Highly Functional Memory Architecture for Main Memory Database

Tomoharu Fukawa (110110)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 14, 2003

Keywords: main memory database, DRAM, memory controller, query processing.

1 Introduction

The response time in database systems is getting large because of the growing gap between speed of a CPU and that of a memory, and the increase in data size. It is thus important to accelerate query processing.

The access time for MMDB is orders of magnitude less for database systems on disk and MMDB is suitable for random access. In order to have an increase in speed up of query processing, in this paper, we propose the data transfer methods which sequentially export a data from a memory to a CPU and obtain a data at an access to memory made one through a pointer. We add these mechanisms to memory controller （MC） and evaluate them in simulations for query processing.

2 Main memory database（MMDB）

Each cell in relational data structure for MMDB is stored address （that is, pointer） to entity due to saving the amount of the memory usage. But a data smaller than pointer is directly stored in it. From such the characteristic of the relation, when the matching operation execute, the identification of data is distinguished by comparison between pointers but the measurement of data is needed by one between entities through the pointer.

3 Data transfer methods

In this section, we describe the fast and large scale data transfer methods.

3.1 Stride Data Transfer（SDT）

By adding the following mechanism to a conventional MC the data in main memory are sent a processor. Before SDT starts, processor sets the number and interval of data to the registers in the MC. Then processor issues a memory request to the MC. the MC sets the address for the memory request to the register in one and sends a read data from DRAM to the processor. After that, it generates the next column address by adding the previous column address and the value of interval and increments the value of the register that indicates the number of data, and then sends a read data to the processor. This process is repeated. The condition of SDT completion is the cases where the number of data read is over one of data that set the registers in the MC, the column address exceeds the size of a bank or a row in DRAM and the column address steps over the page size supported by OS. When an access request to main memory results form cache miss may occur, SDT should suspend until the processor issues a SDT request again.

3.2 Two-Phase Data Transfer （TPDT）

In this method a entity is handled as a character string. It is the fast data transfer method that reduces access time to entity （that is, the access to memory trough pointer at two phase） to save a memory resource. The MC to implement TPDT gets the pointer in the first access to memory but not transfer it to processor and in second access sends a string data read from DRAM to the processor by using the pointer. The completion of TPDT is the cases where the MC detects a NULL character itself or the processor recognizes the difference in entities in the middle of comparison of them.

4 Evaluation／Consideration

We evaluate the effect on the two methods in simulate for query processing. This simulator is the one that executes the assembler source code of a SPARC processor. It assumes the execution cycle of an instruction is 1 CPU clock cycle. We perform each query processing to count the total execution CPU cycle and compare to conventional MC. The relation for evaluation is a Wisconsin Benchmark[2] modified for MMDB. Data read from DRAM are to be inserted the FIFO buffer[3] that reconfigures conventional cache in the processor. When it is applied SDT mechanism, the MC prefetches data read from main memory to FIFO buffer in parallel with the execution of the instructions. Therefore the processor can read date without suffering from the penalty of access to memory.

When it is applied TPDT mechanism, the total number of execution cycles will change with different sizes and the kind of string. If the cache is large enough to accommodate all of entities and pointers to them, it is not effective in introducing TPDT mechanism and if not it is effective. Moreover, if the entity is large size, in conventional MC accesses to main memory may occur in the middle of matching operation but in TPDT methods not.

5 Conclusion

In this paper, we propose the fast and large scale data transfer methods that take advantage of the data structure in MMDB and the characteristics of DRAM to execute fast query processing for large data size in database systems, and add these mechanisms to a conventional MC.

As the result of simulations for query processing, when a cache size is larger than a relation size, which data are reused, it is ineffective in two methods because of temporal and space locality. SDT absorbs the gap between speed of a processor and a main memory in matching operation for data which have no temporal and space locality. In TPDT when the entity are small enough to accommodate one block in the cache, by using FIFO buffer the processor can continue the matching operation with string entities.

Thus we conclude that the two methods improve the performance of query processing.

References

[1] H. Garcia-Molina, K. Salem, "Main Memory Database Systems: An Overview." IEEE Trans. on Knowledge and Data Engineering, Vol.4, No.6, pp.509–516, 1992.

[2] DeWitt. D. J, "The Wisconsin Benchmark:Past, Present, and Future." The Benchmark Handbook, pp.269–316, J.Gray ed., Morgan Kaufmann, 1993.

[3] Khairuddin bin Khalid and Kiyofumi Tanaka, "Implementation of FIFO Buffer Using Cache Memory." IPSJ SIG Notes, ARC, Vol.2002, No.112, pp.83–88, 2002.