www.ijecs.in
International Journal Of Engineering And Computer Science ISSN:2319-7242
Volume 2 Issue 3 March 2013 Page No. 640-641

Increase In Cache Performance Use Of Size As Factor In Replacement Algorithm

Nandnee Jain, Rajesh kumar chakrawarti
Computer science and engineering department
Shri Vaishnav institute of technology and science Indore (m.p.)
nandneejain@yahoo.com, rajesh_kr_chakra@yahoo.com

Abstract— The main requirement of any programmer is unlimited amount of fast memory. But the speed of memory is directly proportional to the cost of memory. But our goal is to achieve more speed in less cost, for these we have introduce memory hierarchy. At the top of this hierachy is small but very fast and costly cache memory. As cache is small memory so it contain only some part of data at any given time. If some new data is to be placed in cache than we have to replace the existing data (in case when cache is packed with data) using some strategies. LRU (Least Recently Used) is one such strategies. In this paper we have a new extension to LRU depending on the size of the block to be replaced.

Keywords—LRU,memory hierarchy.

I. INTRODUCTION

Cache is small but very fast memory as well as it is very costly. Cache act as a interface between main memory and processor for reducing latency period. Today the main concern is how to utilize this small area to maximize the performance. The performance of cache is defined as average time to access the memory.i.e

Average access time= Hit time+ Miss Rate*Miss penalty.

To increase average access time we have to decrease the Miss Rate. We design many strategies to decrease the Miss Rate for example for any new block arrived replace least recently used block (LRU), First in First out (FIFO), Random. This strategy reduces the miss rate but still we have to suffer the miss.

II. REASON FOR SUFFERING MISS IN CACHE

The miss referred to as a failed attempt of reading a data from cache which result in accessing the main memory for bringing the respective data and increasing our access time. You can any strategy but you cannot make the miss rate to 0%. Because there are some miss which are compulsory misses. Consider a scenario where system is just started and you have to initialize your cache, each time request for a data result in miss. So this misses cannot be reduced.

But consider a scenario when cache has no empty space for any new data. So when new block of data is to be placed in cache then we have to replace some old block using any one strategy. As most widely used strategy is LRU so according to this we replace old with new block. Now consider the next request was for data that currently replaced and we suffer from a miss i.e. its unpredictable to determine what the next request all about.

Instead of depending upon these strategies we should add a new angle in replacement algorithm. The Factor of size should also be considered for block that has to be replaced.

III. BASIS FOR INCLUDING SIZE FACTOR IN REPLACEMENT

The basis for using size as a factor comes from real time scenario. Consider that there is rack filled with boxes now we have to keep new box on it. so our approach is to search for the boxes that are not much in use and then we look for size of least recently used boxes and replace one whose size is sufficient enough to place a new box instead of replacing the most heavy one.

As we know that to bring a small data from the memory require less bandwidth & less time as compared to the large data which require more bandwidth & more time which result in increasing average access time of cache memory in case of miss.

IV. PROPOSED ALGORITHM

This new algorithm of replacement is just an extension to earlier strategy LRU. The algorithm uses the LRU to search for least recently used pages than while replacing it compare the size also.

1. Search for 3 least recently used block in the memory.
2. Now compare the sizes of these three blocks and arrange them in increasing order.
3. Now compare the size of new data with these three blocks.
4. Select the block to replace just enough in size which can accommodate the new block.

Now consider if we won't consider size as a parameter to replace the block. Than the block of 1024 bytes will be replaced for new block 0f 200 bytes, When

the next request arrive for this 1024 byte of data than to bring back this data we require more time and more bandwidth.

If we consider size also as factor than consider we select 3 least recently used block of size 1024, 512 and 256.And 1024 is the least recently used among this three than comes 256 and than comes 512.Now a new data of 128 comes than instead of replacing 1024 block we replaced 256 block .Now if next request is for this 256 block than less time and less bandwidth is require for this block in comparison to the block of size 1024.

[1] 1.http://www.eecs.berkeley.edu/~knight/cs267/papers/cache_memories.pdf

[2] . http://oldwww.just.edu.jo/~ali/740/1-cache-implementation.pdf

[3] .http://www.ece.ncsu.edu/arpers/Papers/iccd05-counterrepl.pdf

[4] Memory Data Organization for Improved Cache Performance in Embedded Processor Applications BY PREETI RANJAN PANDA, NIKIL D. DUTT, and ALEXANDRU NICOLAU ,University of California at Irvine

[5] Lecture 19: Cache Replacement Policy, Line Size, Write Method, Soon Tee Teoh CS 147

[6] Computer Architecture A Quantitative Approach by John L Hennessy and David A Patterson.

[7] http://scholarworks.umass.edu/dissertations/AAI8509594/M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.