SQL-ON-HADOOP FOR ANALYTICS + BI: WHAT ARE MY OPTIONS, WHAT'S THE FUTURE?

MARK RITTMAN, COO, RITTMAN MEAD

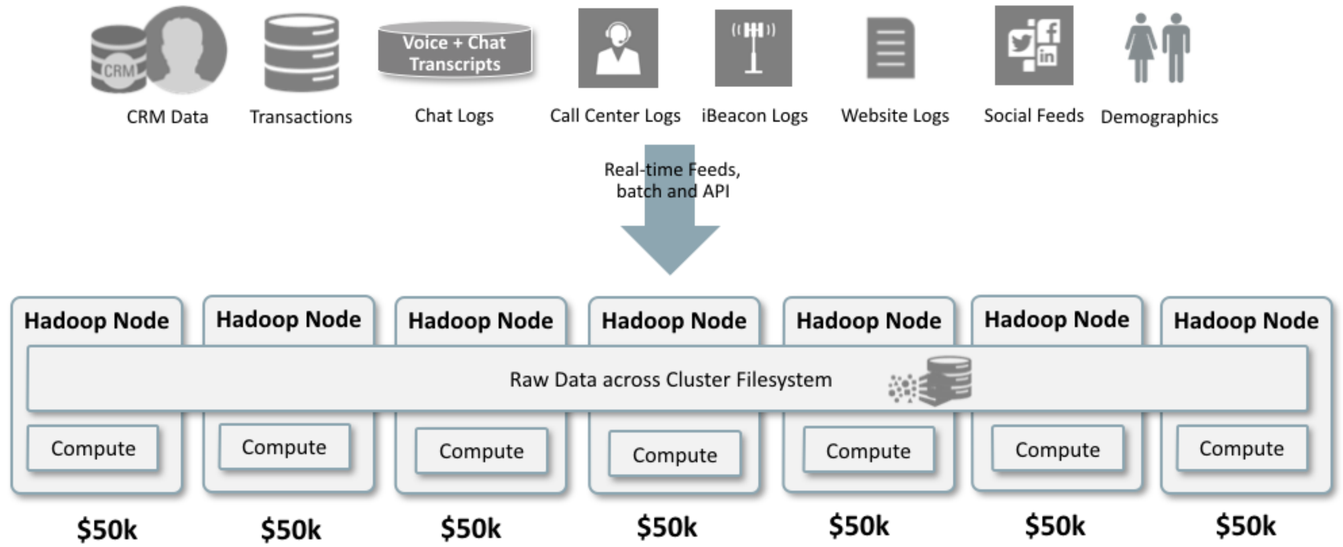
BUDAPEST DATA FORUM, JUNE 2016

Many Organisations are Running Big Data Initiatives

- Many customers and organisations are now running initiatives around “big data”
- Some are IT-led and are looking for cost-savings around data warehouse storage + ETL
- Others are “skunkworks” projects in the marketing department that are now scaling-up
- Projects now emerging from pilot exercises
- And design patterns starting to emerge

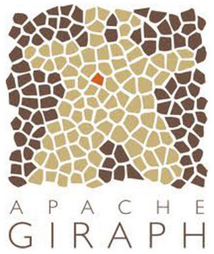
Highly Scalable (and Affordable) Cluster Computing

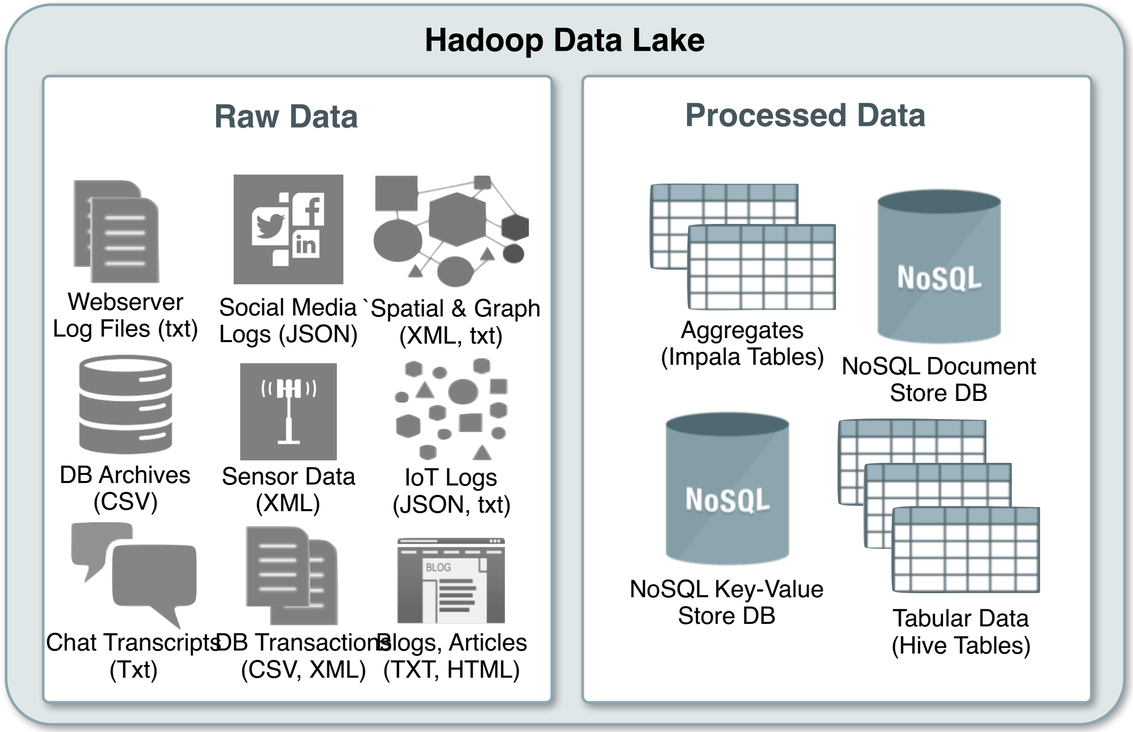
- Enterprise High-End RDBMSs such as Oracle can scale into the petabytes, using clustering
- Sharded databases (e.g. Netezza) can scale further but with complexity / single workload trade-offs
- Hadoop was designed from outside for massive horizontal scalability - using cheap hardware
  - Anticipates hardware failure and makes multiple copies of data as protection
  - More nodes you add, more stable it becomes
  - And at a fraction of the cost of traditional RDBMS platforms



One Platform, Multiple Processing Frameworks

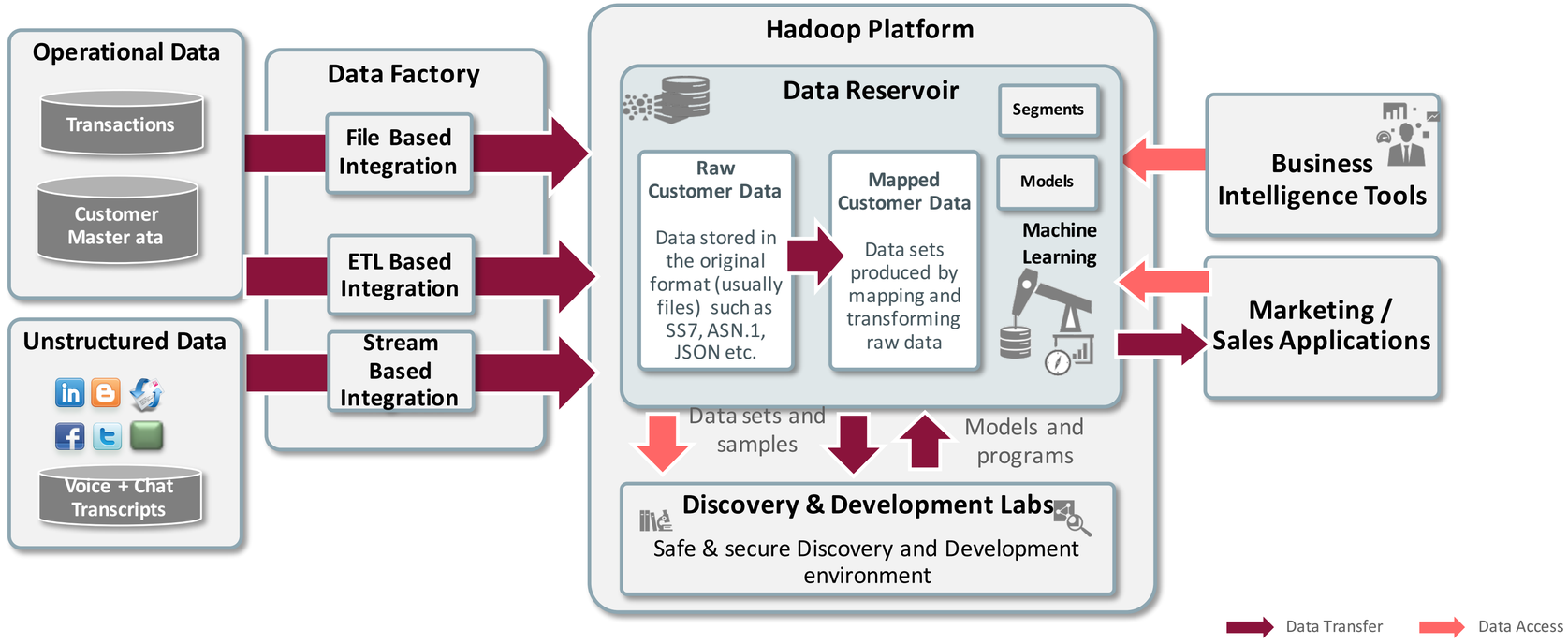
- We can now affordably create a single, massive archive of all our corporate data
- Separate to our OLTP and operational BI tools
- Leave it stored at the individual transaction level
- And then run multiple compute frameworks on it
  - SQL queries through Hive, Impala etc
  - Data processing using Spark, MR
  - Graph Analysis, etc





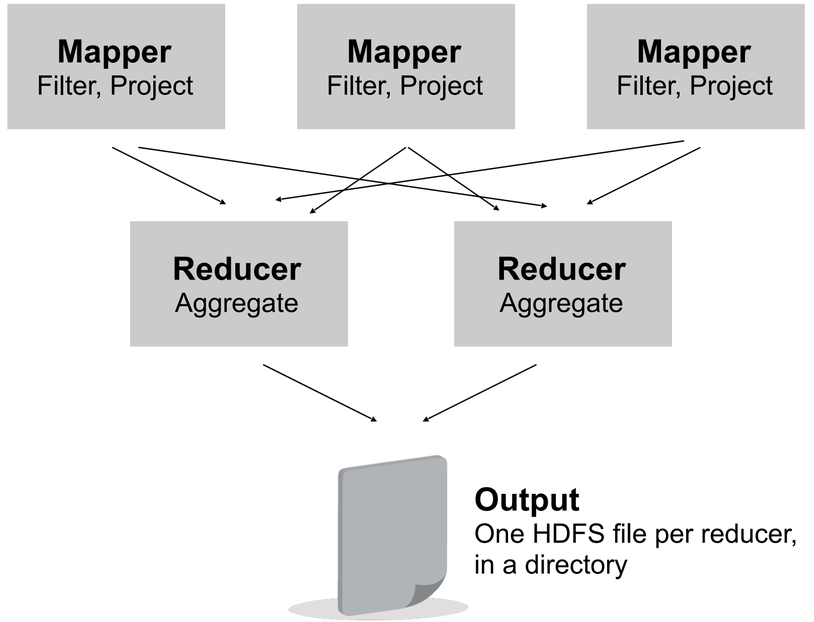
In the Context of BI & Analytics : The Data Reservoir

- Typical implementation of Hadoop and big data in an analytic context is the “data lake”
- Additional data storage platform with cheap storage, flexible schema support + compute
- Data lands in the data lake or reservoir in raw form, then minimally processed
- Data then accessed directly by “data scientists”, or processed further into DW



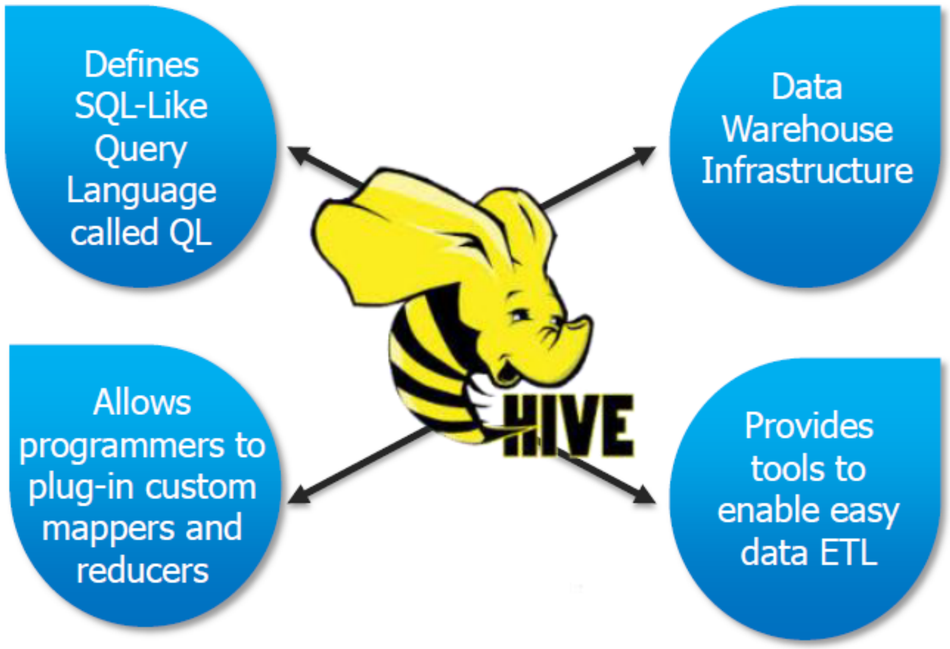
MapReduce - The Original Big Data Query Framework

- Programming model for processing large data sets in parallel on a cluster
- Not specific to a particular language, but usually written in Java
- Inspired by the **map** and **reduce** functions commonly used in functional programming
  - `Map()` performs filtering and sorting
  - `Reduce()` aggregates the output of mappers
  - and a `Shuffle()` step to redistribute output by keys
- Resolved several complications of distributed computing:
  - Allows unlimited computations on unlimited data
  - Map and reduce functions can be easily distributed
  - Originated at Google; Hadoop was Yahoo's open-source implementation of MapReduce, + two are synonymous



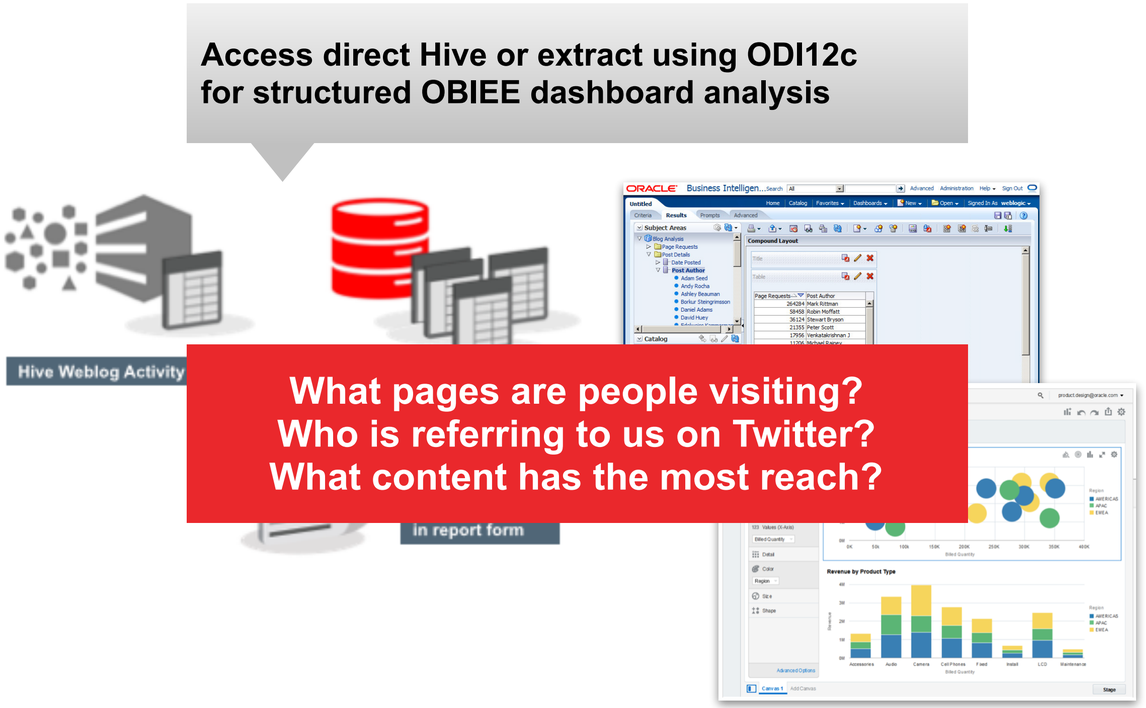
Apache Hive : SQL Metadata + Engine over Hadoop

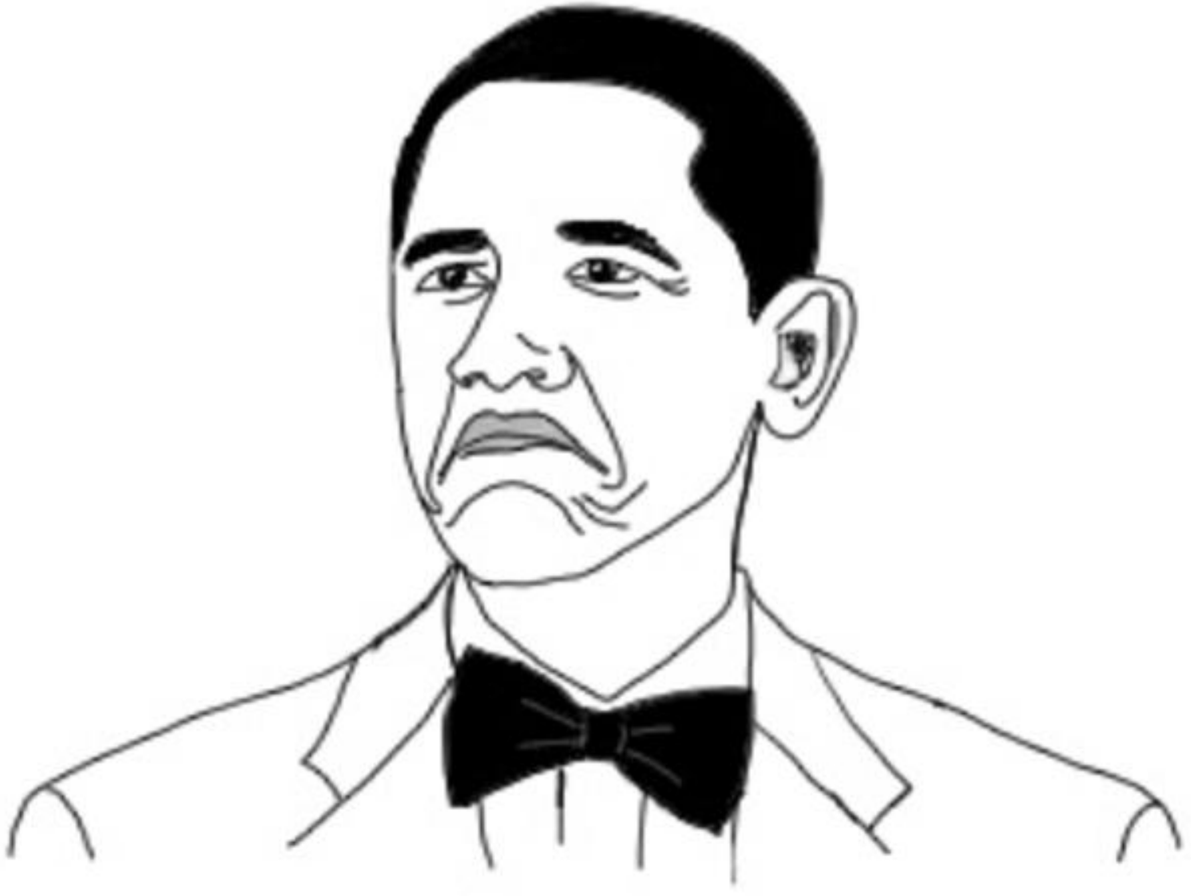
- Original developed at Facebook, now foundational within the Hadoop project
- Allows users to query Hadoop data using SQL-like language
- Tabular metadata layer that overlays files, can interpret semi-structured data (e.g. JSON)
- Generates MapReduce code to return required data
- Extensible through SerDes and Storage Handlers
- JDBC and ODBC drivers for most platforms/tools
- Perfect for set-based access + batch ETL work



Hive Provides a SQL Interface for BI + ETL Tools

- Data integration tools could now load and process Hadoop data
- BI tools could treat Hadoop as just another data source
- Generally use MapReduce and Hive to access data
  - ODBC and JDBC access to Hive tabular data
  - Allows Hadoop unstructured/semi-structured data on HDFS to be accessed like RDBMS





NOT BAD


Hive is slow

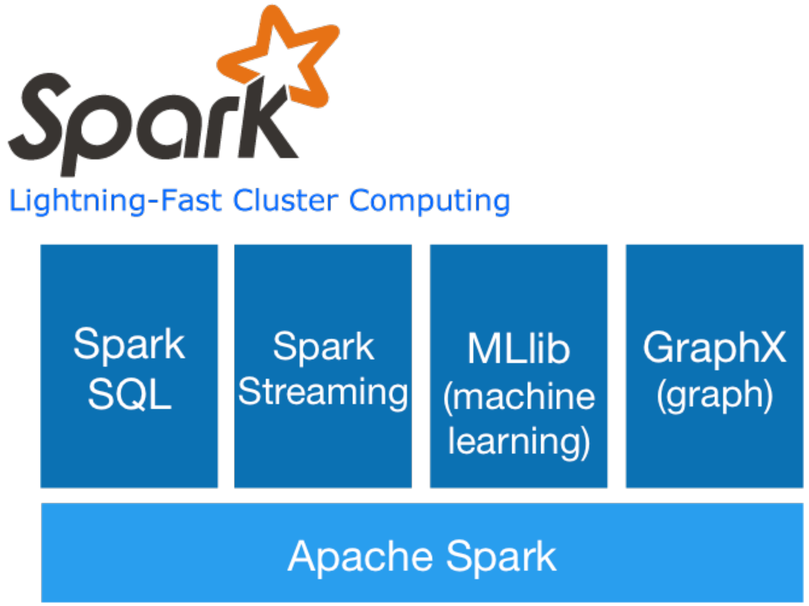
MapReduce is for losers


BEN WICKS' BOOK OF LOSERS
WINNERS AREN'T LOSERS
TRUMP
By Donald Trump


But the future is fast


MAKE AMERICA GREAT AGAIN!
TRUMP
DonaldJTrump.com
Apache Spark Utilizing in-memory Cache for M/R job
Spark
Spark
Iterative algorithms scan through the data each time
With Spark, data is cached in memory after first iteration
Quasi-Newton methods enhance in-memory benefits
Learn more about Advanced Analytics at http://www.alpinenow.com
gettyimages
Sean Rayford

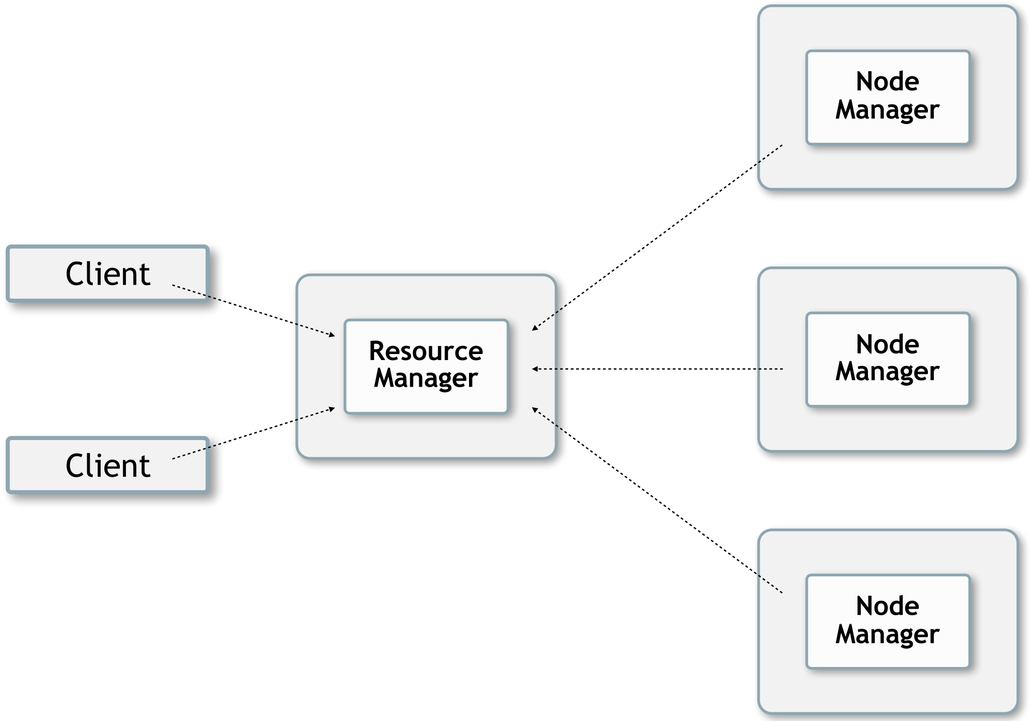
Hadoop 2.0 Processing Frameworks + Tools



MESOS

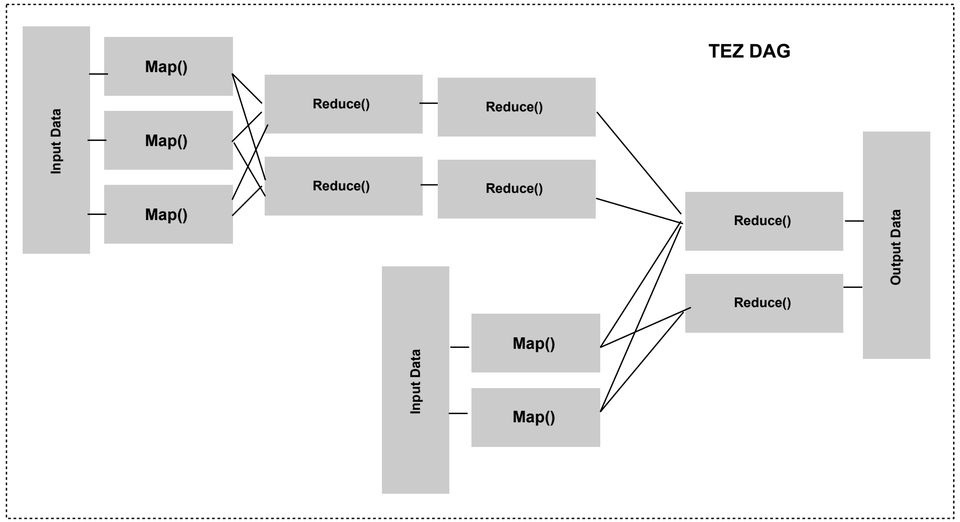
MapReduce 2 and YARN

- MapReduce 2 (MR2) splits the functionality of the JobTracker by separating resource management and job scheduling/monitoring
- Introduces YARN (Yet Another Resource Manager)
- Permits other processing frameworks to MR
  - For example, Apache Spark
- Maintains backwards compatibility with MR1
- Introduced with CDH5+

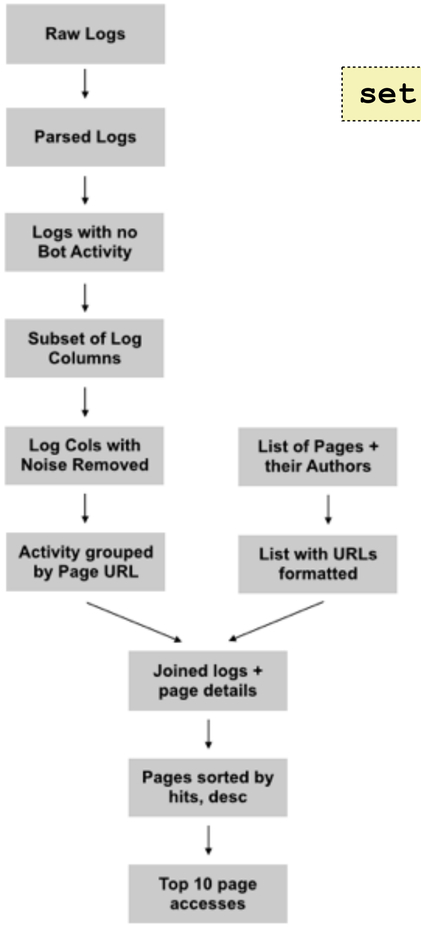


Apache Tez

- Runs on top of YARN, provides a faster execution engine than MapReduce for Hive, Pig etc
- Models processing as an entire data flow graph (DAG), rather than separate job steps
  - DAG (Directed Acyclic Graph) is a new programming style for distributed systems
  - Dataflow steps pass data between them as streams, rather than writing/reading from disk
- Supports in-memory computation, enables Hive on Tez (Stinger) and Pig on Tez
- Favoured In-memory / Hive v2
  route by Hortonworks



Tez Advantage - Drop-In Replacement for MR with Hive, Pig



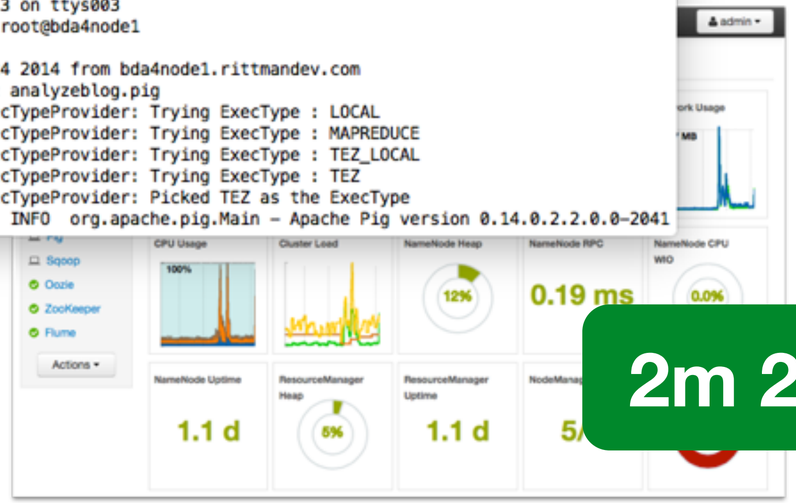

```
set hive.execution.engine=mr
```

```
JobId                    Maps Reduces Alias
job_1417127396023_0145   12   2       logs_base,logs_base_nobots,logs_base_page,logs_
job_1417127396023_0146   2    1       pages_and_post_details,pages_and_posts_trim,pos
job_1417127396023_0147   1    1       pages_and_posts_sorted
job_1417127396023_0148   1    1       pages_and_posts_sorted
job_1417127396023_0149   1    1       pages_and_posts_sorted
```

4m 17s

```
set hive.execution.engine=tez
```

```
markrittman — root@bda4node1:~ — ssh — 97×11
Last login: Fri Dec  5 13:12:13 on ttys003
officeimac:~ markrittman$ ssh root@bda4node1
root@bda4node1's password:
Last login: Fri Dec  5 13:29:24 2014 from bda4node1.rittmandev.com
[root@bda4node1 ~]# pig -x tez analyzeblog.pig
14/12/06 17:16:37 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
14/12/06 17:16:37 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
14/12/06 17:16:37 INFO pig.ExecTypeProvider: Trying ExecType : TEZ_LOCAL
14/12/06 17:16:37 INFO pig.ExecTypeProvider: Trying ExecType : TEZ
14/12/06 17:16:37 INFO pig.ExecTypeProvider: Picked TEZ as the ExecType
2014-12-06 17:16:37,571 [main] INFO  org.apache.pig.Main - Apache Pig version 0.14.0.2.2.0.0-2041
```



2m 25s

Cloudera Impala - Fast, MPP-style Access to Hadoop Data

- Cloudera's answer to Hive query response time issues
- MPP SQL query engine running on Hadoop, bypasses MapReduce for direct data access
  - Mostly in-memory, but spills to disk if required
- Uses Hive metastore to access Hive table metadata
- Similar SQL dialect to Hive - not as rich though and no support for Hive SerDes, storage handlers etc

Enabling Hive Tables for Impala

- Log into Impala Shell, run INVALIDATE METADATA command to refresh Impala table list
- Run SHOW TABLES Impala SQL command to view tables available
- Run COUNT(*) on main ACCESS_PER_POST table to see typical response time

```
[oracle@bigdatalite ~]$ impala-shell
Starting Impala Shell without Kerberos authentication

[bigdatalite.localdomain:21000] > invalidate metadata;
Query: invalidate metadata

Fetched 0 row(s) in 2.18s
[bigdatalite.localdomain:21000] > show tables;
Query: show tables
+-----------------------------------+
| name                              |
+-----------------------------------+
| access_per_post                   |
| access_per_post_cat_author        |
| …                                 |
| posts                             |
|———————————————————————————————--+
Fetched 45 row(s) in 0.15s
```

```
[bigdatalite.localdomain:21000] > select count(*)
                                  from access_per_post;
Query: select count(*) from access_per_post
+----------+
| count(*) |
+----------+
| 343      |
+----------+
Fetched 1 row(s) in 2.76s
```

Apache Parquet - Column-Orientated Storage for Analytics

- Beginners usually store data in HDFS using text file formats (CSV) but these have limitations
- Apache AVRO often used for general-purpose processing
  - Splitability, schema evolution, in-built metadata, support for block compression
- Parquet now commonly used with Impala due to column-orientated storage
  - Mirrors work in RDBMS world around column-store
  - Only return (project) the columns you require across a wide table

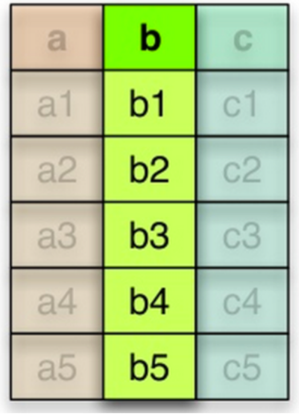
APACHE PARQUET

Columnar storage for the people

Collocating entire columns allows for efficient column projection

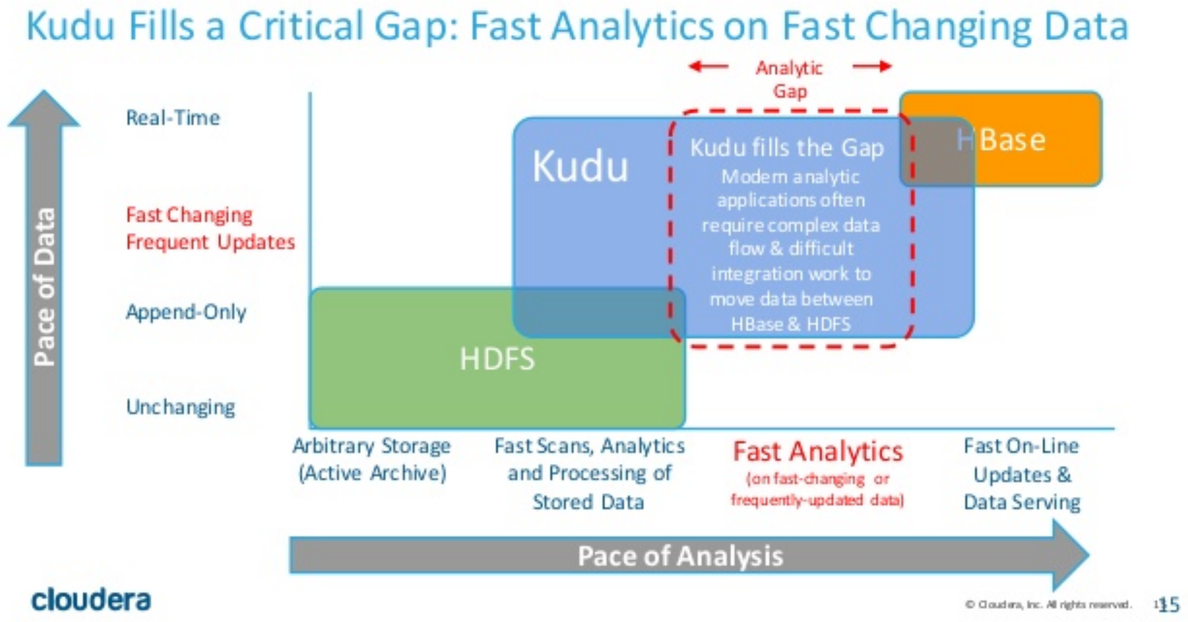
Read off disk only the columns you need

Possibly more importantly: deserialize only the columns you need



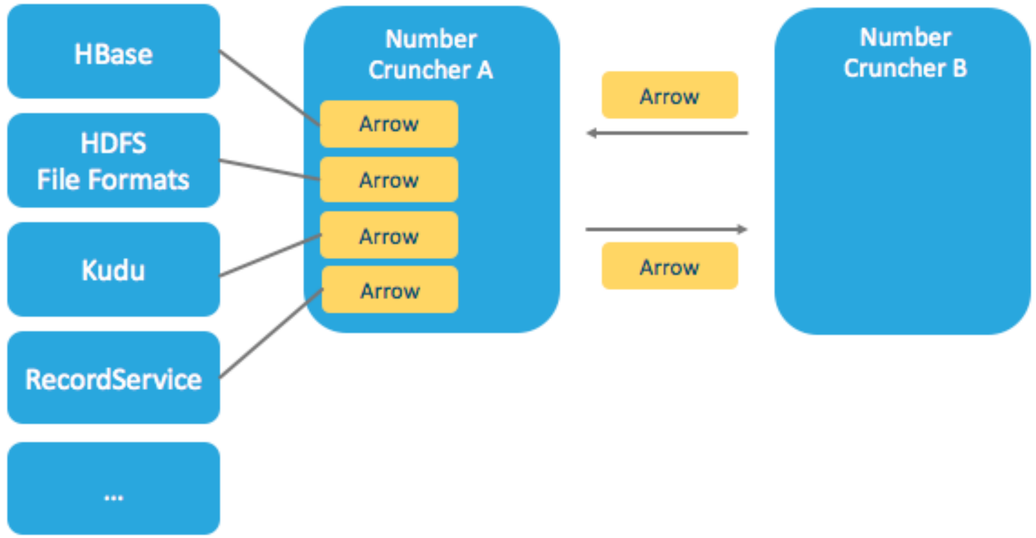
Cloudera Kudu - Combining Best of HBase and Column-Store

- But Parquet (and HDFS) have significant limitation for real-time analytics applications
  - Append-only orientation, focus on column-store makes streaming ingestion harder
- Kudu aims to combine best of HDFS + HBase
  - Real-time analytics-optimised
  - Supports updates to data
  - Fast ingestion of data
  - Accessed using SQL-style tables and get/put/update/delete API



Apache Arrow - Standardising In-Memory Structures

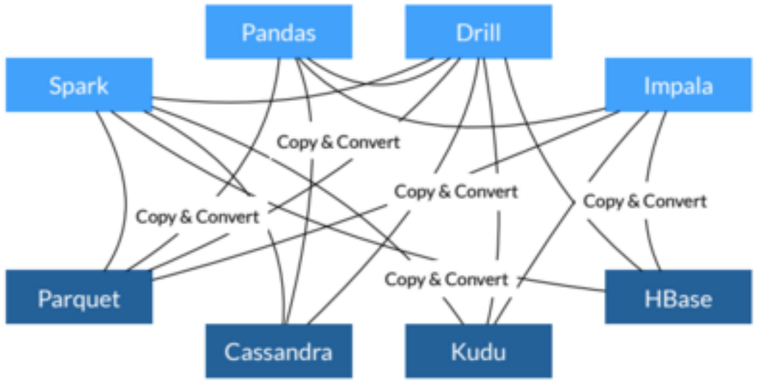
- Many Hadoop tools now use in-memory processing (R, Python, Spark etc)
- But they all work to their own standards
- Considerable overhead in serialising / deserialising data between tools
- Apache Arrow standardised how in-memory data is held
- Considerably reduced overhead and latency between tools



Apache Arrow : Common In-Memory Layer for Hadoop

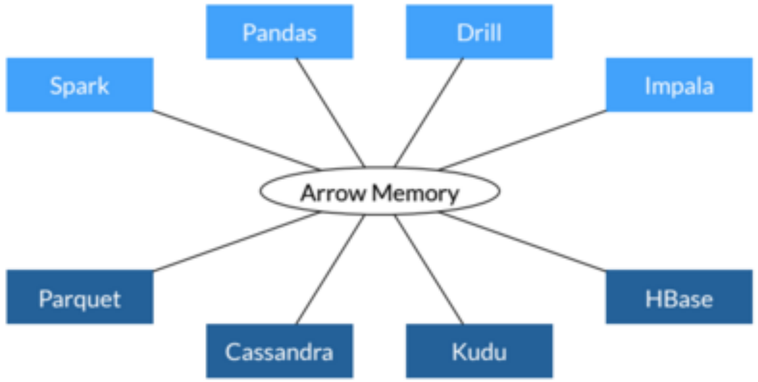
Advantages of a Common Data Layer

Today



- Each system has its own internal memory format
- 70-80% CPU wasted on serialization and deserialization
- Similar functionality implemented in multiple projects

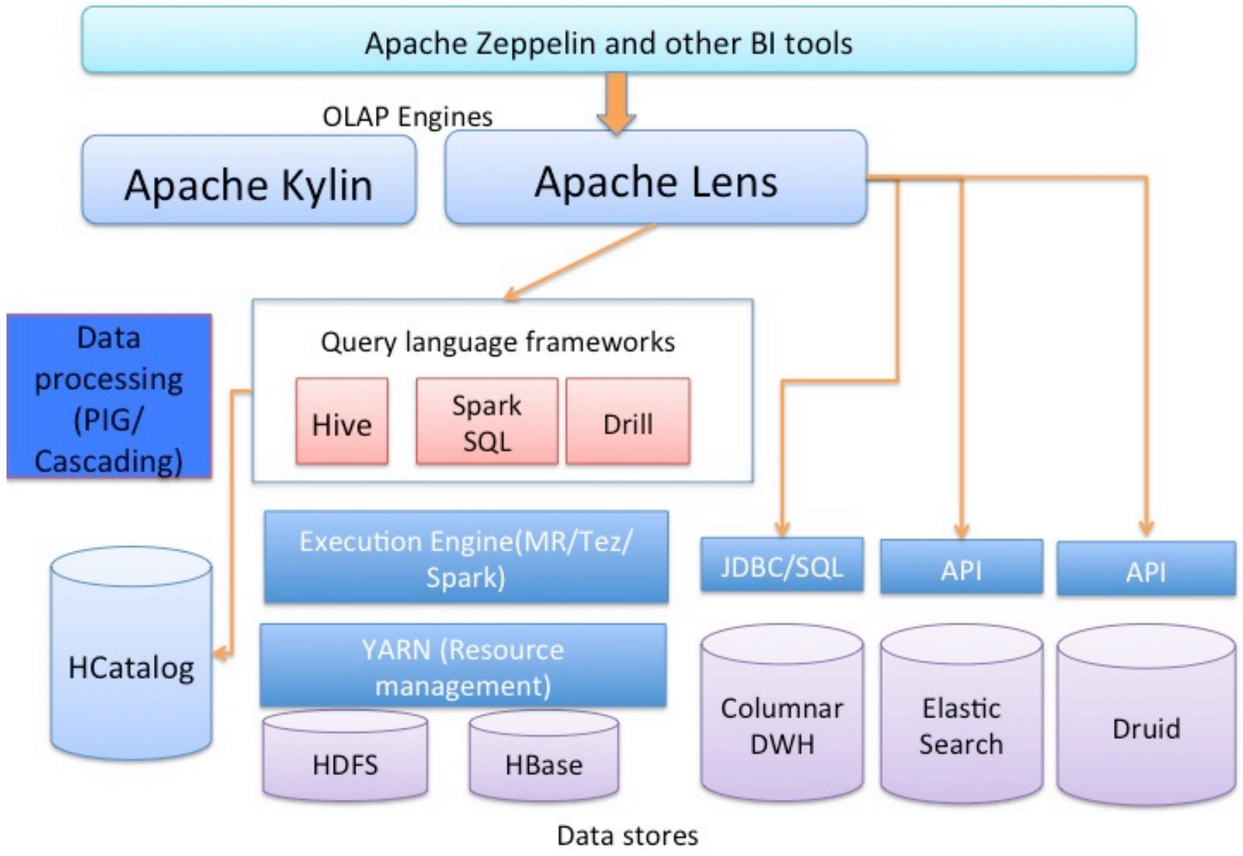
With Arrow

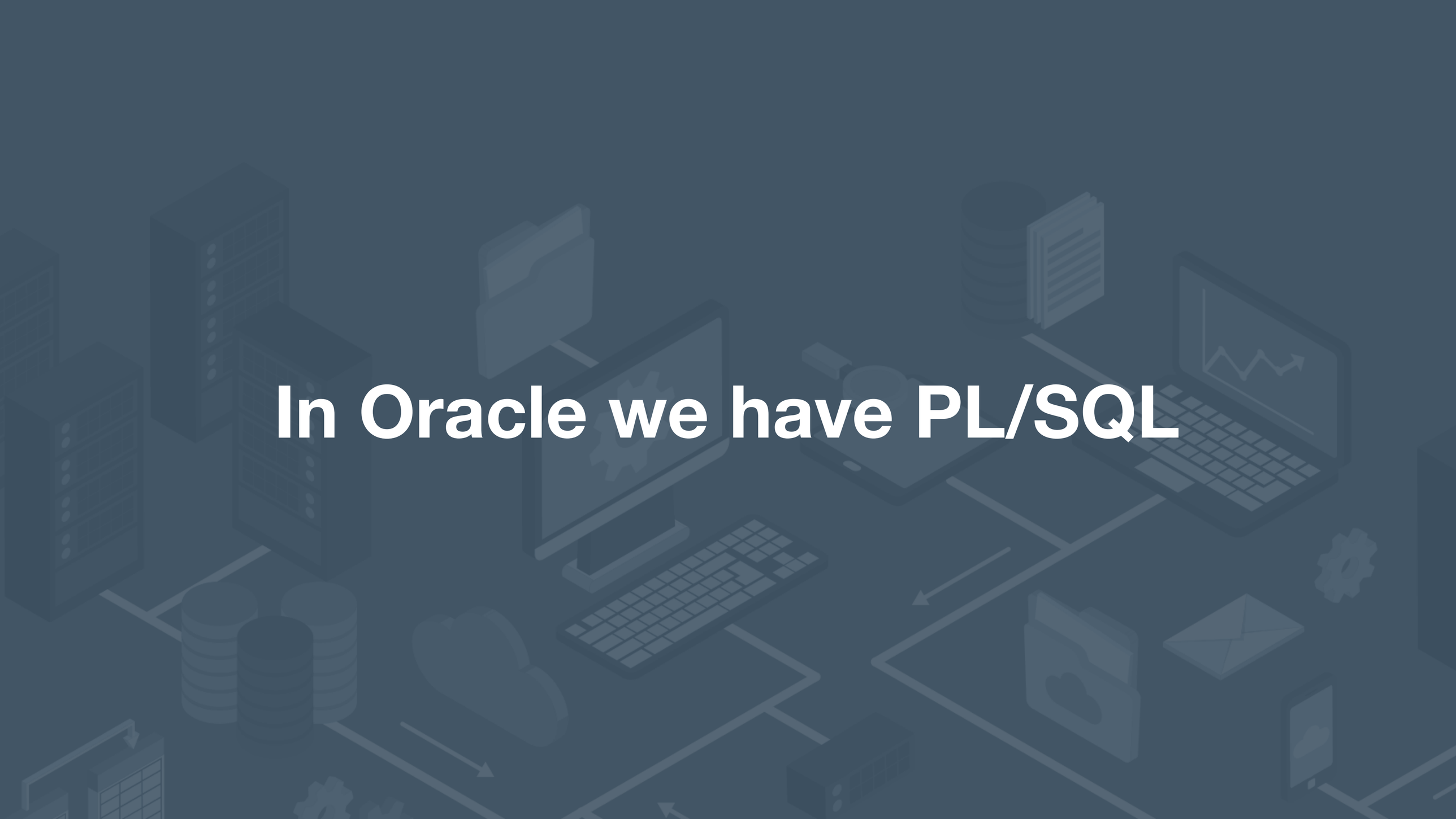


- All systems utilize the same memory format
- No overhead for cross-system communication
- Projects can share functionality (eg, Parquet-to-Arrow reader)

Apache Lens : Logical Dimensional Model for Hadoop

- Similar concept to Oracle BI Server
  - Logical star scheme for business model
  - Maps to federated data sources
  - Integrates and models data for query
  - Can run alongside Kylin MOLAP server
- Part of wider next-gen Hadoop BI stack
  - Apache Zepplin web-based notebook
  - Spark/MR/Tez execution engines
  - Hive and Drill for ETL/ad-hoc SQL
- Slowly the BI stack gets built out...




In Oracle we have PL/SQL

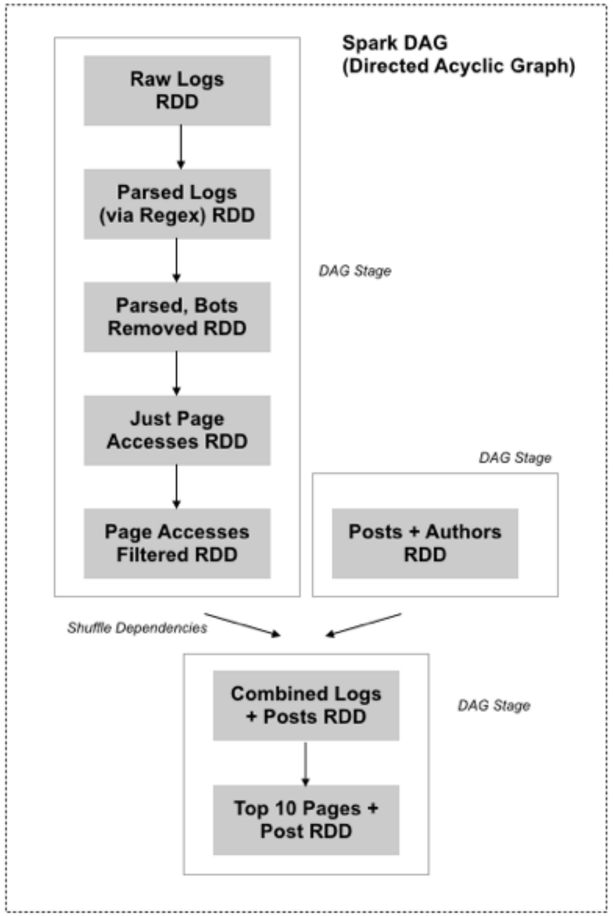

In SQL Server we have TSQL


In Hadoop we have ... Spark

Lightning-Fast Cluster Computing

Apache Spark

- Another DAG execution engine running on YARN
- More mature than TEZ, with richer API and more vendor support
- Uses concept of an RDD (Resilient Distributed Dataset)
  - RDDs like tables or Pig relations, but can be cached in-memory
  - Great for in-memory transformations, or iterative/cyclic processes
- Spark jobs comprise of a DAG of tasks operating on RDDs
- Access through Scala, Python or Java APIs
- Related projects include
  - Spark SQL
  - Spark Streaming



Rich Developer Support + Wide Developer Ecosystem

- Native support for multiple languages with identical APIs
  - Python - prototyping, data wrangling
  - Scala - functional programming features
  - Java - lower-level, application integration
- Use of closures, iterations, and other common language constructs to minimize code
- Integrated support for distributed + functional programming
- Unified API for batch and streaming

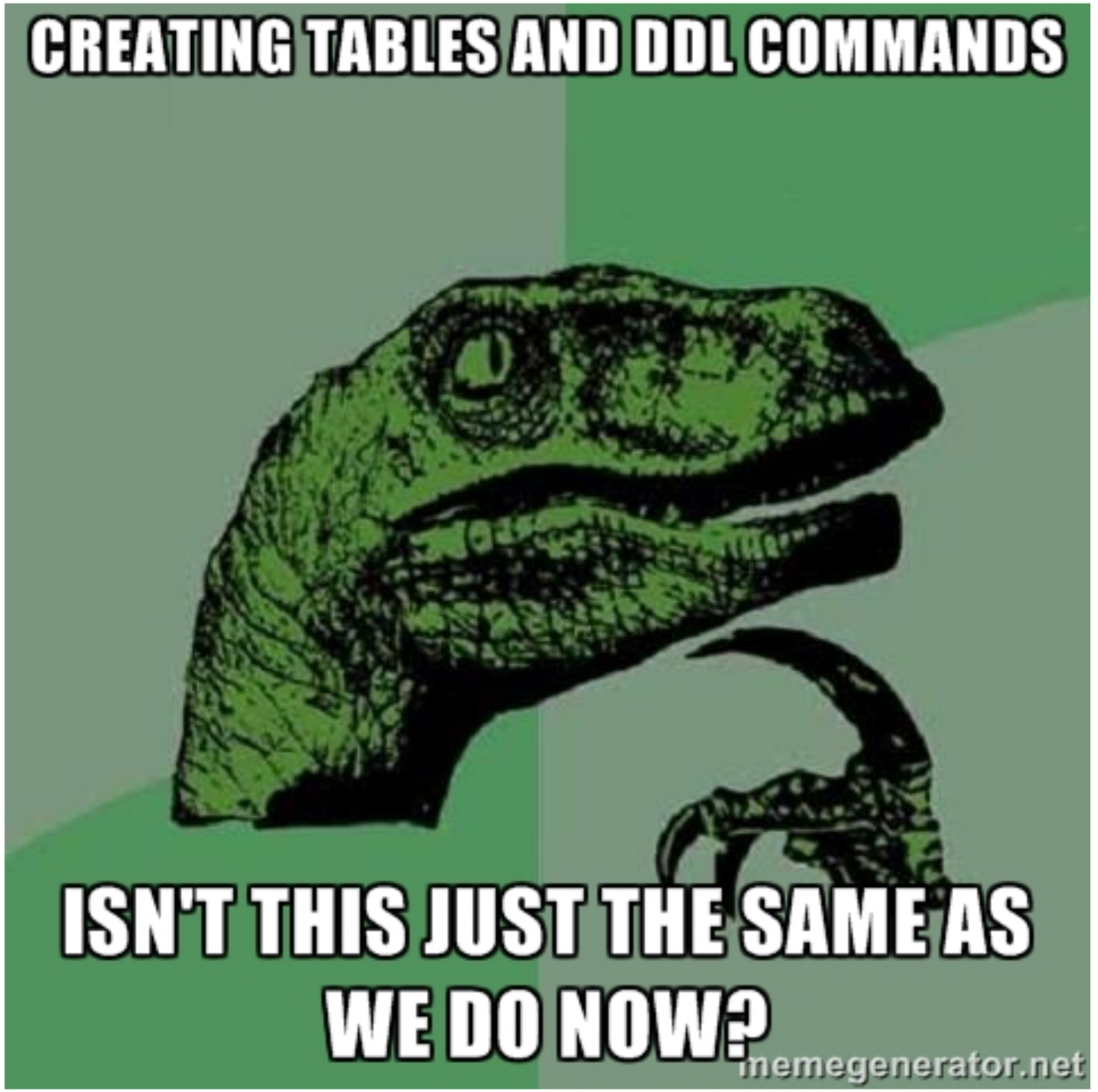
```
scala> val logfile = sc.textFile("logs/access_log")
14/05/12 21:18:59 INFO MemoryStore: ensureFreeSpace(77353)
called with curMem=234759, maxMem=309225062
14/05/12 21:18:59 INFO MemoryStore: Block broadcast_2
stored as values to memory (estimated size 75.5 KB, free 294.6 MB)
logfile: org.apache.spark.rdd.RDD[String] =
MappedRDD[31] at textFile at <console>:15

scala> logfile.count()
14/05/12 21:19:06 INFO FileInputFormat: Total input paths to process : 1
14/05/12 21:19:06 INFO SparkContext: Starting job: count at <console>:1
...
14/05/12 21:19:06 INFO SparkContext: Job finished:
count at <console>:18, took 0.192536694 s
res7: Long = 154563
```

```
scala> val logfile = sc.textFile("logs/access_log").cache
scala> val biapps11g = logfile.filter(line => line.contains("/biapps11g/"))
biapps11g: org.apache.spark.rdd.RDD[String] = FilteredRDD[34] at filter at <console>:17
scala> biapps11g.count()
...
14/05/12 21:28:28 INFO SparkContext: Job finished: count at <console>:20, took 0.387960876 s
res9: Long = 403
```

Spark SQL - SQL within Apache Spark

- Spark SQL, and Data Frames, allow RDDs in Spark to be processed using SQL queries
- Bring in and federate additional data from JDBC sources
- Load, read and save data in Hive, Parquet and other structured tabular formats

```
val accessLogsFilteredDF = accessLogs
                                    .filter( r => ! r.agent.matches(".*(spider|robot|bot|slurp).*"))
                                    .filter( r => ! r.endpoint.matches(".*(wp-content|wp-admin).*")).toDF()
                                    .registerTempTable("accessLogsFiltered")

        val topTenPostsLast24Hour = sqlContext.sql("SELECT p.POST_TITLE, p.POST_AUTHOR, COUNT(*)
                                                    as total
                                                    FROM accessLogsFiltered a
                                                    JOIN posts p ON a.endpoint = p.POST_SLUG
                                                    GROUP BY p.POST_TITLE, p.POST_AUTHOR
                                                    ORDER BY total DESC LIMIT 10 ")

        // Persist top ten table for this window to HDFS as parquet file

        topTenPostsLast24Hour.save("/user/oracle/rm_logs_batch_output/topTenPostsLast24Hour.parquet"
                                , "parquet", SaveMode.Overwrite)
```

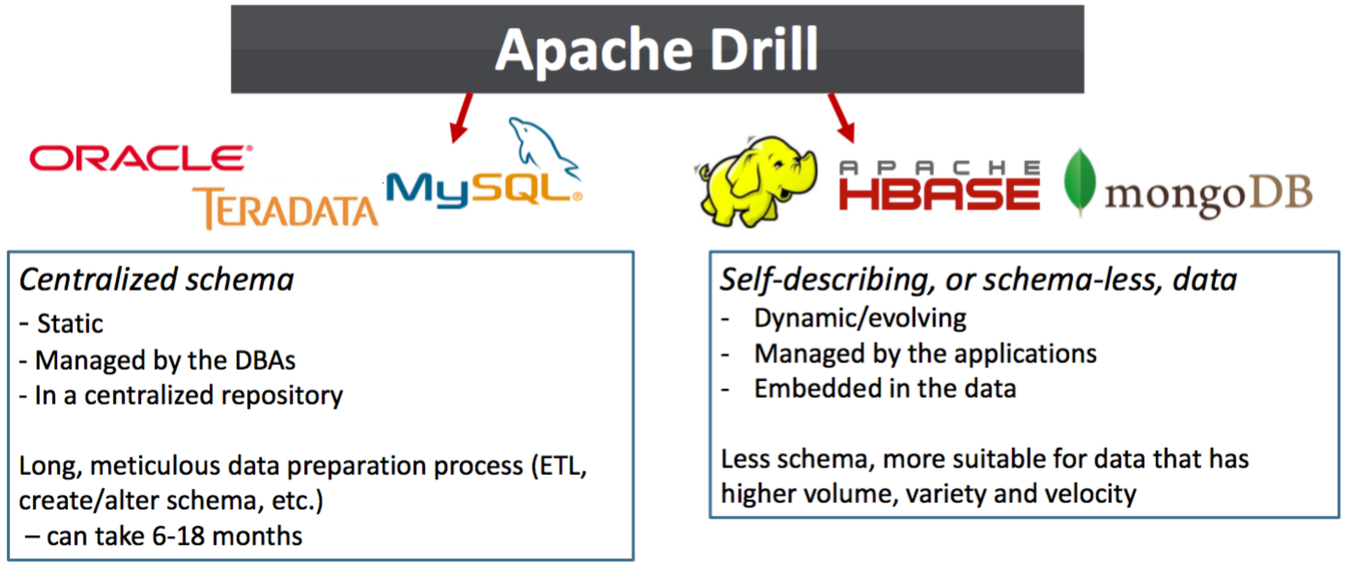

CREATING TABLES AND DDL COMMANDS
ISN'T THIS JUST THE SAME AS WE DO NOW?
memegenerator.net

Apache Drill - Leverage Embedded Metadata in Files

- Apache Drill is another SQL-on-Hadoop project that focus on schema-free data discovery
- Inspired by Google Dremel, innovation is querying raw data with schema optional
  - Automatically infers and detects schema from semi-structured datasets and NoSQL DBs
  - Join across different silos of data e.g. JSON records, Hive tables and HBase database
  - Aimed at different use-cases than Hive -
    low-latency queries, discovery
    (think Endeca vs OBIEE)

Self-Describing Data - Parquet, AVRO, JSON etc

- Most modern datasource formats embed their schema in the data ("schema-on-read")
- Apache Drill makes these as easy to join to traditional datasets as "point me at the data"
  - Cuts out unnecessary work in defining Hive schemas for data that's self-describing
  - Supports joining across files, databases, NoSQL etc

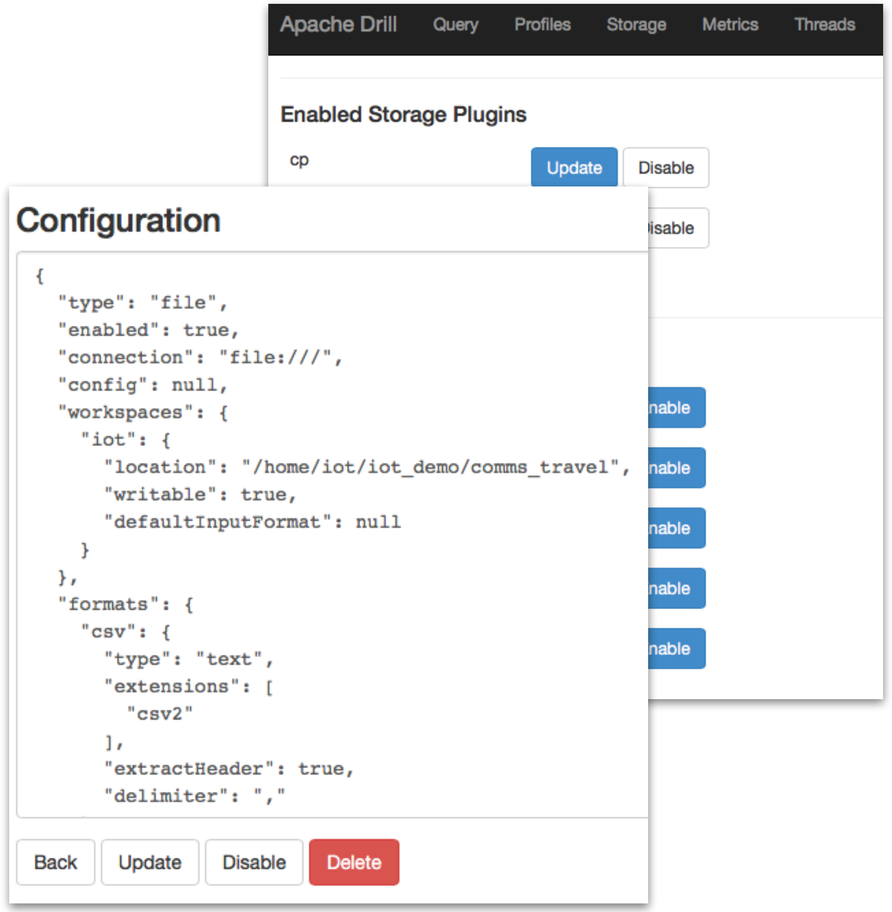


Apache Drill and Text Files

- Files can exist either on the local filesystem, or on HDFS
- Connection to directory or file defined in storage configuration
  - Can work with CSV, TXT, TSV etc
  - First row of file can provide schema (column names)

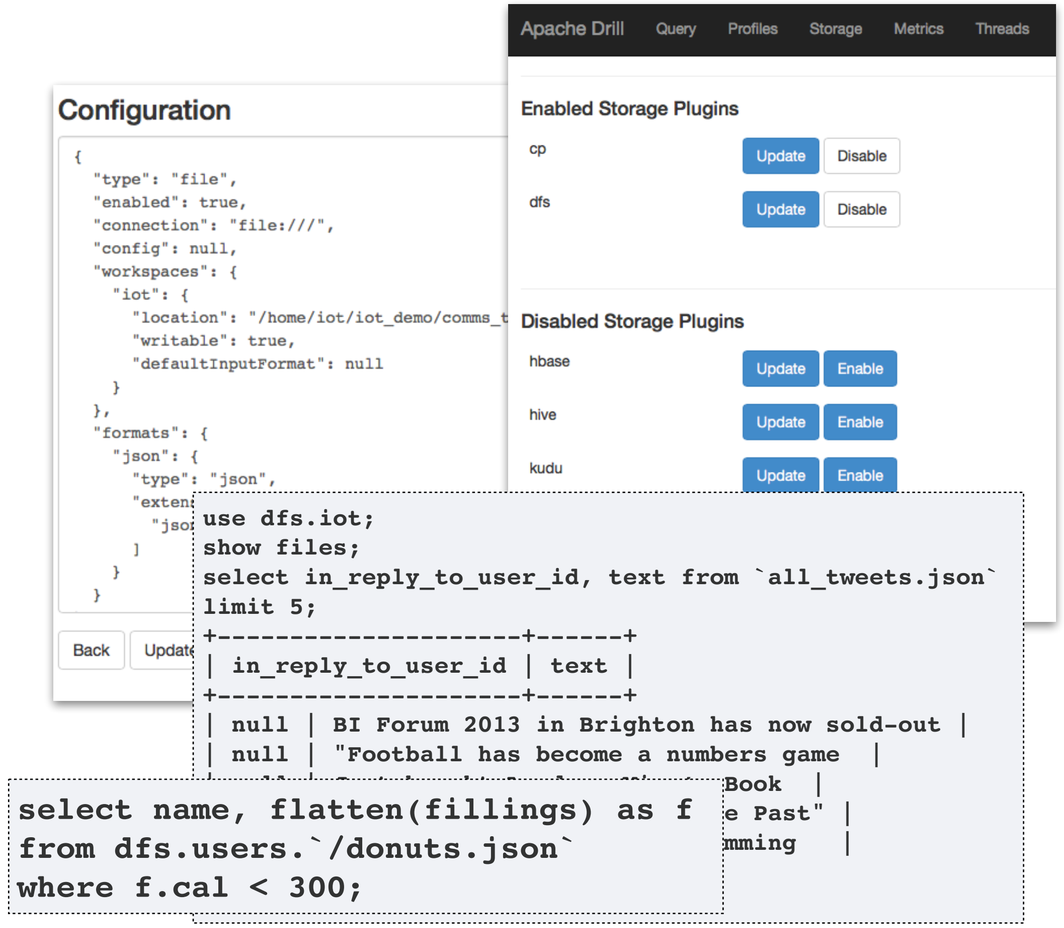
```
SELECT * FROM dfs.`/tmp/csv_with_header.csv2`;
+-------+------+------+------+
| name  | num1 | num2 | num3 |
+-------+------+------+------+
| hello |  1   |
| hello |  1   |
| hello |  1   |
| hello |  1   |
| hello |  1   |
| hello |  1   |
| hello |  1   |
+-------+------+------+------+
7 rows selected (0
```

```
SELECT * FROM dfs.`/tmp/csv_no_header.csv`;
+-----------------------+
|        columns        |
+-----------------------+
| ["hello","1","2","3"] |
| ["hello","1","2","3"] |
| ["hello","1","2","3"] |
| ["hello","1","2","3"] |
| ["hello","1","2","3"] |
| ["hello","1","2","3"] |
| ["hello","1","2","3"] |
+-----------------------+
7 rows selected (0.112 seconds)
```



Apache Drill and JSON Documents

- JSON (Javascript Object Notation) documents are often used for data interchange
  - Exports from Twitter and other consumer services
  - Web service responses and other B2B interfaces
- A more lightweight form of XML that is “self-describing”
- Handles evolving schemas, and optional attributes
- Drill treats each document as a row, and has features to
  - Flatten nested data (extract elements from arrays)
  - Generate key/value pairs for loosely structured data



Apache Drill and Hive, HBase, Parquet Sources etc

- Drill can connect to Hive to make use of metastore (incl. multiple Hive metastores)
- NoSQL databases (HBase etc)
- Parquet files (native storage format - columnar + self describing)

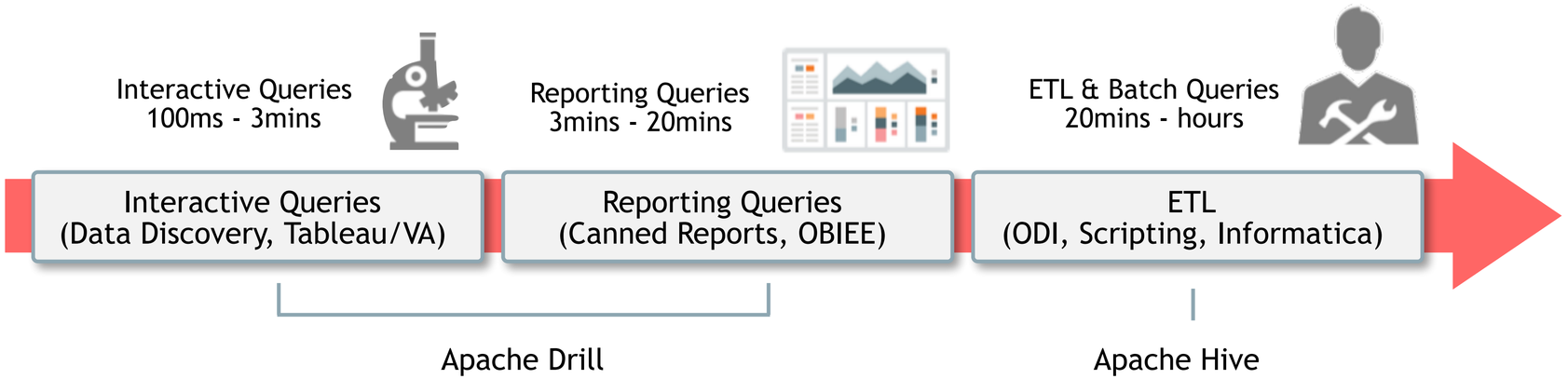
```
SELECT firstname,lastname FROM
hiveremote.`customers` limit 10;`

+------------+------------+
| firstname  |  lastname  |
+------------+------------+
| Essie      | Vaill      |
| Cruz       | Roudabush  |
| Billie     | Tinnes     |
| Zackary    | Mockus     |
| Rosemarie  | Fifield    |
```

```
USE hbase;
SELECT * FROM students;

+-------------+----------------------+--------
|  row_key    |  account             |
+-------------+----------------------+--------
| [B@e6d9eb7  | {"name":"QWxpY2U="}  | {"state
| [B@2823a2b4 | {"name":"Qm9i"}      | {"state
| [B@3b8eec02 | {"name":"RnJhbms="}  | {"state
| [B@242895da | {"name":"TWFyeQ=="}  | {"state
+-------------+----------------------+--------
```

```
SELECT * FROM dfs.`iot_demo/geodata/region.parquet`;

+--------------+--------------+-----------------------+
| R_REGIONKEY  |    R_NAME    |       R_COMMENT       |
+--------------+--------------+-----------------------+
| 0            | AFRICA       | lar deposits. blithe  |
| 1            | AMERICA      | hs use ironic, even   |
| 2            | ASIA         | ges. thinly even pin  |
| 3            | EUROPE       | ly final courts cajo  |
| 4            | MIDDLE EAST  | uickly special accou  |
+--------------+--------------+-----------------------+
```

Apache Drill vs. Apache Hive

- Drill developed for **real-time**, **ad-hoc data exploration** with **schema discovery on-the-fly**
  - Individual analysts exploring new datasets, leveraging corporate metadata/data to help
  - Hive is more about large-scale, centrally curated set-based big data access
- **Drill models conceptually as JSON**, vs. Hive's tabular approach
- **Drill introspects schema** from whatever it connects to, vs. formal modeling in Hive




www.rittmanmead.com

SQL-ON-HADOOP FOR ANALYTICS + BI: WHAT ARE MY OPTIONS, WHAT'S THE FUTURE?

MARK RITTMAN, COO, RITTMAN MEAD

BUDAPEST DATA FORUM, JUNE 2016