NewSQL: Flying on ACID

Thanks to David Maier

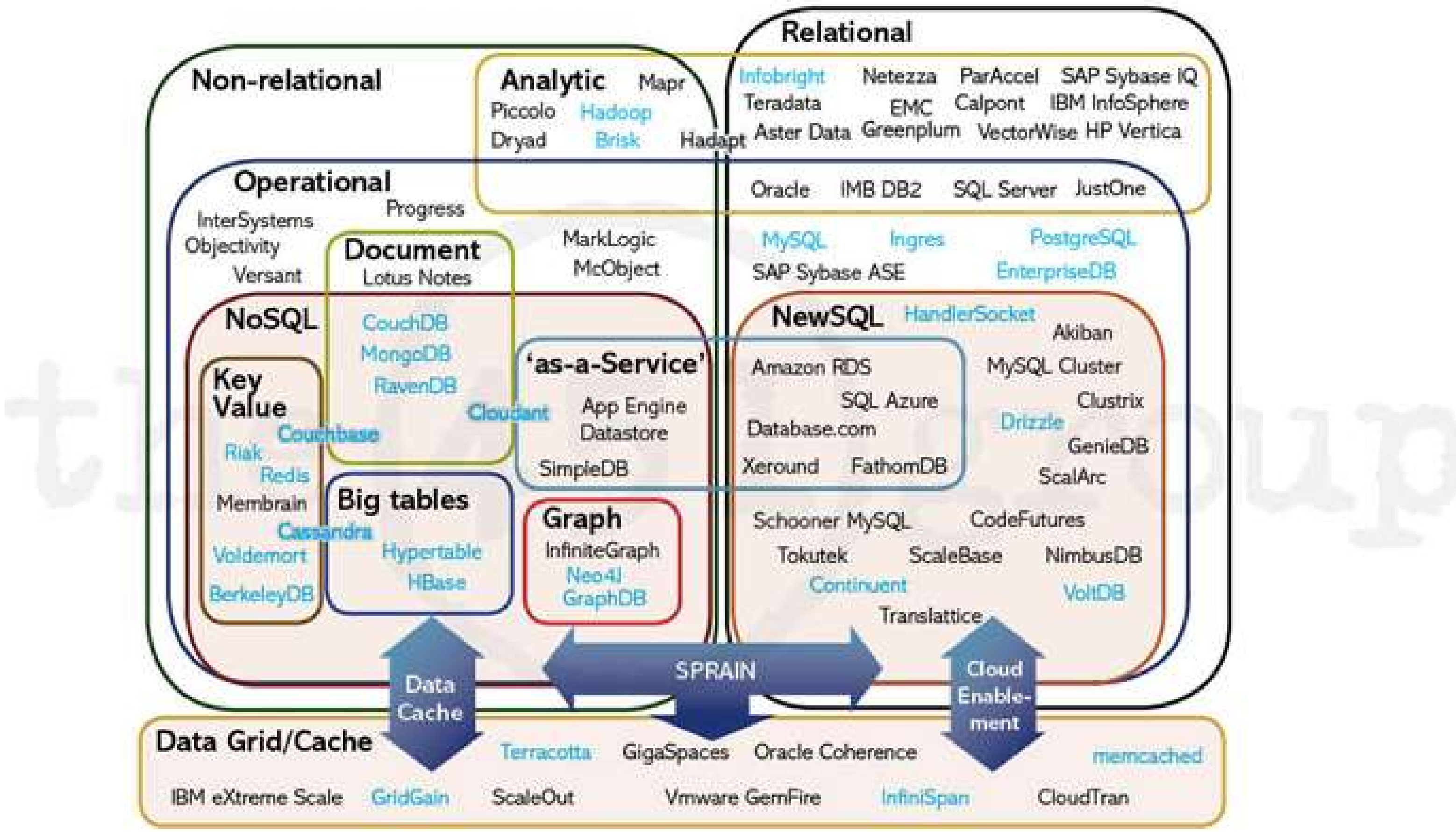
NewSQL

Keep SQL (some of it) and ACID

But be speedy and scalable

Database Landscape

From: the 451 group



OLTP Focus

- On-Line Transaction Processing
- Lots of small reads and updates
- Many transactions no longer have a human intermediary
  - For example, buying sports or show tickets
- 100K+ xact/sec, maybe millions
- Horses for courses

Premises

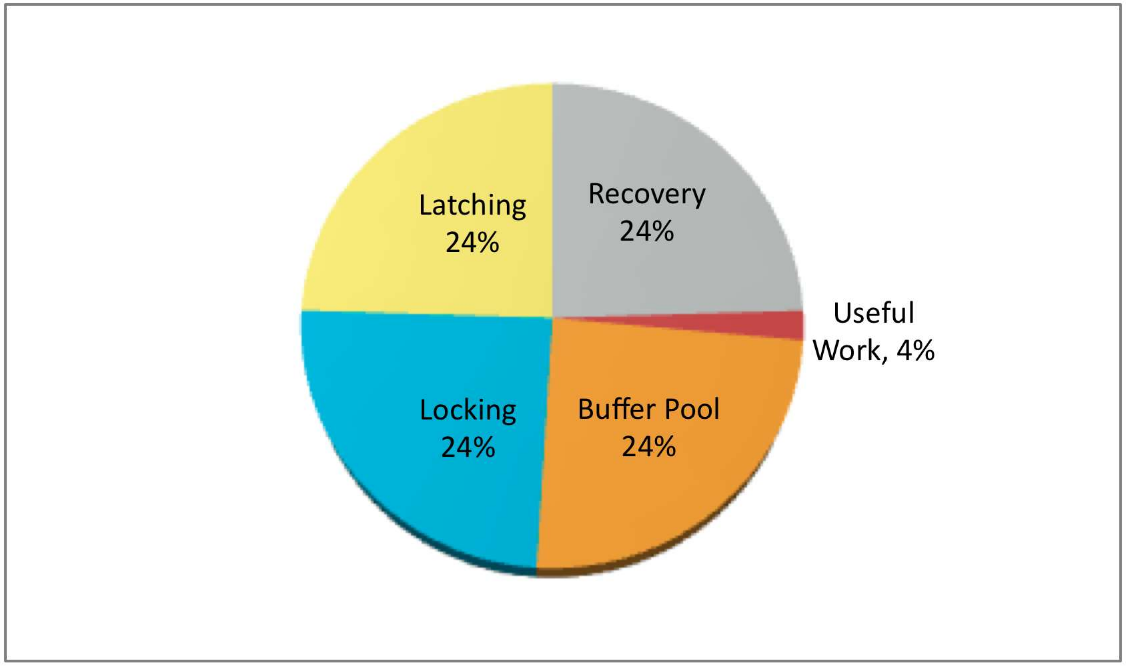
If you want a fast multi-node DBMS, you need a fast single-node DBMS.

If you want a single-node DBMS to go 100x as fast, you need to execute 1/100 as many instructions.

- You won't get there on clever disk I/O: Most of the data is living in memory

Where Does the Time Go?

- TPC-C CPU cycles
- On Shore DBMS
- Instruction counts have similar pattern



What are These Different Parts?

Buffer manager: Manages the slots that holds disk pages

- Locate pages by a hash table
- Employs an eviction strategy (clock scan – approximates LRU)
- Coordinates with recovery system

Different Parts 2

Locks: Logical-level shared and exclusive claims to data items and index nodes

- Locks are typically held until the end of a transaction
- Lock manager must also manage deadlocks

Different Parts 3

Latches: Low-level locks on shared structures

- Free-space list
- Buffer-pool directory (hash table)
- Buffer “clock”

Also, “pinning” pages in the buffer pool

Different Parts 4

Logging: Undo and redo information in case of transaction, application or system failure

- Must be written to disk before corresponding page can be removed from buffer pool

Strategies to Reduce Cost

All data lives in main memory

- Multi-copy for high assurance
- Still need undo info (in memory) for rollback and disk-based information for recovery

No user interaction in transactions

Avoid run-time interpretation and planning

- Compile & register all transactions in advance

Strategies, cont.

Serialize transactions

- Possible, since there aren’t waits for disk I/O or user input

Parallelize

- Between transactions
- Between parts of a single transaction
- Between primary and secondary copies

H-Store & VoltDB

- H-Store is the academic project
  - Brown/Yale/MIT
  - http://hstore.cs.brown.edu/
- VoltDB is the company
  - Velocity OnLine Transactions
  - http://docs.voltdb.com/
  - Community and Enterprise editions

VoltDB Techniques

Data in main memory

- 32-way cluster can have a terabyte of MM
- Don’t need a buffer manager
- No waiting for disk
- All in-use data generally resides in MM for OLTP systems anyway

VoltDB Techniques 2

Interact only via stored procedures

- No round trips to client during multi-query transactions
- No waiting on users or network
- Can compile & optimize in advance
- Results come back all at once – no cursors

Need to structure applications carefully

Discussion Problem

Online course registration

- We're building a VoltDB-style course registration system with these tables:
- Offering(CRN, CourseNum, CName, Limit)
- Registered(CRN, SID)
- Student(SID, First, Last, Status)
- Prereq(CourseNum, PCourseNum, MinMark)
- Transcript(SID, CourseNum, Grade)

Discussion Problem

1- List your stored-procedure (transaction) names

- For each, write a one-sentence description of its input parameters and output.

2 Write pseudocode for registering a student stored procedure:

- **No over-enrollment**: check current enrolled count ≤ Limit before inserting.
- **No phantom seats**: once you've shown a seat is available, concurrent registrations can't steal it.
- **Emphasize** that all reads and the insert happen in one atomic VoltDB transaction so that two concurrent calls can't both see the same "free" seat.

Be prepared to share

- Your list of procedures.
- Your pseudocode snippet.
- A 30-second explanation of how VoltDB's serializable transactions prevent both "over-enroll" and "phantom seat" anomalies.

VoltDB Techniques 3

Serial execution of transactions

- Avoids locking and latching
- Avoids thread or process switches
- Avoids some logging
  - Still need <u>undo buffer</u> for rollback

VoltDB Techniques 4

Multiple copies for high availability

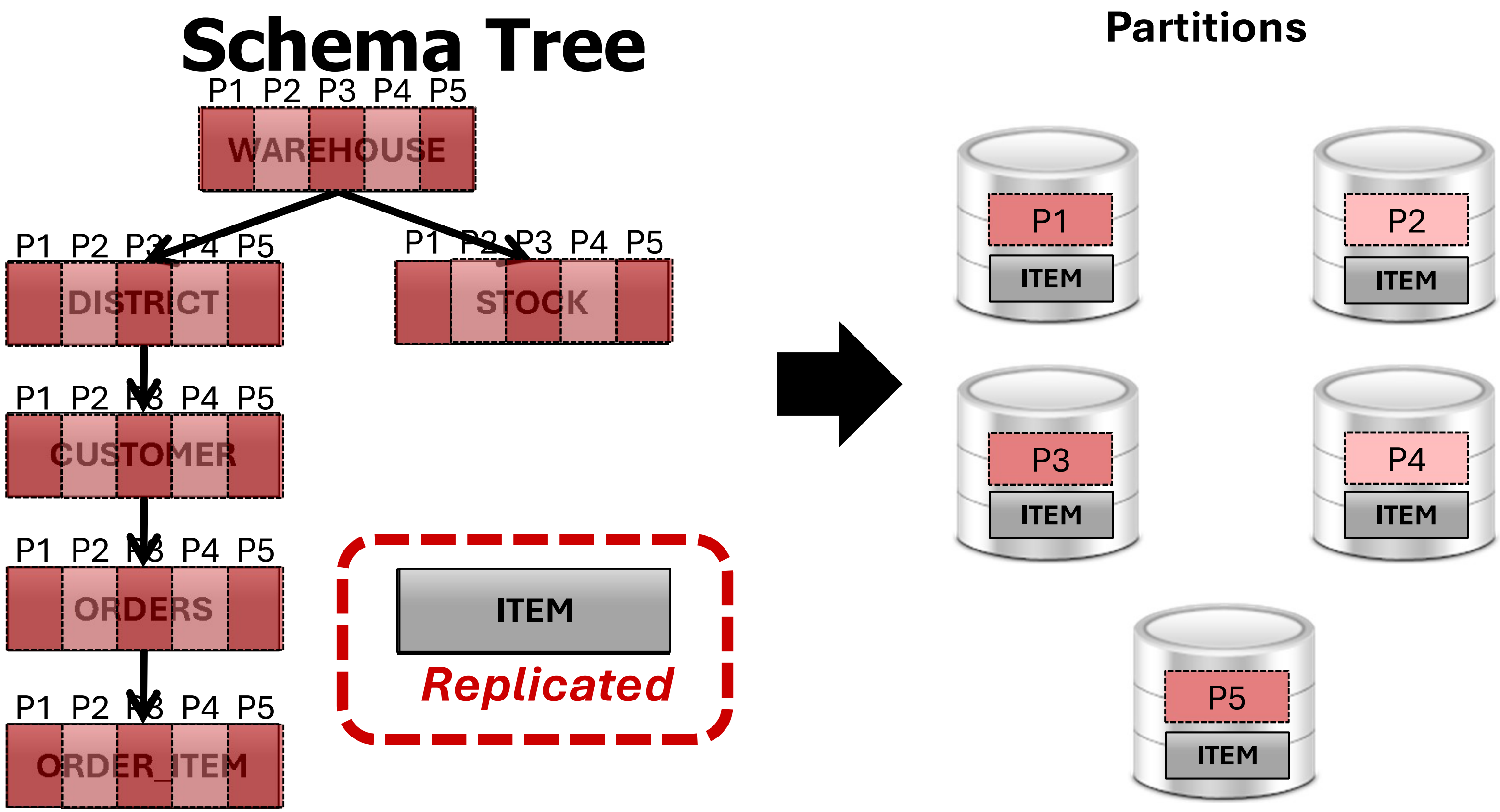
- Can specify k-factor for redundancy: can tolerate up to k node failures
- For complete durability:
  - Snapshot of DB state to disk
  - Log commands to disk
    - Synchronously (higher latency)
    - Asychronously (lower latency, possible loss)

VoltDB Techniques 5

Shared-nothing parallelism: tables can be

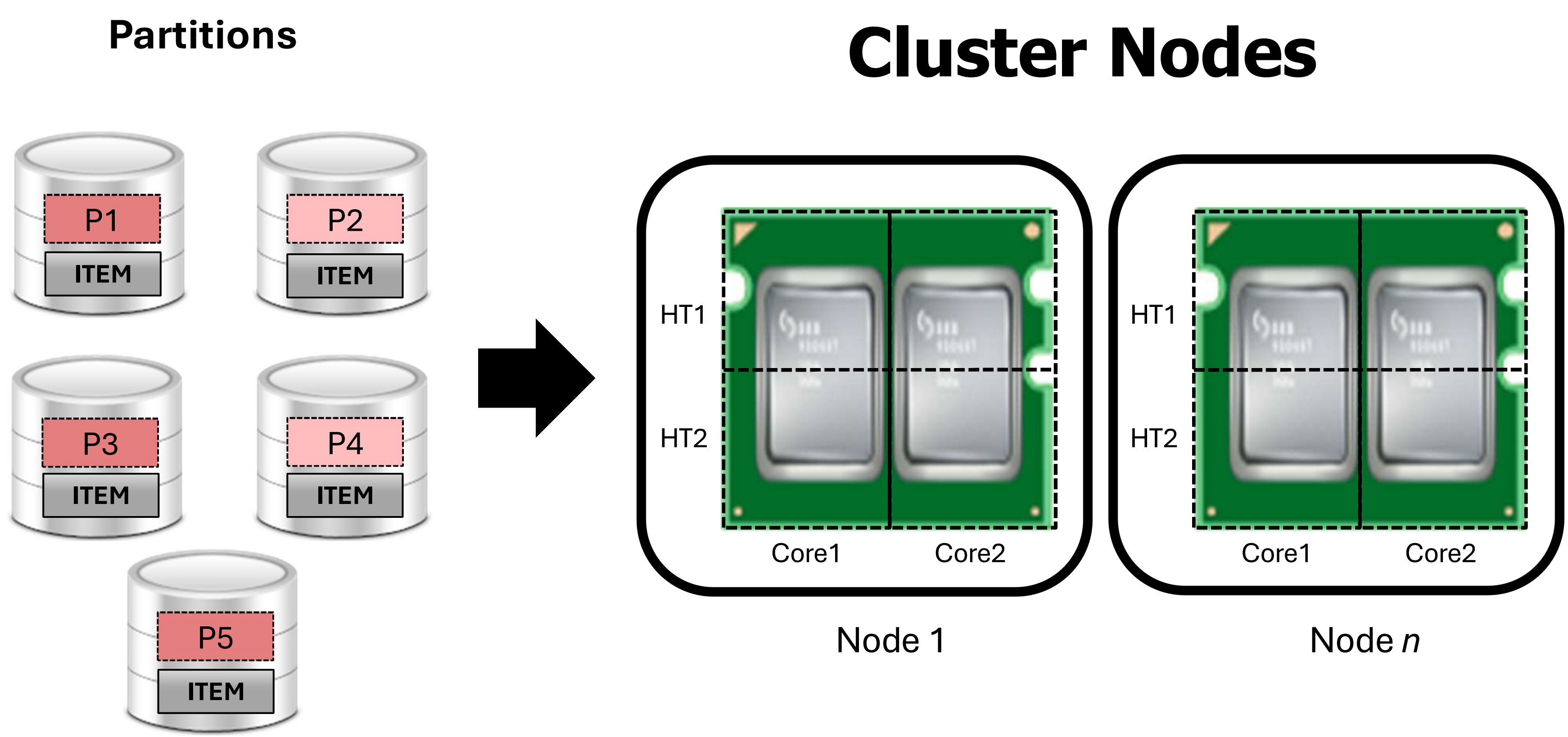
- partitioned (or replicated) and spread across multiple sites.
  - Each site has its own execution engine and data structures
  - No latching of shared structures
  - Does incur some latency on multi-partition transactions

Can have partitions of several tables at each site



Data Placement

- Assign partitions to sites on nodes.



Results

45X conventional RDBMS

7X Cassandra on key-value workload

Has been scaled to 3.3M (simple) transactions per second