A SQL Query Engine

Flexible, Performant, Open

Keys Botzum

Senior Principal Technologist

MapR Technologies

MAPR

Apache Drill: Self-Service Data Exploration and Nested-Data Analytics on Hadoop

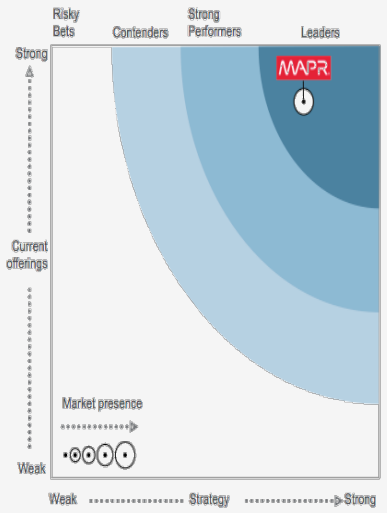
Keys Botzum
Senior Principal Technologist

October 2014

MapR: Best Product, Best Business, Best Customers, Best Partners, Best Investors

Top Ranked

FORRESTER®



Exponential Growth

3X bookings Q1 '13 – Q1 '14

90% software licenses

80% of accounts expand 3X

<2% lifetime churn

>$1B in incremental revenue generated by 1 customer

500+ Customers

LIVE NATION ENTERTAINMENT

hp

beatsMUSIC

SAMSUNG

AADHAAR

CISCO

rubicon PROJECT

comScore

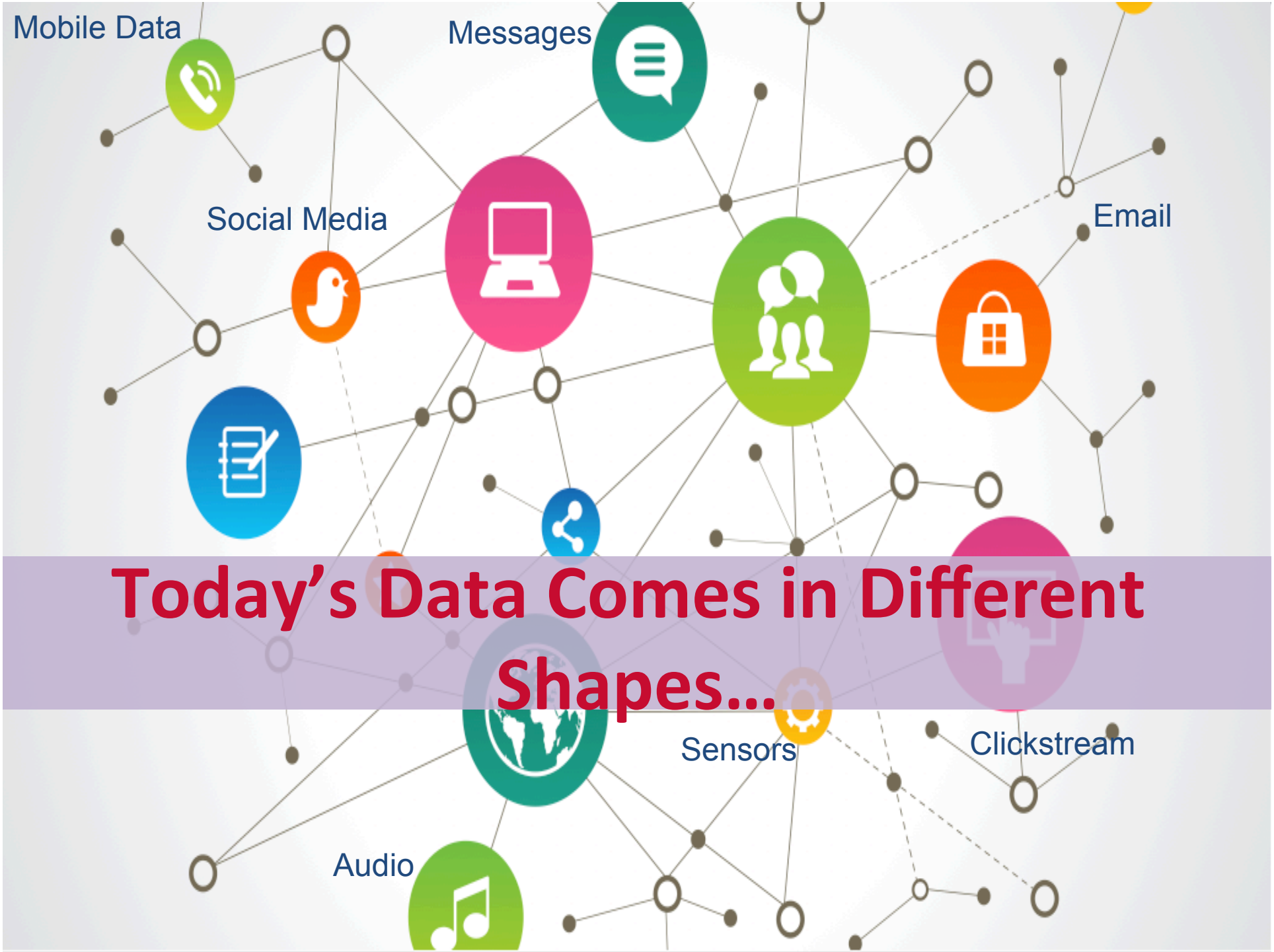
Cloud Leaders

Google

amazon web services™

Agenda

- Problem Statement
- Architecture
- Features
- Under the Hood
- Status and progress


Mobile Data
Messages
Social Media
Email
Today's Data Comes in Different Shapes…
Sensors
Clickstream
Audio

Data landscape is changing

New types of applications

- Social, mobile, Web, “Internet of Things”, Cloud…
- Iterative/Agile in nature
- More users, more data

New data models & data types

- Flexible (schema-less) data
- Rapidly changing
- Semi-structured/Nested data

JSON

```
{
  "data": [
    "id": "X999_Y999",
    "from": {
      "name": "Tom Brady", "id": "X12"
    },
    "message": "Looking forward to 2014!",
    "actions": [
      {
        "name": "Comment",
        "link": "http://www.facebook.com/X99/posts Y999"
      },
      {
        "name": "Like",
        "link": "http://www.facebook.com/X99/posts Y999"
      }
    ],
    "type": "status",
    "created_time": "2013-08-02T21:27:44+0000",
    "updated_time": "2013-08-02T21:27:44+0000"
  }
 }
```

Why Hadoop

But….SQL is key for the enterprise …

QlikView

Hadoop evolving as central hub for analysis

Provides Cost effective, flexible way to store and and process data at scale

Traditional datasets

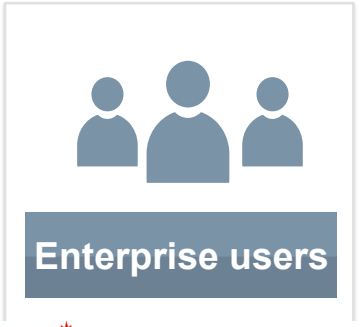
- Comes from transactional applications
- Stored for historical purposes and/or for large scale ETL/Analytics
- Well defined schemas
- Managed centrally by DBAs
- No frequent changes to schema
- Flat datasets

New datasets

- Comes from new applications (Ex: Social feeds, clickstream, logs, sensor data)
- Enable new use cases such as Customer Satisfaction, Product/Service optimization
- Flexible data models/managed within applications
- Schemas evolving rapidly
- Semi-structured/Nested data

SQL and Hadoop

- Opens up Hadoop data to broader audience
  - Existing SQL skill sets
  - Broad eco system of tools
- New and improved BI/Analytics use cases
  - Analysis on more raw data, new types of data and real time data
- Cost savings

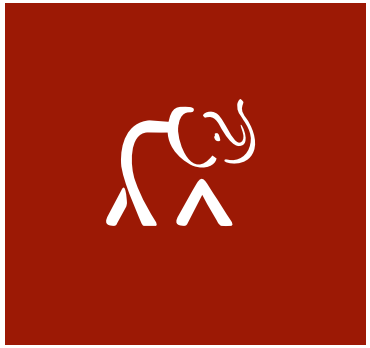


MicroStrategy
Best In Business Intelligence™

tableau SOFTWARE

SAP

platfora



Slap a SQL interface over MapReduce and we're done

Slap a SQL interface over MapReduce and we're done

Hive did that

Slap a SQL interface over MapReduce and we're done

Hive did that

…. Not quite that simple

Interactive at Scale

- Query huge amounts – TBs to PBs!
- Interactive response time
  - A human is waiting

Data is Not Simple Structured

- Structured and unstructured (schema less)
- Nested data
  - Customer → Address → Phone #'s
- Many formats – text, JSON, Parquet, XML, etc.

Variety of Data Sources

- Ideally execute queries against many data sources at the same time!
  - Big Data
    - HBase, Cassandra, HDFS, etc.
  - “Small” Data
    - Oracle, DB2, MongoDB, files, etc..

Why Apache Drill?

- Many partial solutions to previous requirements, but nothing addresses them all
- Other solutions insufficient because
  - Not interactive
  - Nested data not supported
  - Require schema
  - Not full ANSI SQL
  - Memory limited

Architecture

MAPR

Agenda

- Problem Statement
- **Architecture**
- Features
- Status and progress

Drill Inspired by Google's Dremel

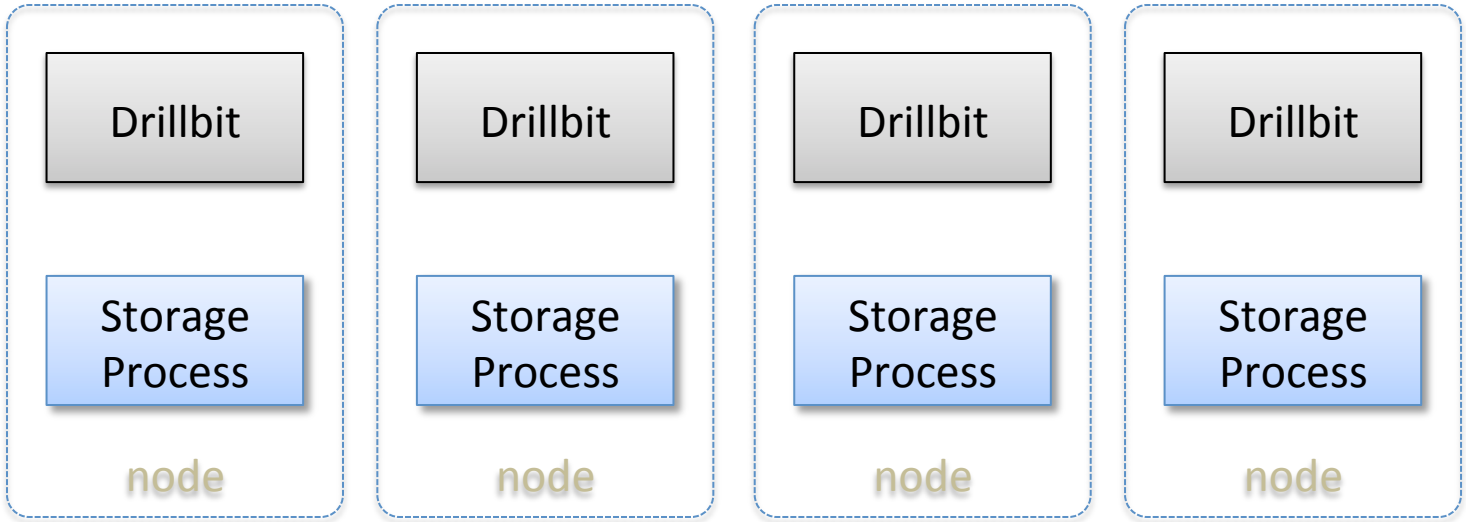
> "Dremel is a scalable, interactive ad-hoc query system for analysis of read-only nested data. By combining multi-level execution trees and columnar data layout, it is capable of running aggregation queries over trillion-row tables in seconds. The system scales to thousands of CPUs and petabytes of data, and has thousands of users at Google."

… http://research.google.com/pubs/pub36632.html

Sergey Melnik, Andrey Gubarev, Jing Jing Long, Geoffrey Romer, Shiva Shivakumar, Matt Tolton, Theo Vassilakis, Proc. of the 36th Int'l Conf on Very Large Data Bases (2010), pp. 330-339

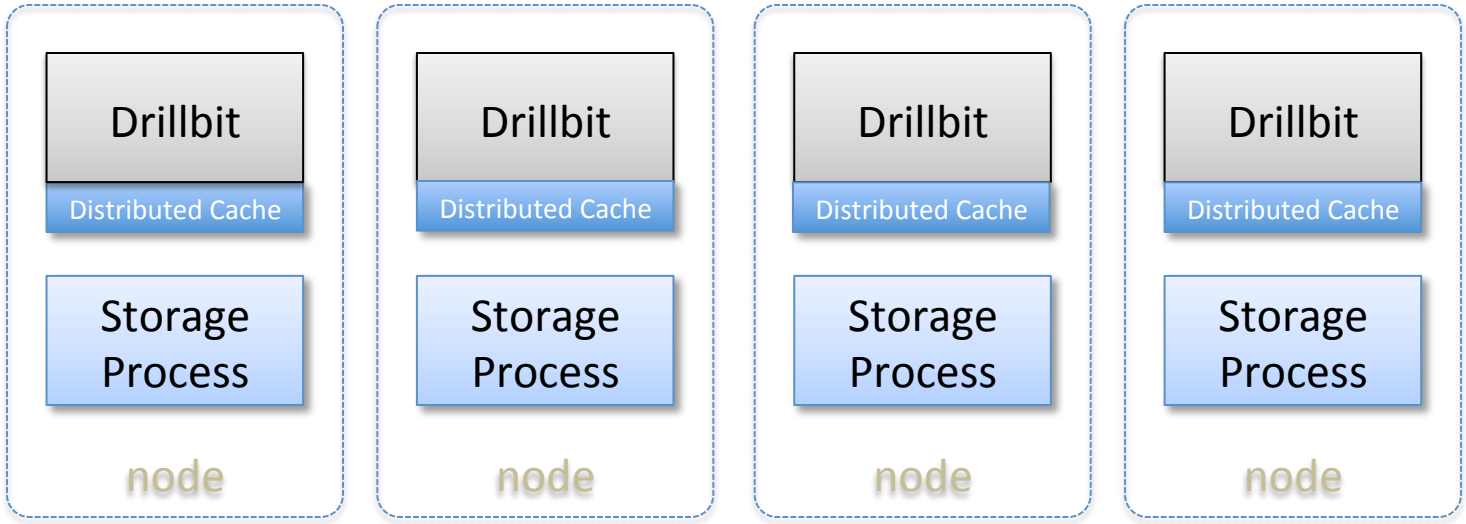
Wire-level Architecture

- Each node: **Drillbit** - maximize data locality
- Co-ordination, query planning, execution, etc, are **distributed**
- Any node can act as endpoint for a query—**foreman**



Wire-level Architecture

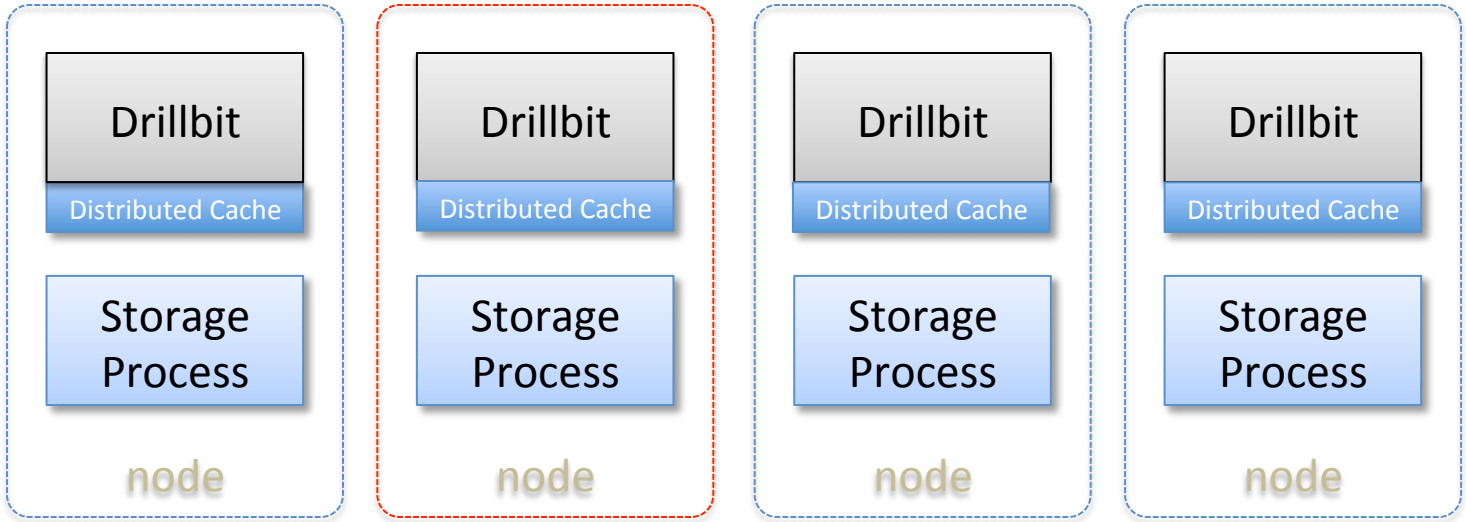
- **Zookeeper** for ephemeral cluster membership info
- **Distributed cache** for metadata, locality information, etc.



Wire-level Architecture

Performant

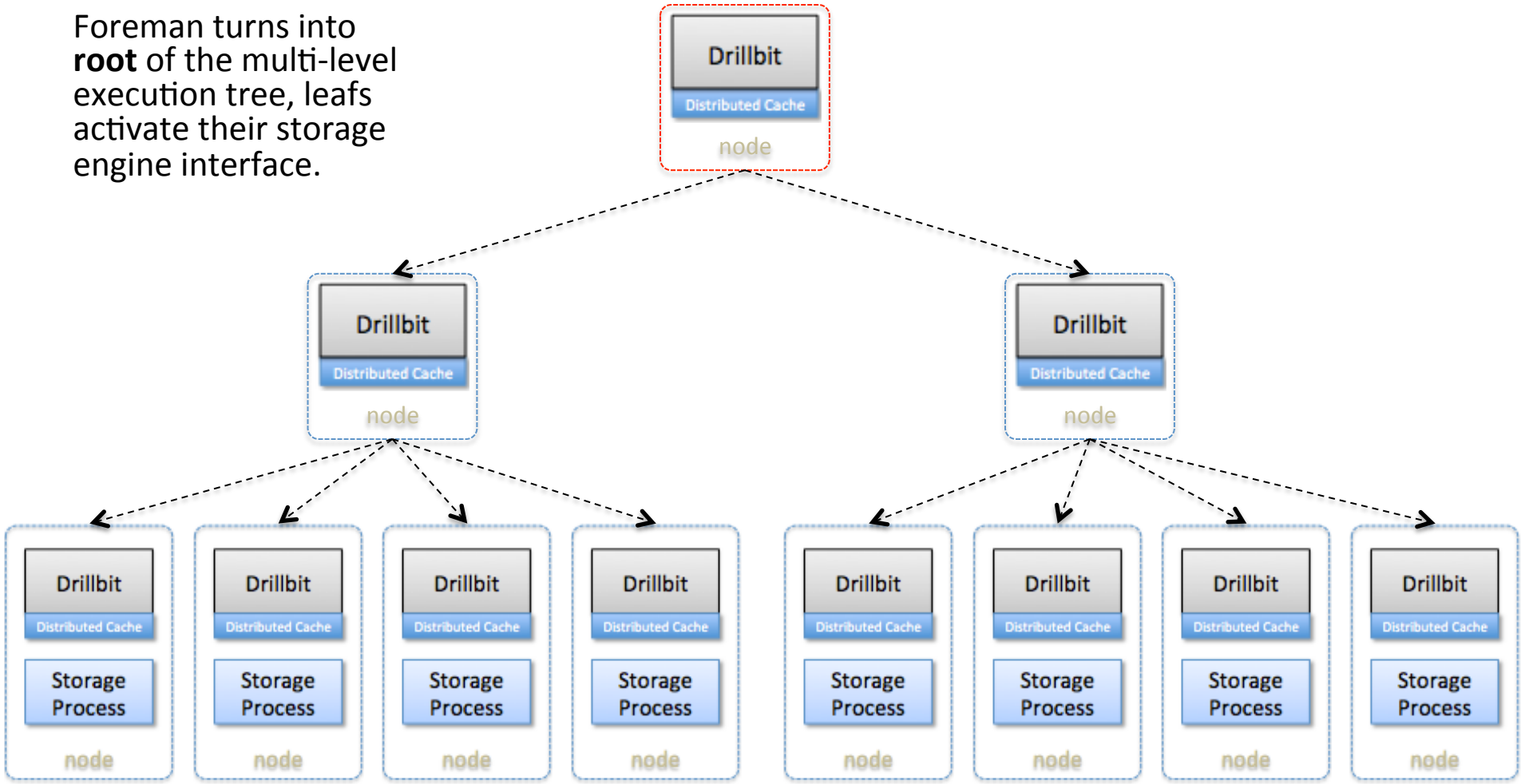
- Originating Drillbit acts as **foreman**: manages query execution, scheduling, locality information, etc.
- Streaming data **communication** that avoids SerDe



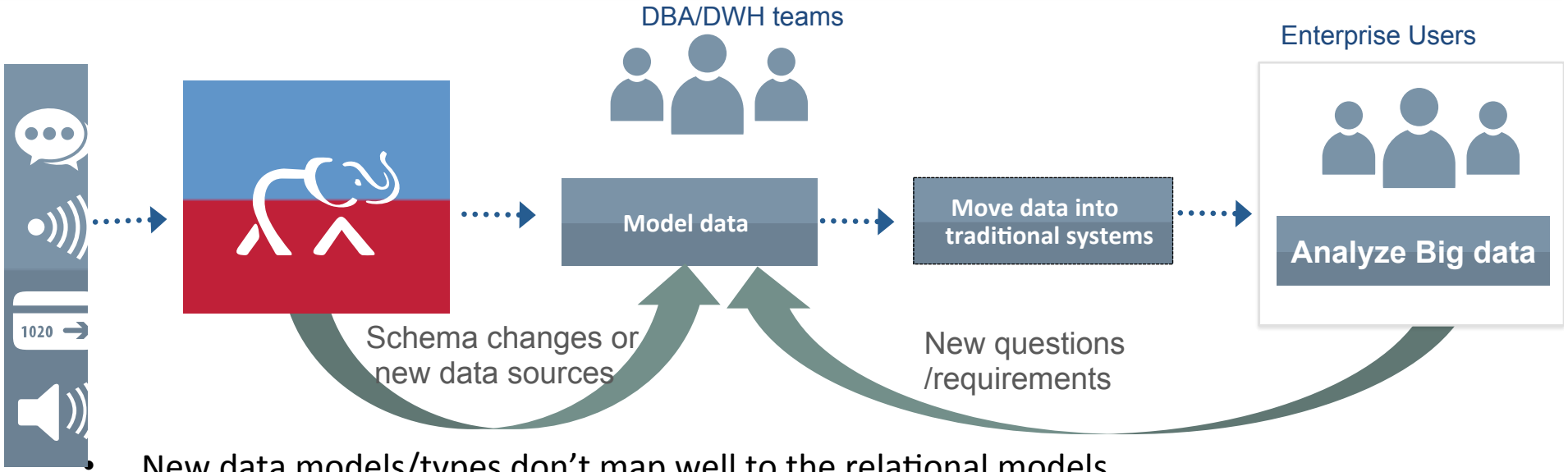
Wire-level Architecture

Performant

Foreman turns into **root** of the multi-level execution tree, leafs activate their storage engine interface.



Existing SQL approaches will not always work for big data needs



- New data models/types don't map well to the relational models
  - Many data sources do not have rigid schemas (HBase, Mongo etc)
    - Each record has a separate schema
    - Sparse and wide rows
  - Flattening nested data is error-prone and often impossible
    - Think about repeated and optional fields at every level…
    - A single HBase value could be a JSON document (compound nested type)
- Centralized schemas are hard to manage for big data
    - Rapidly evolving data source schemas
    - Lots of new data sources
    - Third party data
    - Unknown questions

Apache Drill

Open Source SQL on Hadoop for Agility with Big Data exploration

FLEXIBLE SCHEMA MANAGEMENT

Analyze data with or without centralized schemas

ANALYTICS ON NOSQL DATA

Analyze semi structured & nested data with no modeling/ETL

PLUG AND PLAY WITH EXISTING TOOLS

Analyze data using familiar BI/Analytics and SQL based tools

… and with an architecture built ground up for Low Latency queries at Scale

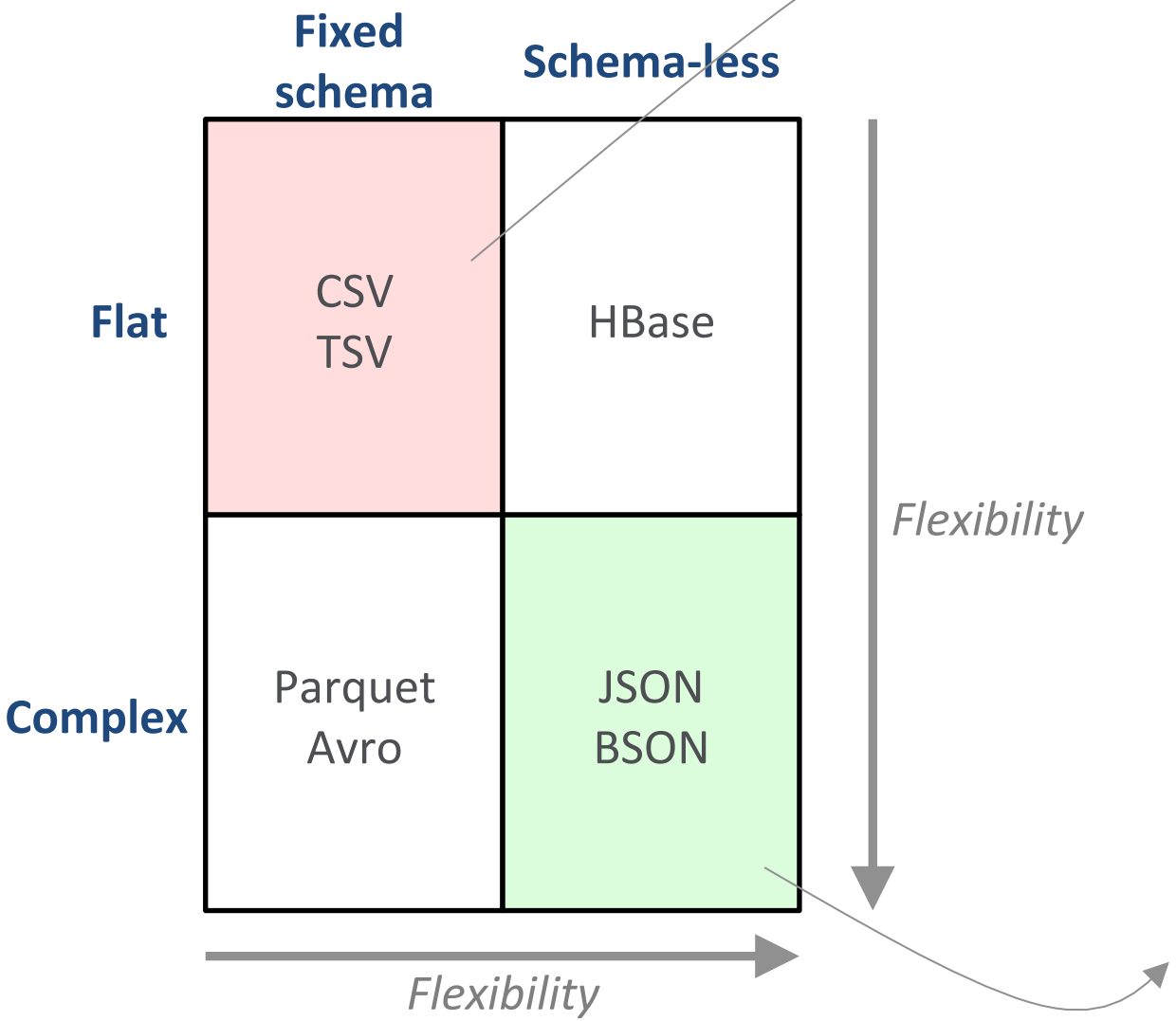
Features

MAPR

Agenda

- Problem Statement
- Architecture
- **Features**
- Status and progress

Modern Data Model



Flexible

RDBMS/SQL-on-Hadoop table

Name	Gender	Age
Mike	M	6
Jen	F	3

Apache Drill table

```
{
  name: {
    first: Michael,
    last: Smith
  },
  hobbies: [ski,
soccer],
  district: Los Altos
}
{
  name: {
    first: Jennifer,
    last: Gates
  },
  hobbies: [sing],
  preschool: CCLC
}
```

Schema Discovery On-The-Fly

Schema Declared In Advance

- Fixed schema
- Leverage schema in centralized repository (Hive Metastore)

Schema Discovered On-The-Fly

- Fixed schema, evolving schema or schema-less
- Leverage schema in centralized repository or self-describing data

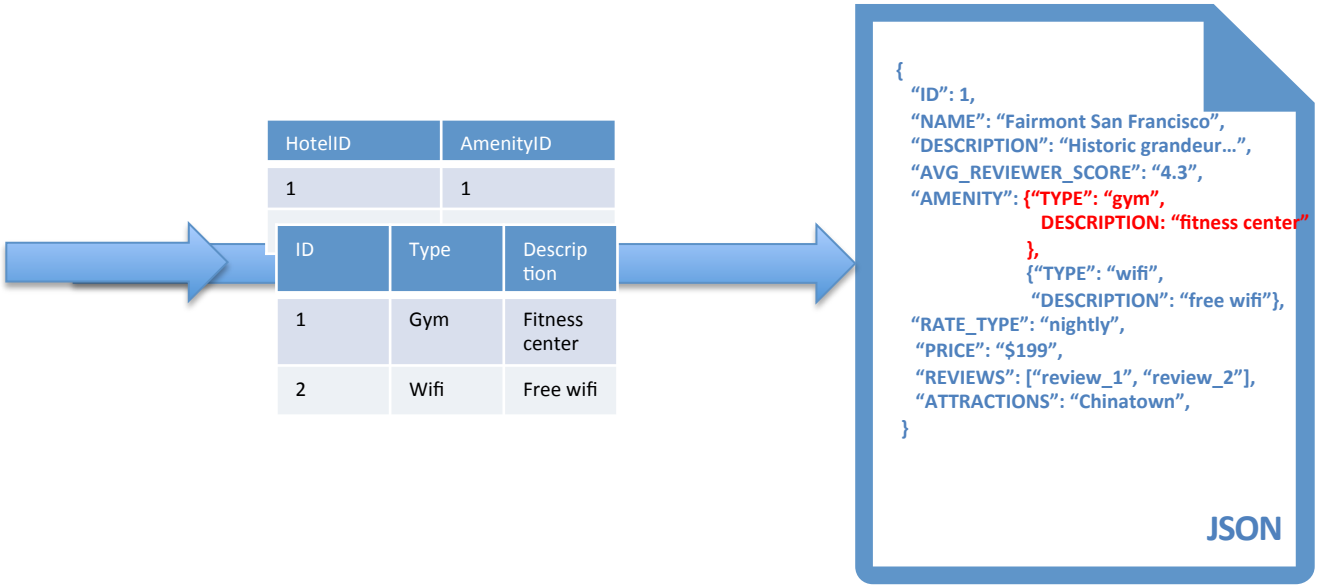
SCHEMA ON WRITE

SCHEMA BEFORE READ

SCHEMA ON THE FLY

Flexible schema management

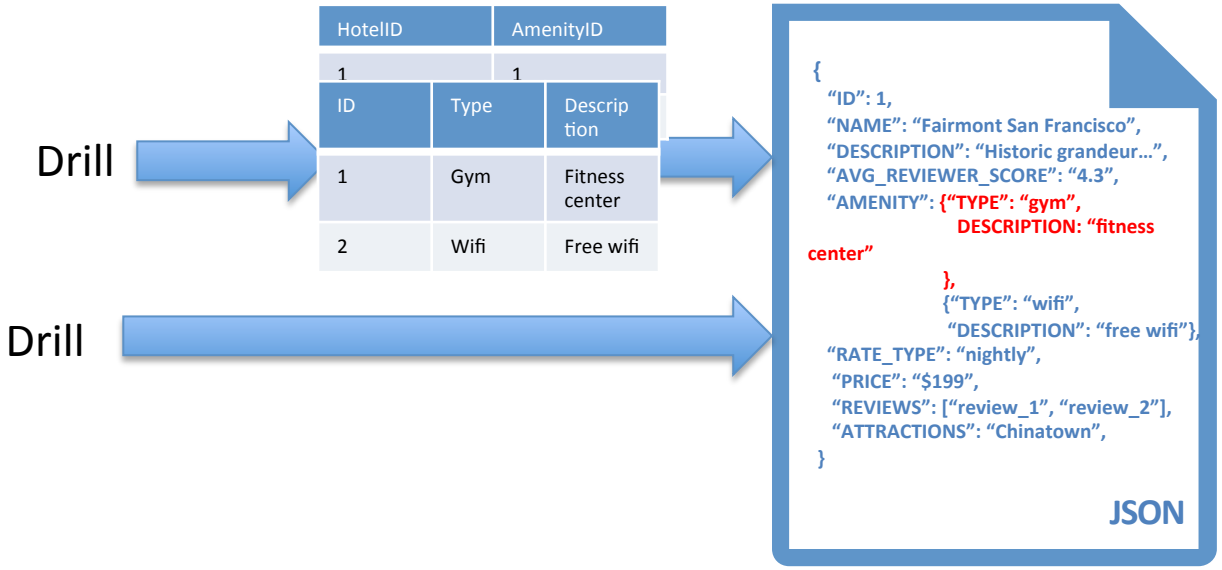
Existing SQL solutions



HotelID	AmenityID
1	1

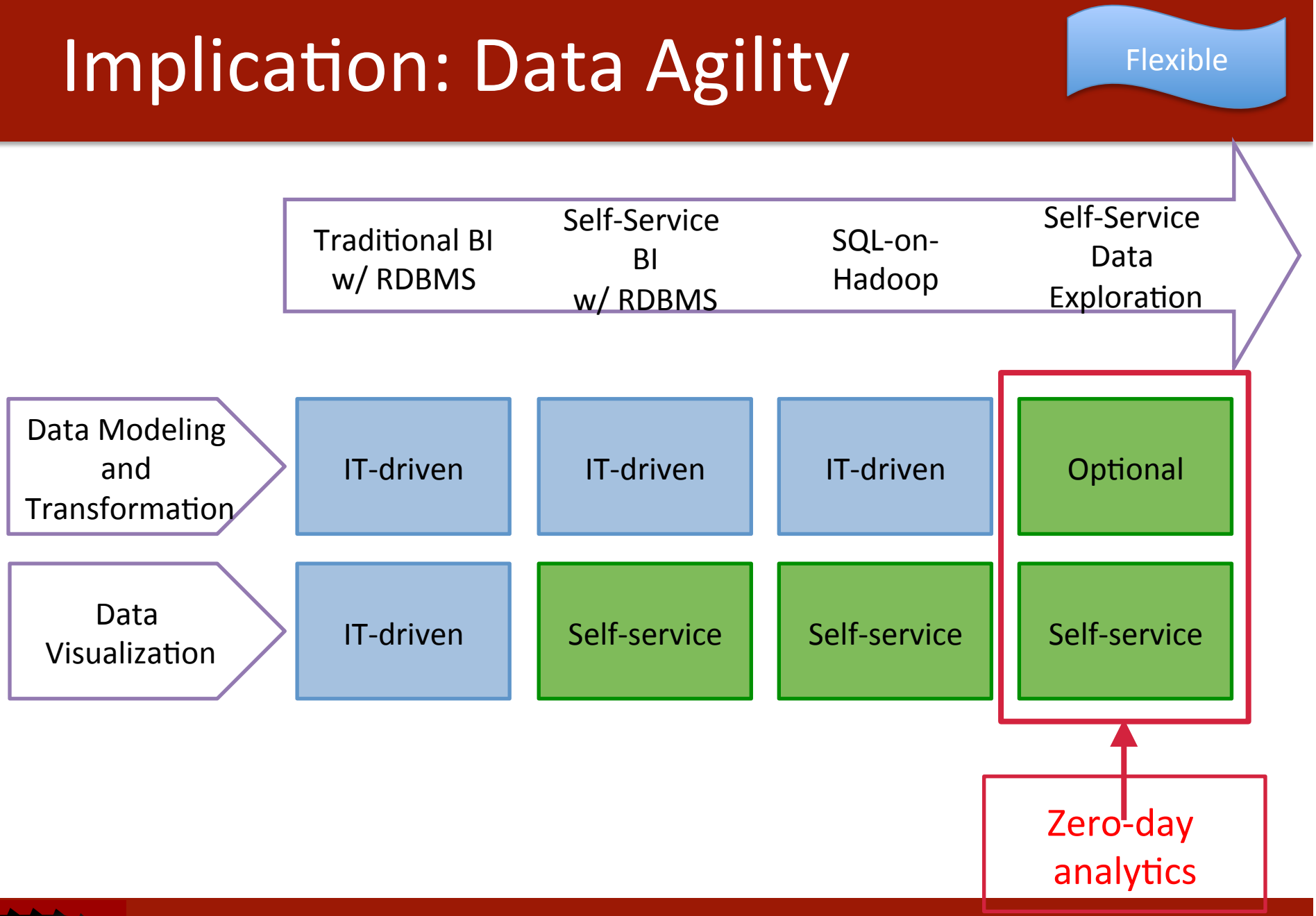
ID	Type	Description
1	Gym	Fitness center
2	Wifi	Free wifi

Flexible schema management



Drill doesn't require any schema definitions to query data making it faster to get insights from data for users. Drill leverages schema definitions if exists.

Implication: Data Agility



Key features

- Dynamic/schema-less queries
- Nested data
- Apache Hive integration
- ANSI SQL/BI tool integration

Querying files

- Direct queries on a local or a distributed file system (HDFS, S3 etc)
- Specify a single file or a directory as 'Table' within query
- Specify schema in query or let Drill discover it. No schema required!
- Example:
- **`SELECT * FROM dfs.`/home/mapr/sample-data/profiles.json``**

dfs	File system as data source
`/home/mapr/sample-data/profiles.json`	Table

More examples

- Query on single file

```
SELECT * FROM dfs.logs.`AppServerLogs/2014/Jan/part0001.txt`
```

- Query on directory

```
SELECT * FROM dfs.logs.`AppServerLogs/2014/Jan` where
errorLevel=1;
```

- Joins on files

```
SELECT  c.c_custkey,sum(o.o_totalprice)
FROM
   dfs.`/home/mapr/tpch/customer.parquet` c
   JOIN
   dfs.`/home/mapr/tpch/orders.parquet` o
   ON c.c_custkey = o.o_custkey
GROUP BY c.c_custkey
LIMIT 10
```

Querying HBase

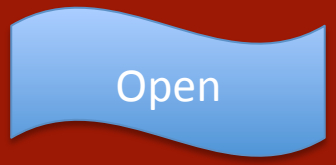
- Direct queries on HBase tables
  - `SELECT row_key, cf1.month, cf1.year FROM hbase.table1;`
  - `SELECT CONVERT_FROM(row_key, UTF-8) as HotelName from FROM HotelData`
- No need to define a parallel/overlay schema in Hive
- Encode and Decode data from HBase using Convert functions
  - Convert_To and Convert_From

Nested data

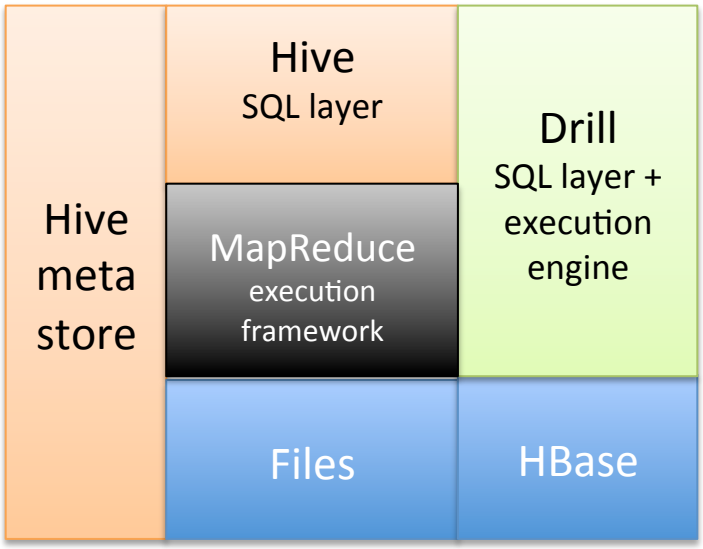
- Nested data as first class entity: Extensions to SQL for nested data types, similar to BigQuery
- No upfront flattening/modeling required
- Generic architecture for a broad variety of nested data types (eg:JSON, BSON, XML, AVRO, Protocol Buffers)
- Performance with ground up design for nested data
- Example:

```
SELECT
  c.name, c.address, REPEATED_COUNT(c.children)
FROM(
  SELECT
      CONVERT_FROM(cf1.user-json-blob, JSON) AS c
  FROM
      hbase.table1
)
```

Apache Hive integration



- Plug and Play integration in existing Hive deployments
- Use Drill to query data in Hive tables/views
- Support to work with more than one Hive metastore
- Support for all Hive file formats
- Ability to use Hive UDFs as part of Drill queries



Hive and Files in Same Query

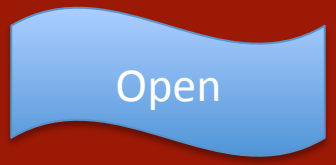
- JSON
- CSV
- ORC (any Hive type)
- Parquet
- HBase tables
- … can combine them

```
Select  USERS.name,
USERS.emails.work  from
  dfs.logs.`/data/logs`  LOGS,
  dfs.users.`/profiles.json`
USERS,
where
   LOGS.uid = USERS.uid   and
   errorLevel > 5
order by  count(*);
```

HBase and Hive in Same Query

- Combine data from Files, HBase, Hive in one query
- No central metadata definitions necessary
- Example:
  - `USE HiveTest.CustomersDB`
  - ```
    SELECT Customers.customer_name, SocialData.Tweets.Count
    FROM Customers
    JOIN HBaseCatalog.SocialData SocialData
    ON Customers.Customer_id = Convert_From(SocialData.rowkey, UTF-8)
    ```
    ```

SQL support



- ANSI SQL compatibility
  - "SQL Like" not enough
  - Fully SQL-92 compliant
    - With extensions to support nested data
- Testing against TPC-H benchmarks
- Full SQL compliance will enable the broadest support for tools that use SQL
- SQL data types
  - `SMALLINT, BIGINT, TINYINT, INT, FLOAT, DOUBLE,DATE, TIMESTAMP, DECIMAL, VARCHAR, VARBINARY ….`
- All common SQL constructs
    - `SELECT, GROUP BY, ORDER BY, LIMIT, JOIN, HAVING, UNION, UNION ALL, IN/NOT IN, EXISTS/NOT EXISTS,DISTINCT, BETWEEN, CREATE TABLE/VIEW AS ….`
    - `Scalar and correlated sub queries`
- Metadata discovery using INFORMATION_SCHEMA

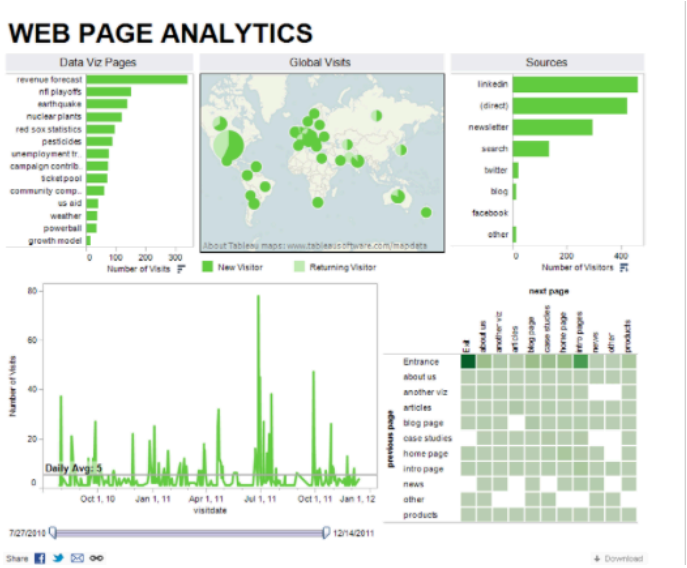
Seamless Integration: ANSI SQL 2003

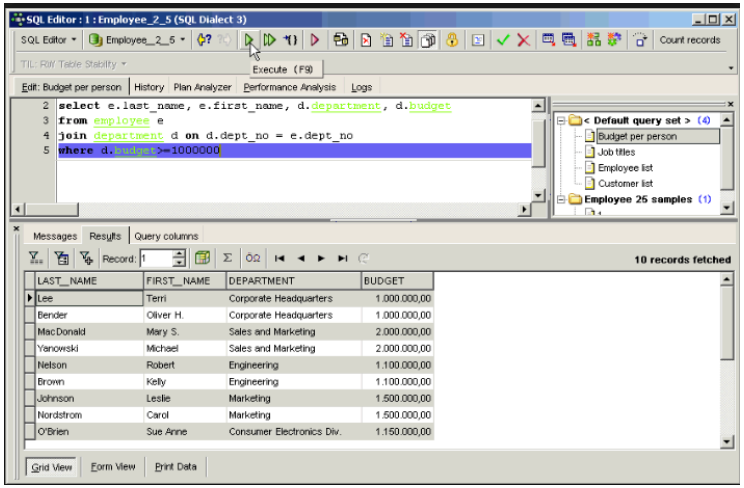
Open

- Users want "standard" SQL rather than SQL-like (HiveQL)
  - Leverage existing expertise
  - Better support for BI/DI tools
- Drill's supports ANSI SQL
  - Extensions to handle complex data
- Current status:
  - Drill 0.5 runs 15 of 22 TPC-H queries unmodified

BI tool integration

Open





- Standard JDBC/ODBC drivers
- Integration Tableau, Excel, Microstrategy, Toad, SQuirreL…

Packaging/install

- Works on all Hadoop distributions
- Easy ramp up with embedded/standalone mode
  - Try out Drill easily on your machine
  - No Hadoop requirement

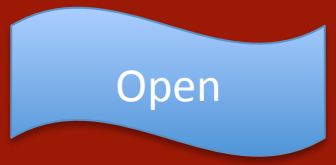
Under the Hood

MAPR

Agenda

- Problem Statement
- Architecture
- **Features**
- Status and progress

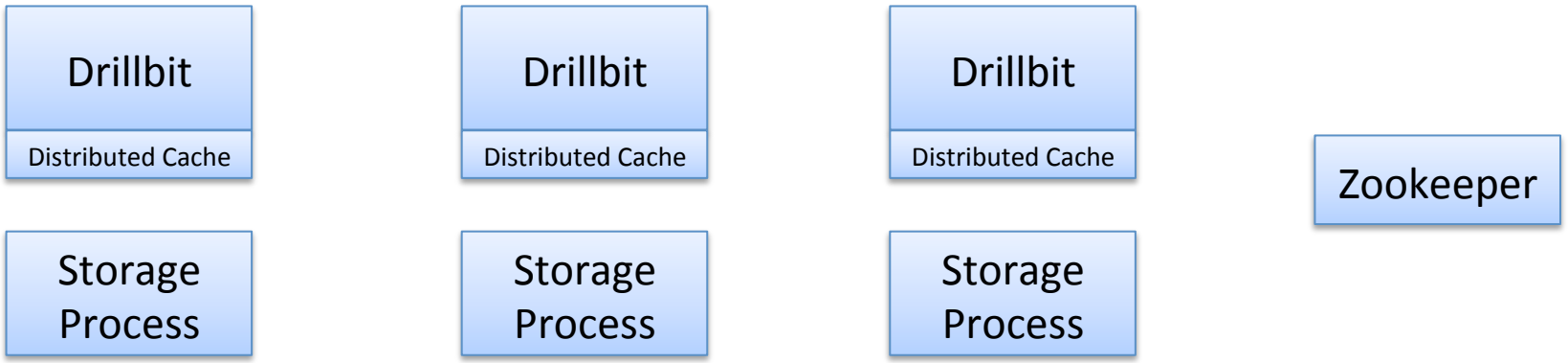
Extensibility at all layers



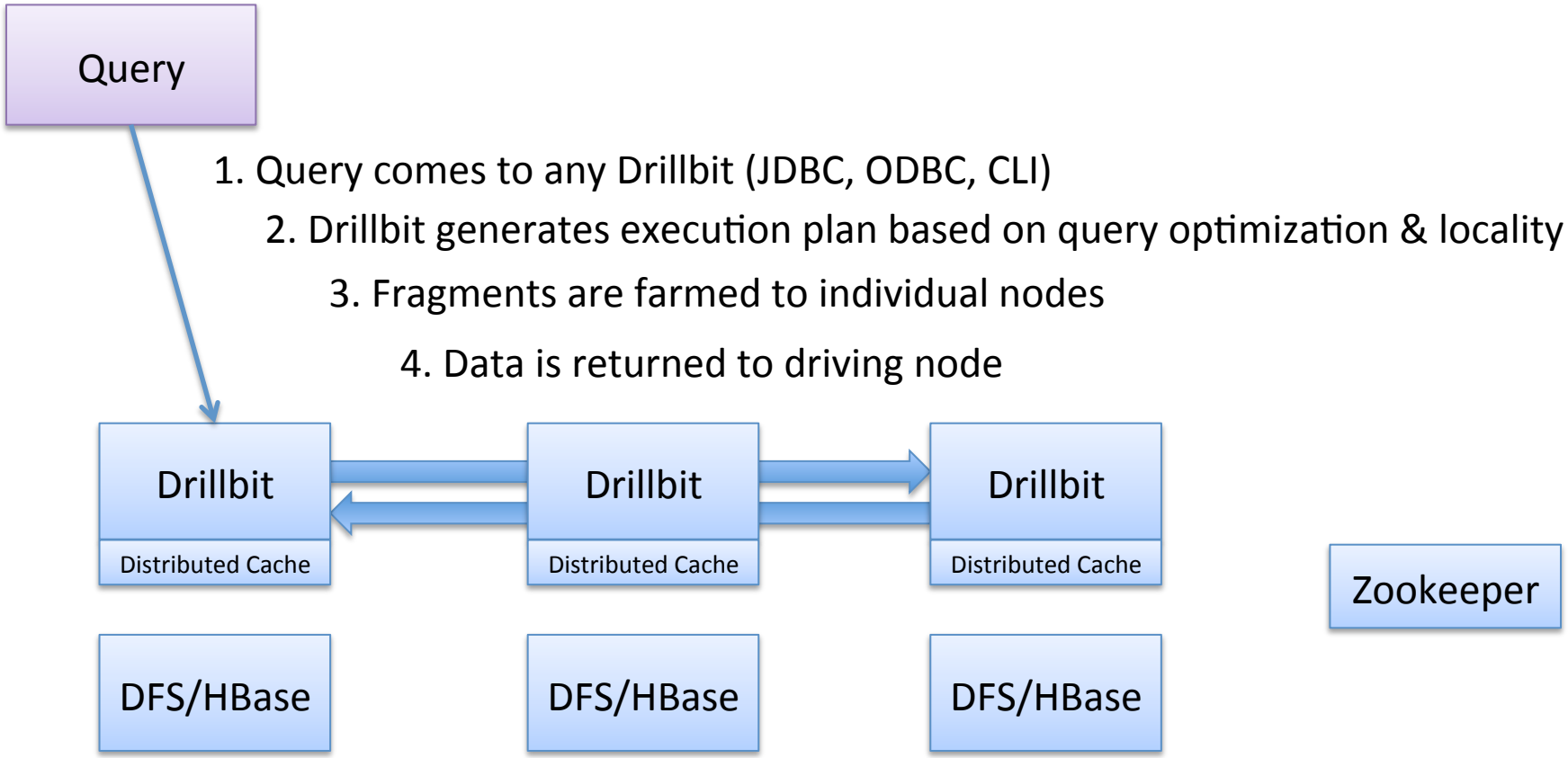
- Well-documented APIs and interfaces
- Data sources and file formats
  - Implement customer scanner to support new file formats
- Functions
  - Define UDFs/UDAFs using high performance Java API
- Query languages
  - SQL is the primary language
  - Implement a custom parser to support domain specific languages
- Optimizers
  - Key standard optimizations out of box
  - API to plugin optimization rules
- Operators
  - Custom operators can be implemented

High Level Architecture

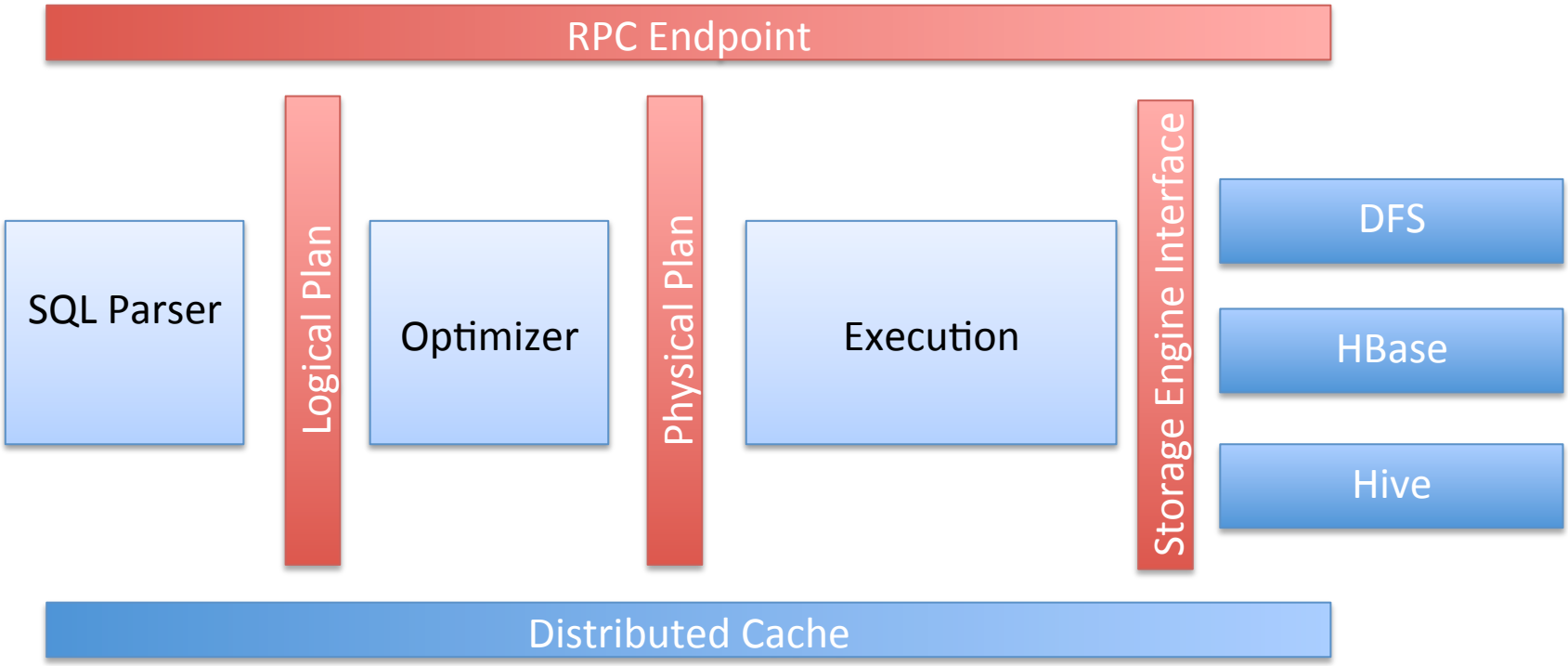
- Drillbits run on each node, designed to maximize data locality
- Drill includes a distributed execution environment built specifically for distributed query processing
- Any Drillbit can act as endpoint for particular query.
- Zookeeper maintains ephemeral cluster membership information only
- Small distributed cache utilizing embedded Hazelcast maintains information about individual queue depth, cached query plans, metadata, locality information, etc.



Basic query flow



Core Modules within a Drillbit



Query Execution

- **Source query**—what we want to do (analyst friendly)
- **Logical Plan**— what we want to do (language agnostic, computer friendly)
- **Physical Plan**—how we want to do it (the best way we can tell)
- **Execution Plan**—where we want to do it

A Query engine that is…

- Optimistic/pipelined
- Columnar/Vectorized
- Late binding
- Extensible

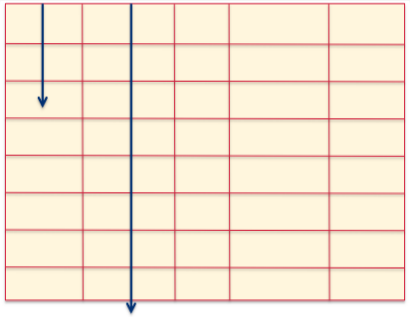
Optimistic Execution

- With a short time horizon, failures infrequent
  - Don't spend energy and time creating boundaries and checkpoints to minimize recovery time
  - Rerun entire query in face of failure
- No barriers
- No persistence unless memory overflow

Efficient Processing

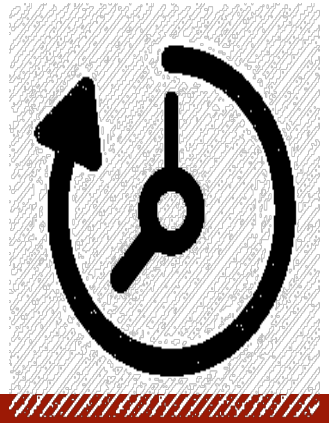
Performant

Columnar Model



Binary Data Representation

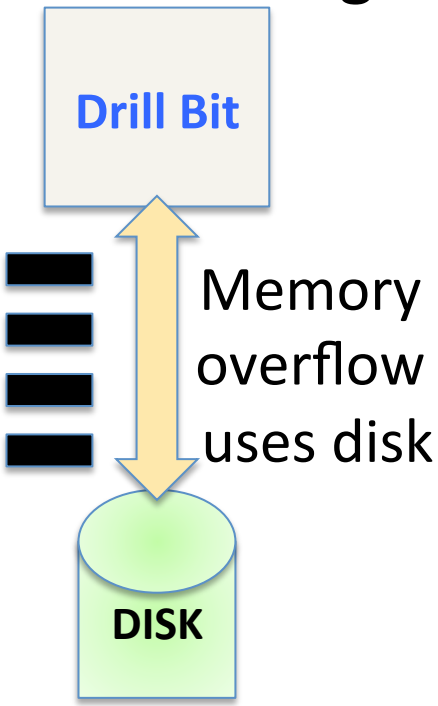
Run Time Compilation



APACHE DRILL

Efficient Processing

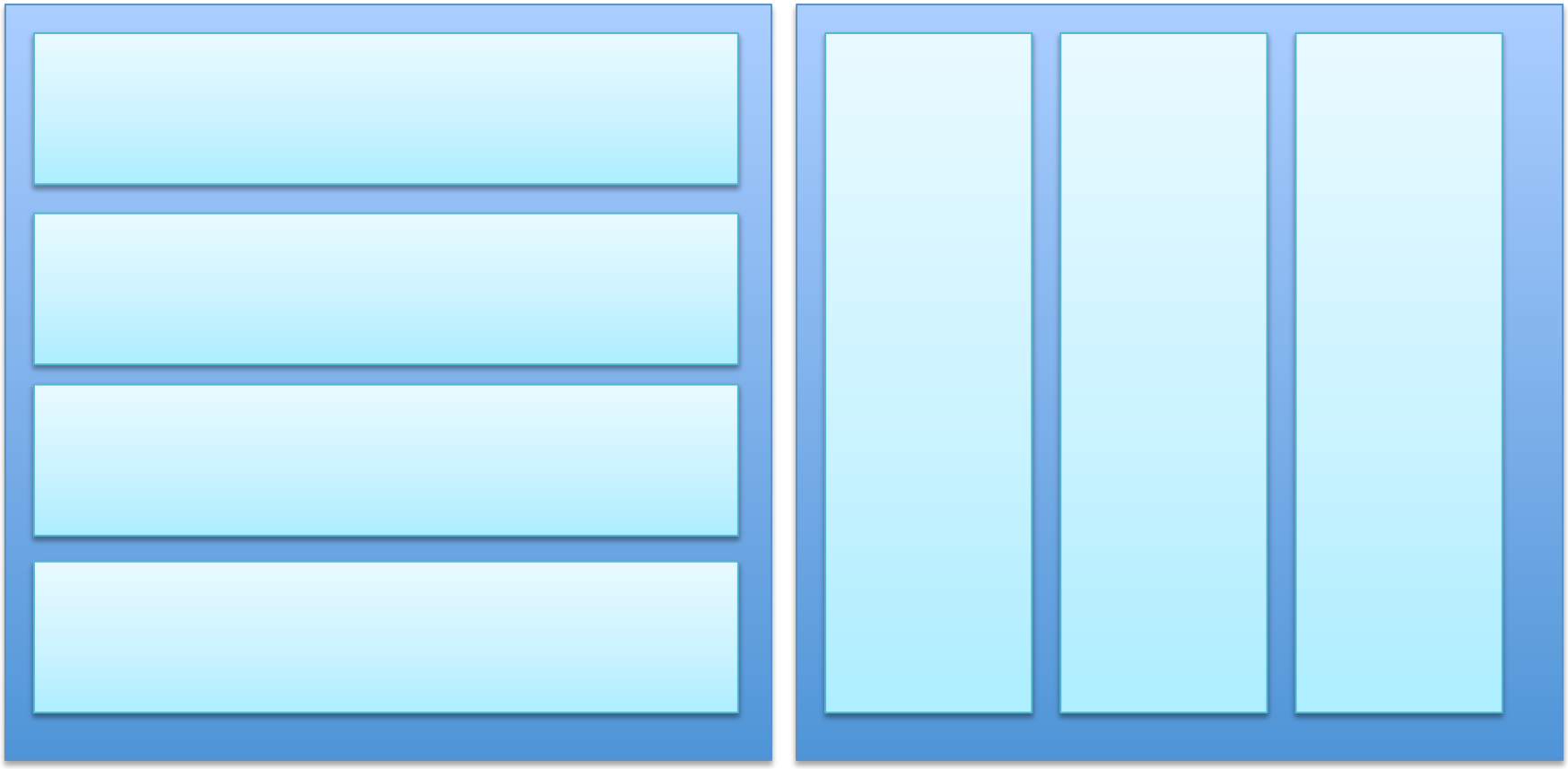
- Random access: sort without copy or restructuring
- Avoids serialization/deserialization
- Off-heap (no GC woes)
- Full specification + off-heap + batch
  - Enables C/C++ operators *(fast!)*
- Read/write to disk
  - when data larger than memory



Runtime Compilation

- Give JIT help
- Avoid virtual method invocation
- Avoid heap allocation and object overhead
- Minimize memory overhead

Record versus Columnar Representation



Record

Column

Data Format Example

Donut	Price	Icing
Bacon Maple Bar	2.19	[Maple Frosting, Bacon]
Portland Cream	1.79	[Chocolate]
The Loop	2.29	[Vanilla, Fruitloops]
Triple Chocolate Penetration	2.79	[Chocolate, Cocoa Puffs]

Record Encoding

Bacon Maple Bar, 2.19, Maple Frosting, Bacon, Portland Cream, 1.79, Chocolate
The Loop, 2.29, Vanilla, Fruitloops, Triple Chocolate Penetration, 2.79, Chocolate, Cocoa Puffs

Columnar Encoding

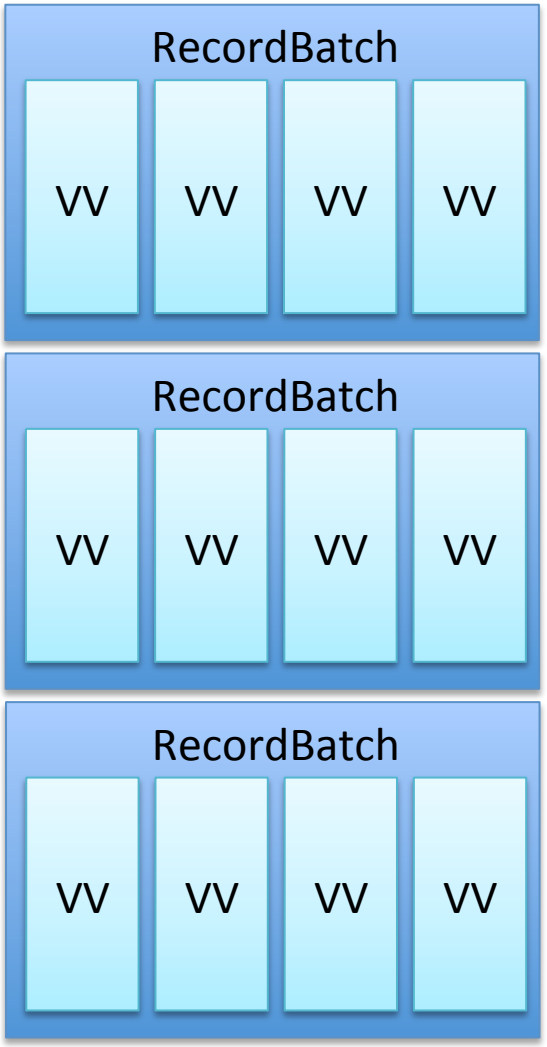
Bacon Maple Bar, Portland Cream, The Loop, Triple Chocolate Penetration
2.19, 1.79, 2.29, 2.79
Maple Frosting, Bacon, Chocolate, Vanilla, Fruitloops, Chocolate, Cocoa Puffs

Example: RLE and Sum

- Dataset
  - 2, 4
  - 8, 10
- Goal
  - Sum all the records
- Normal Work
  - Decompress & store: 2, 2, 2, 2, 8, 8, 8, 8, 8, 8, 8, 8, 8, 8
  - Add: 2 + 2 + 2 + 2 + 8 + 8 + 8 + 8 + 8 + 8 + 8 + 8 + 8 + 8
- Optimized Work
  - 2 * 4 + 8 * 10
  - Less Memory, less operations

Record Batch

- Drill optimizes for BOTH columnar STORAGE and Execution
- Record Batch is unit of work for the query system
  - Operators always work on a batch of records
- All values associated with a particular collection of records
- Each record batch must have a single defined schema
- Record batches are pipelined between operators and nodes



Strengths of RecordBatch + ValueVectors

- RecordBatch clearly delineates low overhead/high performance space
  - Record-by-record, avoid method invocation
  - Batch-by-batch, trust JVM
- Avoid serialization/deserialization
- Off-heap means large memory footprint without GC woes
- Full specification combined with off-heap and batch-level execution allows C/C++ operators as necessary
- Random access: sort without copy or restructuring

Late Schema Binding

- Schema can change over course of query
- Operators are able to reconfigure themselves on schema change events

Integration and Extensibility points

Flexible

- Support UDFs
  - UDFs/UDAFs using high performance Java API
- Not Hadoop centric
  - Work with other NoSQL solutions including MongoDB, Cassandra, Riak, etc.
  - Build one distributed query engine together than per technology
- Built in classpath scanning and plugin concept to add additional storage engines, function and operators with zero configuration
- Support direct execution of strongly specified JSON based logical and physical plans
  - Simplifies testing
  - Enables integration of alternative query languages

How Drill achieves performance

- Execution
  - MPP query engine
  - Pipelined
  - Runtime compiled
  - Columnar execution
  - Late binding
  - Vectorized operators

- Optimization
  - Cost based optimization
  - Pushdown logic to sources
  - Partition awareness
  - ….. And more

Comparison with MapReduce

- Barriers
  - Map completion required before shuffle/reduce commencement
  - All maps must complete before reduce can start
  - In chained jobs, one job must finish entirely before the next one can start
- Persistence and Recoverability
  - Data is persisted to disk between each barrier
  - Serialization and deserialization are required between execution phase

Status and Progress

MAPR

Agenda

- Problem Statement
- Architecture
- Features
- **Status and progress**

Zero to Results in 2 Minutes (3 Commands)

```
$ tar xzf apache-drill.tar.gz

$ apache-drill/bin/sqlline -u jdbc:drill:zk=local

0: jdbc:drill:zk=local>
  SELECT   count(*) AS incidents, columns[1] AS category
  FROM     dfs.`/tmp/SFPD_Incidents_..._Three_Months.csv`
  GROUP BY columns[1]
  ORDER BY incidents DESC;
+------------+------------+
| incidents  |  category  |
+------------+------------+
| 8372       | LARCENY/THEFT |
| 4247       | OTHER OFFENSES |
| 3765       | NON-CRIMINAL |
| 2502       | ASSAULT    | …
35 rows selected (0.847 seconds)
```

Install

Launch shell

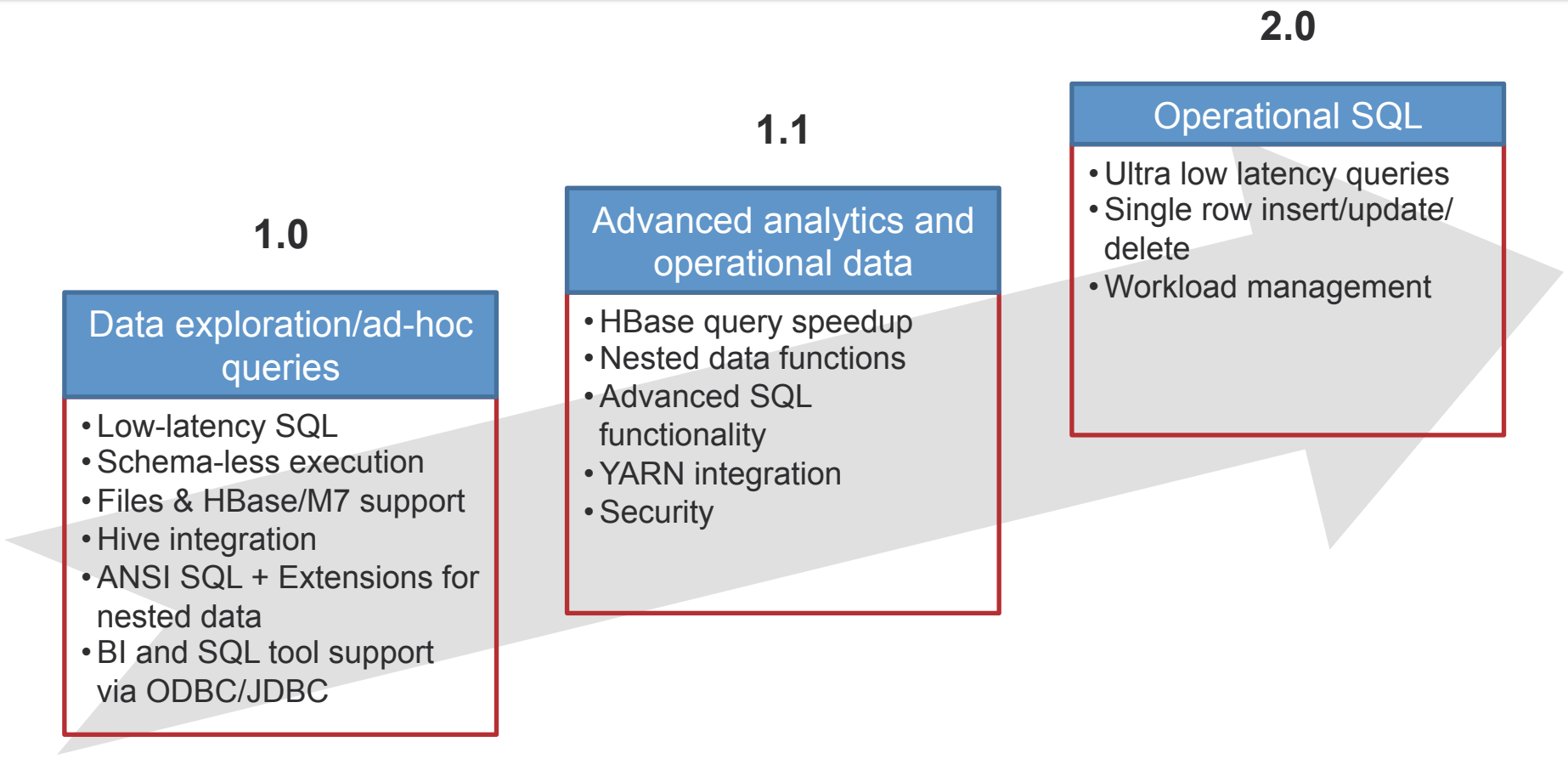
Query

Results

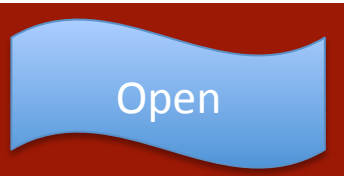
Status

- Heavy active development
- Significant community momentum
  - Many contributors
  - 400+ people in Drill mailing lists
  - 400+ members in Bay area Drill user group
- Current state : Beta, driving to GA early next year
  - Monthly “dot” releases
  - 0.7 out any day

Roadmap


1.0
Data exploration/ad-hoc queries
• Low-latency SQL
• Schema-less execution
• Files & HBase/M7 support
• Hive integration
• ANSI SQL + Extensions for nested data
• BI and SQL tool support via ODBC/JDBC
1.1
Advanced analytics and operational data
• HBase query speedup
• Nested data functions
• Advanced SQL functionality
• YARN integration
• Security
2.0
Operational SQL
• Ultra low latency queries
• Single row insert/update/delete
• Workload management

Interested in Apache Drill?



- http://www.mapr.com/blog/apache-drill-approaches-new-milestone
- Join the community
  - Join the Drill mailing lists
    - drill-user@incubator.apache.org
    - drill-dev@incubator.apache.org
  - Contribute
    - Use cases/Sample queries, JIRAs, code, unit tests, documentation, ...
  - Fork us on GitHub: http://github.com/apache/incubator-drill/
  - Create a JIRA: https://issues.apache.org/jira/browse/DRILL
- Resources
  - Try out Drill in 10mins
  - http://incubator.apache.org/drill/
  - https://cwiki.apache.org/confluence/display/DRILL/Apache+Drill+Wiki

Thank You

@mapr   maprtech

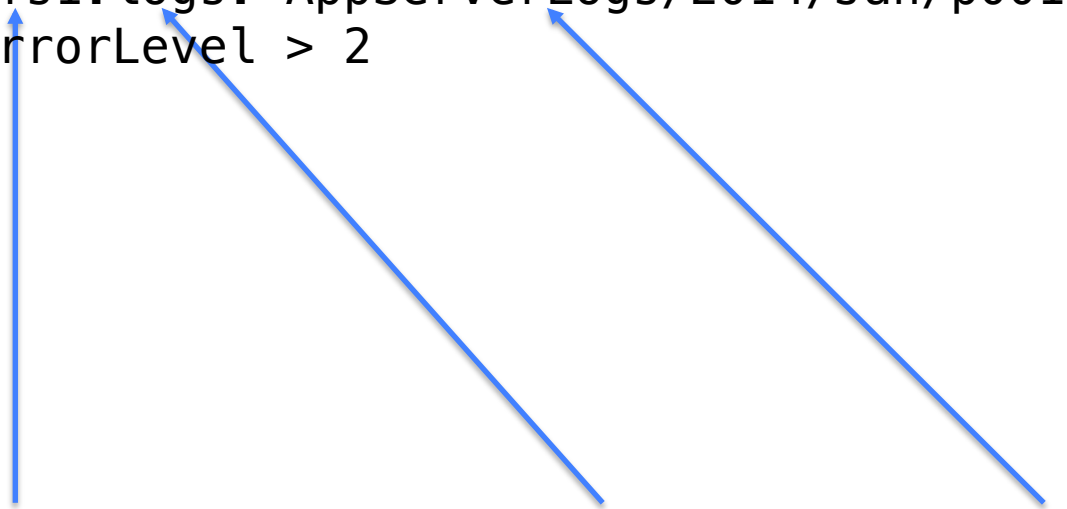
mapr-technologies   MapRTechnologies

kbotzum@mapr.com   maprtech

Appendix

Data Source is in the Query

```
SELECT timestamp, message
FROM   dfs1.logs.`AppServerLogs/2014/Jan/p001.parquet`
WHERE  errorLevel > 2
```



A *storage engine instance*
- DFS
- HBase
- Hive Metastore/HCatalog

A *workspace*
- Sub-directory
- Hive database
- HBase namespace

A *table*
- pathnames
- HBase table
- Hive table

Works with HBase and Embedded Blobs

```
# Query an HBase table directly (no schemas)

SELECT cf1.month, cf1.year
FROM   hbase.table1;

# Embedded JSON value inside column profileBlob inside
column family cf1 of the HBase table users

SELECT profile.name, count(profile.children)
FROM (
  SELECT CONVERT_FROM(cf1.profileBlob, 'json') AS profile
  FROM hbase.users
)
```

Combine Data Sources on the Fly

```
# Join log directory with JSON file (user profiles) to
identify the name and email address for anyone associated
with an error message.

SELECT DISTINCT users.name, users.emails.work
FROM            dfs.logs.`/data/logs` logs,
                dfs.users.`/profiles.json` users
WHERE           logs.uid = users.id AND
                logs.errorLevel > 5;

# Join a Hive table and an HBase table (without Hive
metadata) to determine the number of tweets per user

SELECT   users.name, count(*) as tweetCount
FROM     hive.social.tweets tweets,
         hbase.users users
WHERE    tweets.userId = convert_from(users.rowkey,
'UTF-8')
GROUP BY tweets.userId;
```

Use ANSI SQL with no modifications

```
# TPC-H standard query 4

SELECT
o.o_orderpriority, count(*) AS order_count
FROM orders o
WHERE o.o_orderdate >= date '1996-10-01'
      AND o.o_orderdate < date '1996-10-01' +
interval '3' month
      AND EXISTS(
                  SELECT * FROM lineitem l
                  WHERE l.l_orderkey = o.o_orderkey
                  AND l.l_commitdate <
l.l_receiptdate
                  )
      GROUP BY o.o_orderpriority
      ORDER BY o.o_orderpriority;
```