Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

A Comparative Study of Cache Optimization Techniques and Cache Mapping Techniques

Manu Gupta
Department of Computer Science & Engineering
The NorthCap University, Gurgaon

Jyotsna Singh
Department of Computer Science & Engineering
The NorthCap University, Gurgaon

***Abstract* - For calculation of the processor performance, cache memory has a significant role. Cache replacement policies have a major role in structures that effectively supervise them as processors now require more power with better efficiency along with good performance. Algorithms aiding such replacement policies focus more on assistanting the increased data prerequisites of processors. LRU (Last Recently Used) policy forecasts a re-reference interval and inputs that reveal a different re-reference interval performs poorly under this policy. Performance variance between hypothetical replacement and LRU is large for highly-associative cache as shown by recent studies. The LRU strategy is prone to memory loads where an operational set is bigger than the available size of cache. For improving the performance of the cache other replacement algorithms are designed. This paper presents a lower overhead, high performing cache replacement strategy for processors that utilize the mechanism of LRU replacement and talks about various other techniques as well.**

***Index Terms - Cache optimization; latency; cache miss; memory hierarchy; vectorization*.**

I. INTRODUCTION

To improve the computer system potential, caching is useful to shadow the latency-gap that exists between the CPU and memory by capitalizing on locality in memory accesses.[3]
Auxiliary Memory size is greater than that of cache. In the event of an error, when an instruction is requested, a page from it is replaced in the main memory. The conundrum is highly common to the block replacement in cache memory but the page replacement is graver as page transferences from disk to memory with respect magnitudes lower than block transfers to cache memory from main memory. The effectiveness of data caches in numerical code has not been recognized, however it is demonstrated to be effective for general-purpose utilisations in bridging the processor and memory speeds. Only a small fraction of matrix can be held by a cache ;thus there is a high probability that the data might have been moved from the cache by the time it has to be reused.[3][10][15]
Multiple applications on a single chip becomes possible in modern processors since they contain many cores. The number of cores on a chip is directly proportional to the density on the memory system to withstand the memory requirements of all the executing applications. One of the methods to obtain great performance from various designs is to manage the largest level on-chip cache competently so that off-chip accesses are reduced. Due to high level incorporation, the floating-point arithmetic competence of microprocessors has increased considerably in the last few years. Unfortunately, the increase in processor speed has not been complemented by a similar increase in memory speed. To fully realize the potential of the processors, the memory hierarchy must be proficiently utilized[5][7].

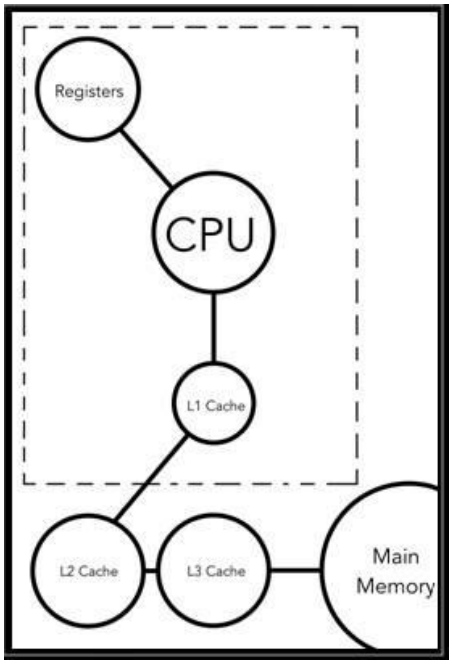

Fig. 1. Memory hierarchy having an on-chip L1 cache, one on-chip L2 cache a third level off-chip cache.

Cache Optimization Techniques
Optimization is a major problem related to the performance of cache which is majorly because of the cache pollution in last level cache. Cache pollution is said to occur when data of the strong locality gets replaced by the data of weak locality. since all the cores of multi-core processor shares the last level cache, affects all of them. To address this issue, a user level control system is introduced. [3][10]

FIFO

- First in, first out literally.
- Older the page more the probability for it to be replaced.

LRU

- Retains the recently used pages.
- Performs better than RR in practice.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

Random Replacement
- Randomly selects pages When low on space, it randomly selects the pages.
- Eliminates overhead cost to tracking the page references. [2]
- Better than FIFO policy.
- Better than LRU for looking memory references.

Second Chance Replacement
- Pages for elimination consider in a round robin manner, page that has been accessed amid successive considerations will not be replaced. Page replaced is one which has not been accessed since its last consideration. [3]

I.a. Data Access Optimizations [1]
- It is composed of code conversions which change the direction in which iterations in a loop nest are executed. This improves temporal locality. [4]

Loop Interchange
- The order of two adjacent loops in a loop nest is reversed in this transformation. It can be useful if the order of loop execution is insignificant. It can be generalised to loop variation by permitting more than two loops to be moved at once and by not necessitating them to be adjacent.
- It also improves vectorization, parallelism and register reuse.
- By reducing the stride of an array based computation, it improves locality.

Loop Fusion
- Under this transformation, the bodies of two contiguous loops that have the same iteration space traversal are combined into a single loop. It is also known as loop fission which breaks a single loop into multiple loops with same iteration space. [4][6]
- Reduces overhead of total loop by approx. factor of 2.
- Fusing two loops forms a single loop which comprises more commands in the body and thus offers enhanced instruction level parallelism.[12]
- Improves data locality.

Loop Blocking
- This transformation enhances the depth of a loop nest with depth n by accumulating more loops to it.
- It is primarily used to improve data locality by enhancing the reuse of data in cache.

Data Prefetching
- When the data are requested early, penalty of misses as well as capacity misses can be hidden.
- It involves overhead.
- It allows the microprocessor to issue a data request before the computation actually requires the data.[18]

I.b. Layout Optimizations [1]
- It talks about modifying the arrangement of data assemblies and variables in memory.
- Effects like cache conflict misses and false sharing are avoided
- It improves spatial locality of code.

Array Padding
- When two arrays are retrieved in an alternating way, the data structures might end up being mapped to the same cache lines, a significant number of conflict misses are introduced.
- In case of cross interference and self-interference, two often array references problems, this method helps to reduce the number of conflict misses.

Group and transpose
- The spatial locality amid the components of different arrays or other data structures can be improved with this layout technique.
- Helps reduce the cross interference misses for large arrays with alternate-access pattern.
- Best utilised with elements which are positioned separately in memory but retrieved together.

Data Copying
- In this technique a non-contiguous data from a block is copied into a contiguous area of memory. Making every word of the block being mapped to its own cache location.
- Is Guarantees high cache utilization and avoids self-interference within the block.
- Copying operation increases the cost and often overrides the benefits of it.

II. CACHE MAPPING TECHNIQUES

Data from main memory could be mapped on to the cache using different mapping techniques and then used by the processor. Performance of processor speed are directly impacted by these techniques.. This paper discusses different cache mapping techniques and their effect on performance. [8]

Direct Mapping
- Every cache block of data is plotted to a memory address; it needs only one contrast to decide where to place the data.
- Access time is fast
- More miss penalty
- Lowest hit rate

Fully Associative
- Data is stored randomly at any place in cache hence requiring many comparators in order to check the tag over every cache block.
- Expensive
- Hit rate is highest
- The process to decide which slot should be freed when new data enters is a challenging.
- Complex than Direct Mapping (Since it requires sophisticated search algorithms.)

N-way set-associative cache
- The cache is divided into sets and every memory address maps to a particular set within the cache.
- Better hit ratios and less conflict misses leads to additional cost.
- One way of improving Performance of direct mapped cache can be improved bykeeping the balance of accessing cache sets. This can be achieved by reducing the size of decoder.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 6 Issue 05, May - 2017

CONCLUSION

In this paper we have compared various cache optimisation techniques along with various mapping techniques. In this paper we have compared techniques based on different parameters such as cache misses, hit rates etc. We saw that LRU performs better than Random Replacement in practice. Different data access optimisation and layout optimisation techniques has also been discussed. We saw that Full Associative mapping technique has the highest hit rate in comparison to the conventional direct mapping technique with lowest hit rate. N-way set associative cache performs better in some cases but at times the additional cost due to better hit ratios override the benefits gained from this technique.

REFERENCES

[1] Markus Kowarschik1 and Christian Wei, "An Overview of Cache Optimization Techniques and Cache-Aware Numerical Algorithms," GI Dagstuhl Research Seminar on Algorithms for Memory Hierarchies, Volume: 2625, Pages 5-12, January 2003.

[2] Pancham, Deepak Chaudhary and Ruchin Gupta, "Comparison of Cache Page Replacement Techniques to Enhance Cache Memory Performance," International Journal of Computer Applications (0975 – 8887) Volume 98– No.19, Pages 1-3, July 2014.

[3] Muhammad Waqas Ahmed, Munam Ali Shah, "Cache Memory: An Analysis on Optimization Techniques," International Journal of Computer and Information Technology (ISSN: 2279 – 0764) Volume 04 – Issue 02, Pages 1-5, March 2015.

[4] Moinuddin K. Qureshi, Yale N. Patt, "Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches," The 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06), Pages 4-5, 2006.

[5] Sparsh Mittal, "A Survey of Cache Bypassing Techniques," Journal of Low Power Electronics and Applications, Page 8, April 2016.

[6] Sa'ed Abed, Mohammad Al-Shayeji, Sari Sultan, Nesreen Mohammad, "Hybrid approach based on partial tag comparison technique and search methods to improve cache performance," IET Comput. Digit. Tech., 2016, Vol. 10, Iss. 2, Pages 1-2, August 2015.

[7] Sara Alouf, Nicaise Choungmo Fofack, Nedko Nedkov, "Performance Models for Hierarchy of Caches: Application to Modern DNS Caches," ValueTools'13, Page 3, December 2015.

[8] Niklas Carlsson, Derek Eager, Ajay Gopinathan, Zongpeng Li, "Caching and optimized request routing in cloud-based content delivery systems."

[9] P. R. Panda, F. Catthoor, N.D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, & A. Vandercappelle, P.G. Kjeldsberg, "Data and Memory Optimization Techniques for Embedded Systems," ACM Transactions on Design Automation of Electronic Systems, Vol. 6, No. 2, April 2001, Pages 149–155.

[10] Cong Thuan Do, Hong-Jun Choi, Jong Myon Kim, Cheol Hong Kim, "A new cache replacement algorithm for last-level caches by exploiting tag-distance correlation of cache lines."

[11] Monica S. Lam, Edward E. Rothberg and Michael E. Wolf, "The Cache Performance and Optimizations of Blocked Algorithms."