 6);
- If it is not collected as a full modification of the cache block, to determine whether the item is switched to write using the allocation strategy; if yes, the implementation of Step 4), otherwise, continue to wait in the memory queue to collect the full-modified cache block.

Step 4) Issue an access request to the lower storage system, and wait for the low-level storage system data to return, the return of the data and the corresponding access to the failure of the queue in the data field of the written in the cache block;

Step 5) Write the cache with the corresponding entry data queue, execute *Step* 6);

Step 6) The failover access instruction is processed and exits from the access

failure queue in the cloud environment.

3.2. Advantages

Compared with the existing write failure processing technology, the adaptive cache write allocation strategy has the following advantages:

- In terms of the write allocation strategy, the invalid storage command sent to the memory after the failure of the queue, directly from the storage management queue. There is no need to wait data in the storage management queue to return, to reduce the number of invalid storage instructions caused by the storage management queue congestion occurs frequently.
- Full modification of the cache block does not need to retrieve the corresponding value from the lower storage system to reduce the unnecessary data transmission. Modify the cache block directly back to the low-level storage system, to avoid the cache port occupied and replace the cache in the useful cache block caused by cache pollution.
- Compared with store buffer technology and other design independent storage instruction buffer, adaptive cache write allocation strategy in the memory failure queue to collect all modified cache block, both to avoid the additional hardware overhead, and to avoid the cost of the instruction fetch buffer and the intervening failure queue are interrogated to ensure data consistency.

4. Prefetching Process in the Information Mart

4.1. Implementations

The current strategies to reduce cache failure rates has greatly improved the performance of storage systems, but because cache's capacity is much smaller than memory, and forced failure is difficult avoid, so cache failure still exists with the performance of the processor system. The optimization of the memory control strategy can reduce the memory access latency of the processor, but the ideal access to the processor is the need to spend memory access delay to get the required data. If the memory fails, the required data has been prefetched back. Consequently, **Figure 2** shows the end-to-end adaptive cache allocation with implementations in the information mart, concerning big data processing, massively parallel processing, and populating.

The implementations exacerbate the number of memory access operations, and have a reduction in the effective bandwidth of the system, resulting in operations. Since it is a guessing operation of fetching data from an access, it should affect the processor's normal access request as little as possible. In order to reduce the impact of operation on the normal memory access operation of the processor, it is necessary to further enhance the efficiency of extraction and the accuracy, thus improving the processor performance.

Based on the analysis of instruction cache and data cache failure behavior, we propose a prefetching strategy combined with the access failure queue state. The

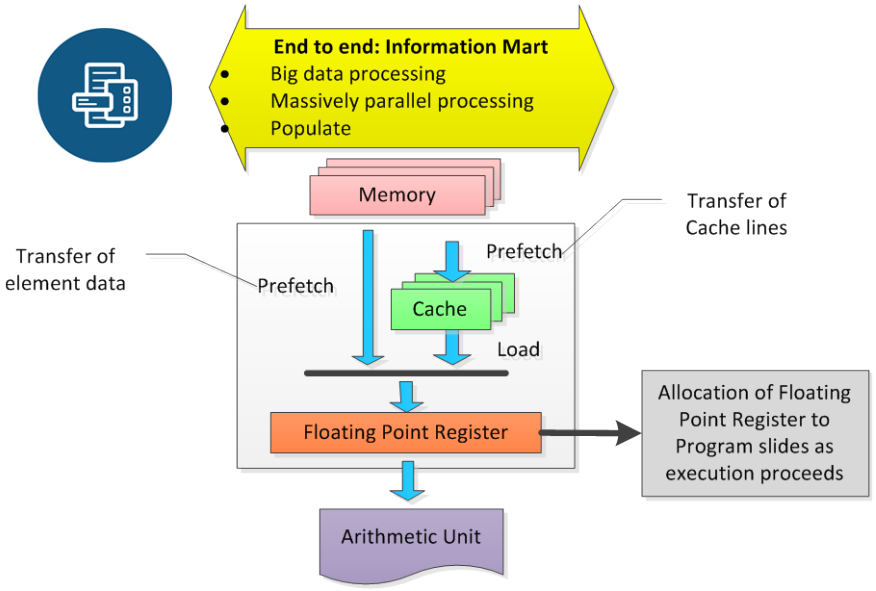

Figure 2. End-to-end prefetching implementation architecture.

timing initiating combination combines the status of the failover queue and reduces the impact on the processor's normal access request. The strategy maintains the order of instruction and data access, which facilitates the extraction of streams. Through the flow filter mechanism to improve the accuracy g, reduce the operation on the system to access the negative impact of bandwidth, effectively improve the performance of the processor system.

4.2. Main Factors

The above process can be done before the processor accesses prefetched data, which are the data required by the processor and only the data needed by the processor. In fact, it is difficult to achieve the ideal situation, mainly because the following main factors:

- Time: As the processor issues a request to the memory for too late, the data can not arrive in time before the processor needs it. The processor must wait for the required data to be returned from the lower storage system, causing the pipeline to halt and performance degradation. Storing data requires adding the necessary memory cells or replacing the data in the cache. If we replace the data in the cache method, it may be the processor to use the data to replace the cache. Data pollution of the original cache data, making this technology not only did not play the original function, but to the overall performance of the system to bring a negative effect.
- Position: The data must be stored at a higher level of storage level, in order to reduce the higher layer of memory access failure rate, so as to achieve shorten the access data delay, improve the overall performance of the processor. There are two ways to deal with the data in the high-level memory location, one is a simple replacement of the original cache data, the other is to increase a certain amount of storage unit. If we use the former, it may be because the

replacement algorithm is not good and cause cache pollution. If the latter, in particular, to increase the storage of some storage unit data, it will not occur before the cache pollution problems.

- Data size: Data granularity is the size of the data that can be transferred by a request. The size of the data granularity can be a word, a cache line, or several cache rows, or even a program data object. In general, the size of the data granularity is closely related to the transmission bandwidth between the two-tier storage hierarchy and the latency of accessing the lower-level storage system, of course it also relates to the size of the upper storage system and the ability of the processor to process the data.

The above main factors are closely related to the internal structure of the processor, such as the size of the internal cache of the processor, the organization mode. The goal is to reduce the cache failure or cache failure without increasing the hardware complexity and additional delay cost, thereby improving the overall performance of the processor.

5. Processor Core-Directed Memory Page Mode Control Strategy

Based on the analysis of the behavior of program memory and the lack of existing dynamic memory page mode, we can have a new dynamic page mode control strategy over the memory mode of controller. When the processor core sends an access request to the memory controller, it preferentially selects the same entry as the last access address. When the memory controller has an unprocessed read request, the current access uses open page mode to continue subsequent read requests.

5.1. Memory Page Mode Control

As for memory page mode control with the dynamic memory page mode, we can adaptively adjust the page mode according to the instructions of the processor core, blending the advantages of the open page policy and the close page policy. The memory controller waits for page mode switching in the absence of an unprocessed read request to avoid subsequent read request processing.

It shows the processing flow of the memory page mode control strategy. The followings are the memory control circuit in the read command after the end of the memory page mode control of the specific steps:

Step 1) After the end of the read command, the page mode control enable bit is judged. The current access uses the close page mode to perform and proceed to *Step* 5); otherwise, *Step* 2);

Step 2) Determine if there is an unprocessed read request in the memory controller. If the memory controller has an unprocessed read request, proceed to *Step* 3); otherwise, *Step* 4);

Step 3) The current access uses open page mode to continue processing subsequent read requests;

Step 4) Pre-charge that are instructed by the processor core as close page,

based on the processor core's page mode guidance information;

Step 5) End of end-to-end data processing.

5.2. Benefits

Compared with the prior art, the processor core-guided memory page mode control strategy has the following advantages:

- The use of dynamic memory page mode control strategy, integration of open page strategy and close page strategy advantages, can adaptively regulate according to the program.
- Comparing the address of the memory entry to the memory queue to the address and row address in the access history table, dynamic guidance memory page mode control, the processor core through the real future access behavior guidance, are better than the existing dynamic memory page mode control strategy with historical information to predict more accurate.
- Processor core in the send request can combine the memory controller in the absence of an unprocessed read request, and the page mode switch can avoid the impact due to pre-charge follow-up.

6. Summary

The data process flow with controlling requires a software platform, which automates and provides visibility into the data flow end to end. In this paper, the behavior of cache write failure is analyzed, and the new cache write failure strategy, *i.e.*, adaptive cache write allocation strategy, is proposed under end-to-end data processing. The strategy which does not need to add additional hardware overhead is easy to implement in the information mart. As a result, we can collect the modified cache block in the failover queue, use the non-write allocation policy for the full modification of the cache block, and are able to switch to the write allocation policy adaptively.

References

[1] Wang, Y.G., *et al.* (2016) Design and Evaluation of the Optimal Cache Allocation for Content-Centric Networking. *IEEE Transactions on Computers*, **65**, 95-107. https://doi.org/10.1109/TC.2015.2409848

[2] Mayuresh, K., *et al.* (2017) ROBUS: Fair Cache Allocation for Data-Parallel Workloads. *Proceedings of the* 2017 *ACM International Conference on Management of Data*, Chicago, 14-19 May 2017.

[3] Xu, M., *et al.* (2016) Analysis and Implementation of Global Preemptive Fixed-Priority Scheduling with Dynamic Cache Allocation. 2016 *IEEE Real-Time and Embedded Technology and Applications Symposium* (*RTAS*), Vienna, 11-14 April 2016, 1-12. https://doi.org/10.1109/RTAS.2016.7461322

[4] Herdrich, A., *et al.* (2016) Cache QoS: From Concept to Reality in the Intel Xeon Processor E5-2600 v3 Product Family. 2016 *IEEE International Symposium on High Performance Computer Architecture* (*HPCA*), Barcelona, 12-16 March 2016, 657-668. https://doi.org/10.1109/HPCA.2016.7446102

[5] Son, D.O., *et al.* (2016) A New Prefetch Policy for Data Filter Cache in Energy-Aware Embedded Systems. 2016 *Information Science and Applications* (*ICISA*),

1409-1418. https://doi.org/10.1007/978-981-10-0557-2_134

[6] Jo, D., *et al.* (2016) Enhanced Rolling Cache Architecture with Prefetch. *IEEE International Conference on Consumer Electronics-Asia* (*ICCE-Asia*), Seoul, 26-28 October 2016, 1-3. https://doi.org/10.1109/ICCE-Asia.2016.7804786

[7] Maurice, C., *et al.* (2017) Hello from the Other Side: SSH over Robust Cache Covert Channels in the Cloud. *NDSS*, San Diego, CA, US.

[8] Tao, M.X., *et al.* (2016) Content-Ceintric Sparse Multicast Beamforming for Cache-Enabled Cloud RAN. *IEEE Transactions on Wireless Communications*, **15**, 6118-6131. https://doi.org/10.1109/TWC.2016.2578922

[9] Liu, F.F., *et al.* (2016) Catalyst: Defeating Last-Level Cache Side Channel Attacks in Cloud Computing. 2016 *IEEE International Symposium on High Performance Computer Architecture* (*HPCA*), Barcelona, 12-16 March 2016, 406-418. https://doi.org/10.1109/HPCA.2016.7446082

[10] Inci, M.S., *et al.* (2016) Cache Attacks Enable Bulk Key Recovery on the Cloud. *International Conference on Cryptographic Hardware and Embedded Systems*, 368-388. https://doi.org/10.1007/978-3-662-53140-2_18

[11] Arteaga, D., *et al.* (2016) CloudCache: On-Demand Flash Cache Management for Cloud Computing. *Proceedings of the* 14*th Usenix Conference on File and Storage Technologies* (*FAST*), Santa Clara, 22-25 February 2016, 355-369.