A case for two-way skewed-associative caches

Rushiprasad Sagare 18114071
Karan Karhale 18116036
Kavya Barnwal 18114039
Rahul Jain 18114061
Gurpreet Singh 18116029
Arpit Mangal 18116018

October 2019

1 Description

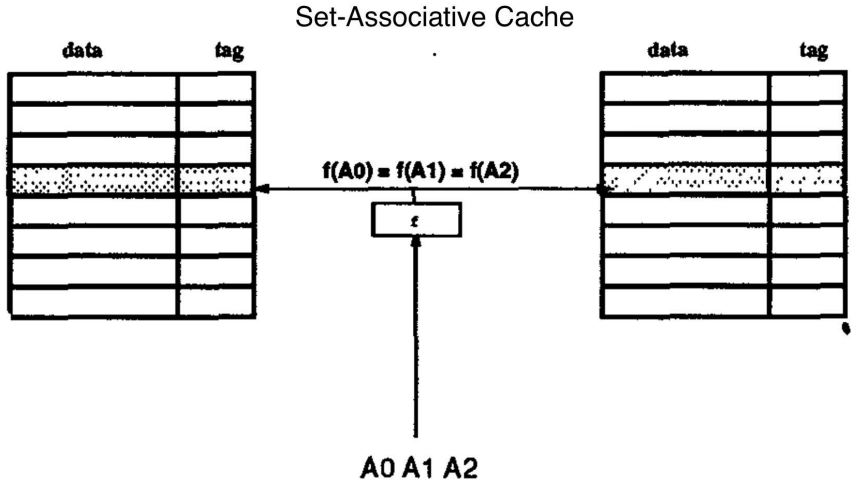
This project focuses on an idea of new organization for multi-bank cache:**The skewed-associative cache** . In which , we will consider the case of two-way skewed associative cache. There is an inherent trade-off between direct mapped caches and associative caches. Direct mapped caches don't have to choose among the cache lines (single cache line) but at the same time suffers from high miss rate.On the other hand, set-associative caches provide has relatively less miss rate by providing flexibility to place in different cache lines, hence less conflict misses.So skewed-associative cache is a new design which helps in improving miss rate while not compromising with the frequency at the same time.

The hardware complexity of two-way skewed-associative cache is same as the two-way set-associative cache ,yet simulations show that it typically exhibits the same hit ratio as a four-way set associative cache with the same size. Then skewed-associative caches must be preferred to set-associative caches. Direct-mapped caches exhibit hit ratios nearly as good as set-associative caches at a lower hardware cost. Moreover, the cache hit time on a direct-mapped cache may be quite smaller than the cache hit time on a set-associative cache.

2 Why is it important ?

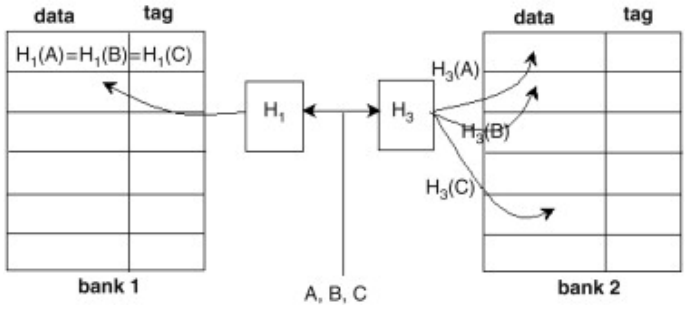
As the gap between the performance of microprocessor and memory performance widens, the penalty for accessing required data from higher level cache increases rapidly. Therefore, the miss rate is a key factor in deciding the performance of the cache from its average memory access time. Since the size of the caches are often constrained so as not to make the processor chip too large.

In response to this problem, it is important to find ways of increasing the hit rate without simply increasing the size of the L2 cache. One way to do this is through a different mapping function, where a certain address is mapped to different blocks in each set, thereby reducing possible collisions.



set associative.jpeg

Figure 1: Set associative cache



hash.jpg

Figure 2: Skewed associative cache

3 Skewed-associative caches

3.1 Principle

Skewed associative caches were proposed in [1,2].A X-way set-associative cache is built with X distinct banks as illustrated in Figure 1. The memory block at address D may be physically mapped onto physical line f(D) of any of the distinct banks. This vision of a set-associative cache fits with the physical implementation: X banks of static RAMs.

For a skewed associative cache (Figure 2), different mapping function is used for each cache bank: A memory block at address D may be mapped onto physical line $f_0(D)$ in bank 0, onto physical line $f_1(D)$ in bank 1, etc.

It has been shown in [1,2] that for general applications skewed-associative caches exhibit an average lower miss ratio than set-associative caches.

3.2 How to choose skewing function

In this section, we will discuss about the properties that might exhibit functions chosen for skewing the blocks in the distinct cache banks in order to obtain a good hit ratio (see [1,3]). We also present the functions which will be used in this report.

1) Equitability
First of all like in classical caches, for each line in the cache, the numbers of memory blocks that may be mapped onto this line must be equal.

2) Inter-bank dispersion
In a usual X-way set-associative cache, when (X+1) memory blocks contend for the same set in the cache: One of the block must be removed as there are only X lines available.

Skewed-associative caches avoid this situation by scattering the data: mapping functions can be chosen so that whenever two memory blocks conflict for a same line in bank i, they have a very low probability to conflict for a location in bank j (Figure 2).

3) Local dispersion in a single bank
Many applications exhibit spatial locality, therefore the mapping functions must also be chosen so that two "almost" neighbor memory blocks do not contend for the same physical line in any cache bank.

The different mapping functions must respect a certain form of local dispersion on their respective bank; the mapping functions f_i must limit the number of conflicts when mapping any region of consecutive memory blocks within bank i.

4) Simple hardware implementation
Mapping functions must have the less complexity so that it will use a small number of gates and delays.

3.3 How it handles conflict misses

Data dispersion
After a single read on a sequence of distinct cache blocks, more blocks are in a skewed-associative cache than on a set-associative cache.
On the other hand, the configuration of data blocks present at the same time in the cache depends on the precise mapping of each data block in the cache. Let us consider a memory block D present in the cache at time t. Among the

other possible locations for a block D, there may be a location occupied by an unuseful block (dead block or block which will not be reused for a long time). Block D may be removed from the cache by a miss. The next time D will be referenced, D can ve mapped in an empty location in bank j, thus increasing the number of useful blocks in the cache.

3.4 Replacement policy for skewed-associative caches: a bottleneck

When a miss occurs in a X-bank caches, the memory block to be replaced must be chosen among X blocks. Different replacement policies may be used. LRU replacement policy or pseudo-random replacement policy are generally used on set-associative caches.

LRU replacement policy is generally considered as the most efficient policy. Implementing an LRU policy on a two-way set-associative cache is quite simple. A single bit tag per cache line is sufficient: when a line is accessed, this tag is asserted and the tag of the second line of the set is deasserted.

Unfortunately, a simple LRU cannot be implemented on a skewed-associative cache. Therefore, some pseudo-LRU replacement policies are used.

4 Evaluation methodology

The purpose of this study is not to measure absolute miss ratios, but, first to show that there is a need for high associativity and at second to evaluate replacement policies for skewed-associative caches. Then,the performance for cache C will be:
P = miss ratio on C / reference miss ratio
where **reference miss ratio** is the miss ratio on a 2-way set-associative cache with LRU replacement for the benchmark and the considered structure.

5 Replacement policies for skewed associative caches

The replacement policies provided here are :

5.1 Single-bit replacement policy

–> A tag bit is associated with each line in bank 0: when the line is indexed, the tag bit is asserted when the data was in bank 0 and deasserted when the data is in bank 1.

5.2 Usefulness policy

–> A tag bit U is associated with each line the two banks. When a line is indexed, on a hit the tag bit is updated with the number of the hitting bank. On a miss, the tag bits of the two selected lines are read: when they agree on value b, the missing line is written in bank 1-b, when they disagree, the number of the target bank is randomly selected.

5.3 Not Recently Used

–> A bit tag RU (Recently used) is asserted when the cache line is accessed. Periodically the bit tags RU of all the cache lines are reset: we experimentally determined that a good period is each "cache size in bytes"/4 accesses to the cache.

When a block misses in the cache, the replaced block is chosen among the X possible blocks in the following priority order:

1) Randomly among the blocks for which the RU tag is clear.

2) Randomly among the blocks for which the RU tag is set, but which have not been loaded in the cache.

3) Randomly among the blocks for which the RU tag is asserted and which have been modified.

5.4 Not Recently Used + Useful

–> Two tag bits Y (for young) and U (for useful) are used. They are asserted just as defined previously, but the choice of the replacement target is now done as with following priority:

1) If one of the block is **old** and the second one is **young** then select the **old** block.

2) If the two U bits agree on value b, then select block in bank 1-b.

3) Randomly select one of the two banks.

5.5 Enhanced Not Recently Used

–> Replacement target is chosen with the following priorities:

1) Randomly select among the **old** blocks.

2) Randomly select among the **young** blocks.

3) Randomly select among the **very young** blocks.

6 Results

1) The two-way skewed-associative cache show approximately same miss ratio as a four-way set associative cache.

2) The improvement of performance of skewed-associative over set-associative caches is mostly due to the inter-bank distribution property.

3) There is no significant hit ratio improvement when increasing the inter-bank dispersion degree over 8 on a two-way skewed-associative cache as there is no significant hit ratio improvement when increasing the associativity degree over 4 or 8 on a set-associative cache.

7 References

1)A. Seznec, "A case for two-way skewed associative caches"
`http://www.archive.ece.cmu.edu/~ece447/s13/lib/exe/fetch.php?media=p169-seznec.pdf`

2)A. Seznec, "Skewed-associative caches"
`https://www.researchgate.net/publication/220758754_Skewed-associative_Caches/link/00b4951bf702aa39bf000000/download`

3)F. Bodin, A. Seznec, "Skewed-associativity enhances performance predictability"
`https://hal.inria.fr/inria-00074177/document`