Understanding, modeling, and improving main-memory database performance

Manegold, S.

Citation for published version (APA):
Chapter 6

Summary and Outlook

Database systems pervade more and more areas of business, research, and everyday life. With the amount of data stored in databases growing rapidly, performance requirements for database management system increase as well. This thesis analyzes, why simply using newer and more powerful hardware is not sufficient to solve the problem. Quantifying the interaction between modern hardware and database software enables use to design performance models that make up vital tools for improving database performance.

6.1 Contributions

Focusing on main-memory database technology, this thesis describes research that aims at understanding, modeling, and improving database performance.

Hardware Trends and MMDBMS performance

We empirically analyzed how main-memory database software and modern hardware do interact. During our study, we found that starting with simple but representative experiments is both necessary and sufficient to identify the essential performance characteristics. Moreover, hardware counters as present in many contemporary CPUs have proved to be indispensable tools for tracking down the most performance-crucial events, such as cache misses, TLB misses, resource stalls, or branch mispredictions. Our experiments disclosed that memory access has become the primary performance bottleneck on various hardware platforms ranging from small of-the-shelf PCs to large high-performance servers. The reason for this can be found in the hardware trends of the last two decades. While CPU speed has been growing rapidly, memory access latency has hardly improved. Thus, the performance gap between CPU and memory speed did widen out. Ongoing and promised hardware development indicates that this trend will last for the foreseeable future. So far, database technology has not payed much attention to these trends; especially in cost modeling, main-memory access has completely been ignored.
Main-Memory Cost Modeling

Using the knowledge gained during our analysis, we developed precise physical cost models for core database algorithms. The main goal of these cost models is to accurately predict main-memory access costs. Furthermore, the new models provide two further innovations compared to traditional database cost models. First, we generalized and simplified the process of creating cost functions for arbitrary database operations. For this purpose, we introduced the concept of data access patterns. Instead of specifying the often complex cost functions for each algorithm "by hand", database software developers only need to describe the algorithms' data access behavior as simple combinations of a few basic access patterns. The actual cost functions can then be derived automatically by applying the rules developed in this thesis. The second innovation addresses the hardware dependency inherent to physical cost models. The principle idea is to have a single common cost model instead of individual cost models for each hardware platform. To achieve this goal, we introduced a novel unified hardware model for hierarchical memory systems. The hardware model allows an arbitrary number of hierarchies, such as several levels of CPU cache, main-memory, and secondary storage. For each level, it stores performance characteristic parameters such as size, access granularity, access latency, access bandwidth, and associativity. Thus, we could design a general cost model that is parameterized by the characteristics as represented in the hardware model. In order to have the cost model automatically instantiated on a new hardware platform, we developed a calibration tool that measures the respective parameters without human interaction. The "Calibrator" is freely available for download from our web site and has become a popular\footnote{more than 12,000 downloads in 2 years} tool to assess computer hardware not only within the database community.

Cache-Conscious Query Processing

Disappointed by the poor performance of standard database technology on supposedly powerful hardware, but enlightened with the insights gained during our analysis and modeling exercises, we developed new hardware-conscious algorithms and coding techniques. Focusing on equi-joins, we introduced radix-algorithms for partitioned hash-join. The basic idea is to reduce memory access costs by restricting random access patterns to the smallest cache size, and thus reduce the number of cache misses. This exercise demonstrates that our cost models — next to being fundamental for database query optimization — serve two further purposes. First, they help us to understand the details and thus enable us to design proper algorithms. Second, the query engine can use the cost functions to tune the algorithms at runtime.

Once memory access is optimized, CPU costs become dominant, mainly as standard database code is too complex to allow CPUs to efficiently exploit their internal parallel processing potentials. We presented coding techniques that achieve significant performance gains by reducing the number of function calls, branches, and data dependencies.
We have performed extensive experimentation on a range of hardware platforms that confirmed both the accuracy and the portability of our cost model. Moreover, the experimental evaluations showed, that combining memory access and CPU optimization yields a performance gain of up to an order of magnitude.

### 6.2 Conclusion

The work presented in this thesis has demonstrated that current main-memory database technology needs to be adapted to achieve optimal performance on modern hardware. Next to data structures, algorithms, and coding techniques, cost models turned out to be a crucial factor that has not received much attention hitherto in main-memory context. Our analysis has shown, that understanding and modeling database performance in a main-memory dominated scenario is not only possible, but also a fundamental requirement for developing and tuning new techniques to improve performance.

While the main-memory dominated scenario looks similar to the classical I/O dominated scenario, but shifted one level up, traditionally successful solutions concerning cost models and algorithms do not work as efficient as one might expect. Small but significant details, such as the increased and now variable number of memory hierarchies, limited associativity and fixed size of hardware caches, and the fact that the cache replacement strategy is not under the control of the DBMS, require modified or completely new techniques. In this thesis, we explain how to solve these problems. Moreover, this thesis represents a new departure for database research and development into taking account of memory, cache, and processor characteristics, hitherto primarily the concern of optimizing compiler writers and operating system researchers.

### 6.3 Open Problems and Future Work

Like almost every research also the work presented here leaves some questions unanswered and even discovers new problems. Some of these open issues shall be mentioned here.

**Portability.** Concerning portability of our cost models, we focused on porting the models to various hardware platforms. Another aspect of portability is whether—and/or to which extend—our models can be successfully applied to other main-memory or even disk-based database systems. While the general framework should be flexible enough to cover the needs of such systems, the main question is whether it will achieve the same accuracy as we experienced in our experiments with Monet. Especially the capabilities to cope with various buffer pool implementations and buffer management strategies in order to accurately predict I/O costs need to be evaluated.
Basic Access Patterns. In the current set of basic access patterns, the interleaved multi-cursor pattern plays a kind of special role, being more complex than the other basic patterns. From an aesthetic point of view, it might be worth to investigate, whether it could be replaced by a compound pattern made up of a proper combination of simple sequential and random patterns that reflects the same knowledge about the pattern performed by algorithms doing partitioning or clustering.

On the other hand, new algorithms or other systems, might need more or different basic access patterns. Similar to the interleaved multi-cursor access pattern, there might be algorithms that allow to deduce more detailed properties of their access pattern(s) than simply being sequential or random. Within the framework we presented, it is simple to add new basic access patterns together with their respective basic cost functions.

In the current setup, basic access patterns are limited to the knowledge and properties that we can derive from the algorithms without knowing anything about the actual data that eventually is to be processed. A possible way to enrich this knowledge, and hence to improve the accuracy of the accompanying basic cost functions would be to combine or parameterize the basic access patterns with some information about the data distributions. The problem is that the cost functions need to be provided statically while the actual data distributions are only known at runtime. A solution might be to use a coarse classification of data distributions and provide cost functions for each class. At runtime, the actual data distribution is matched with one of the classes and the respective cost function can be used.

Logical Costs / Volume Estimation. Focusing on physical cost models, we assumed a “perfect oracle” for estimating intermediate result sizes in this thesis. In practice, any of the numerous techniques for volume, respectively selectivity, estimation proposed in literature (cf., Section 2.2) could pair with our physical cost models to make up a complete database cost model. Recall, logical costs, and hence their models, are independent of the physical design of a DBMS. However, especially in case of Monet, main-memory processing and the decomposed storage model might offer new opportunities to efficiently build and maintain synopses such as histograms or samples. These opportunities need to be explored.

An other idea derives from the bulk-processing approach of Monet. As all intermediate results are materialized, it is simple to create synopses directly from the exact intermediate results with little extra effort. These synopses can then be used for volume estimation of future queries that contain the same subexpressions. By using the exact intermediate results to build the synopses, this approach eliminates the problem that the estimation error grows exponentially throughout the query plan. Two obvious questions remain open to be answered by future research: (1) How can the synopses in this scenario efficiently be updated in case the underlying base tables get updated? (2) How can these synopses be used for volume estimation of queries that do not contain exactly the same subexpressions, but “similar” ones, e.g., a selection predicate on the same attribute(s) but with (slightly) different parameters?