WHAT'S REALLY NEW WITH NEwsQL
Fast  Repetitive  Small
The Last Decade of Database Systems
Early 2000s — Sharding Middleware

- Custom middleware to combine multiple nodes together into a logical database.
- Route queries to correct node.
Middleware Problems

• Developers spend time writing middleware rather than working on core applications.

• Some features are implemented in application code.
Late 2000s — NoSQL

• Forgo transactional guarantees in order to achieve high-availability and high-scalability.
• Non-relational data models.
NoSQL Problems

• Developers write code to handle eventually consistent data, lack of transactions, and joins.
• Not all applications can give up strong transactional semantics.
The Rise of **NEWSQL Systems**
Aslett White Paper

[Systems that] deliver the scalability and flexibility promised by NoSQL while retaining the support for SQL queries and/or ACID, or to improve performance for appropriate workloads.

Matt Aslett – 451 Group (April 4th, 2011)
https://www.451research.com/report-short?entityId=66963
Stonebraker Article

SQL as the primary interface.
ACID support for transactions
Non-locking concurrency control.
High per-node performance.
Shared-nothing architecture.

New SQL: An Alternative to NoSQL and Old SQL for New OLTP Apps

Mike Stonebraker – Blog@CACM (June 16th, 2011)
http://cacm.acm.org/blogs/blog-cacm/109710
A class of modern relational database systems that provide the same scalable performance of NoSQL systems for OLTP workloads while still maintaining the ACID guarantees of a traditional database system.
Distributed Concurrency Control

- SAP HANA
- memsql
- Clustrix
- Google Spanner
- NUODB
- VOLTDB
Distributed Concurrency Control
Distributed Concurrency Control
Distributed Concurrency Control
Distributed Concurrency Control

INTRODUCTION

The Concurrency Control Problem

Concurrency control is the activity of coordinating concurrent access to a database in a multi-user database management system (DBMS). Concurrency control permits users to access a database in a multi-user, multi-programmed fashion while preserving the illusion that each user is executing alone on a dedicated system. The main technical difficulty in achieving this goal is to prevent database updates performed by one user from interfering with database retrievals and updates performed by another. The concurrency control problem is exacerbated in a distributed DBMS (DDBMS) because (i) users may access data stored in many different computers in a distributed system, and (ii) a concurrency control mechanism at one computer cannot instantaneously know about transactions at other computers.

Concurrency control has been actively investigated for the past several years, and the problem for non-distributed DBMS is well understood. A broad mathematical theory has been developed to analyze the problem, and one approach, called two-phase locking, has been accepted as a standard solution. Current research on non-distributed concurrency control is focused on evolutionary improvements to two-phase locking, detailed performance analysis and optimization, and extensions to the mathematical theory.

Distributed concurrency control is much harder to understand, and difficult to implement. These algorithms are usually complex, hard to understand, and difficult to prove correct. Indeed, many are incorrect. Because they are described in different terminologies and make different assumptions.

Permission to copy without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the ACM (the date of the ACM publication appears). To copy otherwise, or republish, requires a fee and/or specific permission.

Phil Bernstein, et al. – Computing Surveys (June 1981)
http://dl.acm.org/citation.cfm?id=356846
Main Memory Systems

http://dl.acm.org/citation.cfm?id=602261
Hybrid Architectures
Hybrid Architectures

Weaving Relations for Cache Performance

Anastassia Ailamaki 1
Department of Computer Science and Engineering
University of Washington, Seattle, WA 98195, USA
ailamaki@cs.washington.edu

Abstract

Relations, database systems have traditionally optimised for
performance and resource usage separately over the
same database platform. This work demonstrates how
weaving together the two paradigms can provide
performance benefits in modern database systems.

Keywords: database systems, hybrid architectures,
performance, resource usage.

1 Introduction

The combination of both the CPU and the memory
model has been extensively studied as the major
database performance bottleneck. To optimise data
transfer to and from main memory, relational DBMSs
have long required methods to deduplicate pages using
the N-way caching model. This paper reports on a
hybrid storage model, which integrates the two
paradigms.

Weaving relations together, we propose a
hybrid storage model. Our experiments show that this
 approach can improve performance by up to 30% in
some cases.

References

XML and SQL." In Proceedings of the 26th VLDB
Conference.

Anastassia Ailamaki, et al. – VLDB (2001)
http://dl.acm.org/citation.cfm?id=672367
Query Code Compilation

memsql

Google Spanner
Query Code Compilation

memsql

Google Spanner

http://dl.acm.org/citation.cfm?id=320457
Recap

- Distributed Concurrency Control
- Main Memory Stores
- Hybrid Architectures
- Query Code Compilation
The Future of OLTP DBMS Research
Nearly Solved Problems

- Fine-grain Elasticity
- Hybrid Architectures
- Larger-than Memory DBs
Application

Primary Storage

Anti-Cache
Application

Primary Storage

Anti-Cache
Application

Primary Storage

Anti-Cache
TPC-C Benchmark
100% Single-Partition Transactions – 100 Warehouses

MySQ L  H-Store+Anti-Caching

TXN/SEC

DATABASE / MEMORY

50,000
40,000
30,000
20,000
10,000
0
1
2
4
8
TPC-C Benchmark

100% Single-Partition Transactions – 100 Warehouses

- MySQL
- H-Store+Anti-Caching
TPC-C Benchmark

100% Single-Partition Transactions – 100 Warehouses

MySQL vs. H-Store+Anti-Caching

30x
Future Research @ CMU

- Many-Core Concurrency
- Non-Volatile Memory
- Geo-replicated DBMS
Spänner
SpäXner
Oracle, Please Acquire Us
OpauSQL™ — Design Principles

- Don’t treat the DB as a black box.
- Machine learning to understand intra- & inter-txn dependencies.
- Introspection of integrity constraints.
OLTP Application Library

• Examine open source software to create a catalog of application properties.

• Automatically infer optimizations by examining DB access patterns.
Conclusion

• Most NewSQL systems are using known ideas to achieve high-performance for OLTP workloads.
• Database systems research is back at CMU.
END
@ANDY_PAVLO