What is Big Data?

All data that is not a fit for a traditional RDBMS, whether used for OLTP or Analytics purposes

Talking Big Data

- You must be able to distinguish when people are talking Big Data, which type of technology will be appropriate for them
  - They may be trying to solve an OLTP scalability problem
  - They may be trying to solve a Big Analytics problem
  - They may not be solving a Big Data problem at all
  - It may simply be a financial or budget decision
- You must understand the Big Data space
  - No one tools is the right fit for all Big Data problem
  - Do not be afraid to recommend the right solution for the problem over the popular solution
  - To do this, you must be aware of the entire ecosystem

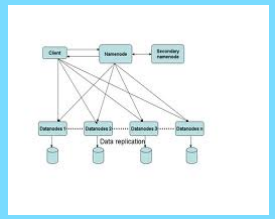
Big Data Technologies

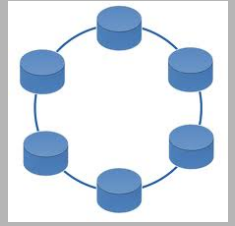
TeraData
Greenplum

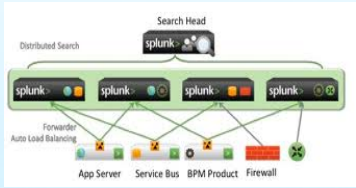
Hadoop

Cassandra
CouchDB
MongoDB

Splunk
Solr







RDBMS
Sharding

SQL &
Map / Reduce

HDFS Storage +
Map / Reduce

NoSQL

Real Time Indexing

Map / Reduce

Relational Database
(highly structured)

Distributed File System
(semi-structured)

Key/Value, Columnar or
Other (semi-structured)

Temporal, Unstructured
Heterogeneous

The Need for New Technology

The Problem

- RDBMS are the square peg for all round holes
- Not all data requires ACID Compliant data stores
  - atomicity, consistency, isolation, durability
- To implement ACID, tradeoffs limit scalability of traditional systems
- First comes Sharding
- Then comes NoSQL

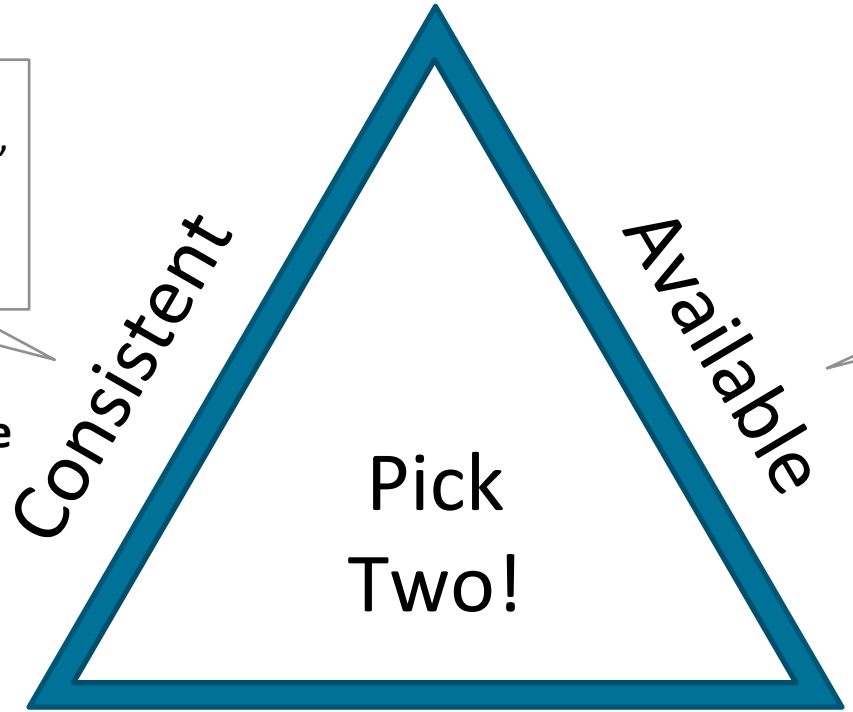
NoSQL Movement

- Grown out of a frustration for traditional RDBMS scalability issues
- Abandons ACID compliance in favor of scalability
- Generally solves for “eventually consistent”
- Far less featured than traditional SQL systems from perspectives of data management, querying, transactions, etc.
- Indexing and other querying optimizations often left to application developers
- Tradeoffs on features are outweighed by scalability concerns for many applications, thus the surge in growth

Understanding CAP

The data, when read, will always be consistent

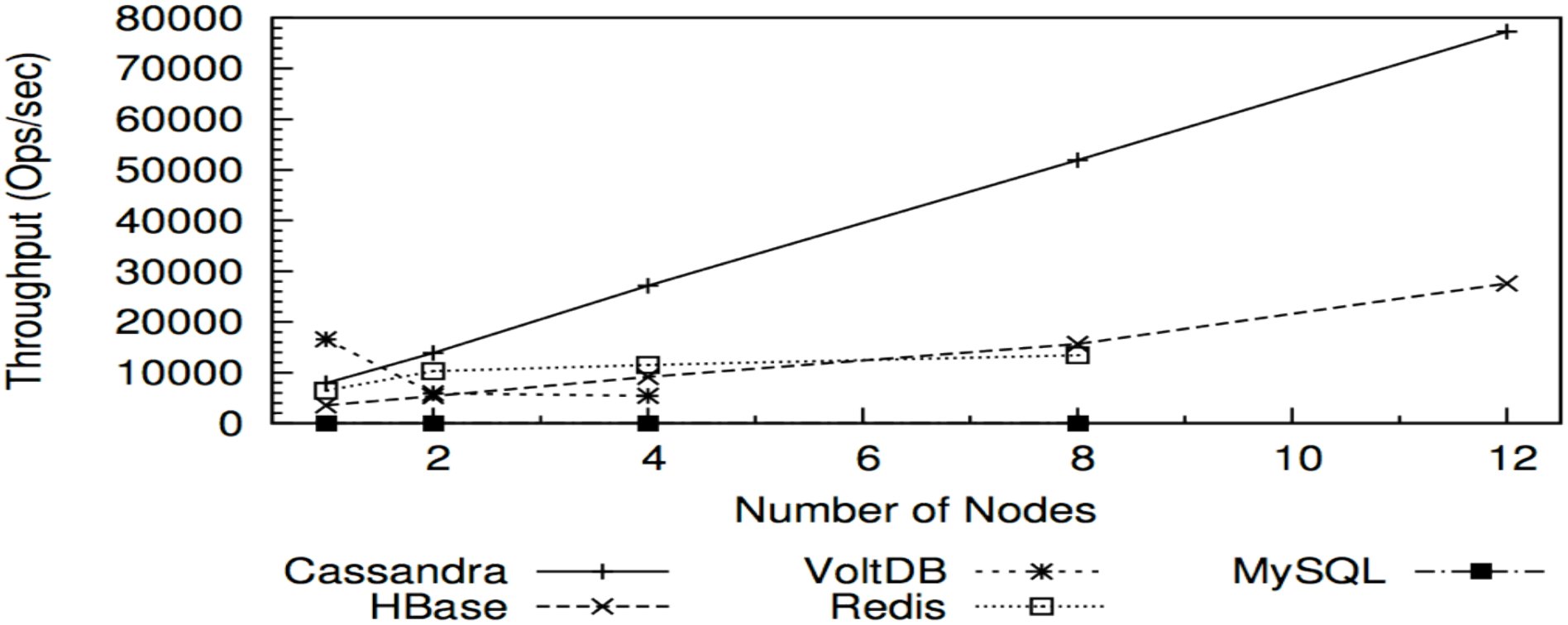
The system is always available



Since network failure is an eventuality in any distributed system, really only debating between Consistency and Availability

The system can handle lost messages, split/ down nodes

VLDB benchmark (RWS)



What is Apache Cassandra?

Cassandra is a highly scalable semi-structured database

- ✓ Fault Tolerant
- ✓ Scalable
- ✓ Open source

Semi-Structured

- Can have predefined structure or be defined at insert-time
- Data can be defined from Java-derived data types

Performance

- Runs extremely fast on spinning disk due to commit log architecture, an append-only log which tracks all write queries with no seeks required.
- Extensive caching for fast reads

Scalability

- Linear scalability; system is scaled by simply adding additional nodes

Fault Tolerant

- Data is replicated to multiple nodes
- Losing one node will not take down the cluster
- Data will be available despite degraded state

What Makes Cassandra Different

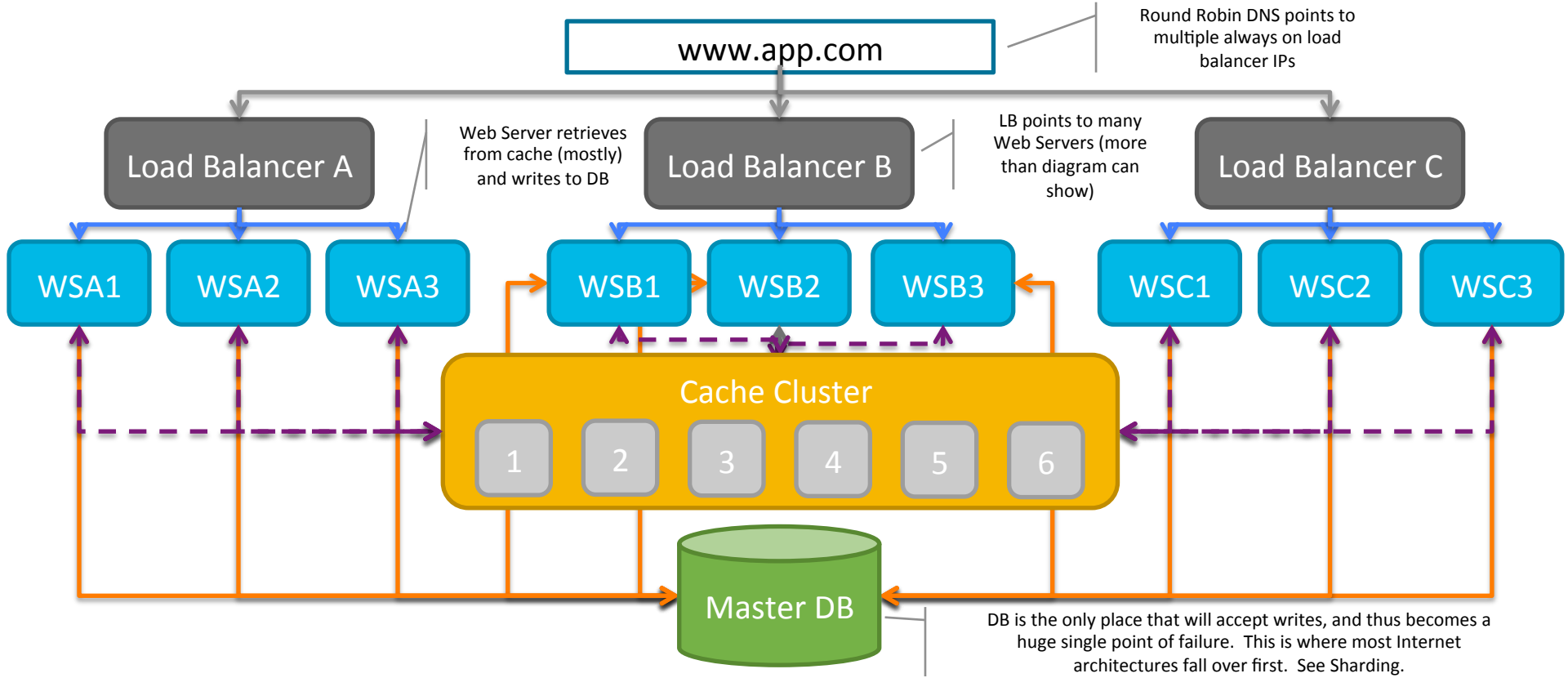
- Scales linearly, which is unique for a database product
- Tunable consistency, allowing speed to be adjusted based on the consistency needs of the application
- Very fast, especially tuned for eventually consistent, allowing blazingly fast writes
- Best used for transactional systems which need fast response times and very high scalability
- Strong commercial backer in DataStax

Understanding Cassandra Data Structure

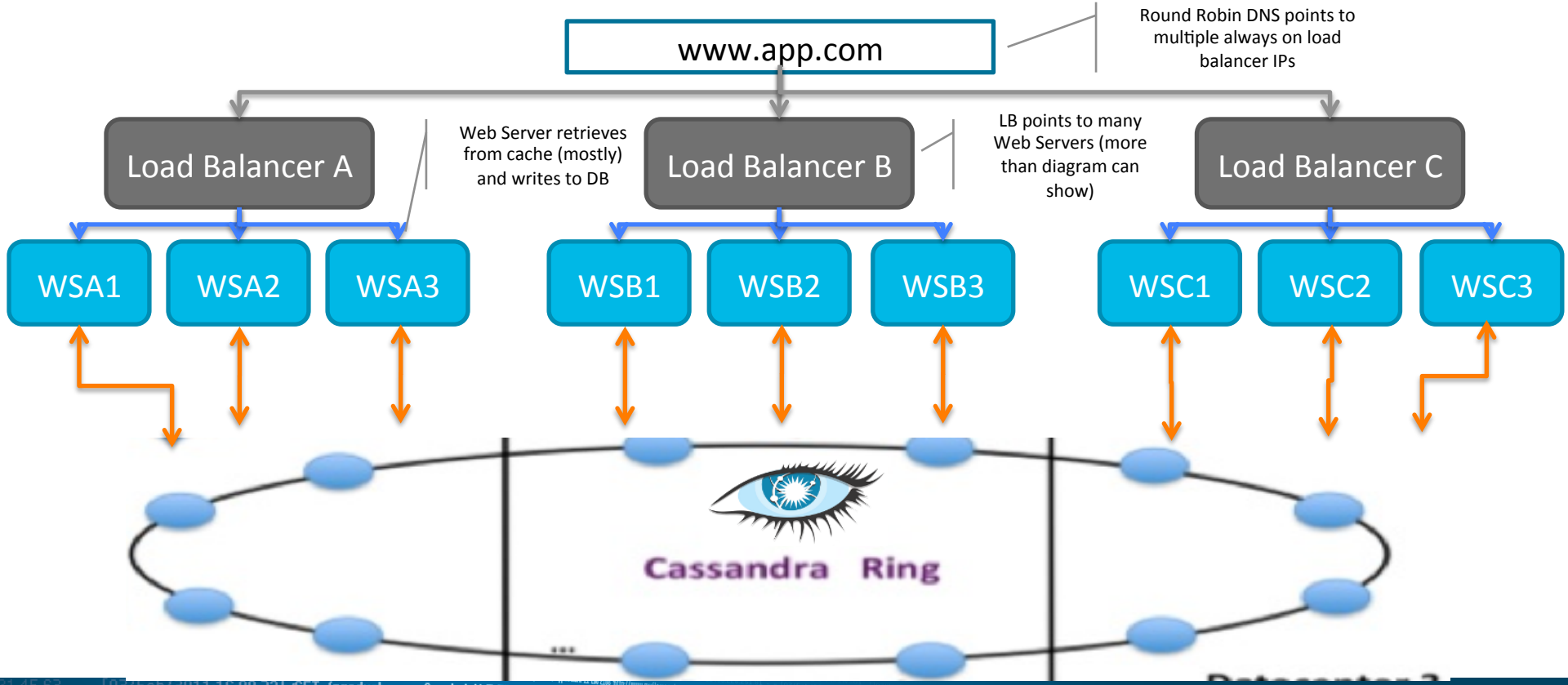
- Nodes are organized into Clusters
- Clusters contain data organized into Keyspaces, similar to a Database Instance or Splunk Index
- Keyspaces can contain multiple “Column Families”, which can be considered analogous to tables
- Note columns cannot be required, and additional columns may be added at any time to a given column family
- A column families can have any number of columns or be completely dynamic columns
- Column names can also be used for storing data

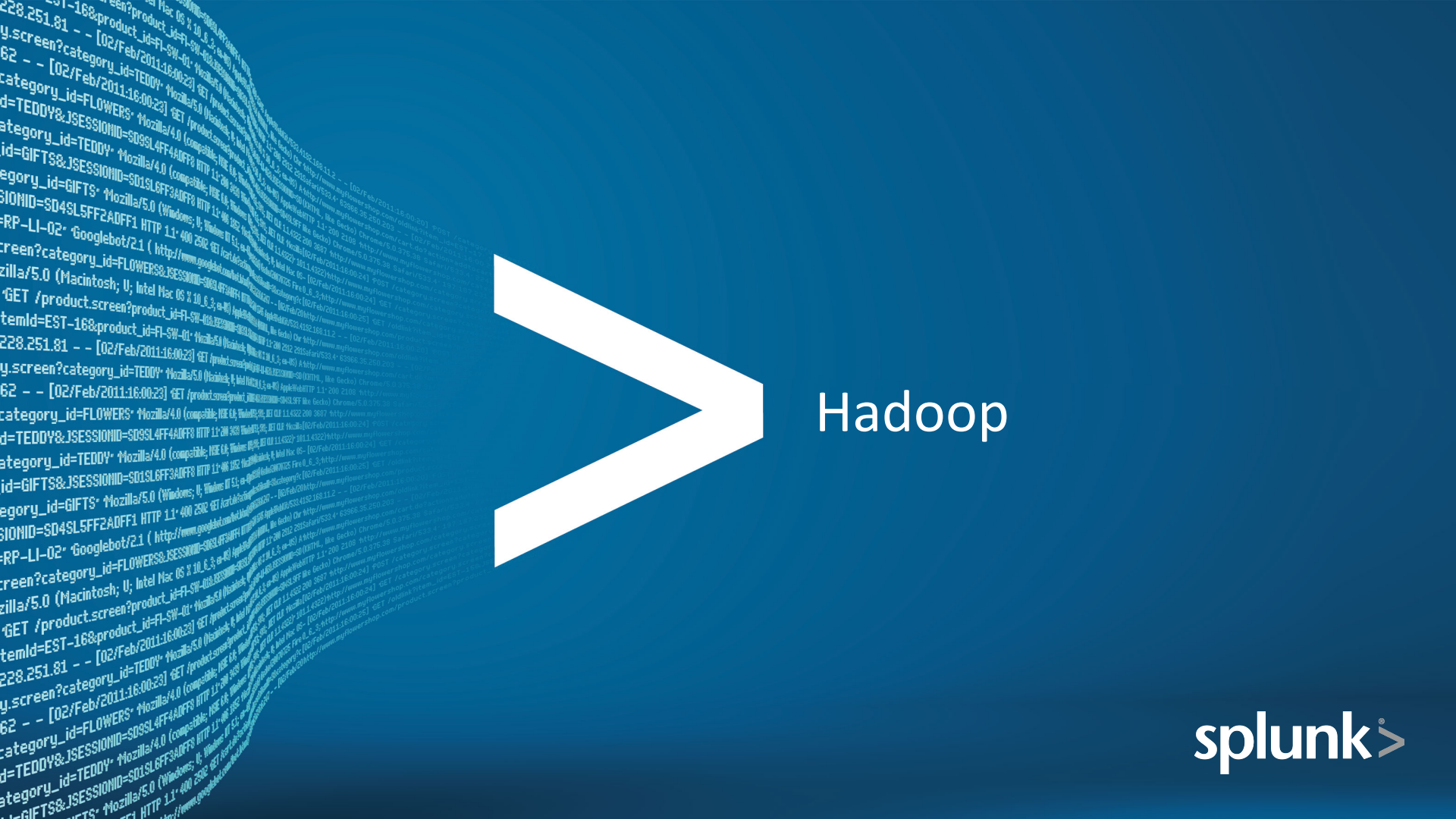
Rowkey	Column Family Employee			Column Family Contact	
EmpID	First Name	Middle Initial	Last Name	Phone	Email
1	Joe		Blow	555-1212	joe@...
2	Sara	M	Name		
3	Srinivas			555-1234	
4				555-4321	jsmith@...

Typical Internet Application Architecture



Typical Internet Application Architecture

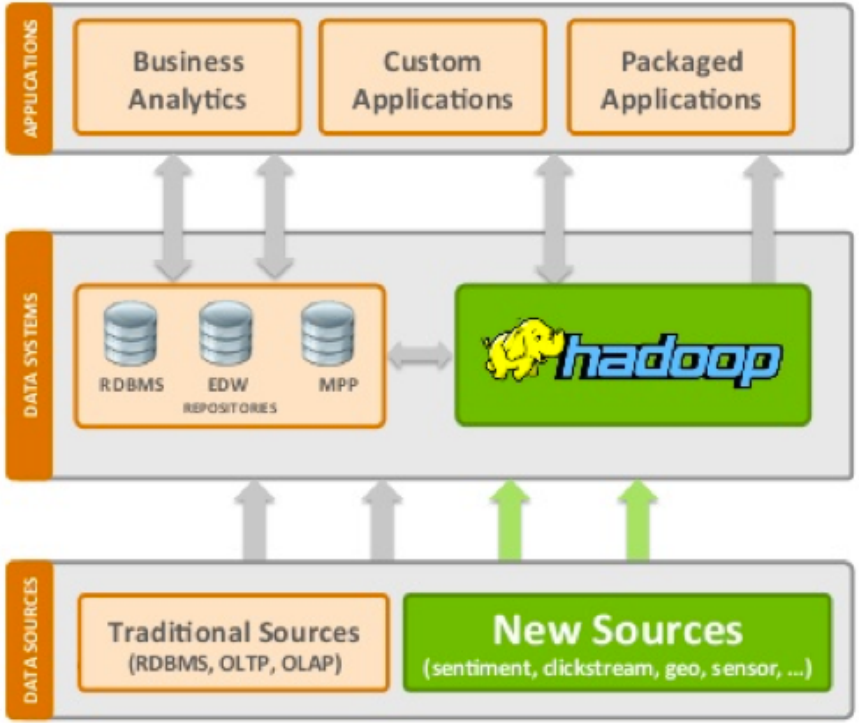



Hadoop
splunk>

What Makes Hadoop Different?

- Ability to scale out to Petabytes in size using commodity hardware
- Processing (MapReduce) jobs are sent to the data versus shipping the data to be processed
- Hadoop doesn't impose a single data format so it can easily handle structure, semi-structure and unstructured data
- Hadoop is a giant storage pool with mostly batch oriented ways to retrieve the large datasets and apply distributed computing methods

Movement away from Legacy EDW

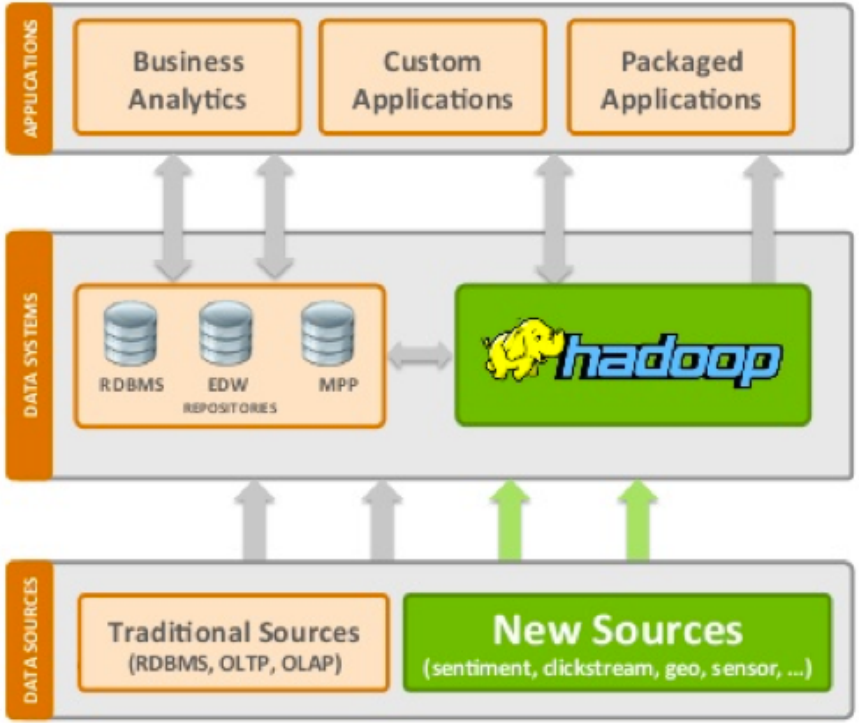


The Costs of storage, SQL licensing and required number of humans for these systems does not work at the scale of data today.

More and more new data sources are added all the time as companies see more value and most of these do not fit a relational model

People want a new way to do their on ad hoc Analysis and not rely on a team of EDW resources yo make data available to them in hard reports

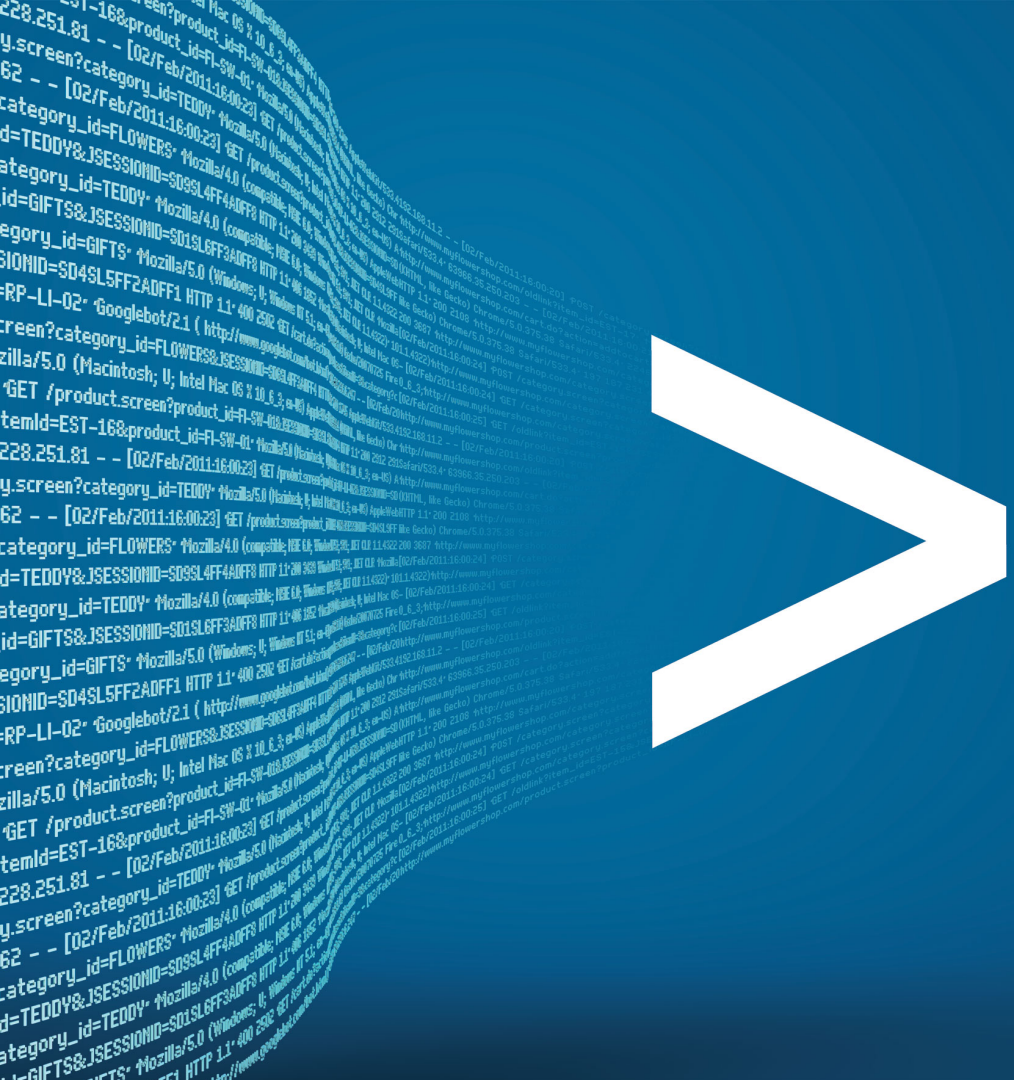
Future end of Legacy ETL $$$$



The Costs of storage, SQL licensing and required number of humans for these systems does not work at the scale of data today.

People are no longer willing to accept the bias of a person of team in the ETL process to decide how to answer the questions that need to be asked

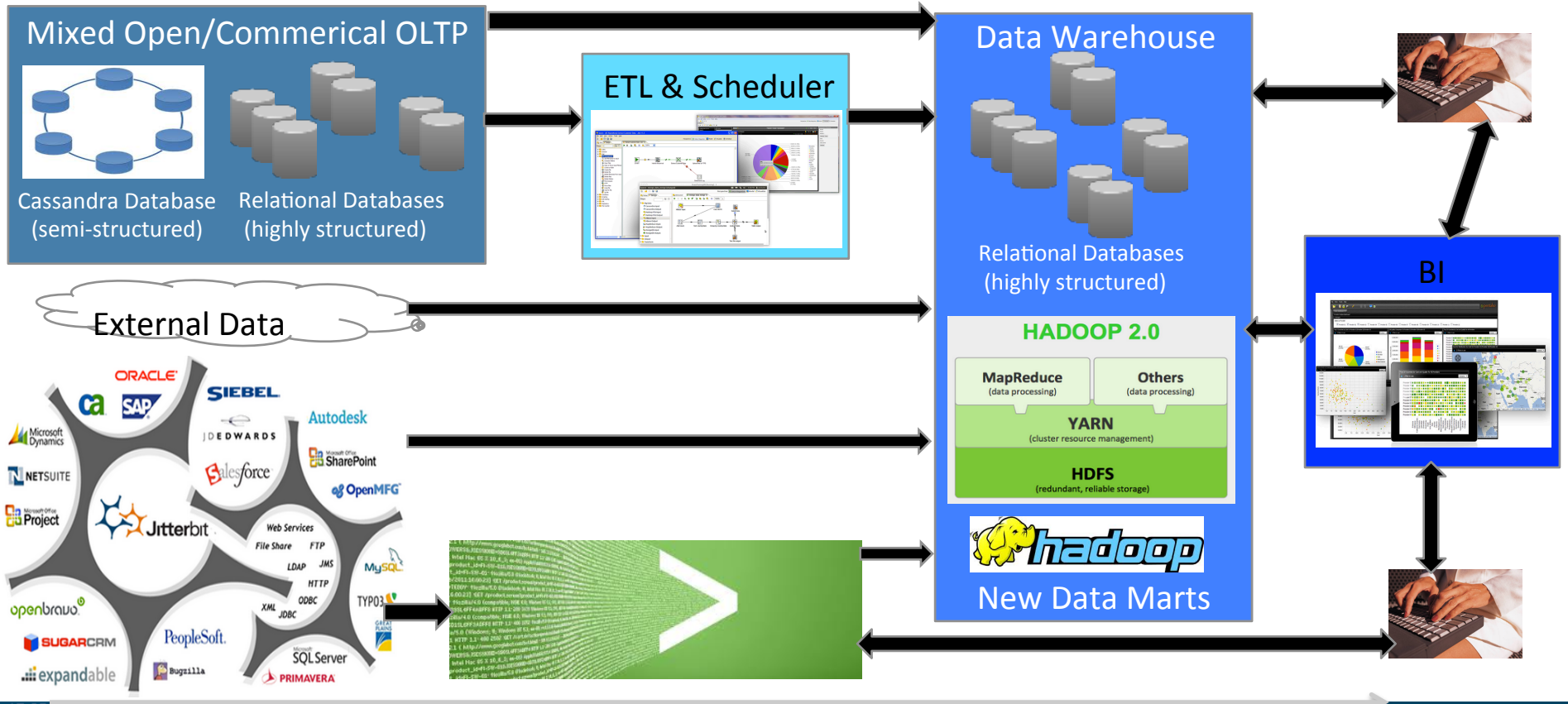
People are now starting to see that the golden nuggets in the data have been ETL's out for years now that they can get access to all the raw data



Solutions

splunk>

The New End to End View



Real World Big Data Solutions Look Something Like This