NoSQL Systems for Big Data Management

Venkat N Gudivada
East Carolina University
Greenville, North Carolina
USA

Outline

RDBMS vs NoSQL

- RDBMS market revenue in 2011 was $26 billion - 2013 IDC Report
- RDBMS revenues predicted to reach $41 billion by 2016 - 2013 IDC Report
- Object-oriented DBMS (OODBMS) of 1980s
- Native XML databases of 1990s
- Special requirements of Web 2.0, Big Data, IoT, and mobile applications
- NoSQL products will generate $14 billion in revenues over the period 2013 – 2018 – Market Research Media
- Worldwide NoSQL market is expected to reach $3.4 billion by 2018 – Market Research Media

Database Requirements for Some Applications

- Flexible data models that evolve over time
- Near real-time query response times
- Concurrent users in the order of millions
- Terabyte scale data volumes spread across multiple data centers to improve locality
- High availability and elastic scaling without system downtime
- Simple data model but fast inserts and lookups are critical for some applications
- In others, updates are almost non-existent and are implemented as a delete followed by an insert
- RDBMS for such applications – the proverbial square peg in a round hole

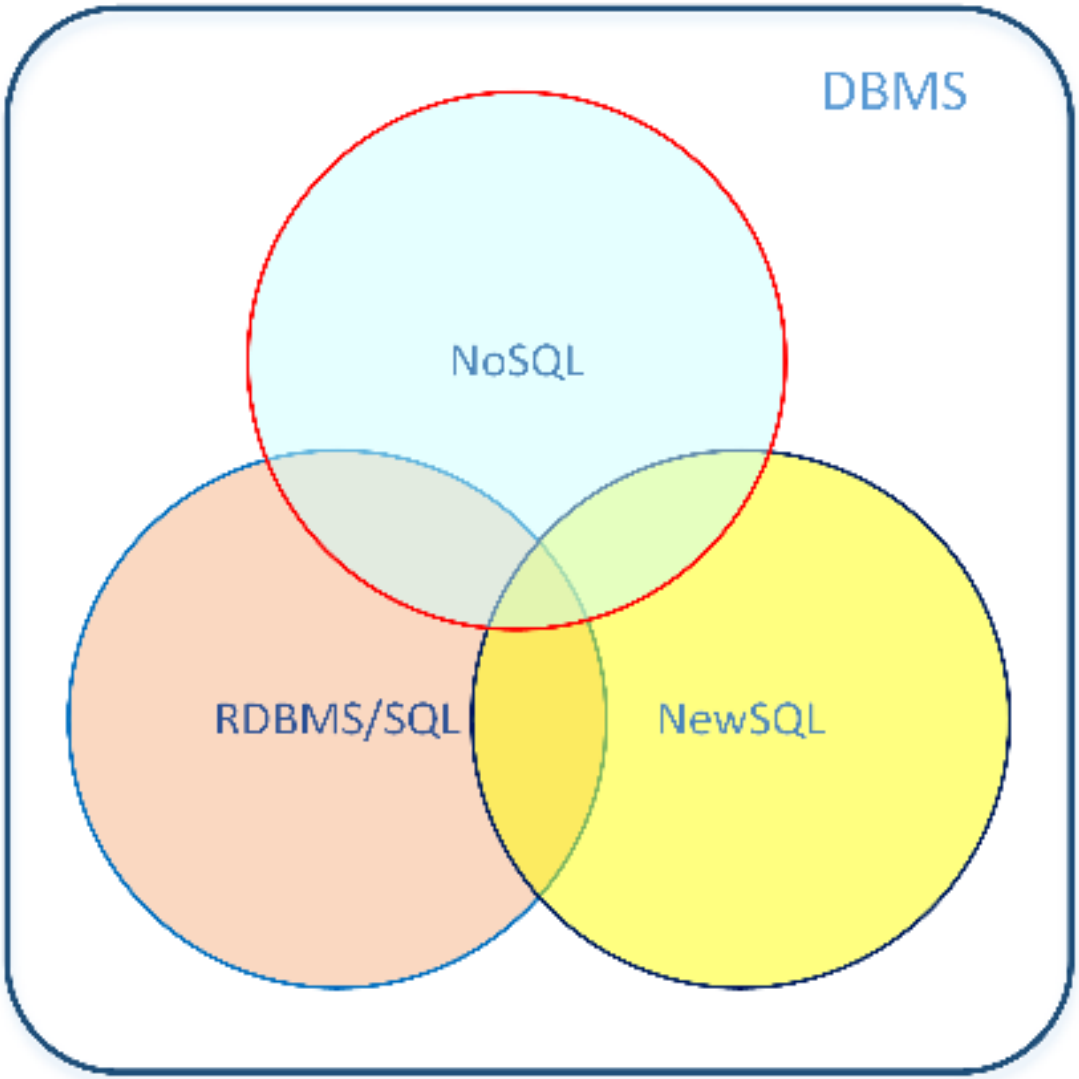
Explosive Growth of NoSQL Systems

- DB-Engines
- NoSQL Databases
- Breakthrough Analysis
- Data Blogger
- EMC Big Data
- IBM Big Data & Analytics Hub
- Forrester Big Data Blogs
- Top 8 Trends in Big Data for 2016
- http://www.dataversity.net/
- http://planetbigdata.com/

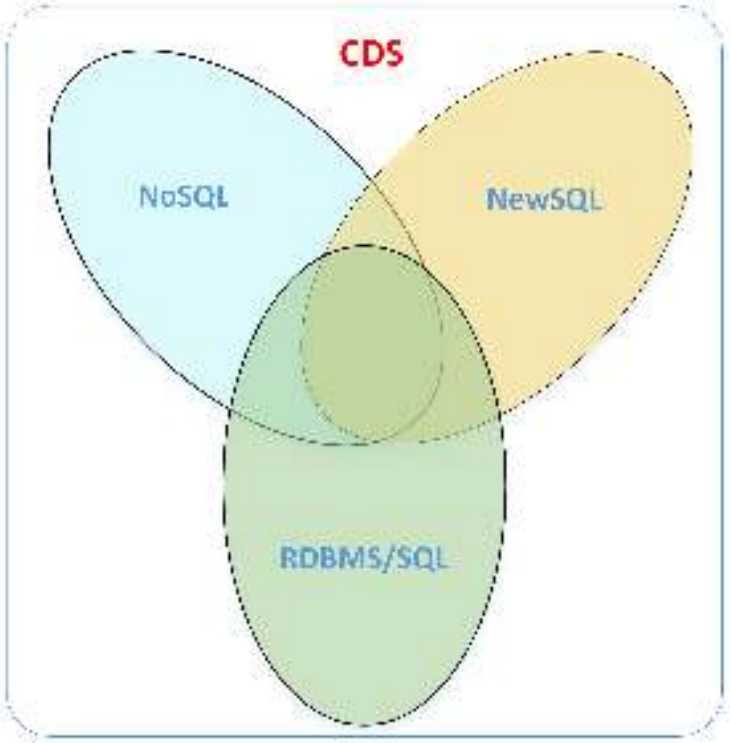
NoSQL Systems

- Performance at scale
- Do not provide all of the RDBMS features
- Principally focus on providing near real-time reads and writes for applications with millions of concurrent users
- Terminology – NoSQL, NewSQL, Not only SQL, and non-RDBMS
- Renaissance in data management - OO DBMS and Native XML
- Consistency, availability, and partition tolerance (CAP)
- Impossible for any distributed system to achieve all these three features simultaneously
- View CAPs as spanning a spectrum rather than as binary values

SQL, NoSQL and NewSQL



SQL, NoSQL and NewSQL



(a)

(b)

NoSQL Systems

- Bigtable, Hypertable, HBase, MongoDB, and Redis strive for consistency and partition tolerance
- DynamoDB, CouchDB, Cassandra, SimpleDB, Voldemort, and Riak emphasize availability and partition tolerance
- RDBMS provide consistency and availability, and not partition tolerance
- Some RDBMS vendors are introducing new features into their products to mitigate the potential risk posed by NoSQL
- Rivalry between the RDBMS and NoSQL vendors
- Exaggeration of features as well as confusion in the current DBMS product space

Web 2.0 Applications

- Web 2.0 – new generation of systems that allow users to interact and collaborate with each other and contribute content
- Social media applications, blogs, wikis, folksonomies, image and video sharing sites, and crowdsourcing – MySpace, Twitter, Facebook, YouTube, and Flickr
- Critically depend on integrated and real-time data from diverse sources
- Preponderance of insert and retrieval operations on a very large scale
- Fraud and anomaly detection, dynamic pricing, real-time order status through supply chain visibility, superior customer service, real-time predictive analytics, user experience personalization, and Web server access log analysis

Big Data

- Data that is too big and complex to capture, store, process, analyze, and interpret using the state-of-the-art tools and methods
- Five Vs: volume, velocity, variety, veracity, and value
- Data volume is in the order of terabytes (2^{40}) or even petabytes (2^{50}), and is rapidly heading towards exabytes (2^{60})
- Stream data processing
- Data heterogeneity
- Secure data provenance
- Predictive models to answer what-if queries, discovering counterintuitive insights and actionable intelligence

Big Data

- Large Hadron Collider experiments – 50 million sensors capturing data about nearly 600 million collisions per second
- 2013 Nobel Prize in Chemistry – measuring and visualizing the behavior of 50,000 or more atoms in a reaction over the course of a fraction of a millisecond
- Square Kilometer Array telescope – goal is to answer fundamental questions about the origin and evolution of the Universe

Mobile Computing and Location-based Services

- Location-based services have become an important channel for marketing, sales, and brand development
- Location-based services leverage users' geographic location information and context in providing relevant and timely services to users
- Mobile devices impose additional requirements such as disconnected operation and delayed synchronization on location-based services
- Applications include locating a nearby business or service (e.g., retail store, ATM, restaurant), receiving alerts about traffic jams and severe weather, mobile advertising, and recommending social events in a city
- Facebook Places, Foursquare, Gowalla, and Where

Functional Inadequacy of RDBMS

- Accommodate increased workload and data volume by using faster CPUs, more memory, and bigger and faster disks – vertical scaling – Oracle Real Application Clusters (RAC)
- Instead, horizontal scaling is needed for economic reasons
- Sparse data and partial record updates are the norm in some Big Data applications
- Complete database schema does not exist up front and evolves over
- ACID-compliant transactions are an overkill as they require only relaxed consistency
- Multiple query methods ranging from simple REST API to complex and ad hoc queries are required
- Horizontal scaling using processors built from commodity hardware

Functional Inadequacy of RDBMS

- Interactive processing of ad hoc queries mandates massive parallel computation using schemes such as MapReduce
- Non-overlapping data partitioning (aka sharding) is necessary to address the data volumes
- Auto-sharding is highly desirable
- Simple data replication is mandatory to ensure high availability
- Built-in support for versioning and compression is needed

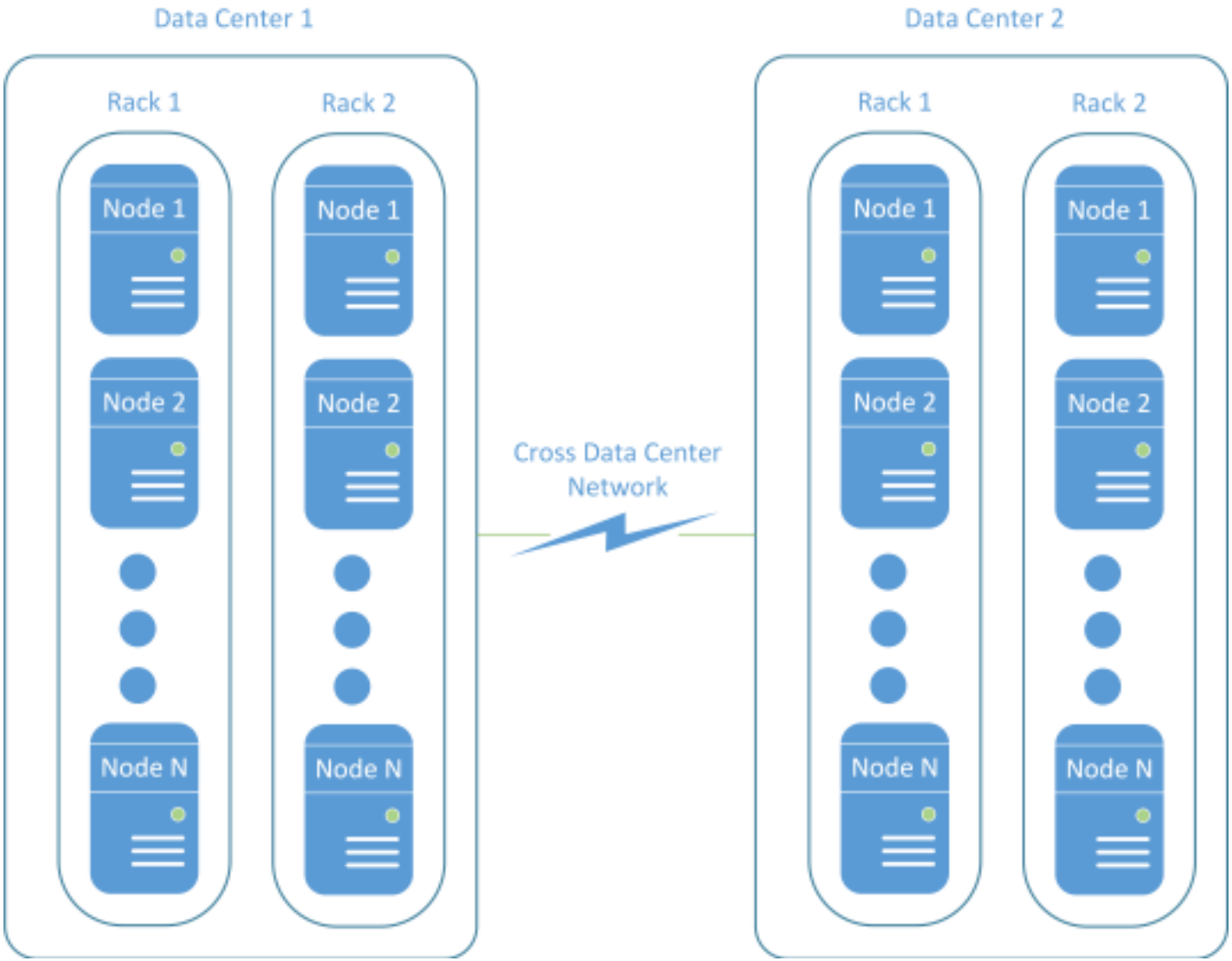
NoSQL Systems

- NoSQL systems significantly vary in functionality from each other
- Riak is highly scalable and available
- MongoDB's defining characteristic is managing deeply nested structured documents and computing aggregates on the documents
- Neo4j excels at managing data that is rich in relationships
- Redis is known for its data structures and elegant query mechanisms
- HBase offers extreme scalability, reliability, and flexibility, however, these features come with considerable complexity

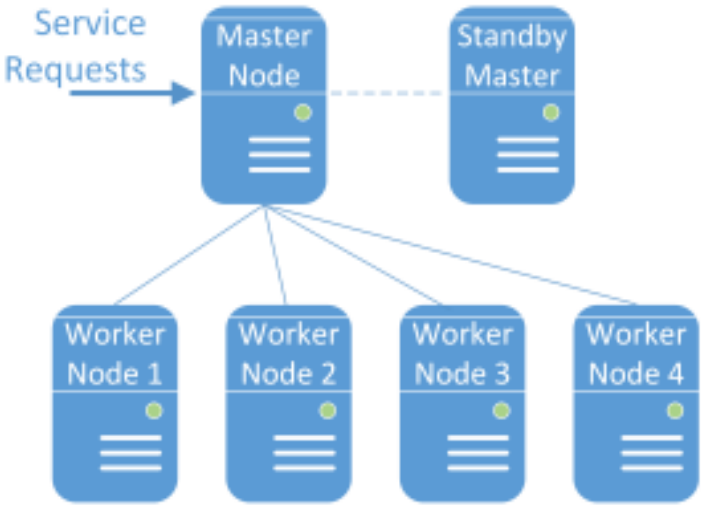
NoSQL Systems

- Use distributed computing architectures
- Feature an assortment of data models query languages and data consistency models
- Each class of systems targets a category of applications
- System architectures are optimized for achieving the primary goal of providing performance at scale
- Command line queries moving towards declarative query languages

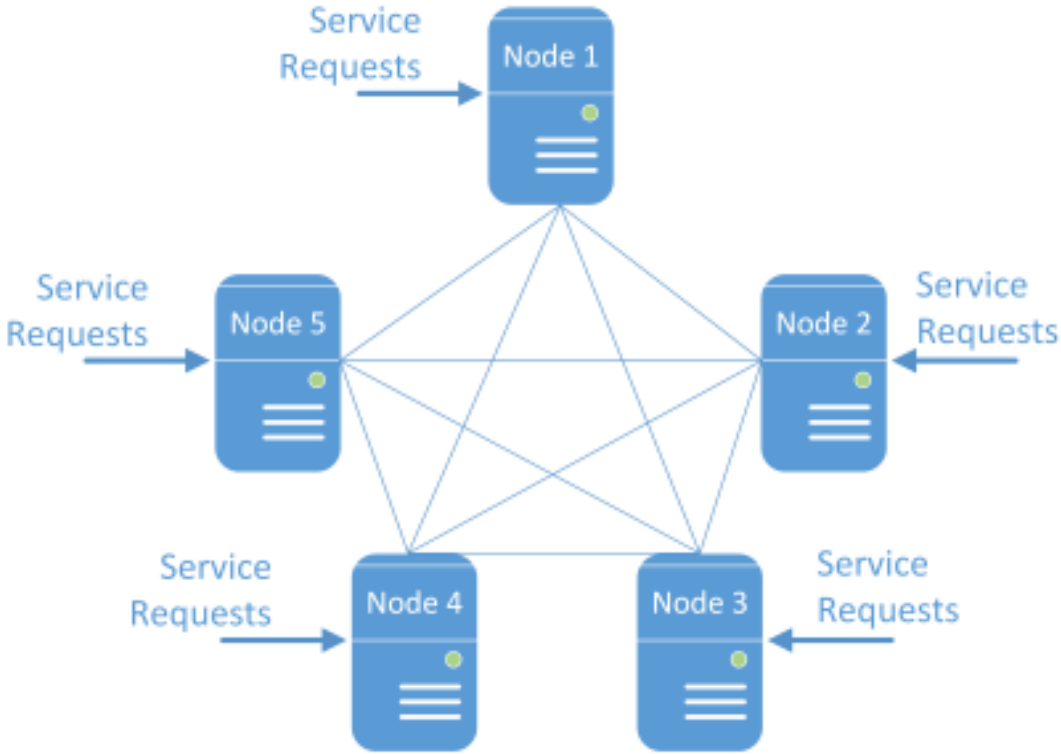
A NoSQL Cluster



Shared-nothing Architecture



(a) Master-Worker Shared-Nothing Architecture



(b) Master-Master Shared-Nothing Architecture

NoSQL Architectures

- Elastic scalability through Database as a Service (DBaaS)
- Amazon's Simple Storage Service (S3) and DynamoDB are two examples of DBaaS
- Sharding and replication are orthogonal concepts – contribute to high availability
- ACID vs. BASE
- Basic availability – support for disconnected client operation and delayed synchronization, and tolerance to temporary inconsistency and its implications
- Data consistency spanning a spectrum from strict to eventual
- Soft state refers to state change without input, which is required for eventual consistency

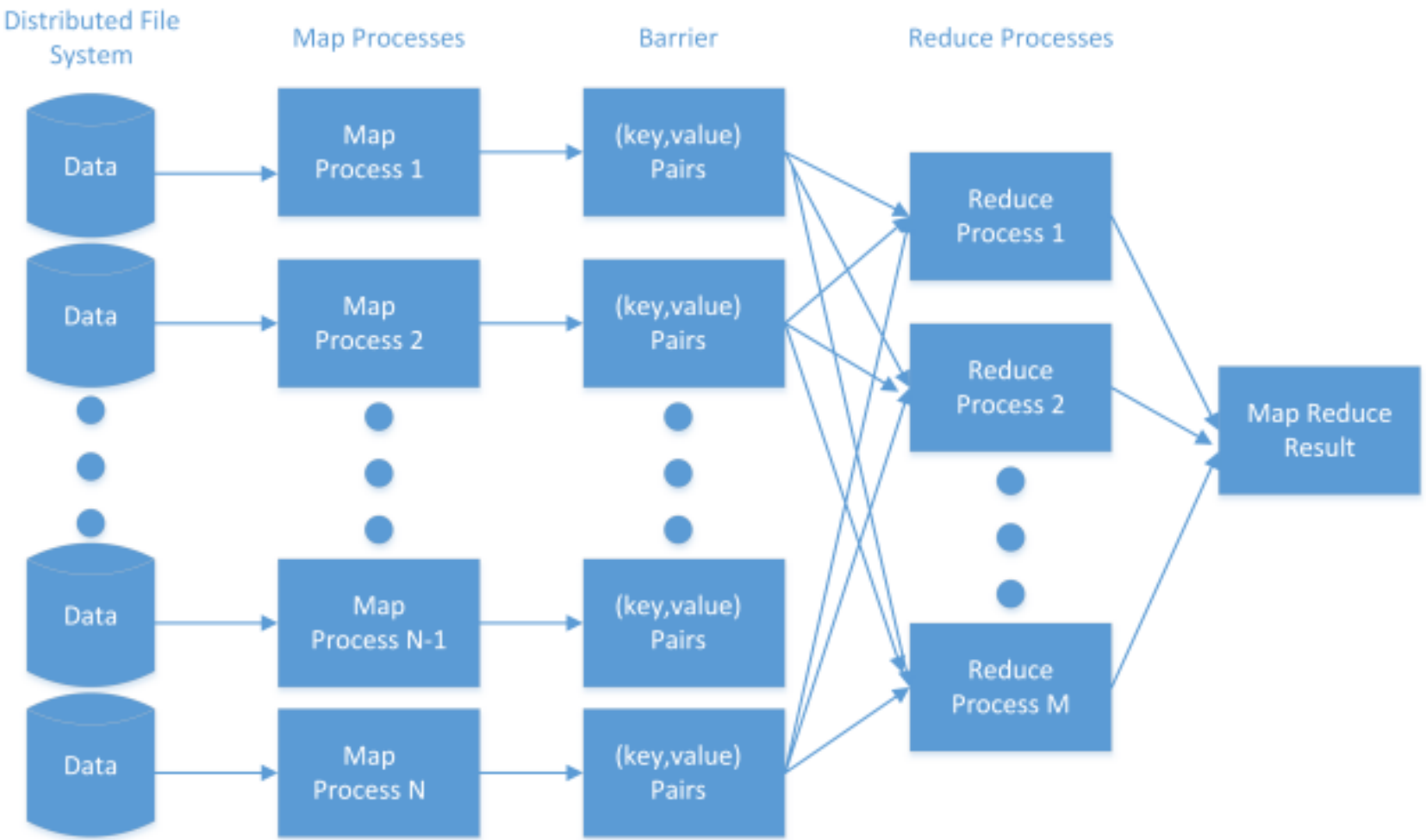
NoSQL Architectures

- Applications can benefit from choosing a right level of data consistency
- Choose a level closer to strong consistency for writes but a weaker consistency for reads – write operations will take longer to complete without contributing to read operation latency
- Type of architecture , data partitioning and replication methods determine the degree of difficulty involved in implementing data consistency models
- Some NoSQL systems provide options for clients to choose a desired data consistency level – tunable consistency

NoSQL Architectures

- Cassandra provides options for choosing consistency levels for reads and writes separately
- Write consistency level specifies on how many replicas the write must succeed before returning an acknowledgment to the client application that the write is successful
- Read consistency specifies how many replicas must respond before a result is returned to the client application

Ad hoc and Compute-intensive Queries



Data Models

- Key-value – optimized for fast inserts and retrievals; updates are almost non-existent and are implemented as a delete followed by an insert
- Column-family – optimized for storing extremely large and sparse data with efficient support for primary key-based retrieval and partial record updates
- Document-oriented NoSQL systems – JSON-style document data model
- Graph model for relationships-rich data
- RDF triples
- XML data models

Conclusions

- Data consistency, high availability, and partition tolerance are the three primary concerns that determine which data management system is suitable for a given application
- Scalability in NoSQL systems comes at the cost of transactional guarantees, application maintained relationships, and simpler data models
- Trade-off between scaling and ad hoc queries – massive scaling entails simple queries; ad hoc queries typically require programming and parallel execution

Conclusions

- Many NoSQL system hide the complexity of sharding from the application developer by providing built-in algorithms for auto-sharding; however, there is an implicit assumption that shards are independent and database transactions are confined to a single shard
- If a transaction spans multiple shards, some systems simply reject such transactions
- If multi-shard transactions are prevalent, suitability of NoSQL systems needs careful analysis
- It is quite difficult to determine which system is the right match
- This problem is further exacerbated by rapidly evolving features

Conclusions

- Type of data to be managed – structured, semi-structured, unstructured, or a mix of these types?
- Type of query language desired - declarative, procedural, or both?
- Are ad hoc queries so prevalent that a native MapReduce framework is required?
- Are the users geographically distributed to warrant data partitioning?
- Is an algorithmic approach needed for data partitioning with automated redistribution to avoid hotspots or performance bottlenecks?
- Are the read or write throughput levels so high for a conventional RDBMS to handle?

Conclusions

- What is the minimal level of consistency essential for reads and writes?
- Is automated failover with no service interruption essential?
- Does the application load fluctuate unpredictably to provision resources dynamically without service interruption?
- What type and level of user authentication and authorization are required?
- What type of replication is needed?
- Is versioning of data required?
- Is cloud hosting a required deployment option?