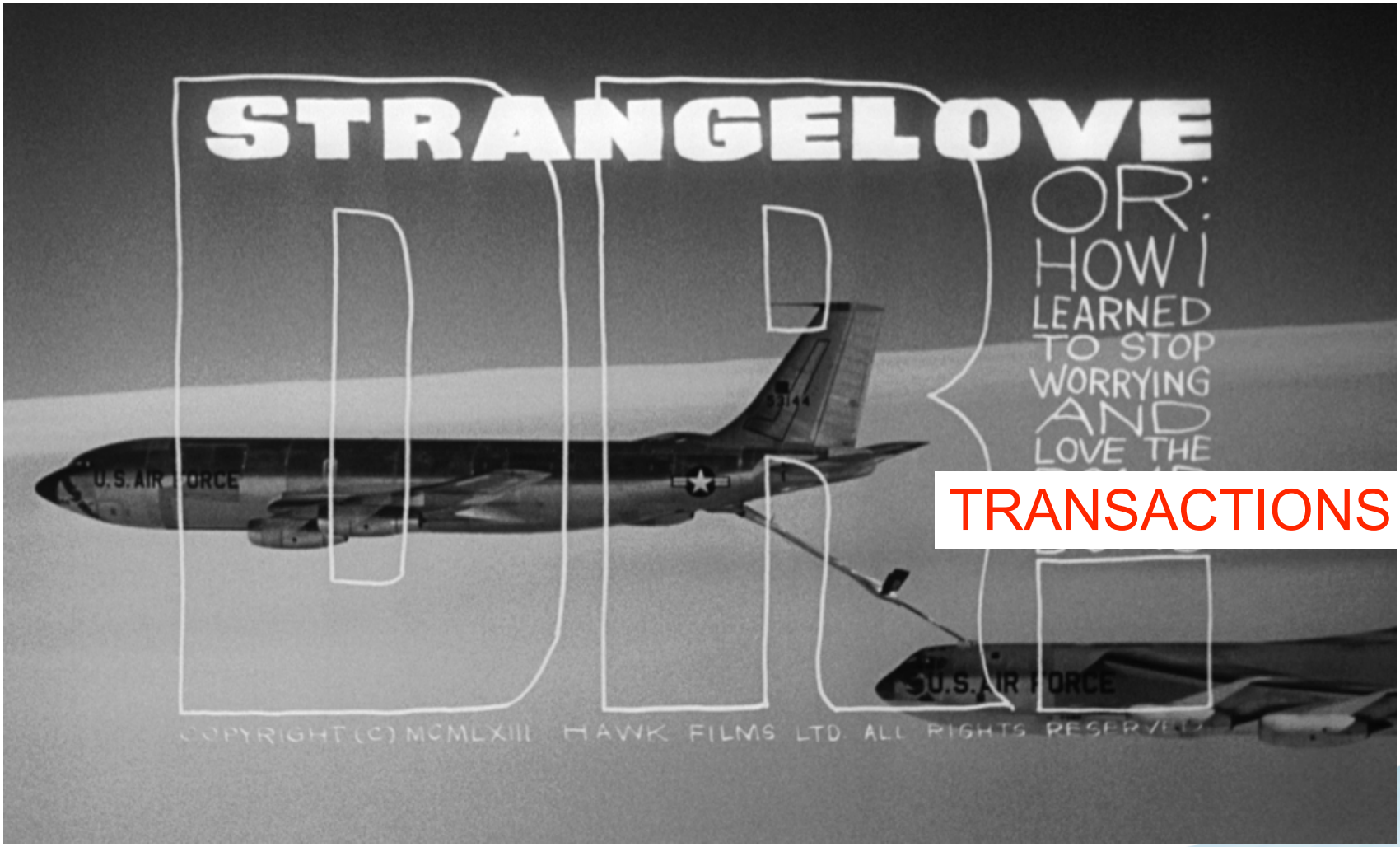
Java. Cloud. Leadership.

Transactions Returning to Big Data (NoSQL)

OR

...

Mark Little
Red Hat, Inc.



TRANSACTIONS

Overview

- Where we are today
- NoSQL and the enterprise
  - Open source perspective?
  - Changes occurring
- The future?
  - Compensation transactions?
  - Cloud-TM, LEADS, ...

Transactions and the masses

- Mass adoption through .NET, J(2)EE, CORBA ...
  - JDBC, JMS, JCA, JTA, OTS
  - Mainstream in the last twenty years
- Open source's contribution
  - MySQL, JBossAS
- Until the 21st Century ACID was good enough
- Then extended transactions
  - XOTS, OASIS BTP, WS-CAF, WS-*

RDBMS

- Evolved nicely over the years
  - Coped with increasing amount of data
  - Coped with increasing distribution of data
- ACID transactions important to enterprises
  - However, one-phase dominates
    - "99% of transactions today are one-phase", Jim Gray, HPTS 1999
  - But two-phase is a requirement for a range of applications

The rise of NoSQL & Big Data

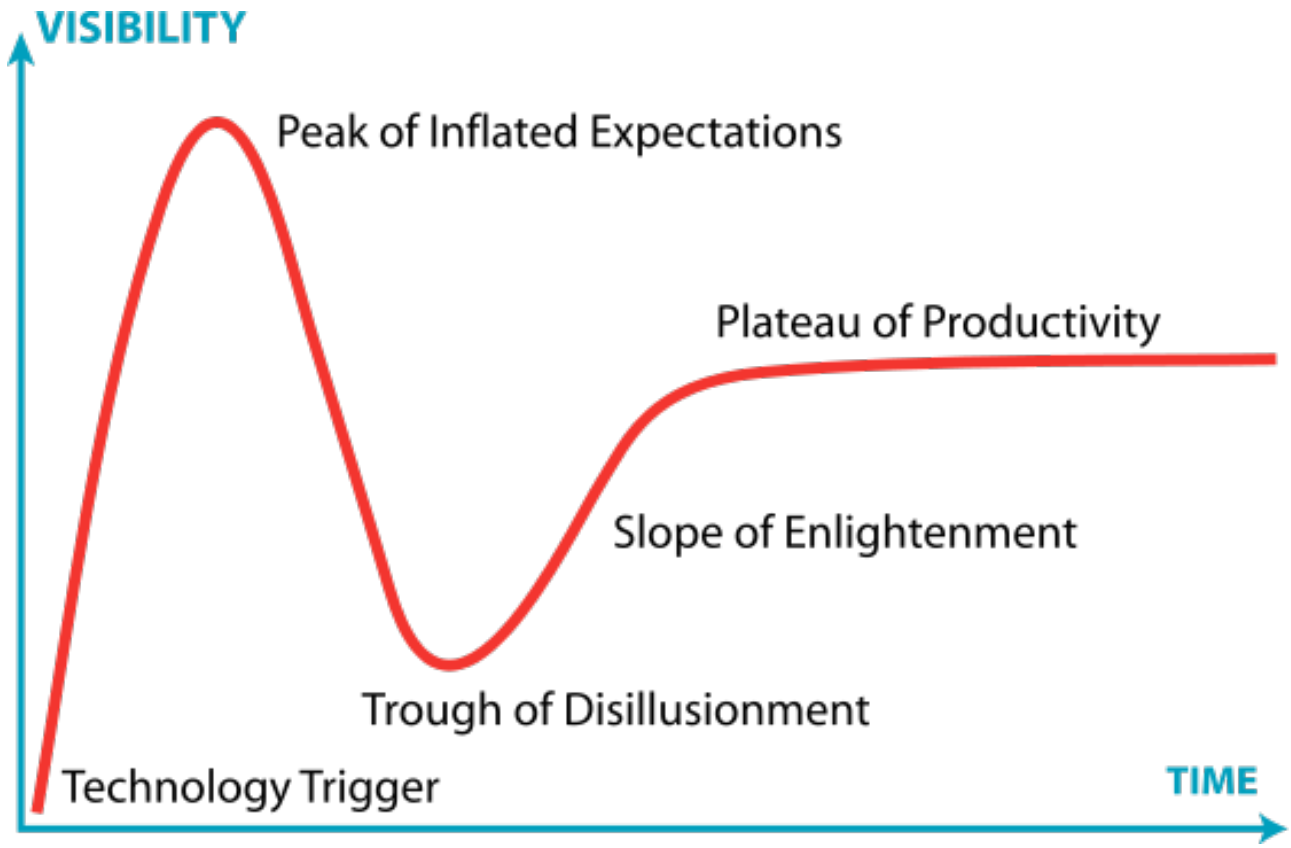
- Data explosion has caused re-evaluation of RDBMS
  - Initially RDBMS augmented with cache
- But ultimately not sufficient to cope with data
  - Hence NoSQL and Big Data growth
  - Range of NoSQL implementations and categories
    - Document, tuple, column, graph
    - See HPTS 2009 and 2011

NoSQL explosion

- > 24 different NoSQL/Big Data implementations
  - Some evolved from cache/data grid
  - Right tool for the right job
- Wouldn't it be nice if ...
  - There was a tool to automatically select the right NoSQL solution for your use case?
  - Or some objective knowledge base?
- On going PhD research

Feature	Cloud SQL Server	Relational Cloud	HadoopDb	CouchDb
NoSQL or SQL	SQL	SQL	SQL.	noSQL
Description	Relational. Uses Microsoft SQL server as the core. Uses a shared nothing architecture. Table groups can be keyed or keyless. If they are keyless then they must be co-located, if they are keyed then they will be partitioned on that key. Uses a system management layer, this is used to control who is active, failed, as well as managing the partitions and other management features that should be automated (12).	This is a relational, transactional database as a service. It uses a central co-ordinator for these transactions. This database promises to take away all the management from the user. Understands the needs of different workloads, to do this the system hosts multiple databases on one server, then it can analyse these and move them about as necessary. Uses an unchanged DBMS at the back end, and database servers that allow the dynamic changes and moves of the data and load balancing. As the system has more than one database on the server they could be from different users but these would never be mixed together (20).	Hybrid of map reduce and DBMS. Has a shared-nothing architecture and its cluster's hardware can be made of commodity nodes. HadoopDb has a split execution environment. A single system is created by connecting multiple independent single node databases deployed across a cluster with the use of Hadoop. Resembles a shared nothing parallel database. Postgres was used as the original database, but now to achieve the performance mentioned a column database is used . HDFS + Map reduce layer. Master slave architecture, with job tracker (11). The system uses a database connector to connect databases they have connectors for MySQL, Postgres and currently a connector for the columnar database VectorWise. This connection allows the SQL queries to run in the database and the result from these is a key value pair for the hadoop layer (18).	CouchDb is a document store and an AP system. It has a Master-master architecture and is an Apache product. It allows the following datatypes within the store: scalar values and compound (document or list) values. These datatypes can be added as the database uses JSON objects and so can save the different types into these. This provides the user with a flexible means of storing different types. The system provides a REST interface so it maybe more suited to the web. The database also provides offline access. (3).
Schema Constraints	Ridged schema constraint- table groups and row groups. Transactions must be done over these. Table groups are the tables on one machine, then the row group is the rows between tables in the table group that have the same partitioning key (12).	Independence from schema layout and foreign keys for partitioning. Partitioning suitable for tables with lots of many-to-many relationships (20).	Uses tables, so that a derivative of the SQL language can be used to query the them. This is also the case as the use of joins, aggregation, selection, etc. are a feature they want to include. The most recent version is using a columnar database, where the database content is stored by column instead of row to improve performance (12).	Schema: collections. Foreign key association. Doesn't guarantee the order of the document (16).

Logging	Cloud SQL's log is shared by all the table groups and shared by the manager in the server (12).	Database have a combined log, which means they all commit together (group committing) (20).	The Catalogue stores information about the partitions, data sets in the cluster and replica locations (18).	No log needed as it uses versioning.
Locking	Uses locking to provide transactions and ACID behaviour (12).	Provides locks for the transactions within the system (but tries to reduce mulit-node locks) (20).	Assumed that the underneath layer of Postgres or VectorWise would use locks for their transactions, so there would be locks. *	There is no locking as couchDb uses MVCC, and requests are run in parallel (16).
Storage	Microsoft SQL Server (12).	Uses a standard DBMS with database servers, supports MySQL, Postgres and JDBC (20).	Uses HDFS as a distributed file system (11).	Disk commit and COW-Btree (1).
Consistency	ACID transactions are used these are constrained to one partition. This makes the consistency strong. Transactions are restricted. Update transactions read and write single values of that partitioning key, this means that transactions are kept to one server. This is restricted further if the table is keyed, the transaction must be done on a table group across a row group (12).	Multi-node transactions, tries to partition the nodes intelligently creating as little multi-node transactions as possible (more costly and larger overhead). Tries to preserve consistency using transactions, so strong consistency (20).	Assumed that as the underlying databases for either version have strong consistency and use ACID transactions that HadoopDB does. *	The database is eventually consistent but provides ACID semantics at document level. Transactions are not provided within this system. It uses MVCC for concurrency control. Version control is used through the use of sequence IDs. The system is append only (3).



redhat. | JBoss by Red Hat

Java. Cloud. Leadership.

NoSQL Backlash

- http://martinfowler.com/bliki/OrmHate.html
- http://blog.engineering.kiip.me/post/20988881092/a-year-with-mongodb
- http://saucelabs.com/blog/index.php/2012/05/goodbye-couchdb/
- http://tech.backtype.com/the-dark-side-of-hadoop
- http://gradlesummit.com/blog/tim_o'brien/2012/12/mongodb_stole_my_lunch_money_and_ruined_my_startup

Common reported problems

- Slow (disk) performance
- Lack of transactions/consistency
- Failures take down entire system
- Hard to configure/optimise
- Poor memory management
- Poor process management (orphans/zombies)
- Reliability

Why?

- Early in the maturity curve
  - Evolution of NoSQL inevitable
- Some of this is definitely lack of understanding
  - Lack of objective knowledge base/tool
- (Unrealistic?) expectations
  - Belief that a NoSQL solution can replace RDBMS completely
  - Realisation that ACID (strong consistency) is a requirement for their applications

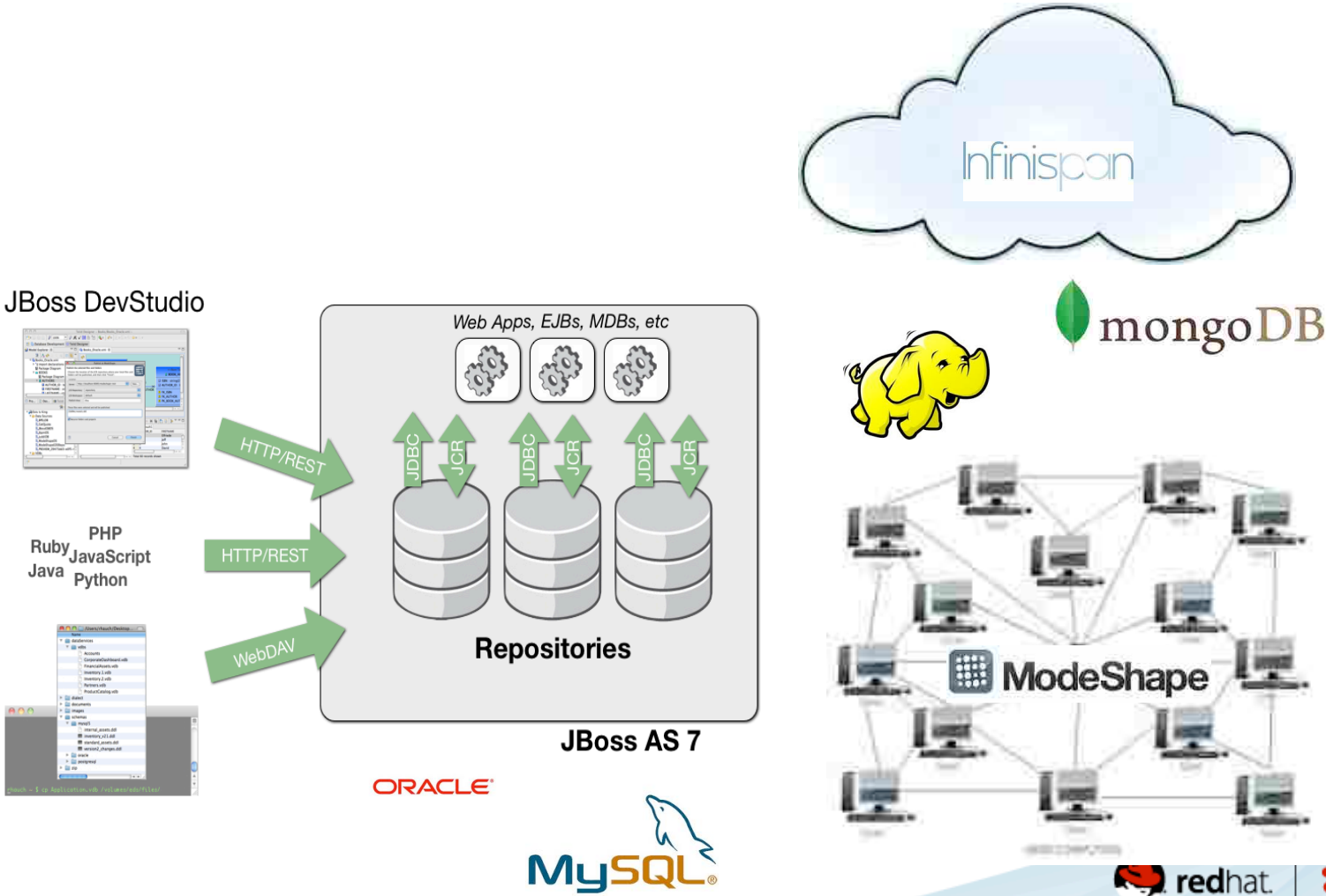
NoSQL in the enterprise

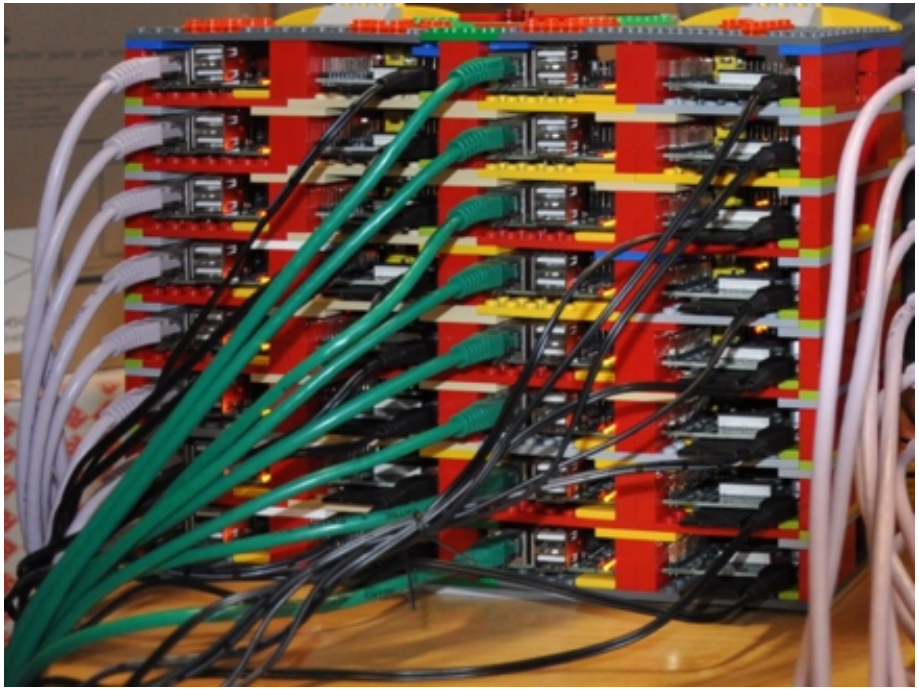
- Many NoSQL-only applications no longer silos
  - Enterprises are looking to add them to existing scenarios
  - But typically lacked global transactions
- Red Hat started with tuple-space cache/data grid/NoSQL implementation (Infinispan)
  - Now have graph/document too (ModeShape)
  - Heavy demand for ACID transactions and compensations

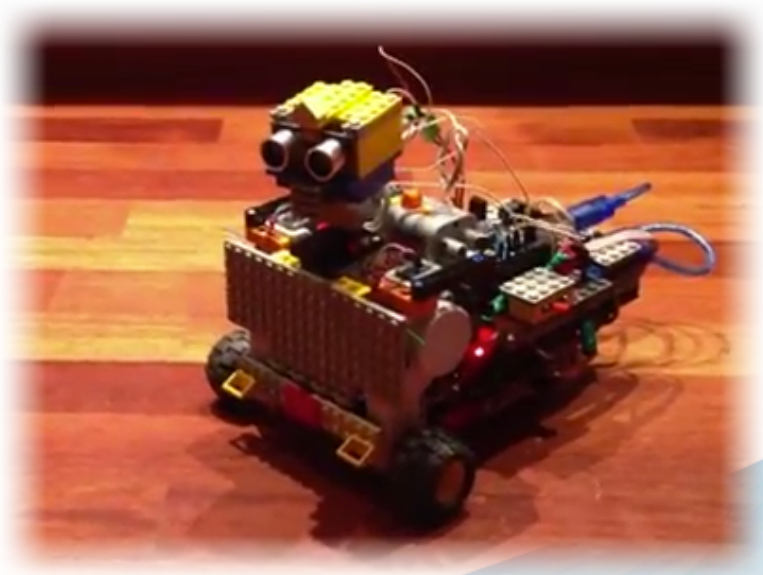
Representative scenarios

- Application developers want to combine
  - RDBMS (xN - frequently)
  - NoSQL (x2 - infrequently but growing)
  - Messaging (xN - frequently)
  - File system updates
- All in the scope of the same ACID transaction
  - Or the same “transaction”?
- Some form of transaction support needed in NoSQL implementations
  - LRCO insufficient

Representative architecture







redhat. | JBoss by Red Hat

Java. Cloud. Leadership.

Adoption of transactions

- Growing realisation that (global) transactions are needed
  - Enterprises are an important use case
  - Not the only reason for change though
- ACID transactions are often easier to understand
  - Despite the performance downside
- Not all-or-nothing
  - Controlled relaxation of ACID
  - Controlled enforcement of consistency

Change takes time

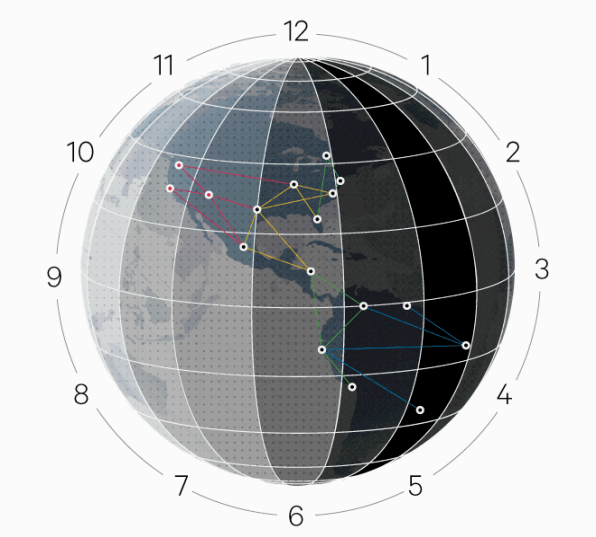
- Slow change for some implementations
  - Added local transactions
    - Most only for the same data item
    - Some don't support automatic rollback/recovery (e.g., Redis)
- At least one implementation requires developers to roll their own transactions
- Telling developers to reimplement their applications to remove need is not an option

NewSQL and OldSQL

- Sometimes NoSQL is not the answer
  - ACID transactions cited by customers as main reason
    - MarkLogic 5 introduced support for XA
    - FoundationDB provides ACID
- Also returning to RDBMS, with core rewrite
  - Postgres
  - MariaDB

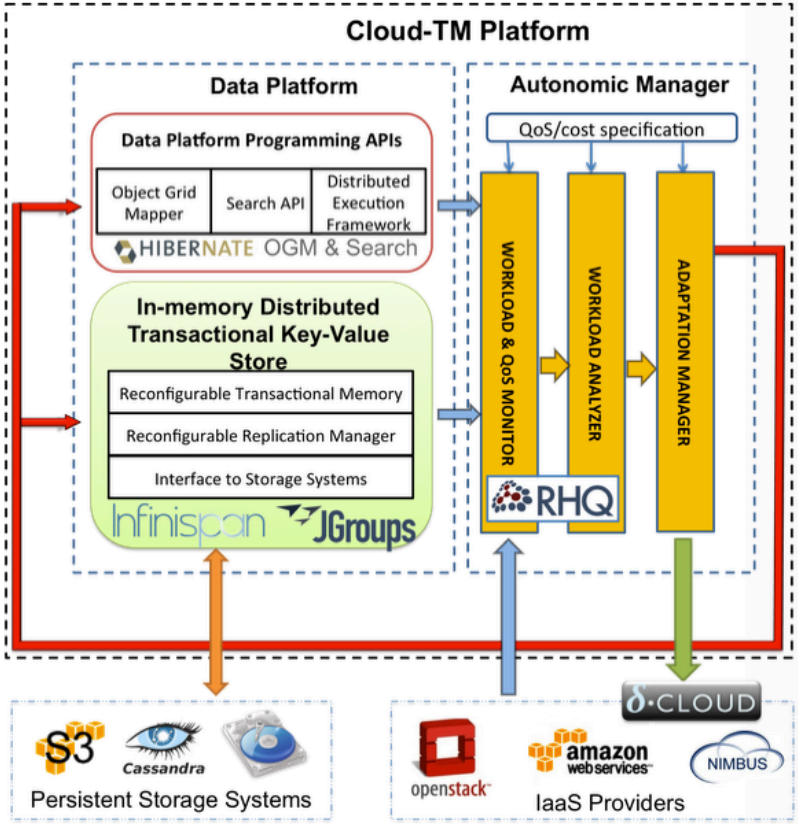
Google's Spanner

- "We believe it is better to have application programmers deal with performance problems due to over use of transactions as bottlenecks arise, rather than always coding around the lack of transactions"



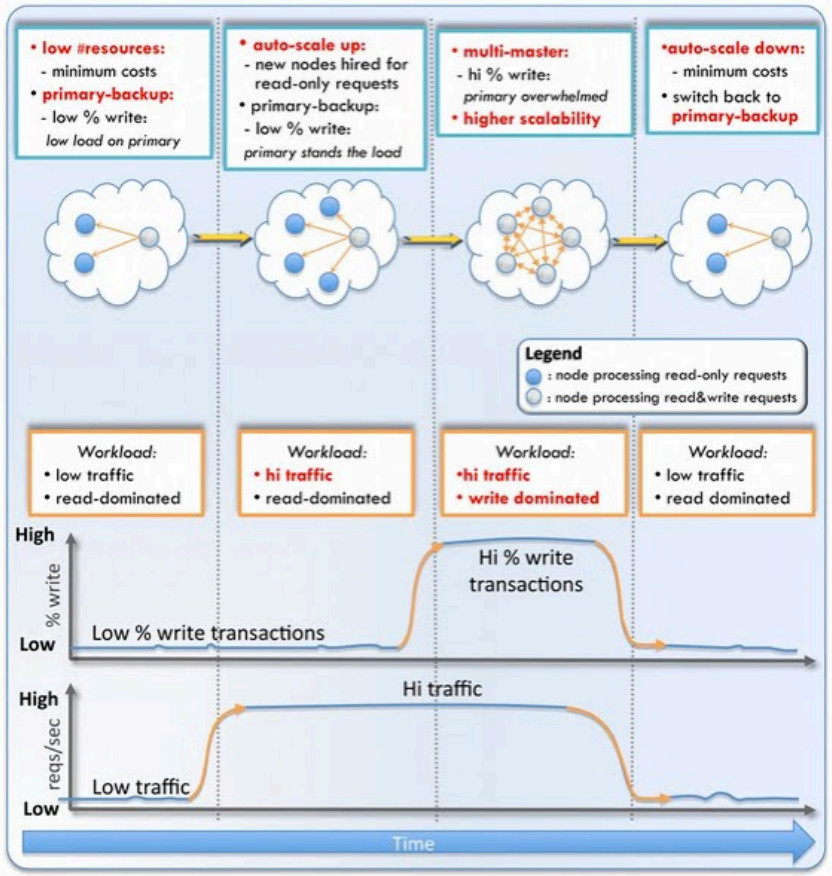
Compensating transactions

- Unlikely that ACID will be standardised for NoSQL
  - Let alone XA
  - Although full ACID is a goal for some, e.g., FoundationDB
- But compensating transactions offer a possible solution for the enterprise problem
- Examples already in use, e.g., Sagas
  - WS-TX, REST Transactions



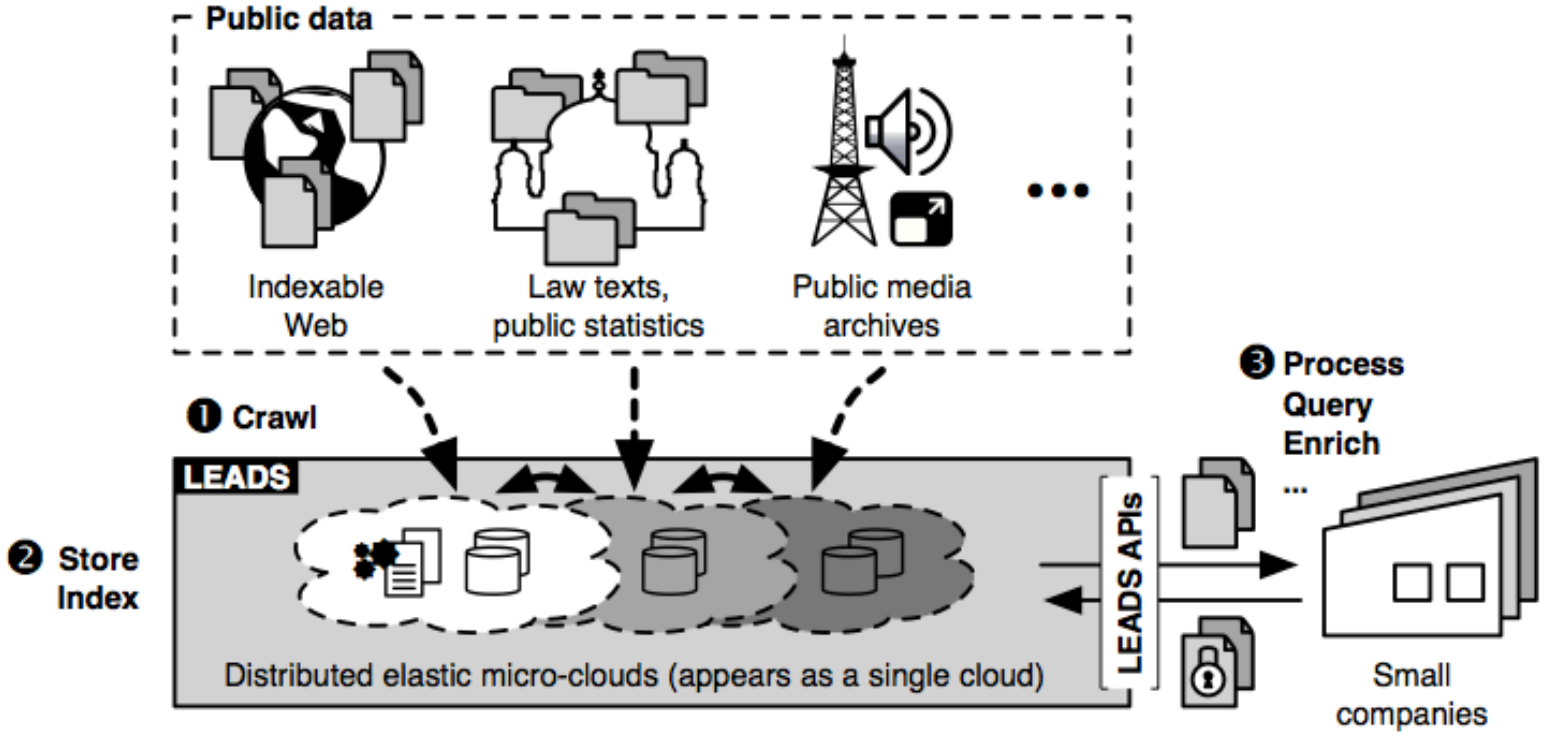
Cloud·TM

A novel programming paradigm for the Cloud



LEADS

LARGE-SCALE ELASTIC ARCHITECTURE FOR DATA AS A SERVICE



Conclusions

- Enterprise applications are important
  - Difficulties with integrating NoSQL and Big Data
  - At least for a subset of applications
- Many NoSQL implementations now support ACID transactions in one way or another
- One size does not fit all
  - Extended transactions?
- Better education on where to use specific technologies