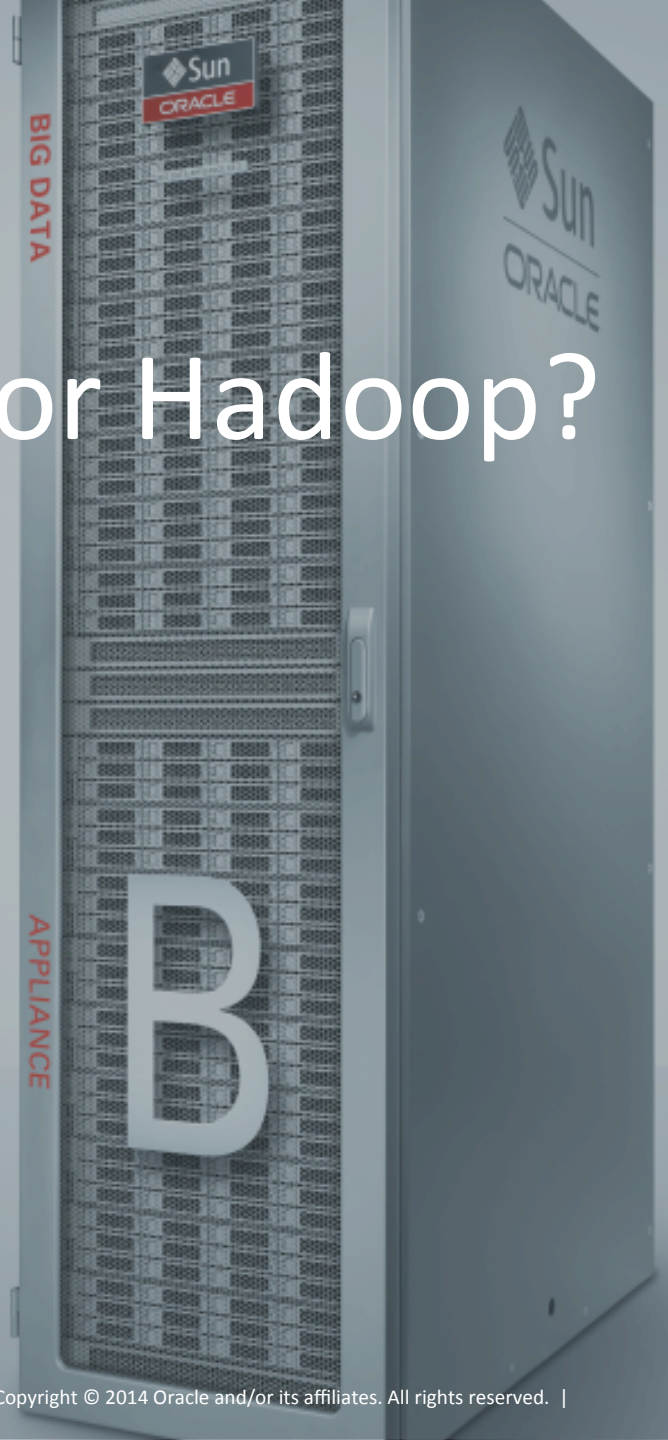
Using RDBMS, NoSQL or Hadoop?

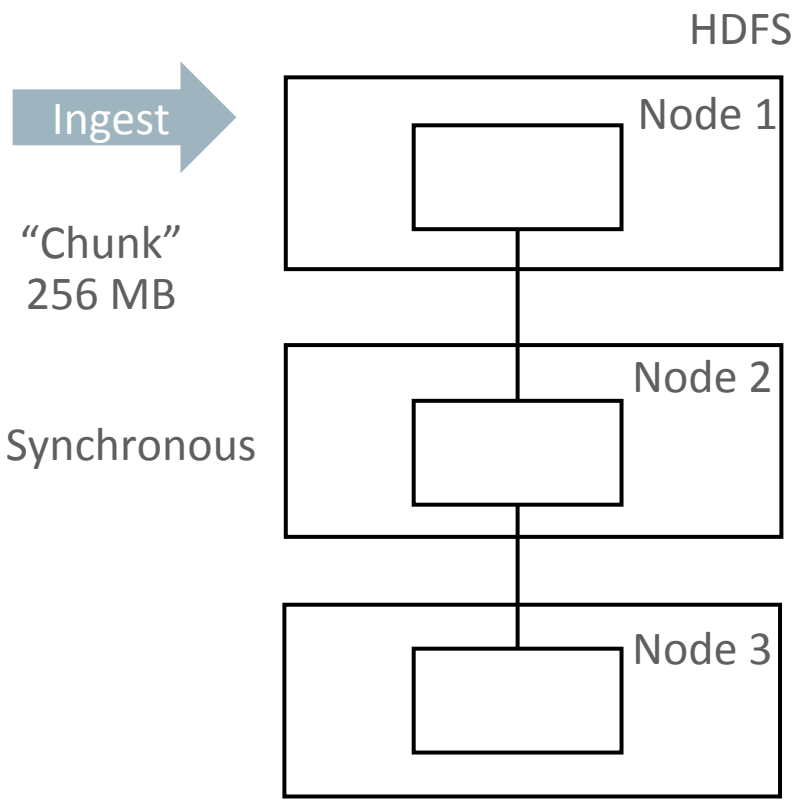
DOAG Conference 2015

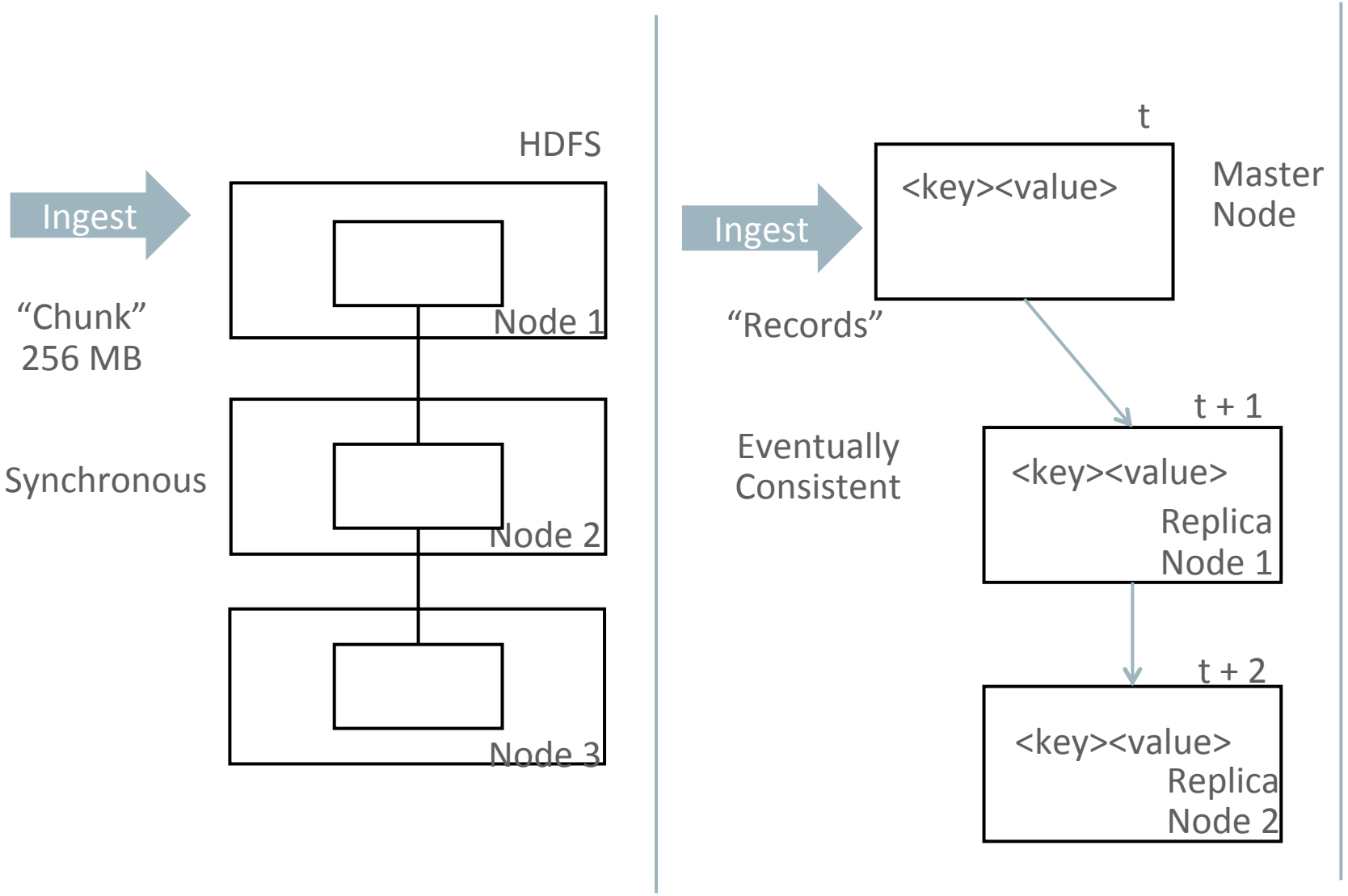
Jean-Pierre Dijcks
Big Data Product Management
Server Technologies

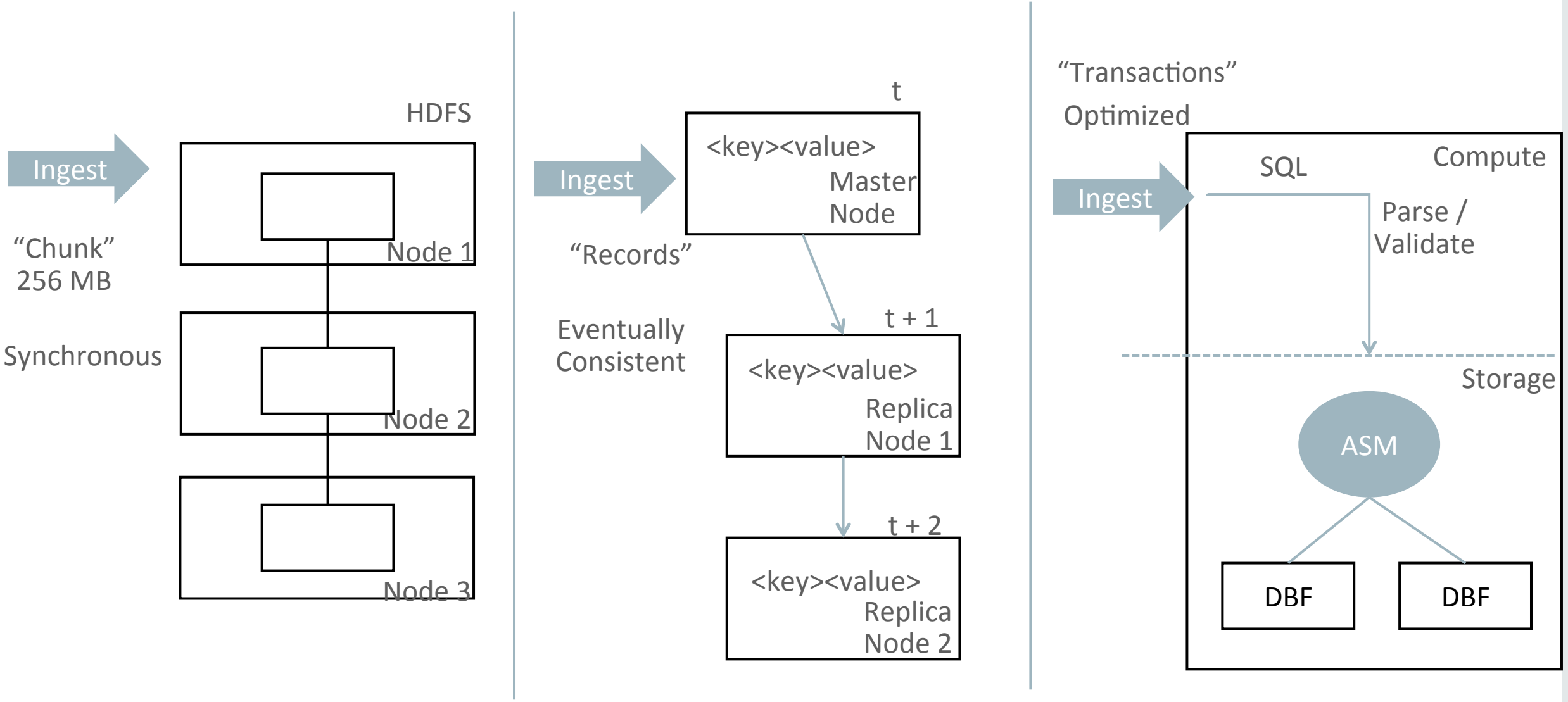


ORACLE

Data Ingest





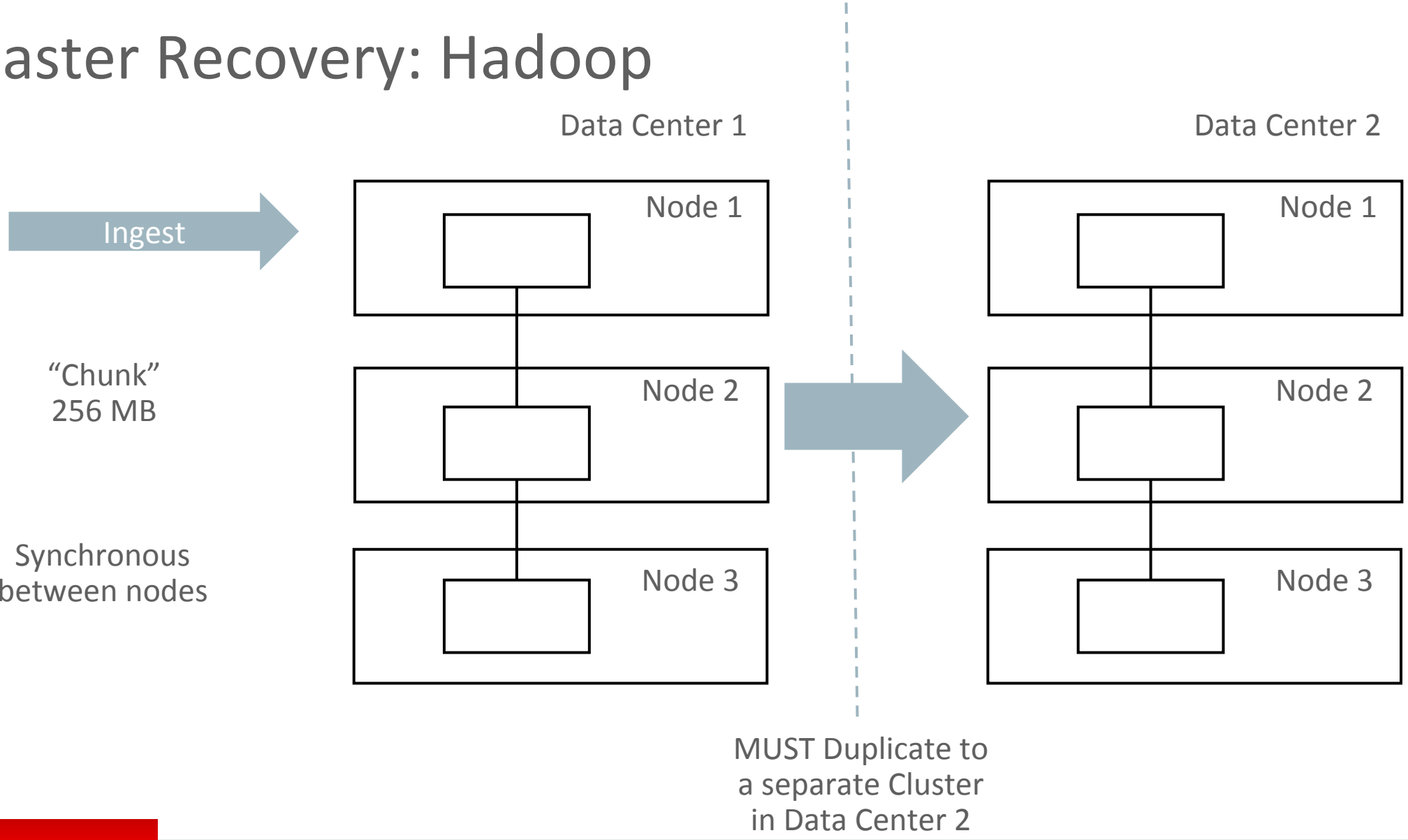


High-level Comparison

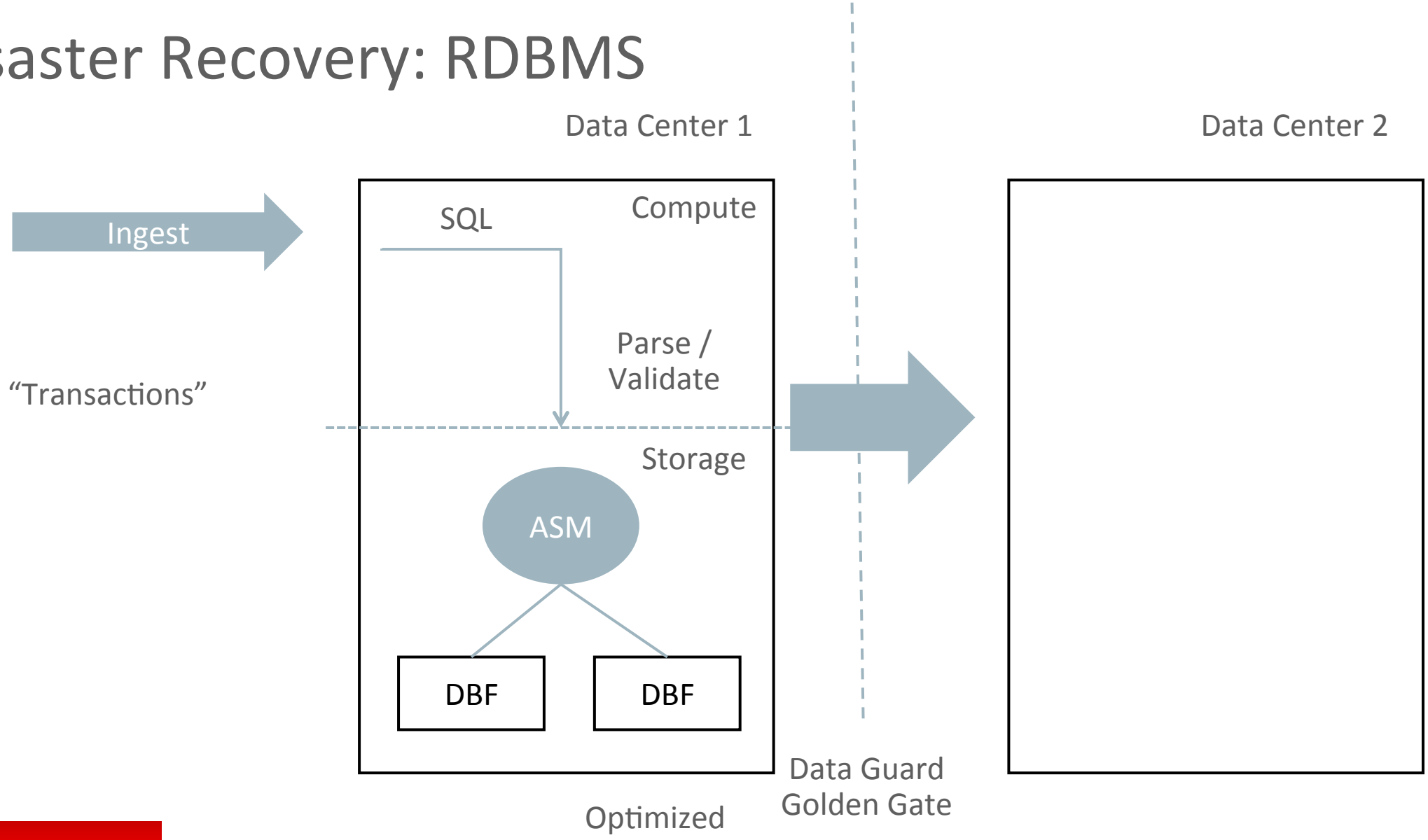
		HDFS	NoSQL	RDBMS
Ingest	Data Type	Chunk	Record	Transaction
	Write Type	Synchronous	Eventually Consistent	ACID Compliant
	Data Preparation	No Parsing	No Parsing	Parsing and Validation

Disaster Recovery - High Availability Across Geography

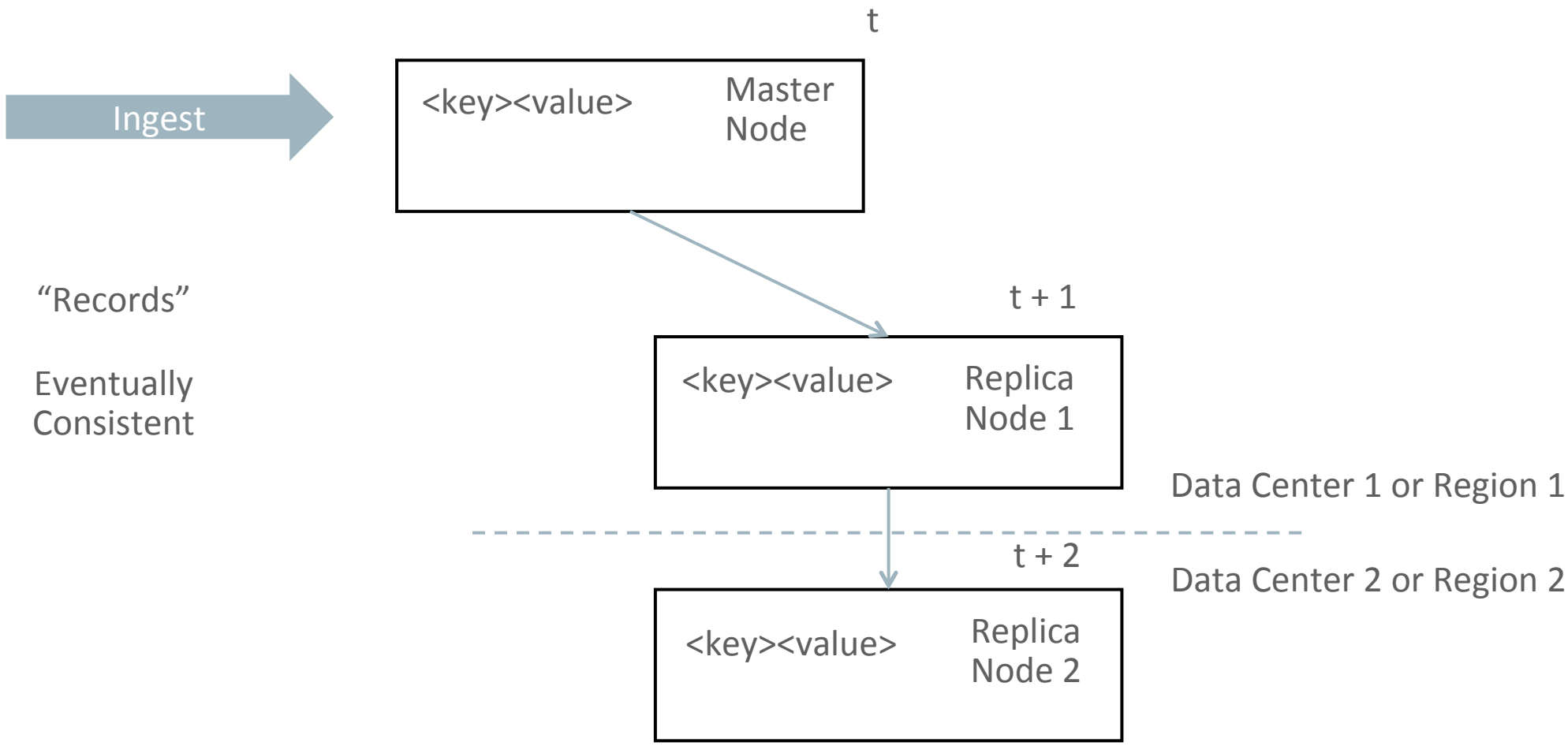
Disaster Recovery: Hadoop



Disaster Recovery: RDBMS



Disaster Recovery: NoSQL Option



Note on HBase

- Don't confuse HBase with NoSQL in terms of geographical distribution options
- Hbase is based on HDFS storage
- Consequently, HBase cannot span data centers as we see in other NoSQL systems

Big Data Appliance includes Cloudera BDR

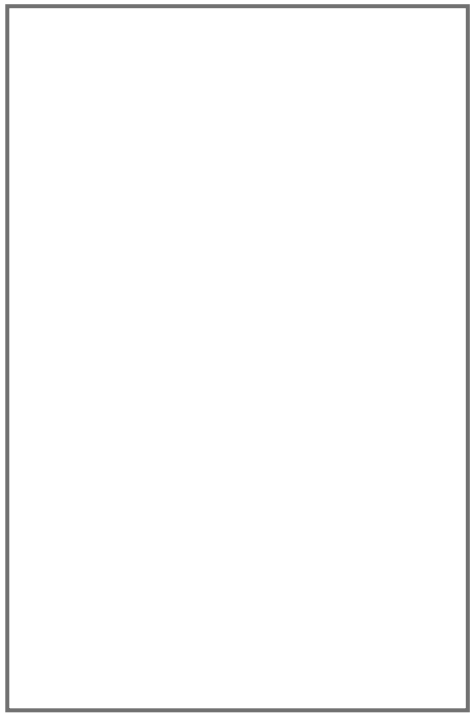
- Oracle packages this on BDA as a Cloudera option called BDR (Backup and Disaster Recovery)
- BDR = DistCP
  - distributed copy method leveraging parallel compute (MapReduce like)
- BDR is NOT proven as Data Guard or GG
  - Most importantly, think of BDR as a batch process not a trickle
- BDR is included (no extra charge) on BDA

High-level Comparison

		HDFS	NoSQL	RDBMS
Ingest	Data Type	Chunk	Record	Transaction
	Write Type	Synchronous	Eventually Consistent	ACID Compliant
	Data Preparation	No Parsing	No Parsing	Parsing and Validation
DR	DR Type	Second Cluster	Node Replica	Second RDBMS
	DR Unit	File	Record	Transaction
	DR Timing	Batch	Record	Transaction

Access Paths to Data Sets

Data Sets and Analytical SQL Queries: Hadoop

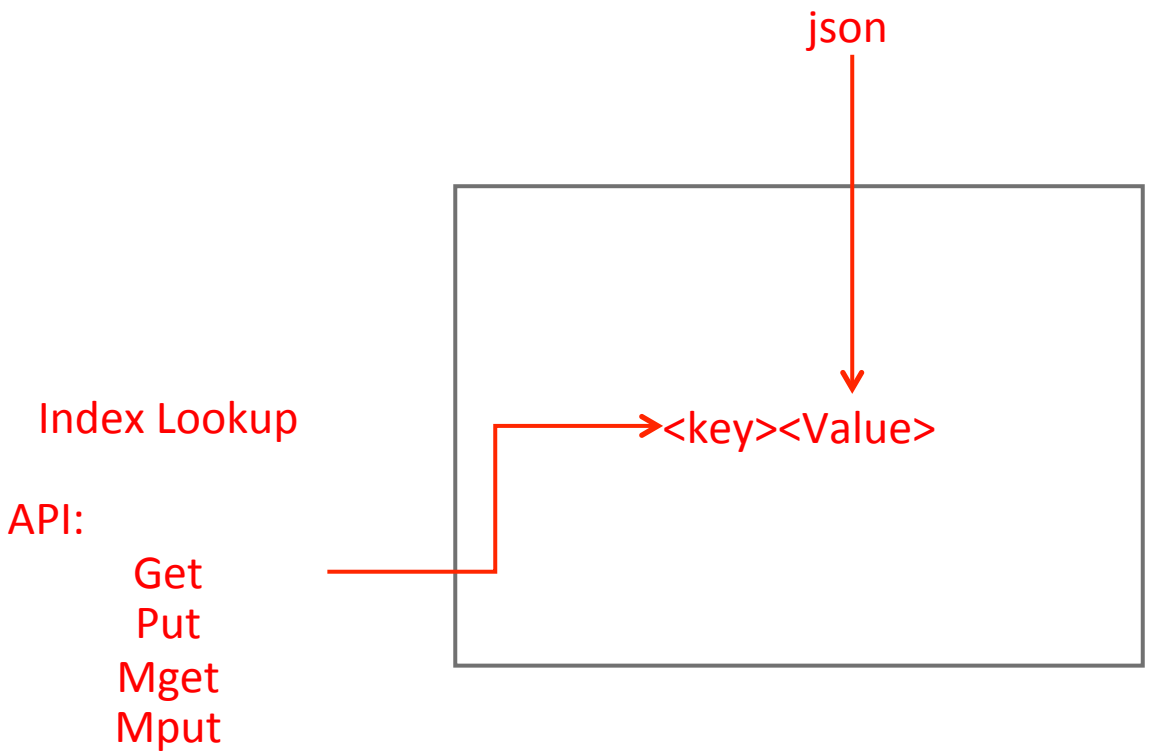


INSERT
- Data is NOT parsed on ingest, example JSON document is not parsed.. Original files loaded an broken into chunks
- Does not like small files

UDPATE
- NO update allowed (append only)
  - Expensive to update a few KB by replacing a 256MB chunk (as part of a file replacement)

SELECT
- Scan ALL data even for a single "row" answer
- SQL access path is a "full table scan"
- Hive optimizes this with
  - Metadata (Partition Pruning)
  - MapReduce in Parallel to achieve scale

Data Sets and Analytical SQL Queries: NoSQL



PUT (Insert)

- Data is typically not parsed, example, JSON document is loaded as a value with a key

GET (Select)

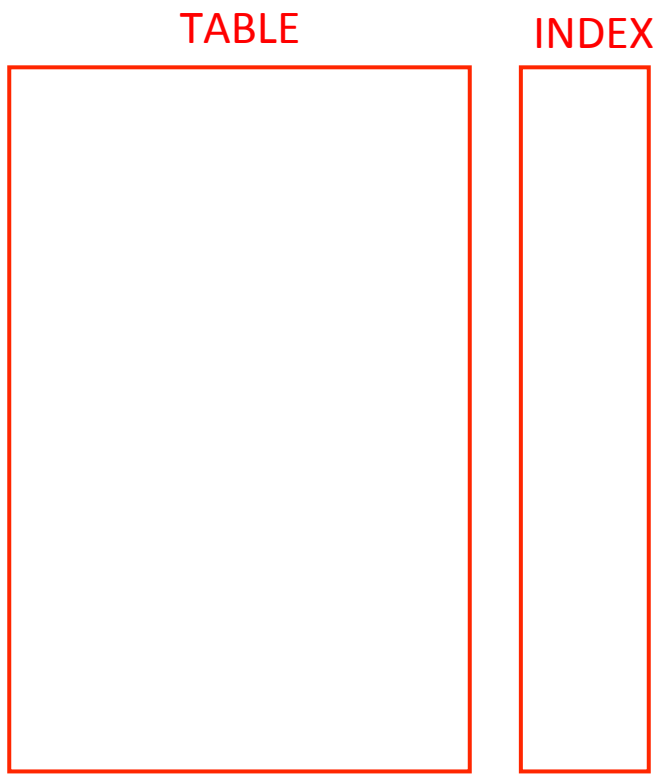
- Data Retrieval through “Index Lookup” based on KEY
  - Only access path is index (primary or secondary)
- If retrieving from Replica may not be consistent (yet)
- Generally do not issue SQL over NoSQL
- Not used for large analysis, instead used to receive and provide single records or small sets based on the same key

Data Sets and Analytical SQL Queries: RDBMS

Data is parsed

Metadata is available on write

Auxiliary structures enable myriad of access paths



INSERT
- Parse data and optimize for retrieval
- Adhere to transaction consistency

UPDATE
- Enables individual records to be updated
- With read-consistency
- Row level locking etc.

SELECT
- Read-consistent guaranteed
- SQL Optimization:
  - Choose the best access path based on the question and database statistics
  - Optimized joins, spills to disk etc.
  - Supports very complex questions

High-level Comparison

		HDFS	NoSQL	RDBMS
Ingest	Data Type	Chunk	Record	Transaction
	Write Type	Synchronous	Eventually Consistent	ACID Compliant
	Data Preparation	No Parsing	No Parsing	Parsing and Validation
DR	DR Type	Second Cluster	Node Replica	Second RDBMS
	DR Unit	File	Record	Transaction
	DR Timing	Batch	Record	Transaction
Acces	Complex Analytics?	Yes	No	Yes
	Query Speed	Slow	Fast for simple questions	Fast
	# of Data Access Methods	One (full table scan)	One (index lookup)	Many (Optimized)

Affordable Scale

Low Predictable Latency

Flexible Performance

Performance Aspects

Ingest and Query

A simple set of criteria

Ingest Performance Considerations

HDFS

- No parsing
- Fastest for Ingesting Large Data Sets, like log files
- Bulk write of sets of data, not individual records
- Slower write on a record-by-record basis than NoSQL

NoSQL

- No parsing
- Fastest for Writing Individual Records to the master
- Direct writes on a per-record basis
- Best for Millisecond Write

RDBMS

- RDBMS does work to data (parsing) on ingest
- Ingest speed is slower than NoSQL on a record by record basis due to parsing
- Ingest speed is slower than Hadoop on a file by file basis due to parsing
- Benefits of RDBMS parsing become evident on query performance…

Query Performance Considerations

HDFS

- Requires parsing
- Slowest on Query Time
  - Due to Full Table Scans on top of parsing
  - No query caching
- Able to run complex queries
  - Requires use of MR or Spark programs
  - SQL immature (SQL-92)

NoSQL

- No parsing
- Fastest on Query Time for "get"
- Consistently fast is the goal
- No syntax available to run complex queries

RDBMS

- Fastest SQL query times because of parsing work done on ingest
  - Completely optimized storage formats for IO
  - Oracle knows how to pick stuff out, metadata on files..
  - Least amount of I/O to retrieve records
  - Advanced Caching
- RDBMS can run more complex SQL queries than NoSQL or Hadoop

SQL on Hadoop vs RDBMS

Hadoop

- Pay on QUERY for GAINS on INGEST
- Full table scan
  - Shorten full table scan by adding more nodes to scan through data
- Problem remains, data isn't parsed
  - All raw data must be read for EACH and EVERY read

RDBMS

- Pay on INGEST (Parsing) for GAINS on QUERY
- Optimized Query Path

Benefits Compared

Hadoop	NoSQL	RDBMS
"Schema on Read" Flexibility Write file on disk without error checking Programmatically stitch disparate data together	Low Latency for simple actions	High performance for lots of workloads without end user knowing Parsed Data Multiple Access Paths Advanced Query Caching In-Memory Formats
Affordable Scale	HOWEVER No API to run analytical query	Consistency - exact same result at exact same time
		Traceability of transactions
		Applications become simpler

Concurrency

Concurrency Comparisons

HDFS

- Most Hadoop systems have small number of users running large number of jobs
  - Batch or very frequent micro batch calculations
- Customers report Impala struggles with concurrency (much like Netezza, or worse)
- Immature resource management, not all solutions work nicely

NoSQL

- Very high concurrency
- Geographically distributed
- Must deal with consistency issues
- Great reader farm for publishing data to for example web apps
- Load balancing built-in

RDBMS

- Hundreds of users, hundreds of queries running at the same time
- Queries are balanced over the system
- Resource Management able to control the whole system
- Proven in many large organizations

Cost

Cost

- Customers want reduce cost by moving from RDBMS to Hadoop
  - Hadoop delivers lowest cost per TB
- But which workloads can move from RDBMS to Hadoop?
  - Dealing with transactions? Stay in RDBMS
  - Running Advanced SQL queries? Stay in RDBMS
  - Large numbers of concurrent users? Stay in RDBMS
- Will hit high performance penalty for moving DW to Hadoop because of full table scans

Can SQL on Hadoop solve the problems without sacrifice?

No.. 1 ½ Attempts

Impala

- Solution to speed up Hive and MR
  - Wrote own optimizer and query engine
- Large speedup but at the cost of:
  - No MR so loses scalability
  - Less SQL capabilities so loses BI transparency
  - System cost increases to run Impala (needs add'l memory) so loses some cost advantage

Impala with Parquet

- Convert files into columnar data blocks (Parquet file format)
- Speed up because of columnar formats etc:
- But at a cost:
  - Must run ETL to Parquet (comparable to ingest into RDBMS)
  - Loose schema on read => flexibility
  - Double the data
- Essentially a new columnar DB on HDFS

Conclusion: Choose the Right Tool for the Right Job

Don’t Use something it’s not meant to be


ORACLE®