Main Memory Adaptive Denormalization

Zezhou Liu Stratos Idreos

Harvard University

{zezhouliu@college, stratos@seas}.harvard.edu

ABSTRACT

Joins have traditionally been the most expensive database operator, but they are required to query normalized schemas. In turn, normalized schemas are necessary to minimize update costs and space usage. Joins can be avoided altogether by using a denormalized schema instead of a normalized schema; this improves analytical query processing times at the tradeoff of increased update overhead, loading cost, and storage requirements.

In our work, we show that we can achieve the best of both worlds by leveraging partial, incremental, and dynamic denormalized tables to avoid join operators, resulting in fast query performance while retaining the minimized loading, update, and storage costs of a normalized schema.

We introduce adaptive denormalization for modern main memory systems. We replace the traditional join operations with efficient scans over the relevant partial universal tables without incurring the prohibitive costs of full denormalization.

1. INTRODUCTION

Normalized schemas are standard in database systems [3][4]. Normalization leads to many desirable characteristics such as enabling efficient and accurate updates, minimizing data redundancy, reducing storage requirements and loading costs [4][6]. However, normalization requires join operations with expensive data access patterns and computational costs to operate over the normalized schema. Despite advances in modern hardware capabilities and join algorithms, the join operator continues to dominate query processing time even in systems that are well-tuned for high performance [1][2][5].

For example, Figure 1 compares a state-of-the-art hardware optimized hash join implementation [1] over a normalized schema against a fast scan (multi-core, numa-aware, SIMD) over a denormalized one. Within the time needed to join just 100 million tuples over the normalized schema we can scan and perform a logical join across more than 100 times the data in the denormalized schema.

Contributions. In this paper we introduce the idea of adaptive denormalization, in which the base data lies in a normalized state while hot data is adaptively and partially denormalized on-demand. This allows our system to operate within a given memory budget and mitigate the negative side-effects of denormalization while still producing the performance gains of denormalized systems. We show that adaptive denormalization achieves performance gains of orders of magnitudes over systems that use strict normalized schemas.

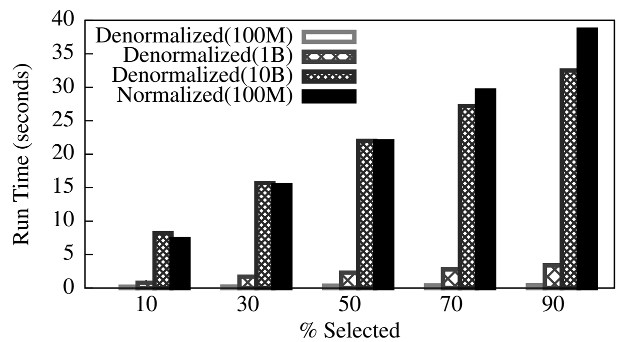

Figure 1: In the time it takes to join inputs of 100 million rows in a normalized schema, we can perform a (logical) join by scanning over 10 billion rows in denormalized schema. The disparity is larger when a higher percentage of rows are selected.

2. ADAPTIVE DENORMALIZATION

Adaptive denormalization achieves the best characteristics of both normalization and denormalization by exploiting embarrassingly parallel scans over a denormalized schema to process join queries while still achieving the efficient space utilization, updates, and loading time characteristics found in normalized schemas.

Adaptive denormalization maintains data in a normalized state and denormalizes only regions of the data as they are queried and to only data that has not yet been denormalized by previous queries. As a result, future queries on any previously queried range can be answered with scans, thereby avoiding expensive join operations. These denormalized regions form partial universal tables – auxiliary data structures that are maintained alongside the underlying normalized data.

By denormalizing only the data that is touched, we limit the extra storage requirements to only attributes of interest. Furthermore, adaptive denormalization operates within the given memory budget by dropping regions of the partial universal table in response to memory pressures. Moreover, denormalizing during query processing amortizes the overhead and cost of denormalization across many queries. Loading costs are the same as in normalized schemas. Since the denormalized data is logically separated into partial universal tables, updates can be applied lazily to only the partial universal tables that are required by the query, fur-

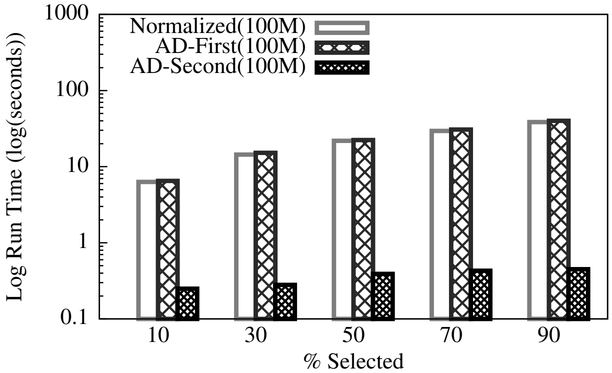

Figure 2: Adaptive denormalization (AD) improves significantly over repeated join patterns without penalizing the first join queries.

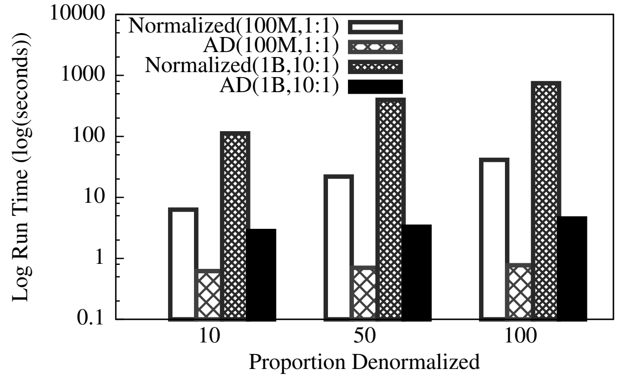

Figure 3: Adaptive denormalization (AD) achieves significant benefits even when the required data is only partially denormalized.

ther mitigating update costs that might otherwise be prohibitive in a fully denormalized system. These properties allow us to leverage the benefits of both denormalization and normalization without suffering the negative characteristics.

3. ANALYSIS

In this section we present a brief experimental analysis to show the performance advantages of adaptive denormalization over traditional join operations over normalized schemas.

Experimental Setup. We run our experiments on an in-memory column store prototype with modern multi-core scan and hash joins implementations. We evaluate our system on a 4-way Intel Xeon E7-4820 configuration with 64 hardware threads and 1 TB of main memory. We evaluate our system on synthetic workloads to meet certain selectivity, input size, and join output size criteria. We use 8-byte ints generated from random uniform distributions, which we tune to control the sizes of input columns and join outputs.

Performance Gains of Adaptive Denormalization. A query benefits from adaptive denormalization if its results are partially contained in the universal tables, since the query can be evaluated with fast sequential scans and only require joins for the parts where the data is being queried for the first time. Joins can be avoided altogether if the universal table contains all the data necessary to answer the query. Figure 3 reveals that this strategy results in orders of magnitudes in speed-up, especially when the join operation is large and the data is already contained in the universal table. For example, the joins over a normalized schema between 100M tuples with 100M tuples for a join output of 100M (output:input ratio of 1:1), requires 38.6 seconds, whereas a scan over the equivalent denormalized 100M output tuples requires only 0.45 seconds.

This speed-up is further magnified as the size of the join output increases. In Figure 3 we see that at a join output size of 1B (output:input ratio of 10:1), the traditional operator takes 6 minutes, whereas adaptive denormalization takes only 5.1 seconds. Figure 3 also shows that there are benefits even when only part of the query is contained in the current denormalized tables.

Small Overhead of Adaptive Denormalization. The overhead of adaptive denormalization can be separated into performance and storage overhead. The performance overhead results from the additional book-keeping required to track which parts of the queries can be answered using the partial universal table and which parts still need to be joined. Figure 2 compares the performance of a join over a normalized schema with that of our modified adaptive denormalization join when the universal table is empty and a join cannot be avoided (Normalized vs AD-First). In these cases, the overhead still represents only 5-10% of total query time.

Storage overhead results from the additional storage required for the partial universal tables. However, adaptive denormalization operates within the given memory budget by dropping regions of the partial universal table if they are no longer needed by the workload. This is an inexpensive solution since future reconstruction is relatively cheap as shown in the aforementioned performance overhead. Our lazy update technique also makes update overhead small; further details are left for a full future paper.

4. CONCLUSION & FUTURE WORK

Joins have been one of the primary performance bottlenecks in relational database systems for the past five decades. In this paper, we show how to effectively eliminate join costs for the hot part of the workload. We present adaptive denormalization to provide a way to achieve the best of both schemas: faster query processing via scans instead of joins as in a denormalized schema and at the same time minimum loading, storage and update overheads as in a normalized schema. Arbitrary joins, not just equi-joins, can benefit from this technique as long as subsequent queries share the same join condition. We show how adaptive denormalization in a modern column-store achieves gains of orders of magnitude over traditional strict systems that rely on normalized data or on a priori full denormalization.

Our ongoing plans include work on handling updates with limited impact on read and write performance and evaluating our system on real workloads. We also work on handling some limitations of adaptive denormalization, such as cases when performing a scan on the Universal Table is slower than its equivalent join.

Acknowledgements. The authors of this paper would like to thank Michael Kester and Wilson Qin, members of the Harvard DASlab, for their help on this project.

This work is partially supported NSF Grant No. IIS-1452595.

5. REFERENCES

[1] C. Balkesen, G. Alonso, J. Teubner, and M. T. Özsu. Multi-core, main-memory joins: Sort vs. hash revisited. *PVLDB*, pages 85–96, 2013.

[2] C. Balkesen, J. Teubner, G. Alonso, and M. T. Özsu. Main-memory hash joins on multi-core cpus: Tuning to the underlying hardware. In *ICDE*, pages 362–373, 2013.

[3] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, pages 377–387, 1970.

[4] G. Gottlob, R. Pichler, and V. Savenkov. Normalization and optimization of schema mappings. *VLDB*, pages 277–302, 2011.

[5] V. Raman et al. Constant-time query processing. In *ICDE*, pages 60–69, 2008.

[6] G. Sanders and S. Shin. Denormalization Effects on Performance of RDBMS. In *HICSS*, pages 9–17, 2001.