The NoSQL Ecosystem
Executive summary

* NoSQL is about using the right tool for the job
My bias

* Started working on Cassandra in 2009 after looking at the alternatives
* Co-founded Riptano in April 2010
NoSQL at OSCON

- Introduction to MongoDB
- Scaling Sourceforge with MongoDB
- Hadoop, Pig, and Twitter*
- (Plus the Neo4J and Cassandra tutorials Monday and Tuesday)
Why NoSQL? 1

- Relational databases don’t scale
Why NoSQL? 2

• The relational model maps poorly to some problems

• Sub-category: almost all NoSQL databases are schema-free or schema-optional to some degree
Why NoSQL? 3

- Relational databases are slow
Myth 1

- “NoSQL is for people who don’t understand {SQL, denormalization, query tuning, ...}”

- Similarly: “Only users of [database X] are turning to NoSQL databases, because X sucks.”
“BASE is diametrically opposed to ACID. Where ACID is pessimistic and forces consistency at the end of every operation, BASE is optimistic and accepts that the database consistency will be in a state of flux. Although this sounds impossible to cope with, in reality it is quite manageable and leads to levels of scalability that cannot be obtained with ACID.”

“BASE: An Acid Alternative,” Dan Pritchett, eBay
Scale forces tradeoffs
Myth 2

“NoSQL is nothing new because we had key/value databases like bdb years ago.”
Myth 3

“Only huge sites like Facebook and Twitter need to care about scalability.”
The downside to NoSQL-as-identifier
Evaluating NoSQL databases

* Data model / query language
* Scalability / availability
* Persistence
Data model

- **Document**
  - CouchDB, MongoDB, Riak

- **ColumnFamily**
  - Cassandra, HBase

- **Graph**
  - Neo4j, AllegroGraph, Objectivity InfiniteGraph

- **Collections**
  - Redis

- **Key / value**
  - bdb, bitcask, Memcached, Tokyo Cabinet
Document queries

- CouchDB
  - js map/reduce creates [materialized] views that may be queried

- MongoDB
  - b-tree indexes allow querying documents by field

- Riak
  - link-walking or [runtime] js map/reduce
ColumnFamily queries

SELECT * FROM tweets
WHERE user_id IN (SELECT follower FROM followers WHERE user_id = ?)
### Persistence

- **Classic B-tree**
  - bdb, TC, MongoDB
- **Append-only B-tree**
  - CouchDB
- **On-disk linked lists**
  - Neo4J
- **Pluggable**
  - Riak, Voldemort
- **SSTable**
  - Cassandra, HBase
- **Memory-only**
  - Memcached, VoltDB
- **Memory w/ checkpoint**
  - Membase, Redis
Durable

- bdb
- Cassandra
- CouchDB
- Neo4J
- Riak*, Voldemort*
pathExists(a, b, 4)

MySQL
Neo4j
Neo4j

<table>
<thead>
<tr>
<th># persons</th>
<th>query time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 000</td>
<td>2 000 ms</td>
</tr>
<tr>
<td>1 000</td>
<td>2 ms</td>
</tr>
<tr>
<td>1 000 000</td>
<td>2 ms</td>
</tr>
</tbody>
</table>
The Log-Structured Merge-Tree, Bigtable: A Distributed Storage System for Structured Data
Scalability

* Master-driven vs distributed replicas
Nodes B, C, and D store keys in range (A, B) including K.
CAP

- Consistency
- Availability
- Partition tolerance
Multi-DC with distributed replicas

Key K

Wednesday, July 21, 2010
CA

* Scalaris
* VoltDB
Conclusion

“If you’re deploying memcache on top of your database, you’re inventing your own ad-hoc, difficult to maintain NoSQL data store”