ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

An Overview of B+ Tree Performance

Amulya Singh
Department of computer science
Jamia Hamdard
New Delhi, India

Mr. Bhavya Alankar
Department of computer science
Jamia Hamdard
New Delhi, India

Abstract:Over the years, difference in speed between Central Processing Units or CPUs and main memory has become so much that lots of applications, including DBMSs, spend a lot of time waiting for data to be fetched from main-memory. B+ trees have been observed poorly utilizing the cache memory, which is leading to the development of many cache-conscious indices. Earlier simulation models were used to study cache conscious indices, the trend has recently gone towards measuring performance on real computer architectures, which means the performance is measured on real systems, instead of simulation models. In this study, we study the performance of the pB+ tree on the Itanium 2 processor, focusing on various implementation options and their effect on the performance.

Keywords: DBMS, B+ Tree, CPU

INTRODUCTION

Over the past 20 years, the Central Processing Unit or the CPU speeds have been increasing at a faster pace as compared to the memory speeds. As a result, many modern applications or software spenda lot of their time waiting for data to be fetched or delivered from the memory. DBMSs are not an exception, and the recent studies [1, 3] show that slightly less than half ofthe Central Processing Unit or the CPU time used by the commercial Database Management Systems or the commercial DBMSs is spent on important computations. One of the main parts or the components of the Database Management Systems or DBMSs are the indices, which in past have been optimized to minimize or decrease disk input-output or I/O. Database Management Systems or DBMSs do pretty well in hiding the disk Input-output latency or disk I/O latency, but with the increasing gap between CPU and memory speeds, indexing performance is becoming increasingly bound or affected by memory speeds. This hasled to the development of many memory oriented or memory based or "cache-conscious" indices, which tend to use memory more efficiently. The main focusis the implementation of one such index, the "Prefetching B+-tree" (pB+-tree), and measurements of its performance on a modern server CPU [9].

THE MAIN MEMORY BOTTLENECK

These days, the modern computers are based or built on von Neumann architecture[7], in which the Central Processing Unit or the CPU and storage (main memory) are kept separate.Over the last twenty years, the CPU speed has been increasing by 61%, and on the other hand, the main-memory speed has only been increasing by just 10% per year [8]. The speed difference between the two has become so big, in fact, that in many applications Central Processing Units or the CPUs spend a lot of their time waiting for data to be fetched or delivered from the main-memory [5].

The commonly used method which deals with this main – memory bottleneck is the use of cache – memory (or cache), which is a limited size, high speed memory, stored on the same chip as the Central Processing Unit or the CPU [6].

CACHE-CONSCIOUS INDICES

DBMSs or Database Management Systems are not exempted from the main memory bottleneck. A recent follow-up study showed that the condition hasn't improved [3],which indicates that commercial Database Management Systems are not being engineered to cope up with the main-memory bottleneck.

PROBLEM STATEMENT

As mentioned in the introduction, these days' modern computers are based or built on the von Neumann architecture[7], in which the CPU and main-memory are kept separate.This separation between the two has created the so called "Von Neumann bottleneck", because (for many workloads) CPU are now capable of processing data much faster than it can be delivered from main-memory. The term was founded by John Backup in his 1977 ACM Turing award lecture [2]

CACHE MEMORY

Hill and Smith [4] classify cache-misses into 3 categories

- Compulsory misses happen when the cache-line is first at referenced.
- Capacity misses happen when the cache-line, which is being accessed has been replaced because the cache was full.
- Conflict misses happen when the cache-line accessed has been replaced because of the limited associativity, but would be found in a fully associative cache.

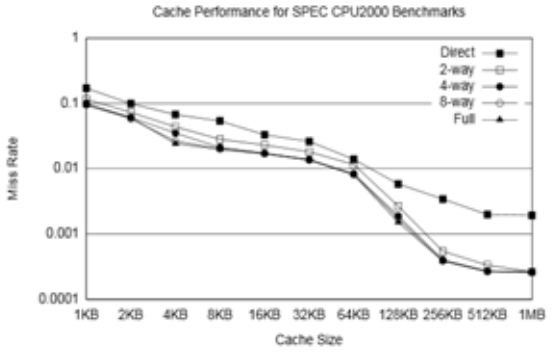

Figure 2.1: The average of miss rates for various SPEC CPU2000 benchmarks.

Out of these three types of misses, only compulsory misses can't be avoided. The amount or the number of capacity misses can be decreased by increasing the size of the cache, and the amount or the number of conflict misses can be reduced by increasing the associativity of the memory; a fully associative memory has none.

Modern Central Processing Units or the CPUs have up to three levels of cache, each positioned or placed closer the Central Processing Unit or the CPU core. The Itanium 2 processor has 3 levels of cache, and the latest generation has2 Central Processing Unit or CPU cores [9]. It is noteworthy that two – third of the die area is occupied by cache, which shows that it plays an important role in themodern-day computing.

PROPOSED SOLUTION

The Itanium 2 processor instruction set includes a pre-fetch instruction, which has three different types of prefetching hints:

The Exclusive hint: This controls whether the cache-line should be marked as dirty, in which, it is written back to the cache-level below when replaced.

Data Alignment

Data alignment is extremely important when you are working with integer data on the Itanium 2 processor.An eight byte window must hold all the stores and holds.If an unaligned integer data operation is issued, the CPU will throw an error or exception and call a handler inside the OS.

IMPLEMENTATION CHOICES

- Prefetching hints
- The prefetching loop
- Node prefetching

CONCLUSION

In this study we have studied the performance of the pB+ tree on the Itanium 2 processor. Its main focus was on the variety of implementation options or choices in study we faced and their impact on performance. Whereas some of these options have a slight effect on the performance, others affect it a lot; when all these benefits or gains are put altogether, the performance implications of using prefetching can be considered or are considerable.

REFERENCES

[1] A. Ailamaki, D. J. DeWitt, M. D. Hill, and D. A. Wood. DBMSs on a modern processor: Where does time go? In Proceedings of VLDB, pages 266–277, Edinburgh,Scotland, 1999.

[2] J. Backus. Can c-programming be liberated from the von Neumann style? A Functional style and its algebra of programs.Comm. ACM, 21(8):613–641, Aug. 1978.

[3] M. Becker, N. Mancheril, and S. Okamoto. DBMSs on a modern processor: "Wheredoes time go?" revisited. Technical report, CMU, 2004.

[4] M. D. Hill and A. J. Smith. Evaluating associativity in CPU caches.IEEE Transactions on Computers, 38(12):1612–1630, 1989.

[5] S. A. McKee. Reflections on the memory wall. In Proceedings of the Conference on Computing Frontiers, page 162, Ischia, Italy, 2004.

[6] A. J. Smith. Cache memories .ACM Computing Surveys, 14(3):473–530, 1982.

[7] J. von Neumann. First draft of a report on the EDVAC.IEEE Annals of the History of Computing, 15(4):27–75, 1993.

[8] J. L. Hennessy and D. A. Patterson. Computer Architecture: A Quantitative Approach. 2002. BIBLIOGRAPHY 49

[9]http://skemman.is/stream/get/1946/7489/19942/1/MSc_ArniMar-Jonsson.pdf