Cache Conscious Data Layouting for In-Memory Databases

Holger Pirk, CWI, Amsterdam <holger.pirk@cwi.nl>

The Data Access Performance of an In-Memory Database largely depends on the arrangement of the data in memory: the Layout. This is due to the principle of Data Locality, which states that the contemporary access to data items that are stored close to each other in memory is significantly faster than accessing data items that are distributed all over the memory. The importance of Data Locality for access performance has been recognized by researchers and spawned an interesting development: Column-oriented DBMSs. For OLAP applications, that usually access the values of many tuples but few attributes, traditional row-oriented DBMSs perform suboptimally because they do not provide good Data Locality for these applications. They do, however, provide good data locality for OLTP applications which is why today most transactional systems are still row-oriented. For mixed workloads, applications that include OLTP as well as OLAP queries, neither approach is optimal. A Hybrid DBMS that stores each data item in the most suitable way increases data locality and, thus, performance. To decide where to store each data item, a good layout with respect to properties of the underlying hardware and the stored data itself has to be found.

The goal of our work is to find a good layout at acceptable costs/time. This poses two major problems. The first Problem is the high optimization effort: Even when limiting the considered layouts to vertically partitioned ones, the number of layouts still grows exponentially with the number of attributes of the relational schema. Thus, extensive searching for the optimal layout is not an option. The second problem is the cost estimation: to evaluate the suitability of a layout for a given workload the costs of executing the workload on the layout have to be determined. Actually executing the workload on the layout, however, leads to unacceptable costs.

We will present our Solution that is based on the existing Generic Cost Model for Hierarchical Memory Systems. The model was extended to allow an accurate prediction of the costs of a workload on a layout, it takes hardware and data properties into account. We evaluated our model by estimating the costs of a number of microbenchmarks and comparing them to their C-implementation. For these simple cases, the accuracy of the model is very high. Based on that model, we developed two optimization algorithms that exploit mathematical properties of the model to find the (analytically) optimal layout at reasonable optimization costs. We applied our optimization algorithms to a benchmark that is based on the SAP Sales and Distribution Benchmark. In a prototypical implementation of a hybrid DBMS, it shows a performance improvement by a factor 2.3 compared to a row and 1.8 compared to a column-store.

The next step is to integrate hybrid storage into a full-fledged DBMS and evaluate the performance benefits for real applications.