Editorial Scope

Topics

Foundations and Trends® in Databases covers a breadth of topics relating to the management of large volumes of data. The journal targets the full scope of issues in data management, from theoretical foundations, to languages and modeling, to algorithms, system architecture, and applications. The list of topics below illustrates some of the intended coverage, though it is by no means exhaustive:

- Data models and query languages
- Query processing and optimization
- Storage, access methods, and indexing
- Transaction management, concurrency control, and recovery
- Deductive databases
- Parallel and distributed database systems
- Database design and tuning
- Metadata management
- Object management
- Trigger processing and active databases
- Data mining and OLAP
- Approximate and interactive query processing
- Data warehousing
- Adaptive query processing
- Data stream management
- Search and query integration
- XML and semi-structured data
- Web services and middleware
- Data integration and exchange
- Private and secure data management
- Peer-to-peer, sensor net, and mobile data management
- Scientific and spatial data management
- Data brokering and publish/subscribe
- Data cleaning and information extraction
- Probabilistic data management

Information for Librarians

Foundations and Trends® in Databases, 2016, Volume 8, 4 issues. ISSN paper version 1931-7883. ISSN online version 1931-7891. Also available as a combined paper and online subscription.
Main Memory Database Systems

Franz Faerber
SAP
franz.faerber@sap.com

Alfons Kemper
Technische Universität München
alfons.kemper@in.tum.de

Per-Åke Larson
Microsoft Research
palarson@acm.org

Justin Levandoski
Microsoft Research
justin.levandoski@microsoft.com

Thomas Neumann
Technische Universität München
thomas.neumann@in.tum.de

Andrew Pavlo
Carnegie Mellon University
pavlo@cs.cmu.edu
Contents

1 Introduction 2

2 History and Trends 6
  2.1 History 6
  2.2 Trends 10

3 Issues and Architectural Choices 12
  3.1 Data Organization and Layout 12
  3.2 Indexing 18
  3.3 Concurrency Control 23
  3.4 Durability and Recovery 28
  3.5 Query Processing and Compilation 31

4 Systems 35
  4.1 SQL Server Hekaton 35
  4.2 H-Store and VoltDB 47
  4.3 HyPer 62
  4.4 SAP HANA 86
  4.5 Other Systems 101

5 Active Research and Future Directions 106
  5.1 Cold Data Management 106
Abstract

This article provides an overview of recent developments in main-memory database systems. With growing memory sizes and memory prices dropping by a factor of 10 every 5 years, data having a “primary home” in memory is now a reality. Main-memory databases eschew many of the traditional architectural pillars of relational database systems that optimized for disk-resident data. The result of these memory-optimized designs are systems that feature several innovative approaches to fundamental issues (e.g., concurrency control, query processing) that achieve orders of magnitude performance improvements over traditional designs. Our survey covers five main issues and architectural choices that need to be made when building a high performance main-memory optimized database: data organization and storage, indexing, concurrency control, durability and recovery techniques, and query processing and compilation. We focus our survey on four commercial and research systems: H-Store/VoltDB, Hekaton, HyPer, and SAP HANA. These systems are diverse in their design choices and form a representative sample of the state of the art in main-memory database systems. We also cover other commercial and academic systems, along with current and future research trends.
Research and development of main-memory database systems started in the early eighties [37], with several commercial systems appearing in the nineties (e.g., TimesTen [146], P*Time [28], DataBlitz [16]). Many of these systems were used in targeted, performance-critical applications, mainly in telecommunications and finance. The price and capacity of memory during this time period limited applicability of many of these engines, thus main-memory systems did not - at the time - succeed as a general data processing solution.

Recently, two trends have made this field interesting again: memory prices and multi-core parallelism. For the last 30 years memory prices have dropped by a factor of 10 every 5 years. A server with 32 cores and 1 TB of memory now costs around $40K. Machines such as these make it feasible to fit most (if not all) of the world’s OLTP workloads\(^1\) comfortably into memory at a reasonable price. In addition, modern CPUs provide a staggering amount of raw parallelism. Vanilla CPUs contain at least 8 cores, and it is common for modern servers to contain two to four CPU sockets (16 to 32 cores). Core counts continue to rise, with Intel currently shipping a Xeon CPU with 18 cores [1].

Such parallelism coupled with the ability to (practically) store data completely in memory has brought about a recent flurry of research and development into main-memory databases. The result has been astounding. Prominent research systems such as H-Store [142] and

\(^1\)The focus of this survey is primarily on main-memory OLTP databases.
HyPeR [72] reinvigorated research into main-memory and multi-core data processing techniques. Most major database vendors now have an in-memory database solution, such as SAP HANA [137], Oracle TimesTen [74], and Microsoft SQL Server Hekaton [38]. In addition, a number of startups such as VoltDB [143] and MemSQL [2] have carved out a niche in the database vendor landscape.

The result of this research and development is a new breed of database system with a radically different design when compared to a traditional disk-based relational system. These systems abandon many of the “textbook” design tenets in favor of new (or revisited) approaches to achieve high performance on modern hardware. For instance, the following examples provide an idea of how different these systems are:

- **Data organization and indexing.** A pervasive trend in all systems is to avoid page-based indirection through a buffer pool and store only records in memory. Indexes usually store direct pointers to records. Several systems also implement novel indexing methods that optimize for CPU cache efficiency [85] as well as multi-core parallelism using latch-free designs [89].

- **Concurrency control.** Most systems avoid pessimistic lock-based concurrency control due to blocking and context switch overheads. Instead, some systems use a multi-version concurrency control variant [11, 77, 74, 111], while others use partitioned serial execution to achieve high performance [72, 142, 143].

- **Durability and recovery.** Aries-style redo/undo logging and recovery is rarely used. Instead, most systems opt for a form of redo-only logging (or command logging) coupled with periodic database snapshots to recover from a crash or restart [38, 74, 142].

- **Query processing and compilation.** To avoid the overhead of virtual function calls, degradation of branch prediction, and byte interpretation, several systems abandon the “get next” iterator processing model. Instead, queries are compiled into highly optimized machine code and run directly over in-memory records [38, 47, 109].
Introduction

Latch contention plays a large part in limiting the scalability of the traditional engine.

The list above resembles a set topics once thought to be picked over and “closed” in the database research literature. However, over the past several years there has been significant innovation in these and other core areas with the reemergence of main-memory database systems, making it a vibrant and exciting technology space.

The innovations made in main-memory databases come with meaningful performance gains. Figure 1.1 provides an evaluation of the thread-level scalability of the Microsoft Hekaton engine compared to the traditional SQL Server engine running a typical customer workload. Hekaton achieves a roughly 15.7X performance improvement at 12 cores, while the scalability of the traditional engine is limited due to the overheads inherent in a disk-based architecture running a memory-bound workload.

This article provides an overview of the research and development of modern main-memory database systems. We focus our sur-

\footnote{Graph taken from the original paper [38]}
vey mainly on transaction processing engines through the lens of four main-memory systems: (1) *H-Store/VoltDB* [142, 143], a pioneering research system from academic research groups at MIT, Brown, Brandeis, and Yale that subsequently became a commercial database; (2) *Hekaton* [38], Microsoft SQL Server’s main-memory OLTP engine; (3) *HyPeR* [72], a prominent research system from TU Munich that aims to support both high performance OLTP and OLAP workloads in the same engine; and (4) *SAP HANA* [137], the first main-memory optimized engine to ship from a major database vendor. As we will see, each of these systems are diverse in their design choices and form a representative sample of the state of the art in main-memory database systems.

Our coverage focuses on five issues that influence the architecture and design of the system: (a) data storage and layout, (b) indexing and data structure design, (c) concurrency control, (d) durability and recovery, and (e) query processing and compilation. We also summarize the design of other modern commercial and academic main-memory systems.

The organization of this survey is as follows. Chapter 2 summarizes the past research and development of main-memory database systems prior to the “modern era” (starting around 2007). Chapter 3 summarizes the issues used in our system survey. Chapter 4 provides case studies describing the design of our four representative systems. Chapter 5 concludes this survey by summarizing interesting current and future research trends in main-memory databases.
References


References


References


References


References


References


References


References


References


References


References


References


<table>
<thead>
<tr>
<th>Reference</th>
</tr>
</thead>
</table>

Full text available at: [http://dx.doi.org/10.1561/1900000058](http://dx.doi.org/10.1561/1900000058)


Full text available at: http://dx.doi.org/10.1561/1900000058