Analysis of Database Workloads on Modern Processors

Dawei Liu[1,2], Shan Wang[1,2]
[1] School of Information, Renmin University of China, Beijing 100872, China
[2] Key Laboratory of Data Engineering and Knowledge Engineering, MOE of China Beijing, China
{liudawei,swang}@ruc.edu.cn

ABSTRACT

This paper describes my Ph.D. work on LAMA[1], a joint project with HP Lab China. The goal of my research on the project is to exploit the characteristics of modern processors and multi-level memory hierarchies, thereby improving the database performance on next generation processor and improving the performance of next generation memory-oriented database. In this paper I will give an overview of my PhD work associated with it in terms of what progress has been made so far and directions for further research.

1. BACKGROUND

The continued evolution of modern processor creates new opportunities for improving database performance. However, traditional database researchers dedicate to improving database performance through I/O optimization fail to utilize processor resources efficiently. Modern processors with multi-level memory hierarchies, superscalar out-of-order execution, simultaneous multi-threading, multi-cores, such as Itanium II, impose a great opportunities to improve database performance. Therefore, exploiting the characteristics of modern processor has become an important topic of database system research [26]. In this research area, accurately characterizing the database workload behavior on modern processor is an important issue. An in-depth understanding that how database workloads interact with the processor and deep memory hierarchy can identify a set of characteristics, which would be extremely valuable for further research on performance optimization for these environments. The issues to be evaluated are described as follows.

1.1 Processor Issue

Prior research has shown that DBMSs tend to achieve low IPC (instructions-per-cycle) on modern processors [17], that is processors are inefficiently used. Previous research calculated the time spent on CPU cache misses by measuring the actual number of misses. This methodology accurately described DBMS's performance on the machines used at that time (Intel Pentium II/Pentium Pro). What we are interested in, however, is the case of today's CPUs (such as Itanium II), since modern processor leads to significantly increased processor speed. For example, the Itanium II processor can execute up to six instructions concurrently. One might expect that query-intensive workloads such as data mining, multimedia information retrieval, and OLAP could provide modern CPUs the opportunity to reach near optimal-IPC efficiency. Therefore, accurately characterize the database workloads on modern processors is significant to both the database and computer architecture communities.

1.2 MMDB Issue

Since [1](1999), the execution time breakdown of database systems on modern computer platforms, has been analyzed extensively. The primary motivation of these studies is to improve the performance of Disk Resident Databases (DRDB), which form the main stream of database systems until now. The typical benchmark used in those studies is TPC-C [25].

However, the continued hardware advancement has "moved-up" on the memory hierarchy - the larger and larger on-chip and off-chip caches, the steady increased RAM space, the commercial availability of huge flash memory (solid-state disk) on top of regular disk, etc. To catch such trend, we should also move-up the target of workload characterization research along the memory hierarchy; Unlike the performance of a DRDB which is I/O bound and may be optimized by high-level mechanisms such as indexing, the performance of a Main Memory Database (MMDB) is basically CPU and memory bound. Therefore, investigating the MMDB workloads on modern processor is an important topic.

1.3 MMDB Benchmark Issue

Database researchers have developed several systems of performance assessment, such as the Hypermodel [24], OO1-Benchmark [20], OO7-Benchmark [13], BUCKY benchmark [14] and industrial standards like the family of TPC benchmarks.

However, none of the available benchmarks offers the coverage needed for MMDB's performance evaluation. Prior

[1]LAMA stands for Large Scale Data Management, a joint project between Renmin University of China and HP Lab China

Proceedings of SIGMOD2007 Ph.D. Workshop on Innovative Database Research 2007(IDAR2007) ,June 10, 2007, Beijing, China.

research used TPC-C to evaluate MMDBs throughput, however TPC-C, which is currently the OLTP benchmark supported by TPC, focuses on DRDB performance evaluation. Several constraints specified in the TPC-C standard are not suitable for MMDB performance evaluation. For example, according to the TPC-C standard specification, the number of New-Order transactions per minute and the number of warehouses must be in the ratio between 9.0 and 12.86. This constraint is designed aiming to ensure a certain amount of disk I/Os in transaction processing. As another example, the size of datasets under test must scale up with the numbers of users increasing. This is not suitable for MMDB benchmarking too, because in a DRDB, if a System Under Test (SUT) can support the population of 120 warehouses , it will support 1200 concurrent users according to the specification. However, in the case of MMDB, although a SUT can support 1200 concurrent users, it may be impossible to support the population of 120 warehouses due to the space limitation. Therefore, with the popularity of MMDB [9], such as AMOS [5], PRISMA [18], Dali [6], Times Ten [23], Databitz [8], MonetDB [16] etc, developing a specialized benchmark has become a pressing issue.

1.4 Summary

As discussed above, my research will focus on exploring the following subjects:

(1) Characterize DRDB workloads on modern processor: investigate how DRDB interacts with the deep memory hierarchy under OLTP and OLAP workloads running on a modern processor, such as Itanium II - based platform.

(2) Characterize MMDB workloads on modern processor: explore its deep memory behavior different from that of DRDB; explore the difference of column-oriented and row-oriented storage models in CPU and cache utilization; explore index influence on deep memory utilization.

(3) Study on how to benchmark a main memory oriented database.

Organization: The rest of this paper is organized as follows. Section 2 contains a brief discussion of related work. The methodology is presented in section 3. A brief discussion and analysis of the experimental results is given in Section 4. Conclusions and future research directions are presented in section 5.

2. RELATED WORK

There have been several published studies concentrating on characterizing the database workloads. In the first literature [22] Shreekant et al. studied a relational DBMS running an on-line transaction processing (OLTP) workload, they concentrated on multiprocessor system issues, such as assigning processes to different processors to avoid bandwidth bottlenecks. A.M.G.Maynard et al. [2] using TPC-A and TPC-C on another relational DBMS showed that commercial workloads exhibit large instruction footprints with distinctive branch behavior, typically not found in scientific workloads and that they benefit more from large first-level caches. M.Rosenblum et al. [12] showed that, although I/O can be a major bottleneck, the processor is stalled 50% of the time due to cache misses when running OLTP workloads. Other interesting studies evaluated database workloads, mostly on multiprocessor platforms. Most of these studies evaluate OLTP workloads [19][7][10], a few evaluate decision support (DSS) workloads [10] and there are some studies that use both [11][15]. All of the studies agree that the DBMS behavior depends upon the nature of the workloads (DSS or OLTP), that DSS workloads benefit more from out-of-order processors with increased instruction-level parallelism than OLTP, and that memory stalls are a major bottleneck. Ailamaki et al. [1] introduces a framework of analyzing query execution time on a DBMS running on a server with a Intel Xeon processor and memory architecture, and then examines four commercial DBMSs running on workstation. Furthermore, they studied OLTP workloads and found that DBMS performance on modern processors have narrowed the primary memory-related bottlenecks to L1 instruction and L2 data cache misses.

All these research work were based on the processor at that time. In a most recent research [4], G.Saylor et al. used cycle time accounting to outline the performance constraints of a general OLTP workload on a large ccNUMA platform and describe targeted optimization on the platform.

3. METHODOLOGY

In this section, we first introduce a simple analysis framework proposed by Ailamaki et al.[1], which is the most widely used method to analyze the query execution time breakdown of DBMSs on modern computer platforms. It is important to note that we used the same metrics to facilitate experimental analysis but the focus is different. We then describe the experimental setup and the methodology we used.

3.1 Query Execution Time Breakdown

The pipeline is the basic model of modern processor, which receives an instruction then executes it and stores results into memory. The pipeline works in several sequential stages, and each involves some functional components. An operation at one stage can overlap with operations at other stages. When an operation can not be able to complete immediately, this will cause delay ("stall") in pipeline. Modern processor will cover the stall time by doing other work through using three techniques: non-blocking caches, out-of-order execution and speculative execution with branch prediction.

Even with these techniques, the stalls cannot be fully overlapped with useful computation. Thus, the time to execute a query (T_Q) includes a useful computation time (T_C), a stall time because of memory stalls (T_M), a branch misprediction overhead (T_B), and resource-related stalls (T_R). The latter are due to execution resources not being available, such as functional units, buffer space in the instruction pool, or registers. As discussed above, some of the stall time can be overlapped (T_{OVL}). Thus, the following equation holds [1]:

$$T_Q = T_C + T_M + T_B + T_R - T_{OVL} \quad (1)$$

Table 1 shows the time breakdown into smaller components based on Itanium II architecture. The D_{TLB} and I_{TLB} (Data or Instruction Transaction Lookaside Buffer) are page