Database Techniques for New Hardware

D

Xiongpai Qin
Renmin University of China, China

Yueguo Chen
Renmin University of China, China

INTRODUCTION

In the last decade, computer hardware progresses with leaps and bounds. The advancements of hardware include: widely application of multi-core CPUs, using of GPUs in data intensive tasks, bigger and bigger main memory capacity, maturity and production use of non-volatile memory etc.

Database systems immediately benefit from faster CPU/GPU and bigger memory, and run faster. However, there are some pitfalls. For example, database systems running on multi-core processors may suffer from cache conflicts when the number of concurrently executing DB processes increases. To fully exploit advantages of new hardware to improve the performance of database systems, database software should be more or less revised (Ailamaki, 2004).

This chapter introduces some efforts of database research community in this aspect. And the following section gives a brief introduction to some new hardware technologies that database systems could utilized.

BACKGROUND

Multi-Core CPU

Improving the performance of CPU through increasing its clock frequency becomes more and more difficult. Researchers and engineers bring forth multi-core technology. In a typical multi-core CPU, 2, 4, 8 or more cores are integrated on one chip. Putting several cores on one die allows for higher communication speeds between the cores, which will benefit many computing tasks.

The cores have their own private caches (Level1 Cache, or L1 Cache), and share some larger but slower caches (L2 Cache). They access the shared main memory for parallel data processing. Database systems are basically multi-threaded, and they can benefit from multi-core CPUs without any modification. However, to fully utilize the cores to boost database performance, there is much work to do.

GPU for General Tasks

GPU is traditionally used to accelerate the specific task of graphic rendering. GPU venders have integrated many computing units in a single die, and optimized the bandwidth to process large volume of graphic data. The highly parallelism of GPUs is exploited to speed up data intensive tasks as well, and GPU becomes GPGPU (General Purpose GPU).

Since GPU is designed primarily for graphic processing tasks instead of general tasks, the architecture of a GPU is rather different from CPU. It has its own unique thread hierarchy and memory hierarchy, which should be taken into account when using GPU for data processing tasks.

Bigger Memory

The price of memory is going down, now people can install more memory in a single server. It is rather common for a single server to possess a memory capacity as large as hundreds of giga-

DOI: 10.4018/978-1-5225-2255-3.ch169

bytes, or even up to terabytes. For moderate-size applications, it is possible to load the whole dataset into memory for fast access.

Non-Uniform Memory Access

Non-Uniform Memory Access (NUMA) machine is becoming more and more common. The NUMA architecture consists of a small number of processors, each having its own memory and I/O channels. Each group of CPUs is called a 'node'. Memory that is local to a node is called local or near memory, while memory outside of a node is called foreign or far memory (Golding, 2010). Accessing foreign memory is much slower than accessing local memory. NUMA architecture requires changes of memory management.

Non-Volatile Memory, Solid State Disks, and Changes of the Memory Hierarchy

Non-volatile memory (NVRAM) is a type of computer memory that can keep the information even after the power if turned off. There are several flavors of NVRAM, including flash memory, and phase change memory (PCM) etc.

Flash memory devices use two different logical technologies, i.e. NOR and NAND, to map data. NOR flash reads and writes data in specific memory locations. It provides high-speed random access and it can retrieve a single byte. NAND flash reads and writes sequentially at high speed in small blocks called pages. In general, reading data from flash memory is much faster than writing data into it.

Phase-change memory (PCM) is a form of computer RAM (Random Access Memory). It stores data by altering the state of the matter from which the device is made. PCM is byte addressable. It incurs very low power consumption. And It has the potential to provide inexpensive, high speed, and high volume non-volatile storage.

A solid state disk (SSD) uses integrated circuit as memory to store data permanently. SSDs have no moving parts, they are more resistant to physical shocks, run silently, have lower access time and latency, and consume much less power. SSD is usually made of flash memory. With the price of flash memory continues to decline, the price of SSDs is also declining over time.

With advent of non-volatile memory and SSDs, traditional memory hierarchy has been enriched with new intermediate levels. Recent and hot data could be kept in high speed memory levels, which is near to CPU, and historical and cold data could be migrated to low speed memory or disks.

MAIN FOCUS OF THE ARTICLE

Database researchers have devised many new techniques to utilize characteristics of above introduced hardware to improve performance of database systems. we classify these techniques into several categories, i.e. storage, index, query processing, transaction processing & concurrency control, and recovery.

Storage and Compression and Migration

Data Layouts

Traditional database systems are designed around disks. Data is primary stored in disks, and it is loaded into memory buffer for later process. Now the memory capacity is so big, and it could be used as primary store of data. The whole dataset or most data can be kept in memory for faster access. Outdated or unused data is occasionally evicted to disks to make room for data that will be accessed soon (Zhang, Chen, Ooi, Wong, Wu, & Xia, 2015). The design principle is called "anti-caching". In these scenarios, the optimization focus has shifted to how to efficiently exchange data between main memory and CPU cache for faster processing.

For a relational table, data could be stored in three styles of layouts: row-wise, columnar, and