Yes or NoSQL

Andy Street, Casey Link, David Mazary, Jonathan Berkhahn, Val Komarov

CS4284 Systems & Networking Capstone. Spring 2011. Faculty advisor: Ali R. Butt

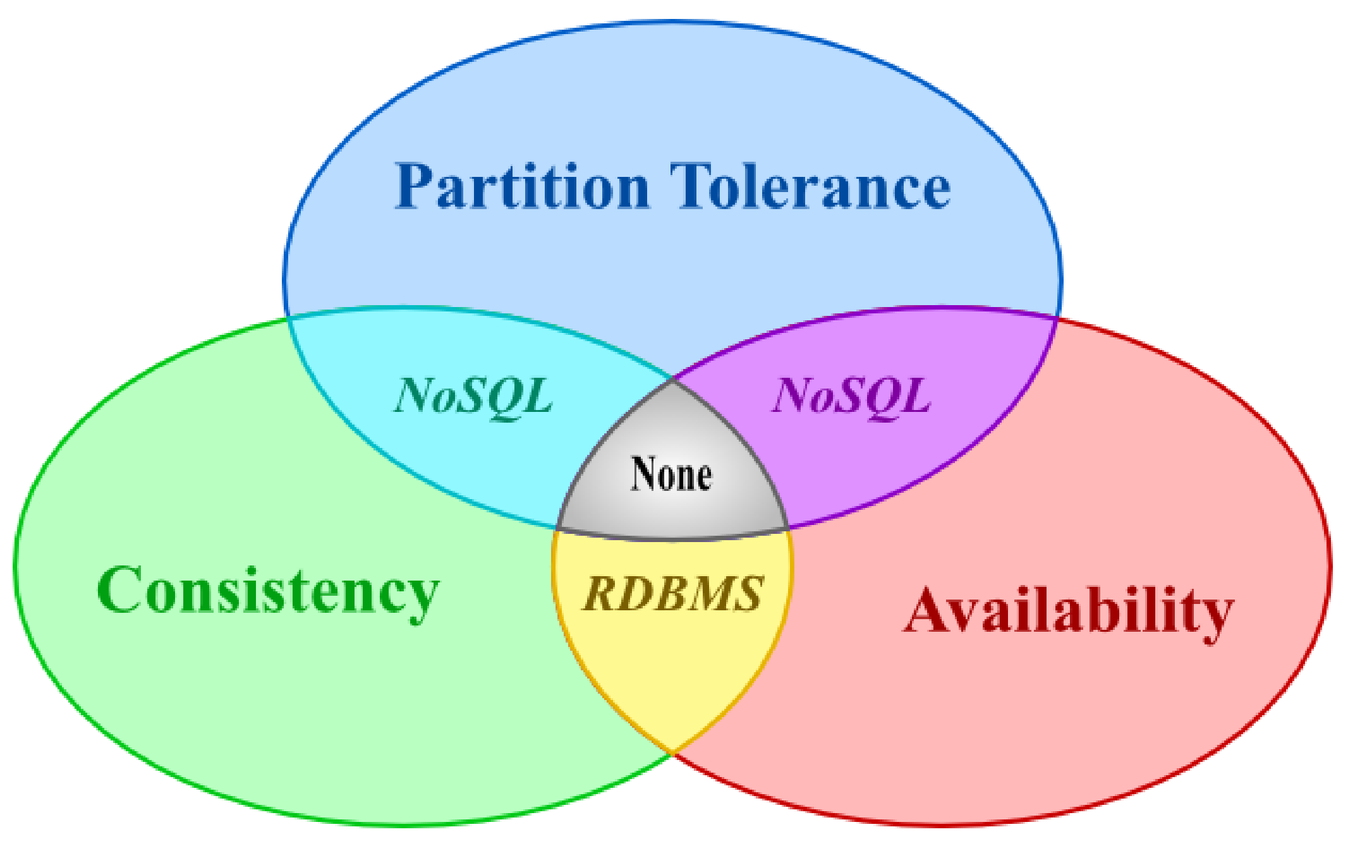
VirginiaTech
Invent the Future

Motivation

- NoSQL databases are a new technology gaining traction
  - Sacrifice an attribute of a traditional RDBMS
  - Better suited to large data tasks
- Businesses are using larger and larger data sets
  - Amazon, Facebook, Google are key users of NoSQL
  - More conservative businesses could benefit from adopting NoSQL technologies

Background

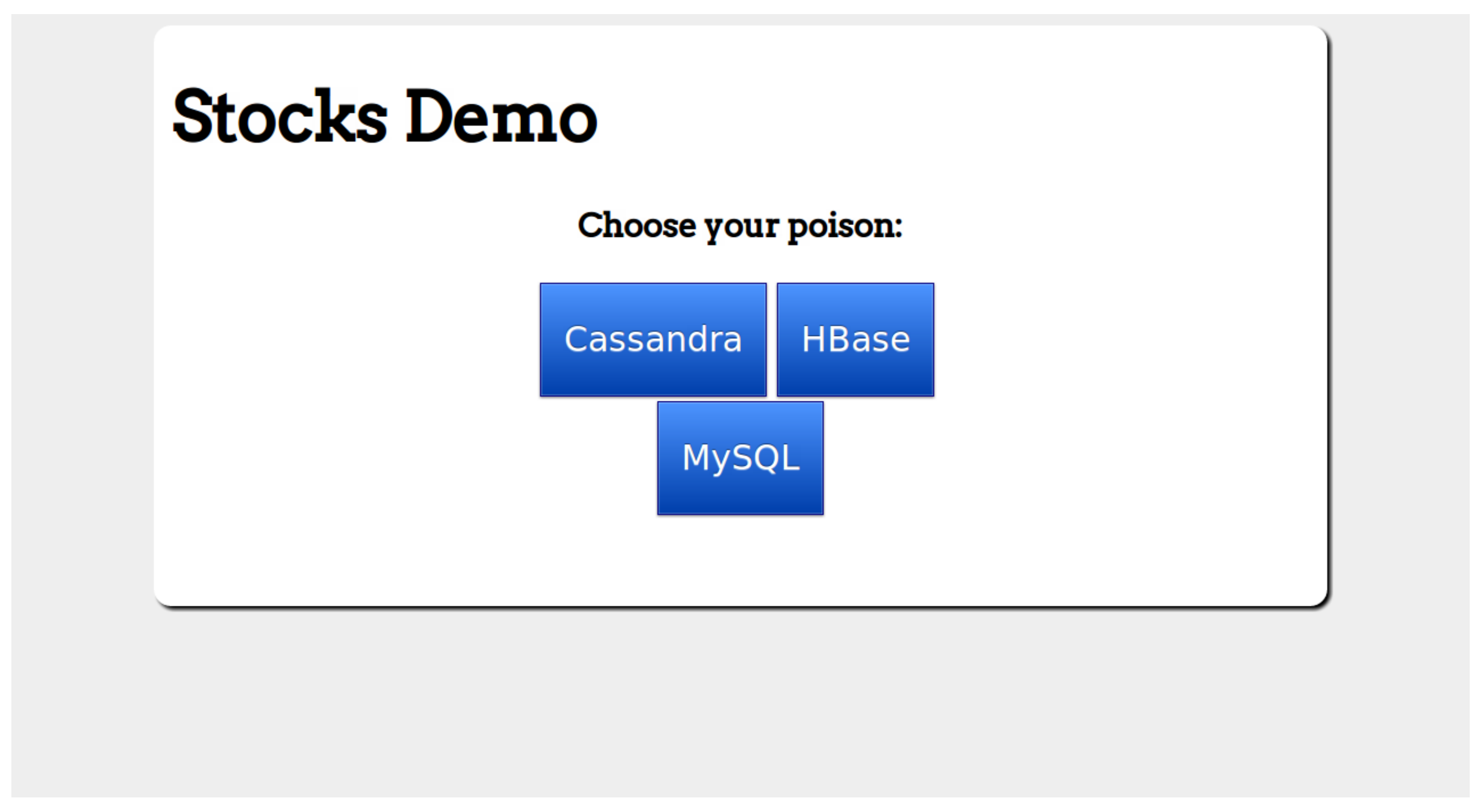
- Google's BigTable set the stage for this new breed of structured storage software.
  - Stores high volume of sparse data with fault tolerance
- Amazon Dynamo describes their "highly-available key-value store"
  - Achieves reliability using a Distributed Hash Table
- These and other NoSQL ideas were implemented as open-source projects
  - Apache Cassandra, Apache HBase, MongoDB, Redis, …
- NoSQL databases gain scalability (partition tolerance) by sacrificing either consistency or availability, corresponding to Eric Brewer's Consistency, Availability, Partition tolerance (CAP) theorem.



Brewer's CAP Theorem

Use in a real world application

- Documented the developer experience using:
  - Cassandra – Based on BigTable and Dynamo
  - HBase – Based on BigTable
  - MySQL – Traditional RDBMS
- Developed a web application involving insertion and querying of data, with the ability to toggle the storage backend.



Implementation Evaluation

- Schema design requires a deep understanding of the very different storage architectures.
- Cassandra and MySQL have the most mature language bindings, while HBase is centered around Java development.
- Cassandra clusters are easiest to create since they are DHT-based, followed by MySQL sharding, and HBase has the most involved setup, emulating the full Google BigTable platform.

```
Symbol {
  Date {
    open:
    close:
    low:
    high:
    volume
  }
}

+-----------------+-------------------------+
| row             | column families         |
+-----------------+-------------------------+
|                 | price:         | volume: |
+-----------------+----------------+---------+
| <symbol><date>  | price_open     |         |
|                 | price_high     |         |
|                 | price_low      |         |
|                 | price_close    |         |
+-----------------+----------------+---------+
```

Sample schemas for Cassandra and HBase

Performance Evaluation

Yahoo Cloud-Serving Benchmark

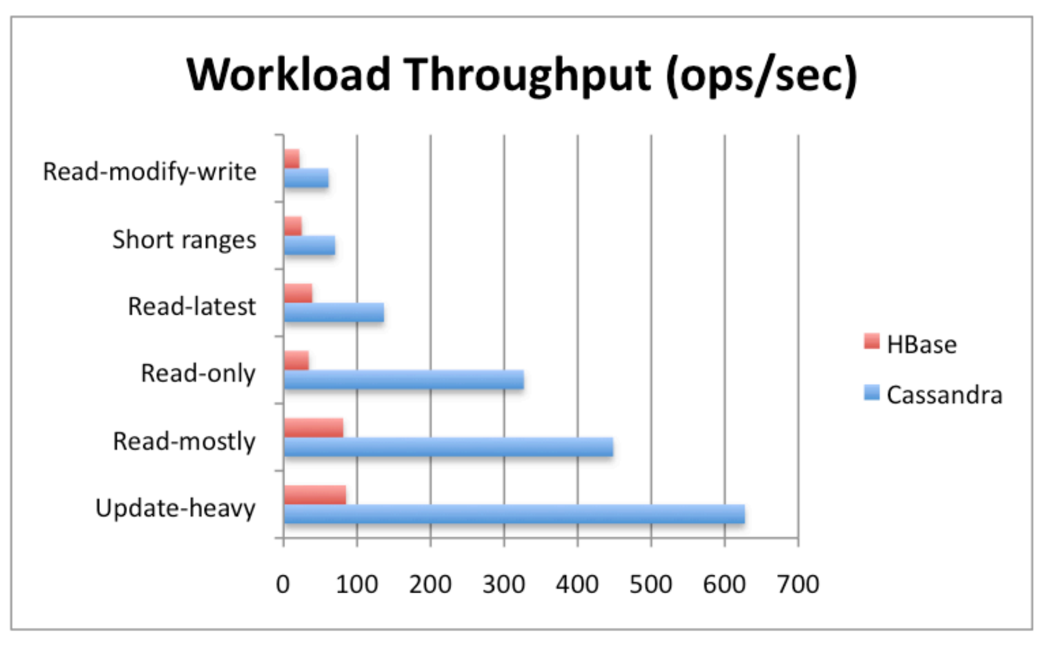
- A Yahoo! Research project to generate workloads for testing structured storage systems
- Updated and ran these workloads against HBase 0.90.2 and Cassandra 0.7.4

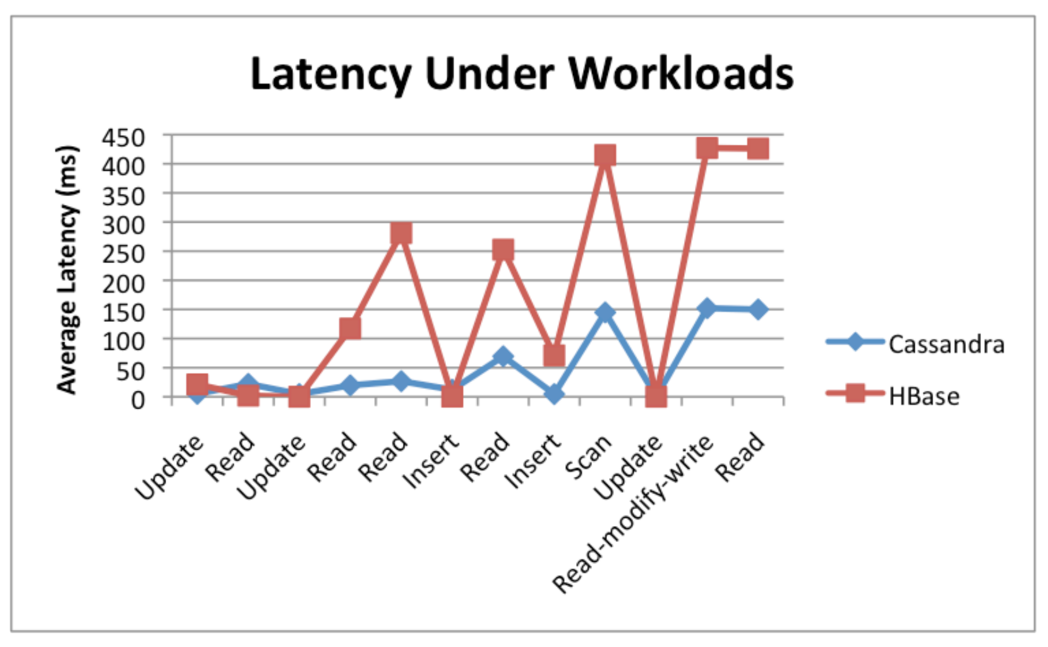
Workload	Operations	Application
Update heavy	Read/update: 50/50	Session store recording recent actions
Read mostly	Read/update: 95/5	Photo tagging: Add a tag is an update, but most operations are read tags
Read only	Read: 100	User profile cache, where user profiles are constructed elsewhere
Read latest	Read/insert: 95/5	User status updates: people want to read the latest
Short ranges	Scan/insert: 95/5	Threaded conversations, where each scan is for a post in a given thread
Read-modify-write	Read/read-modify-write: 50/50	User database, where user records are read and modified by the user or to record user activity

Proportions of Operations and the Applications of YCSB Workloads

Results

- Cassandra demonstrated favorable throughput and latency.
- HBase is designed for consistency rather than availability
- Tradeoff between performance and application design goals





Future Work

- Create applications built on NoSQL using different workflows
- Evaluate a broader range of the many NoSQL databases

In partnership with Booz | Allen | Hamilton