Main Memory Databases

Matthias Meixner
Technische Universität Darmstadt, Germany

INTRODUCTION

For a long time, hard disks were the only technology that could store enough information to hold a database and offered random access at the same time. Therefore, conventional database management systems were tuned to take this technology to the maximum. But in recent years, main memories have become cheaper and grown to a point that for some fields of application it allows one to keep the whole information of a database in main memory and therefore speed up operation. This article focuses on the differences in conventional databases that affect both performance and internal structure of a database management system.

BACKGROUND

When storing information in main memory, many design decisions that were based on disk storage are not valid anymore, and different steps have to be taken to achieve maximum performance since main memory and disk storage have very different access performance characteristics. The access time of main memory is of orders of magnitude smaller than for disk storage, but on the other hand main memory is volatile, while disk storage is not. Disk accesses exhibit a high fixed cost per access due to seek time. Therefore, to achieve good performance, accesses should be sequential and transfer large amounts of needed data, i.e., data layout on disk is critical. In contrast, the access time of main memory is less dependent on the location, and therefore data layout is far less critical, although this is also changing since the access time of on-processor caches is improving faster than the access time of main memory. These differences have effects on almost every aspect of database management. We will discuss this in the following sections.

MAIN THRUST

Main Differences to Disk Resident Databases

Both conventional disk resident database management systems (DRDBMSs) and main memory database management systems (MMDBMSs) process data in main memory, and both keep a (backup) copy on disk. If the cache of a DRDBMS is large enough, sooner or later the whole database will be in cache, and an MMDBMS needs to store a backup copy on disk so that it is able to recover from failures. So what is the main difference between a DRDBMS and an MMDBMS? The key difference is that in an MMDBMS the primary copy of the database lives permanently in main memory (Garcia-Molina & Salem, 1992), and this has important implications for the algorithms and data structures used. Even if the whole database of a DRDBMS is cached in main memory, it will not provide best performance since a DRDBMS is not tuned for this case. A DRDBMS cannot rely on data being present in main memory at all times. Therefore, each data access has to go to the buffer manager to make sure data is still in main memory. Index structures in DRDBMSs are designed for disk access and may trade computing power and storage efficiency for a lower number of disk accesses since disk access is the processing-time dominating factor in DRDBMSs. This overhead incurs even if all data is cached in main memory. In MMDBMSs the situation is different: Since data is guaranteed to stay present in main memory, index structures and all the other parts of the database do not need to consider disk access and can be tuned for low computational cost.

Performance and Data Structures

In DRDBMSs disk accesses have traditionally been the bottleneck since in the time required for one single disk access, a processor can perform up to several million instructions. Therefore, the most important optimizations in disk-based systems are to reduce the number of disk accesses, to prefer sequential access, and to keep the processor busy while waiting for I/O. To reduce the number of disk accesses, caching is used, and special index structures, like the B+ tree, were developed. Data that is used together in a database is grouped on disk to be able to access it using one single sequential read. A high degree of concurrency is employed to keep the processor busy while other transactions are waiting for I/O, and therefore a small locking granularity down to record level locking is used.

In the case of MMDBMSs all these optimizations are not relevant anymore; instead, performance is only determined by the CPU efficiency of the algorithms used.