WiredTiger

info@wiredtiger.com

A data management engine

Big memory and data focus

Scalable

Transactional

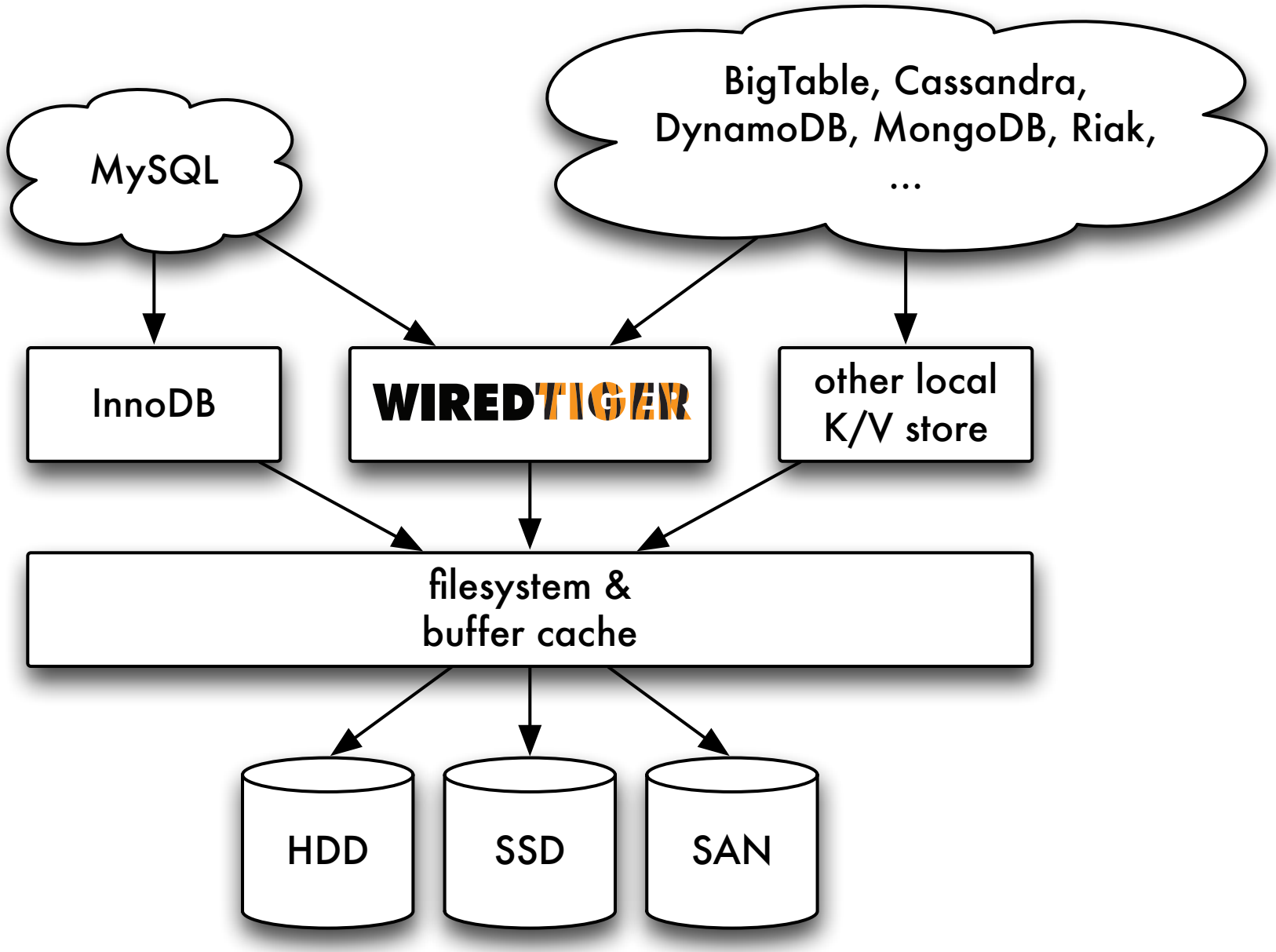
Pluggable architecture

Embedded or Server

NoSQL, Open Source

WIREDTIGER

Positioning



Access methods

- B+tree row-store
  - write-optimized, known performance
- Column-store
  - read-optimized
  - store column groups separately
- Log-structured merge trees, bloom filters
  - write-optimized
  - sustained throughput for random inserts

WIREDTIGER

Mix-and-match

Combine row-store, column-store, LSM

- sparse, wide table
- column-store primary
- row-store or LSM indexes

All transactional, of course.

In-memory vs. On-disk format

Completely separate

Allow both to be efficient

- simplify new features, tuning, upgrades

Willing to pay the transformative cost

In-memory

Focus on transaction cost/performance:

- multicore scaling
- tree pointers, not block addresses
- lockless, non-blocking algorithms
  - skip lists, hazard pointers
- no update in-place
- minimize cache footprint
  - column-store, LSM, compression
  - variable size tree nodes

On-disk

Focus on store footprint:

- column-store lends itself to compression
- big, variable-sized chunks
  - block compression
  - efficient I/O transfers
- compress everything:
  - RLE, key-prefix, dictionary lookup, Huffman
  - no indexing information

WiredTiger

info@wiredtiger.com