APACHE PHOENIX ❤ APACHE DRILL

Drillix: Combined Operational & Analytical SQL at Scale

Salesforce.com, February 19, 2016

Who?

- CTO and Cofounder of Dremio
- Open Source:
  - PMC Chair Apache Arrow
  - PMC Chair Apache Drill
  - PMC Calcite, Incubator
  - Contributor on Parquet
  - Past contributor on HBase
- jacques@apache.org
- @intjesus

Apache Drill

- Open source SQL query engine for non-relational datastores
  - JSON document model
  - Columnar
  - First Production Release (Drill 1.0) released in May 2015

- Key advantages:
  - Query any non-relational datastore
  - No overhead (creating and maintaining schemas, transforming data, …)
  - Treat your data like a table even when it’s not
  - Keep using the BI tools you love
  - Scales from one laptop to 1000s of servers
  - Great performance and scalability

Drill Integrates With What You Have

Any Non-Relational Datastore

- File systems
  - Traditional: Local files and NAS
  - Hadoop: HDFS and MapR-FS
  - Cloud storage: Amazon S3, Google Cloud Storage, Azure Blob Storage
- NoSQL databases
  - MongoDB
  - HBase
  - MapR-DB
  - Hive
- And you can add new datastores

Any Client

- Multiple interfaces: ODBC, JDBC, REST, C, Java
- BI tools
  - Tableau
  - Qlik
  - MicroStrategy
  - TIBCO Spotfire
  - Excel
- Command line (Drill shell)
- Web and mobile apps
  - Many JSON-powered chart libraries (see D3.js)
- SAS, R, …

Apache Drill Provides the Best of Both Worlds

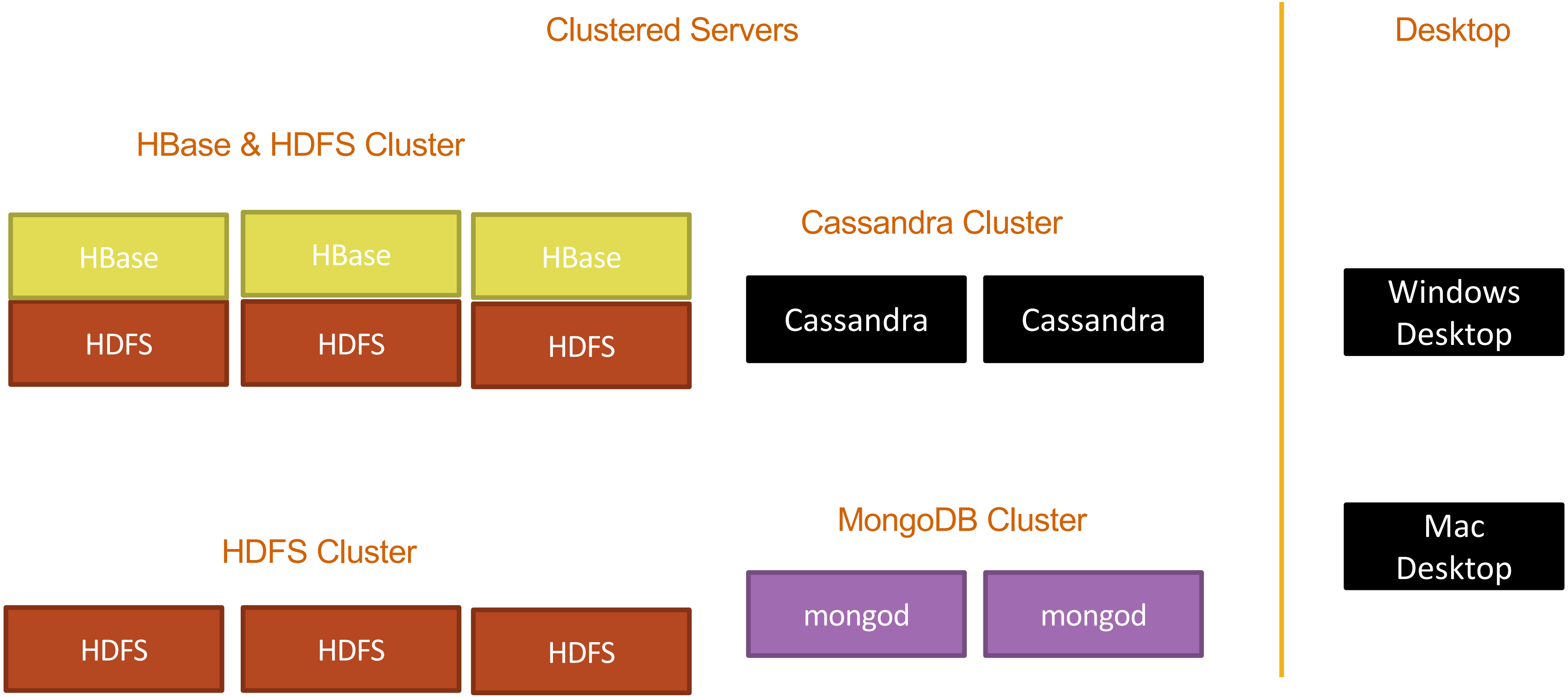
Acts Like a Database

- ANSI SQL: SELECT, FROM, WHERE, JOIN, HAVING, ORDER BY, WITH, CTAS, ALL, EXISTS, ANY, IN, SOME
- VarChar, Int, BigInt, Decimal, VarBinary, Timestamp, Float, Double, etc.
- Subqueries, scalar subqueries, partition pruning, CTE
- Data warehouse offload
- Tableau, ODBC, JDBC
- TPC-H & TPC-DS-like workloads
- Supports Hive SerDes
- Supports Hive UDFs
- Supports Hive Metastore

Even When Your Data Doesn’t

- Path based queries and wildcards
  - select * from /my/logs/
  - select * from /revenue/*/q2
- Modern data types
  - Map, Array, Any
- Complex Functions and Relational Operators
  - FLATTEN, kvgen, convert_from, convert_to, repeated_count, etc
- JSON Sensor analytics
- Complex data analysis
- Alternative DSLs

Data Lake, More Like Data Maelstrom



Clustered Servers

Desktop

HBase & HDFS Cluster

HBase HBase HBase

HDFS HDFS HDFS

Cassandra Cluster

Cassandra Cassandra

Windows Desktop

HDFS Cluster

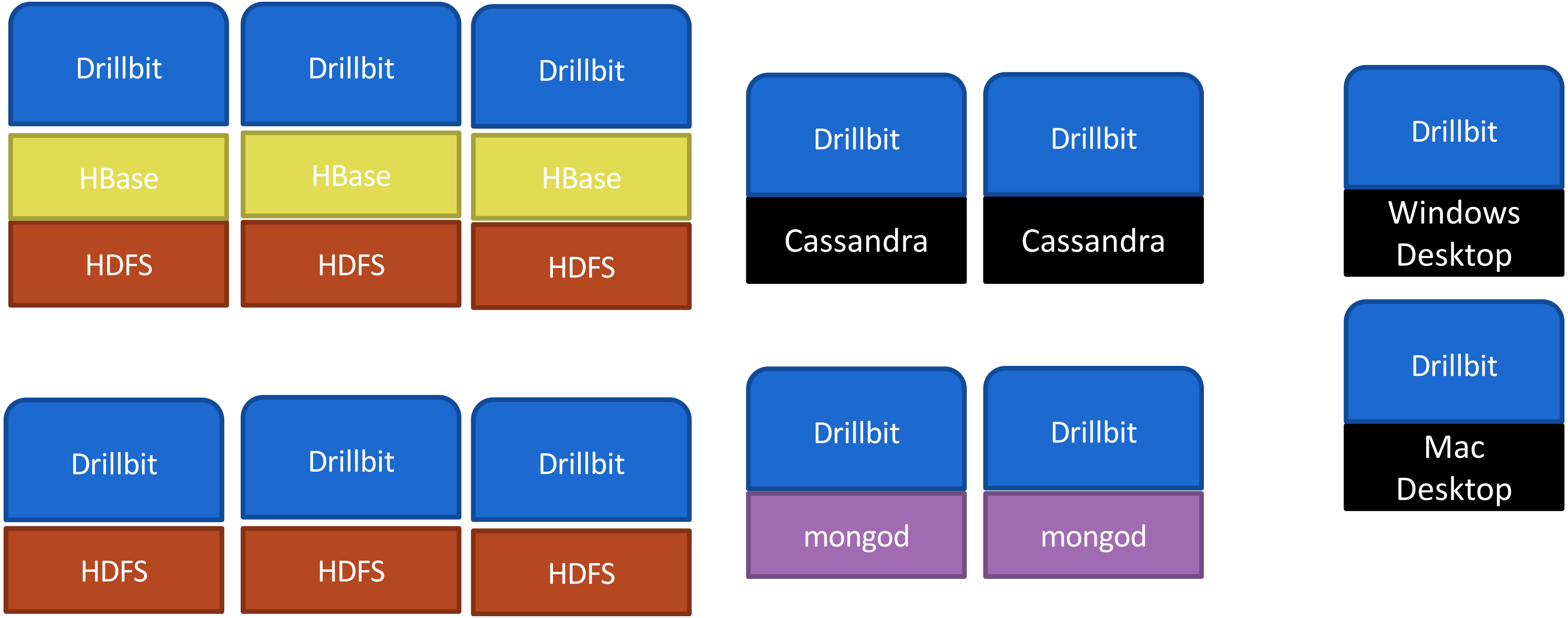
HDFS HDFS HDFS

MongoDB Cluster

mongod mongod

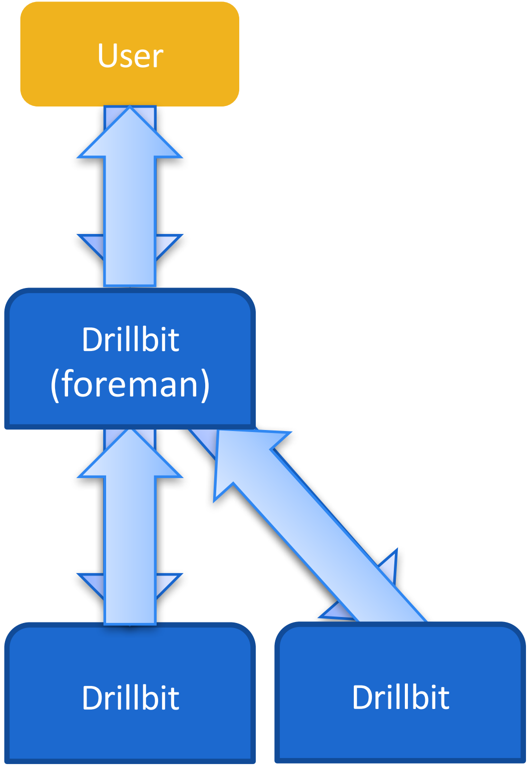
Mac Desktop

Run Drillbits Wherever; Whatever Your Data



Connect to Any Drillbit with ODBC, JDBC, C, Java, REST

1. User connects to Drillbit
2. That Drillbit becomes Foreman
   - Foreman generates execution plan
   - Cost-based query optimization & locality
3. Execution fragments are farmed to other Drillbits
4. Drillbits exchange data as necessary to guarantee relational algebra
5. Results are returned to user through Foreman



Why Drillix?

Phoenix Provides

- Strong Operational SQL capabilities
  - CREATE, INSERT, UPDATE, SELECT
- Broadcast and merge joins
- Transactions (soon)
- Extremely powerful HBase Integration
  - SkipScan, Coprocessors, etc
- Built on Calcite to do query parsing and optimization

Drill Provides

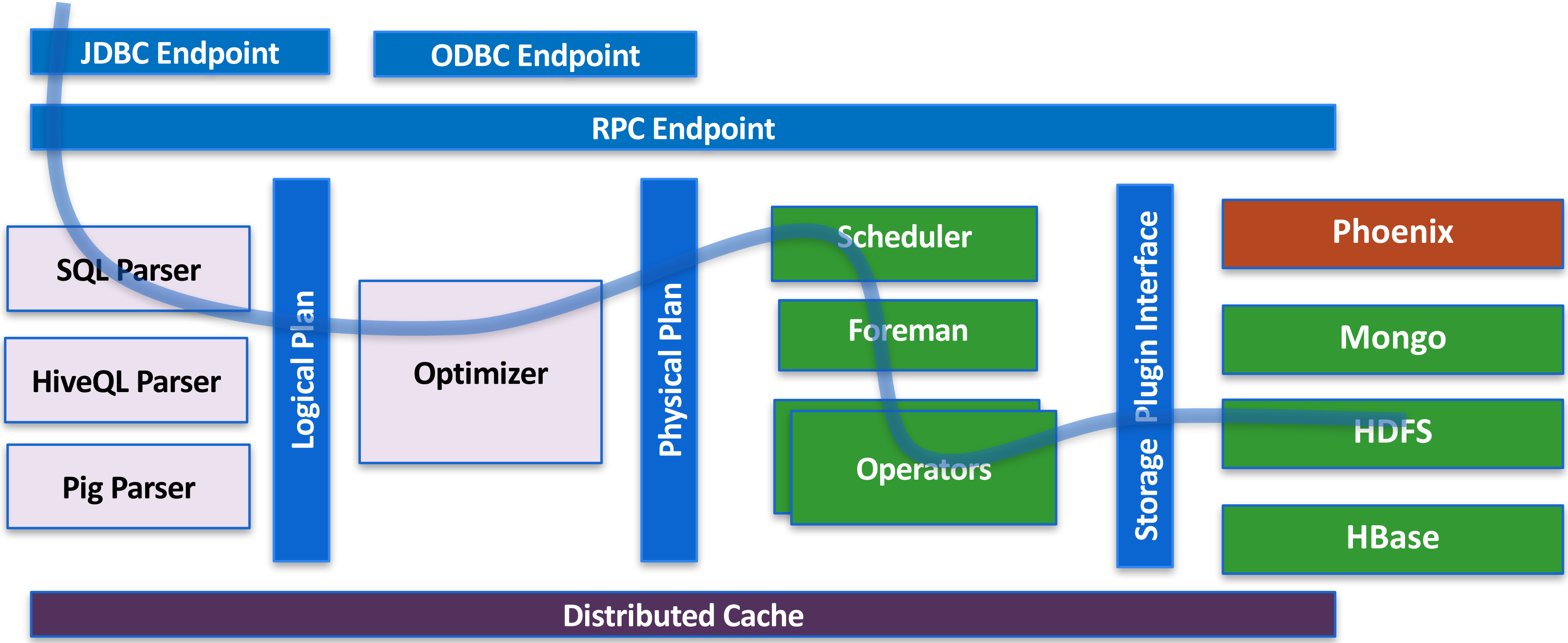
- Powerful Analytical SQL
  - Window Functions, all join types, many phase queries)
- Highly optimized columnar engine
- Powerful JSON Capablities
- Metadata and storage agnostic
- Window functions
- Built on Calcite to do query parsing and optimization

Let’s use Calcite to provide more powerful analytical capabilities on top of Phoenix

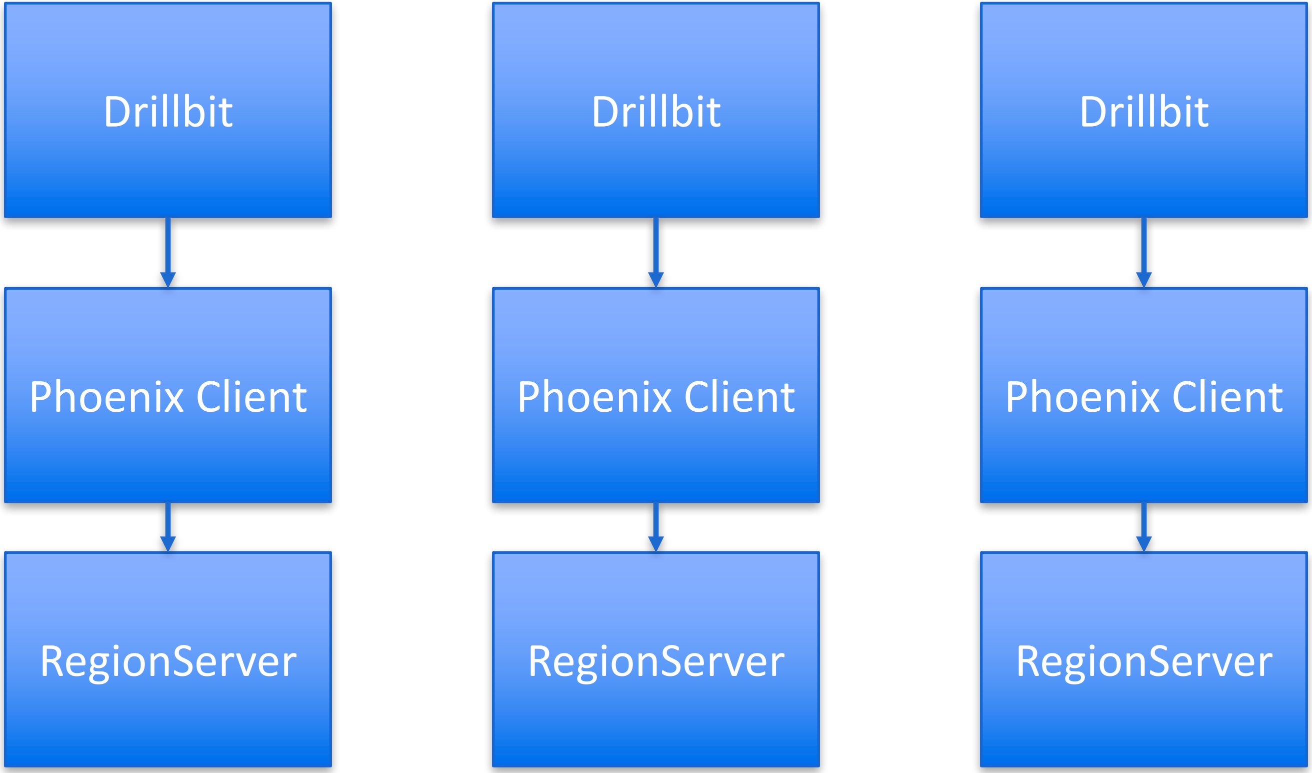
Calcite: A library to build your next Database

- Standard SQL
  - – Parsing, Validation, etc
- Query Optimization
  - – Query Plans
  - – Volcano Based Transformations
- JDBC & Metadata
- Vast library of powerful optimization rules

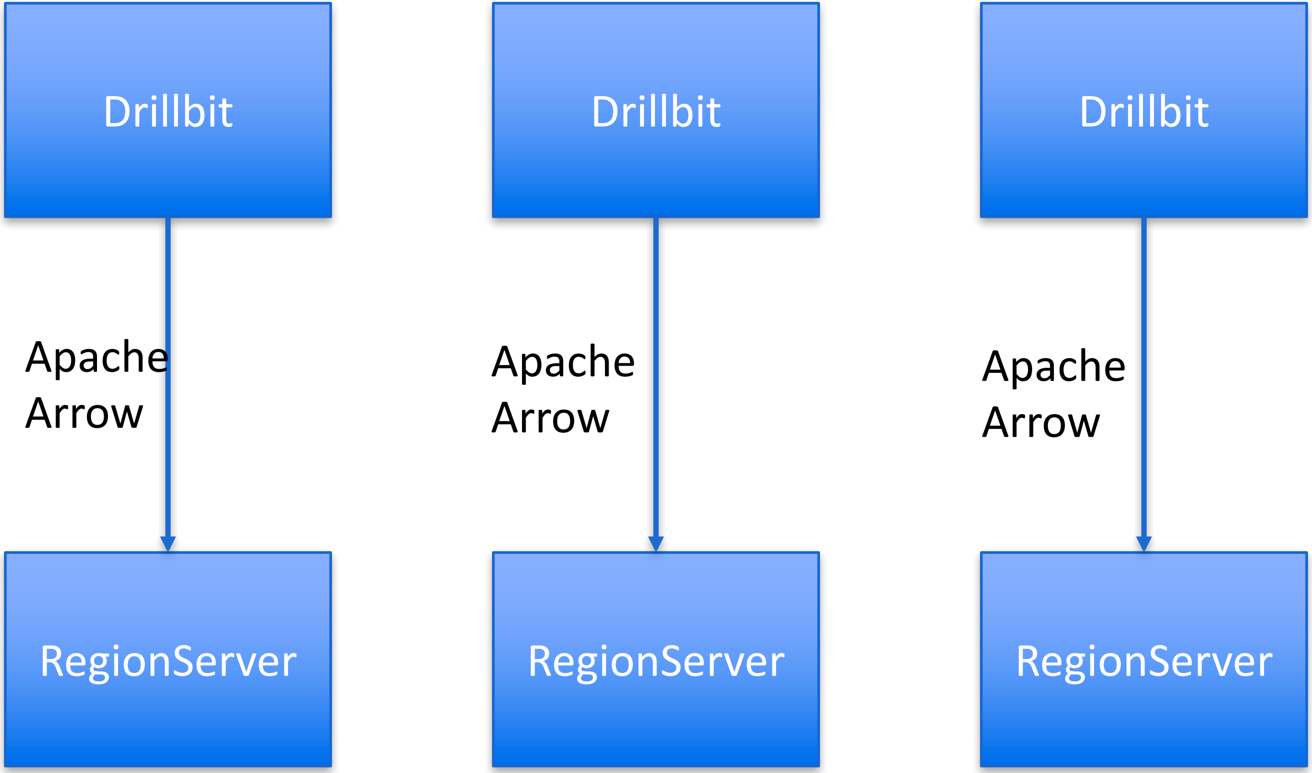
Query Execution: Where does Phoenix Fit?



Phase 1: Parallel Reading (using JDBC)



Phase 2: Parallel Reading (using Arrow)



Drillix: Current Status

- Initial Code available in Github
- Supports basic reading
- Next:
  - First release
  - Support for cross system join planning
  - Apache Arrow integration

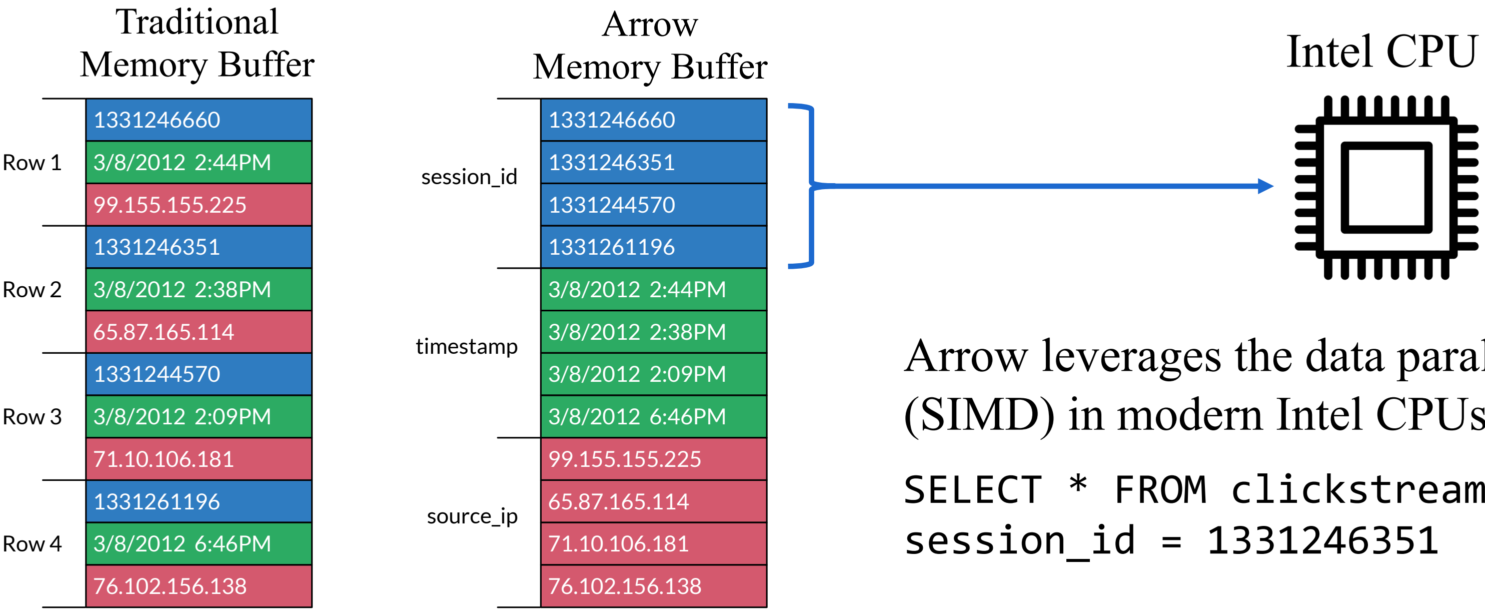
How: Arrow

Introducing Apache Arrow

- New Top-level Apache Software Foundation project
  - Announced Feb 17, 2016
- Focused on Columnar In-Memory Analytics
  1. 10-100x speedup on many workloads
  2. Common data layer enables companies to choose best of breed systems
  3. Designed to work with any programming language
  4. Support for both relational and complex data as-is
- Developers from 13 major open source projects involved
  - A significant % of the world's data will be processed through Arrow!

Performance Advantage of Columnar In-Memory

	session_id	timestamp	source_ip
Row 1	1331246660	3/8/2012 2:44PM	99.155.155.225
Row 2	1331246351	3/8/2012 2:38PM	65.87.165.114
Row 3	1331244570	3/8/2012 2:09PM	71.10.106.181
Row 4	1331261196	3/8/2012 6:46PM	76.102.156.138



Traditional Memory Buffer

Row 1
1331246660
3/8/2012 2:44PM
99.155.155.225

Row 2
1331246351
3/8/2012 2:38PM
65.87.165.114

Row 3
1331244570
3/8/2012 2:09PM
71.10.106.181

Row 4
1331261196
3/8/2012 6:46PM
76.102.156.138

Arrow Memory Buffer

session_id
1331246660
1331246351
1331244570
1331261196

timestamp
3/8/2012 2:44PM
3/8/2012 2:38PM
3/8/2012 2:09PM
3/8/2012 6:46PM

source_ip
99.155.155.225
65.87.165.114
71.10.106.181
76.102.156.138

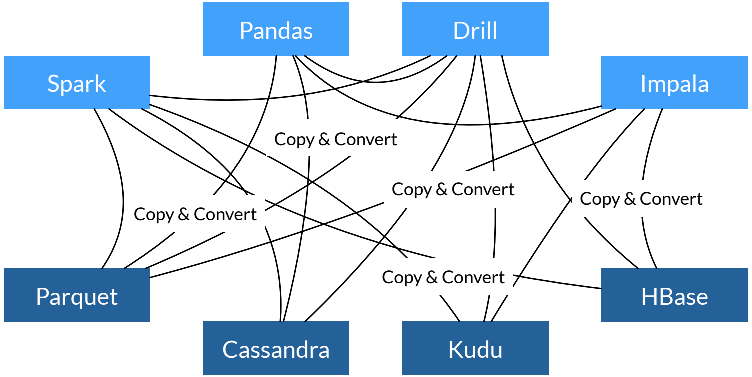
Intel CPU

Arrow leverages the data parallelism (SIMD) in modern Intel CPUs:

```
SELECT * FROM clickstream WHERE
session_id = 1331246351
```

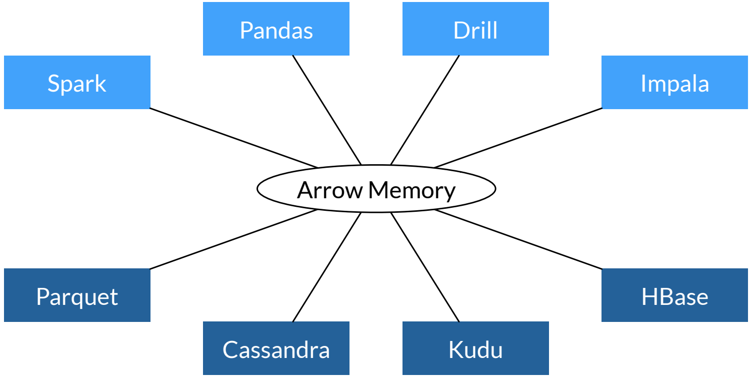
Advantages of a Common Data Layer

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)

Who’s Behind Apache Arrow?

- Creators and/or key developers of 13 major open source Big Data projects
- Employees of:
  - Salesforce, Amazon, Cloudera, Databricks, Datastax, Dremio, Hortonworks, MapR, Twitter

- Calcite
- Cassandra
- Drill
- Hadoop
- HBase
- Ibis
- Impala
- Kudu
- Pandas
- Parquet
- Phoenix
- Spark
- Storm

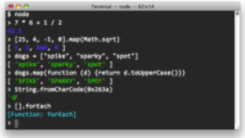
Current Status

- C, C++, Python and Java implementations currently underway
  - Additional languages (eg, R, JavaScript) and projects also expected to adopt Arrow by EOY
- Likely included in Drill, Ibis, Impala, Kudu, Parquet and Spark by EOY

How: Drill

SQL-on-Everything with Apache Drill

CLI



```
$ node
> 7 * 6 + 1 / 2
42.5
> [25, 4, -1, 0].map(Math.sqrt)
[ 5, 2, NaN, 0 ]
> dogs = ["spike", "sparky", "spot"]
[ 'spike', 'sparky', 'spot' ]
> dogs.map(function (d) {return d.toUpperCase()})
[ 'SPIKE', 'SPARKY', 'SPOT' ]
> String.fromCharCode(0x263a)
'☺'
> [].forEach
[Function: forEach]
> 
```

Tableau, Excel, Qlik, …

Web/Custom Applications

REST

Apache Drill

(1-1000 nodes)

NoSQL

HBase
MongoDB
Kudu

Search

Elasticsearch

Files

NAS (NetApp, etc.)
HDFS

IaaS/PaaS

Amazon S3

Relational

Oracle
MySQL
SQL Server

Apache Drill: Open Source Schema-Free SQL Engine

Open Source Apache Project

- Contributors from many companies including Dremio, MapR and Hortonworks
- 3-year engineering effort, 200K+ lines of code

Innovative Schema-free Engine

- Point-and-query vs. schema-first
- No data loading, schemas or ETL
- Handles complex (eg, JSON) data natively

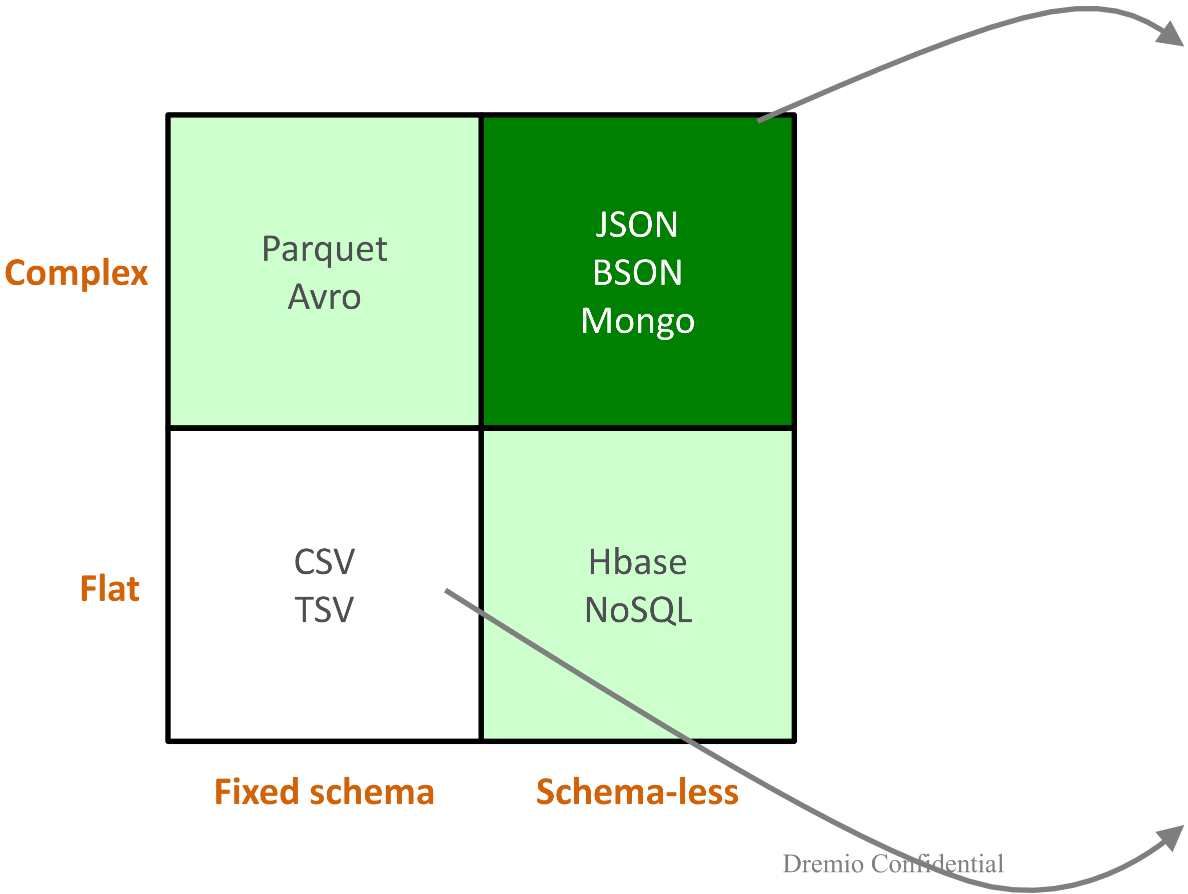
Extreme Scale & Performance

- Scales from one laptop to 1000s of servers
- High performance via columnar execution & dynamic query compilation

Extensible Architecture

- Pluggable high-speed datastore connectors (eg, MongoDB, Amazon S3)
- Custom operators and UDFs

JSON Model, Columnar Speed (with Arrow)



Apache Drill table

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

RDBMS/SQL-on-Hadoop table

Name	Gender	Age
Michael	M	6
Jennifer	F	3

Drill Supports *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-free
- Leverage schema in centralized repository or self-describing data

SCHEMA ON WRITE

SCHEMA BEFORE READ

SCHEMA ON THE FLY

ORACLE
Microsoft SQL Server
IBM DB2
HIVE
APACHE DRILL

Apache Drill is Not Just SQL-on-Hadoop

	Drill	SQL-on-Hadoop (Hive, Impala, etc.)
Use case	Self-service, in-situ, SQL-based analytics	Teradata offload
Deployment model	Standalone or co-located with NoSQL/Hadoop	Hadoop service
User experience	Point-and-query	Ingest data → define schemas → query
Data model	Schema-free JSON (like MongoDB)	Relational (like Postgres)
Data sources	NoSQL, Cloud Storage, Hadoop, SaaS, local files (including multiple instances)	A single Hadoop cluster
Data management	Logical, by IT or end-users (self-service)	Physical, by IT only
1.0 availability	Q2 2015	Q2 2013 or earlier

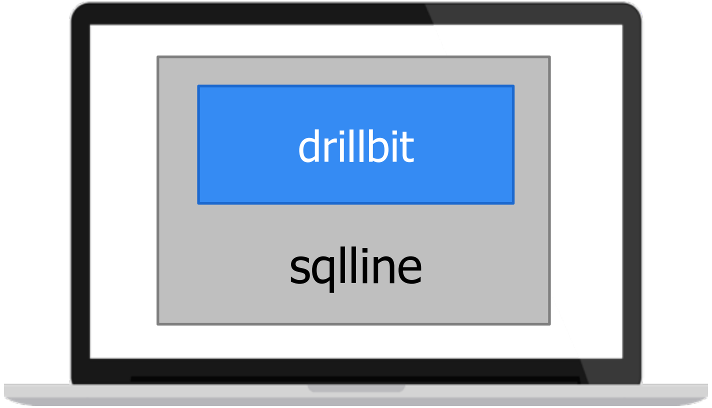
Omni-SQL ("SQL-on-Everything")

> ***Drill: Omni-SQL***
> *Whereas the other engines we're discussing here create a relational database environment on top of Hadoop, Drill instead enables a SQL language interface to data in numerous formats, without requiring a formal schema to be declared. This enables plug-and-play discovery over a huge universe of data without prerequisites and preparation. So while Drill uses SQL, and can connect to Hadoop, calling it SQL-on-Hadoop kind of misses the point. A better name might be SQL-on-Everything, with very low setup requirements.*

Andrew Brust, ZDNet

Deployment Modes

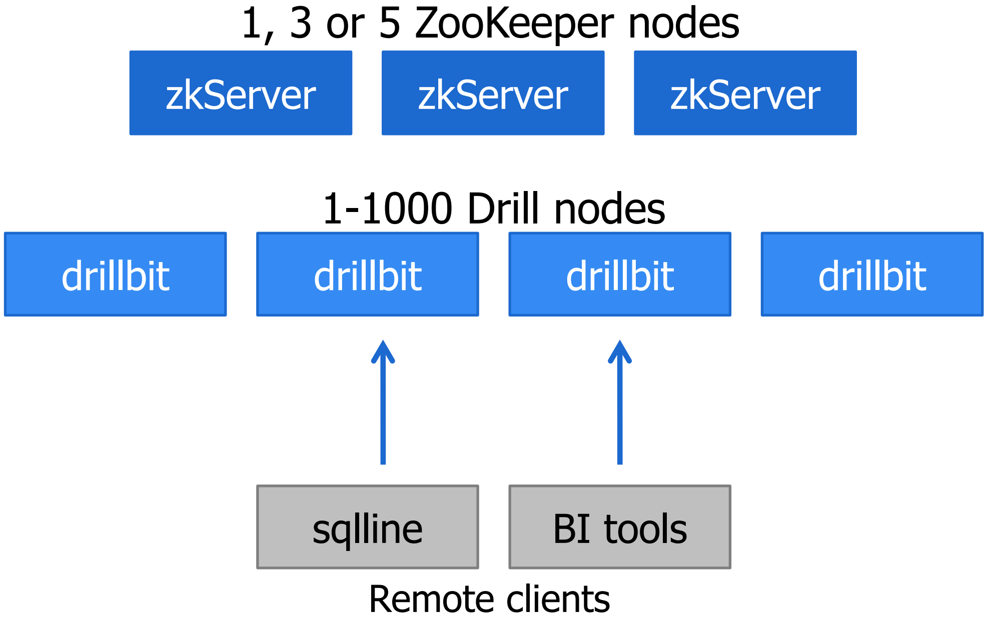
Embedded Mode



drillbit

sqlline

Distributed Mode (aka Drill Cluster)



1, 3 or 5 ZooKeeper nodes

zkServer zkServer zkServer

1-1000 Drill nodes

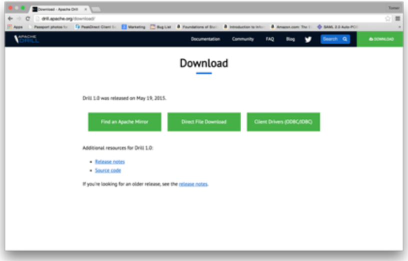
drillbit drillbit drillbit drillbit

sqlline BI tools

Remote clients

Instant Drill

From the browser: http://drill.apache.org/download



Download

Drill 1.0 was released on May 19, 2015.

Additional resources for Drill 1.0:

- Release notes
- Source code

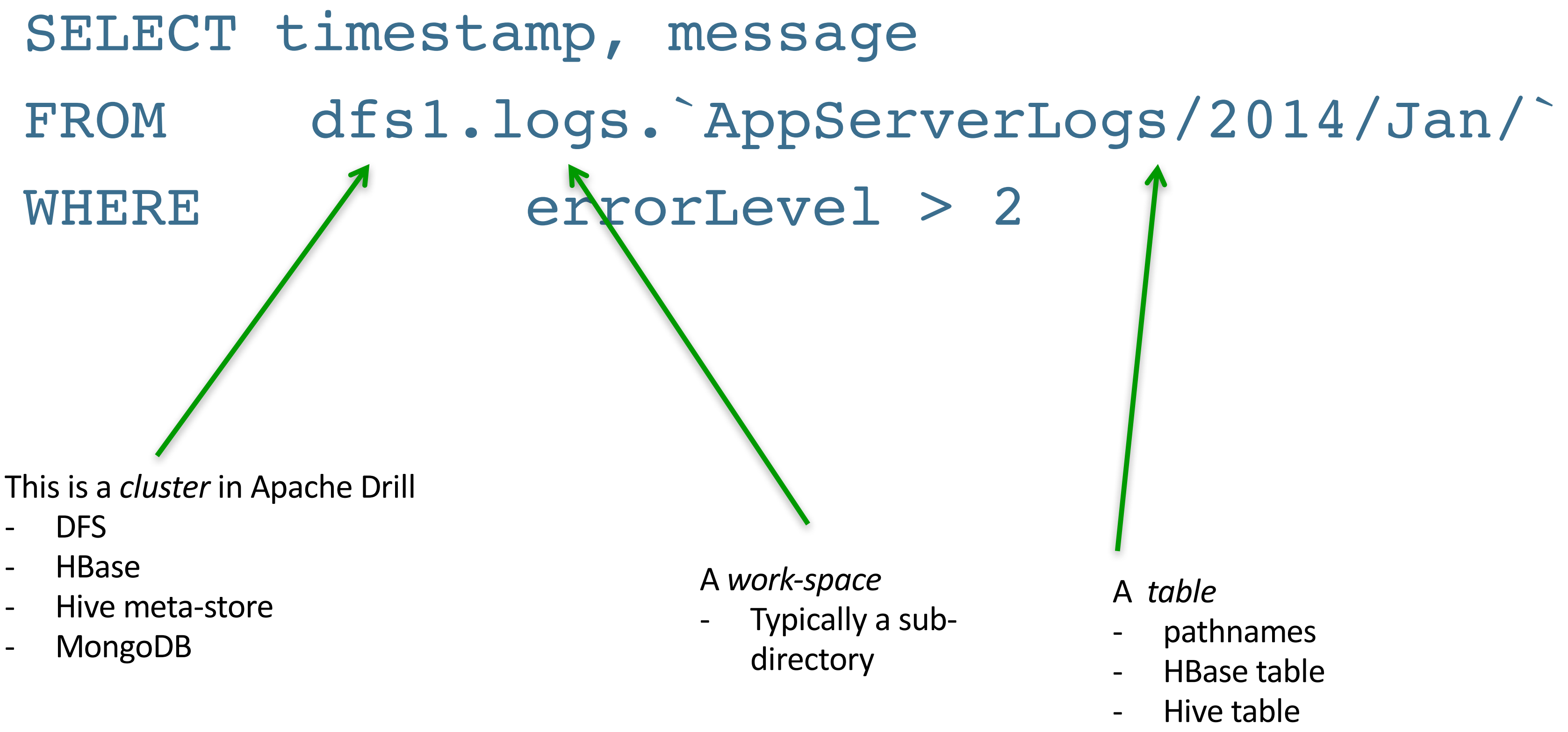
If you're looking for an older release, see the release notes.

From the command line:

```
$ curl -L http://www.dremio.com/drill-latest.tgz | tar xz
```

(Also make sure you have JDK 1.7+ installed…)

Everything looks like a table…



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

This is a *cluster* in Apache Drill

- DFS
- HBase
- Hive meta-store
- MongoDB

A *work-space*

- Typically a sub-directory

A *table*

- pathnames
- HBase table
- Hive table

Query a file or a directory tree

```
-- Queries on files
   SELECT errorLevel, COUNT(*)
   FROM dfs.logs.`AppServerLogs/2014/Jan/log.json`
   GROUP  BY errorLevel;


-- Queries on entire directory tree
   use dfs.logs;

   SELECT errorLevel, COUNT(*)
   FROM AppServerLogs
   GROUP  BY errorLevel;
```

Directories are implicit partitions

```
sales
 ├──── 2014
 |     ├──── q1
 |     ├──── q2
 |     ├──── q3
 |     └── q4
 └── 2015
       └── q1
```

```
-- Direct
   SELECT dir0, SUM(amount)
   FROM sales
   GROUP BY dir1

-- View
   CREATE VIEW sales_q2 as
   SELECT dir0 as year,
   amount
   FROM sales
   WHERE dir1 = ‘q2’
```

Interpret binary data on the fly

CONVERT_FROM and CONVERT_TO allows access to common Hadoop encodings:

- Boolean, byte, byte_be, tinyint, tinyint_be, smallint, smallint_be, int, int_be, bigint, bigint_be, float, double, int_hadoopv, bigint_hadoopv, date_epoch_be, date_epoch, time_epoch_be, time_epoch, utf8, utf16, json

- E.g. CONVERT_FROM(mydata, ‘int_hadoopv’) => Internal INT format
- E.g. CONVERT_FROM(mydata, ‘JSON’) => UTF8 JSON to Internal complex object

Access complex data with SQL

```
{
  name: “Jacques”,
  wife: “Sarah”,
  address: {
    city: “Santa Clara”,
    state: “CA”
  },
   dogs: [
    {name: “William”, age: 19}
    {name: “Kate”, age: 10}
    {name: “Philip”, age: 3}
  ]
}
```

Reference subfields using dot notation

```
SELECT t.address.state FROM t
```

Reference array items using json index

```
SELECT t.dogs[0] FROM t
```

Mix Both

```
SELECT t.dogs[0].name FROM t
```

Make Complex Data Relational using FLATTEN

```
SELECT name, FLATTEN(dogs) FROM t
```

```
{name: “Jacques”, dog: {name: “William”, age: 19}}
{name: “Jacques”, dog: {name: “Kate”, age: 10}}
{name: “Jacques”, dog: {name: “Philip”, age: 3}}
```

=> 3 records, repeating value for non-flattened columns

Grab fields you didn’t know existed

- In many JSON datasets, map keys are data, not metadata

```
{
  sessionid: 1234,
  pages: {
    "/home/": {time: 15, scroll: 70},
    "/store/": {time: 30, scroll: 50},
    "/return/": {time: 45, scroll: 100},
    "/support/": {time: 30, scroll: 10}
}
```

```
SELECT sessionid, count(*) from (
     SELECT sessionid, FLATTEN(KVGEN(pages)) FROM t
) WHERE scroll > 50 and time >30
GROUP BY sessionid
HAVING count(*) >= 1
```

Extract embedded JSON

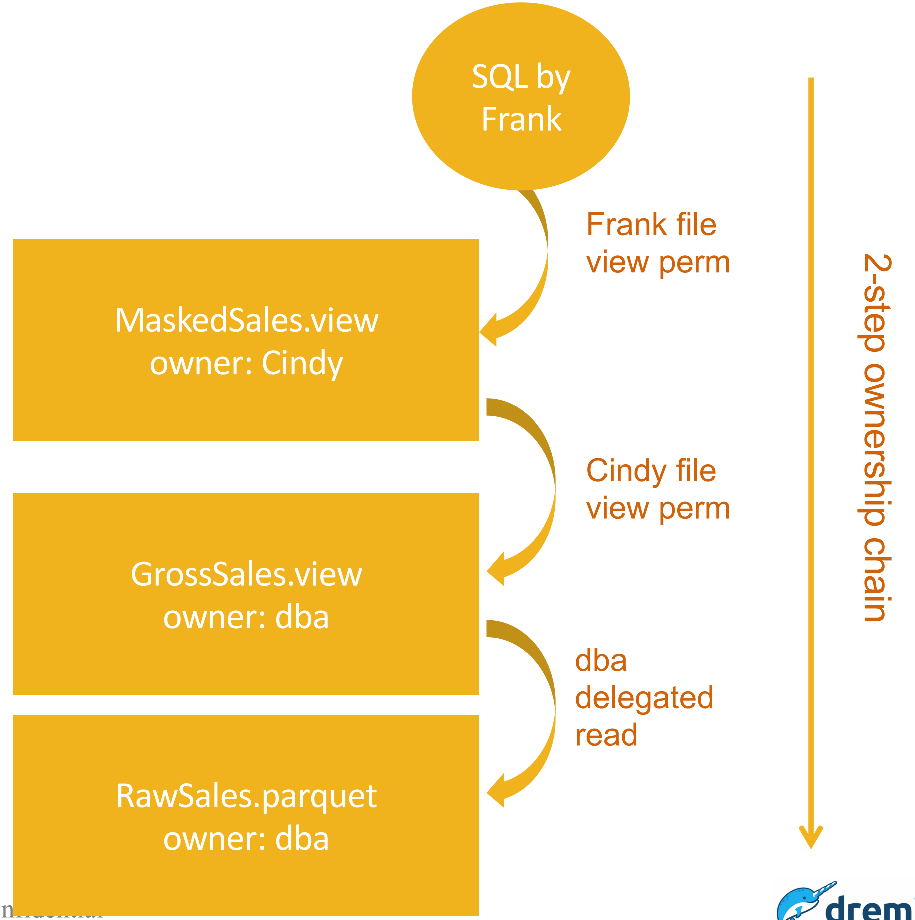
```
-- embedded JSON value inside column donutjson inside column-family cf1  of an hbase table donuts
SELECT
    d.name, COUNT(d.fillings)
FROM (
    SELECT convert_from(cf1.donutjson, JSON) as d
    FROM hbase.donuts);
```

Advanced: Analyze Drill’s JSON profiles

```
SELECT
  t3.majorFragmentId,
  t3.opProfile.operatorType opType,
  sum(t3.opProfile.peakLocalMemoryAllocated) aggPeakMemoryAcrossAllMinorFragments
FROM
  (SELECT
       t2.majorFragmentId,
       flatten(t2.minorFragProfile.operatorProfile) opProfile
   FROM
       (SELECT
             t1.majorFragment.majorFragmentId majorFragmentId,
             flatten(t1.majorFragment.minorFragmentProfile) minorFragProfile
         FROM
             (SELECT flatten(fragmentProfile) as majorFragment from `profile.json` t0) t1
         ) t2
   ) t3
WHERE t3.opProfile.operatorType = 6
```

Secure your data without an extra service

- Drill Views
- Ownership chaining with configurable delegation TTL
- Leverages existing HDFS ACLs
- Complete security solution without additional services or software



Technical Nitty Gritty

Core Drill Architectural Goals

- Go fast when you don’t know anything
  - And do “the right thing”
- Go faster when you do know things

Drill goes fast

First Read JSON (cpu bound)

- ~5TB semi-relational dataset on 30 nodes
- Encoded in Extended JSON
  - $date, $numberLong, etc
- Raw Execution:
  - Group by aggregation on ~80% of data: 140s, >900mb/s/node
  - 6 way join on 100% of data: 180s, >875mb/s/node

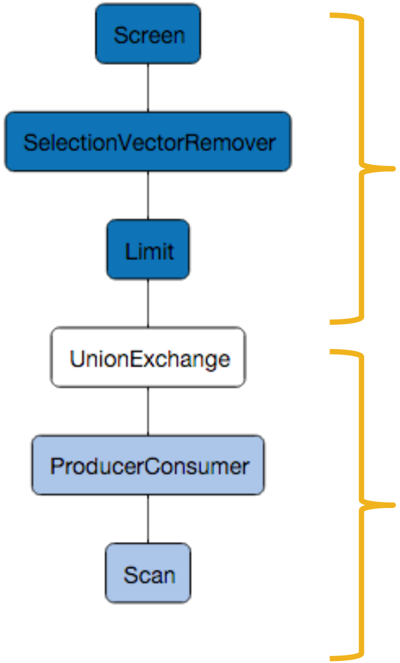
Columnar Encoded Formats (e.g. Parquet)

- Even faster

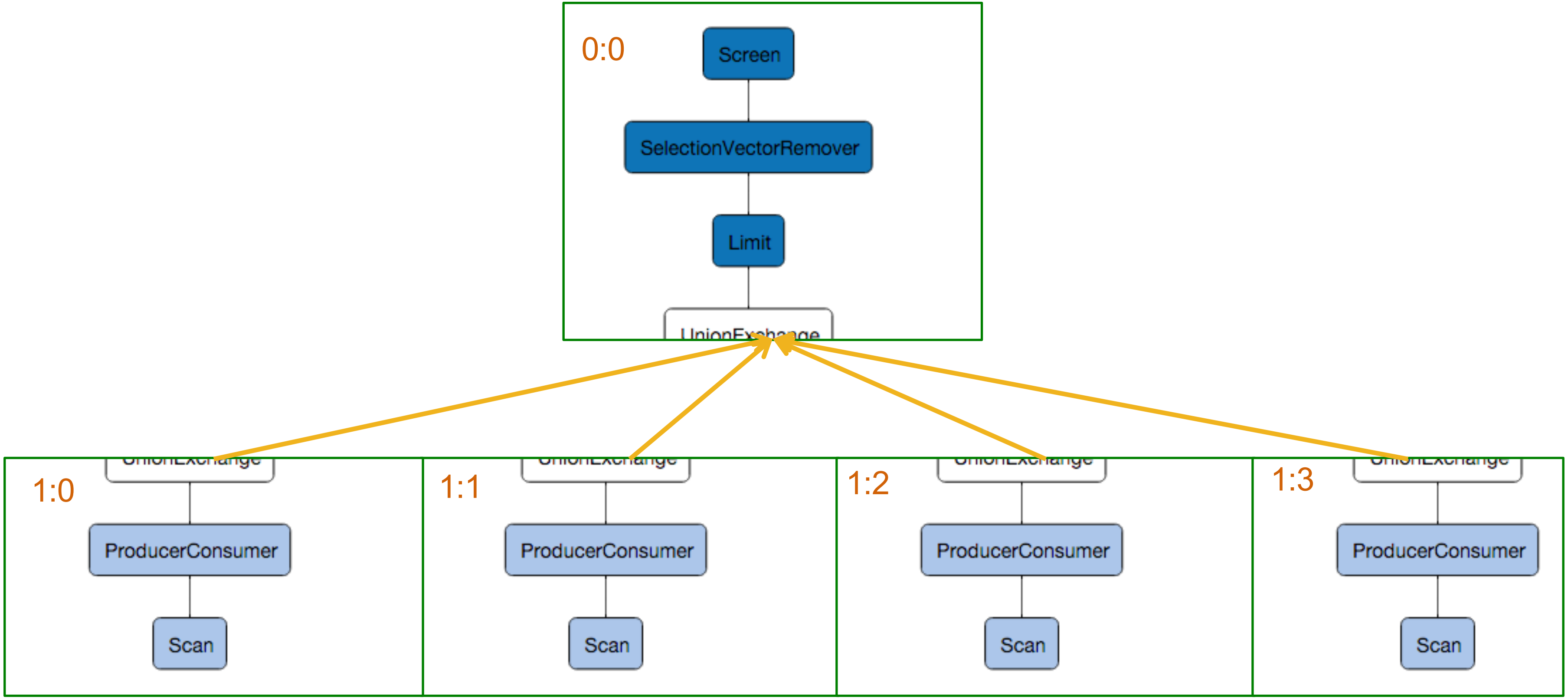
An optimistic, pipelined purpose-built DAG engine

- Three-Level DAG
- Major Fragments (phases)
- Minor Fragments (threads)
- Operators (in-thread operations)

```
> explain plan for select * from customer limit 5;
…
00-00    Screen
00-01      SelectionVectorRemover
00-02        Limit(fetch=[5])
00-03          UnionExchange
01-01            ProducerConsumer
01-02              Scan(groupscan=[ParquetGroupScan […
```

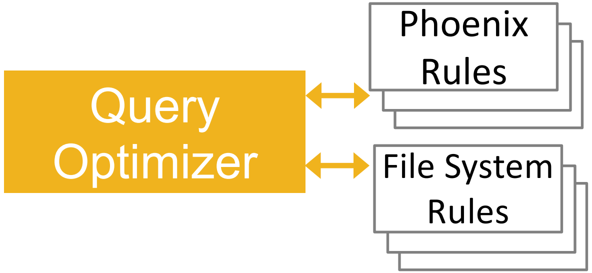


Each phase (MajorFragment) gets parallelized (MinorFragment)



Drill plans with a parallel-aware extensible cost-based optimizer

- Volcano-inspired cost based optimizer
  - CPU, IO, memory, network (data locality)
- Pluggable rules per storage subsystem
- Exchanges provide parallelization:
  - Hash, Ordered, Broadcast and Merging
- Join and aggregation planning
  - Merge Join vs Hash Join
  - Partition-based join vs Broadcast-based join
  - Streaming Aggregation vs Hash Aggregation
- Relies heavily on the awesome Apache Calcite project



Drill uses statistics where possible

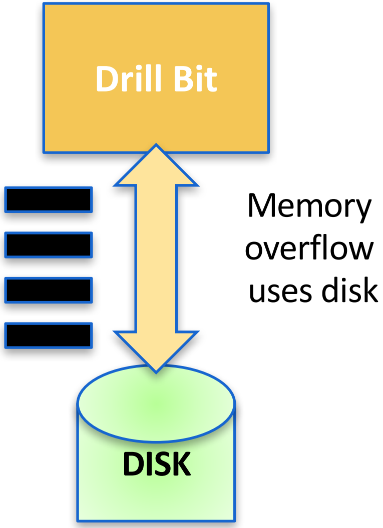
- Different Storage systems have different capabilities
- All: Basic estimation of row count
  - Allows the first 70% of critical optimization decisions for most Hadoop workloads
  - Either exact numbers or approximations based on best effort
- Some file formats
  - Better than basic stats in file, such as Parquet
    - Substantial effort to make certain formats ‘nearly native’
  - Drill adding support for extended attributes for enhanced file-held statistics

Drill moves data quickly

- Highly Optimized Native Drill Readers:
  - JSON, Vectorized Parquet, Text/CSV, Avro
  - Also works with all Hive supported formats
- Drill supports partition pruning
  - Adding direct physical property exposure soon for highly optimized cases
- Drill parallelizes to maximum level format allows
  - Also balances data locality and maximum parallelization
- Bespoke Asynchronous Zero-Copy RPC Layer
  - Built specifically for Drill's internal data format

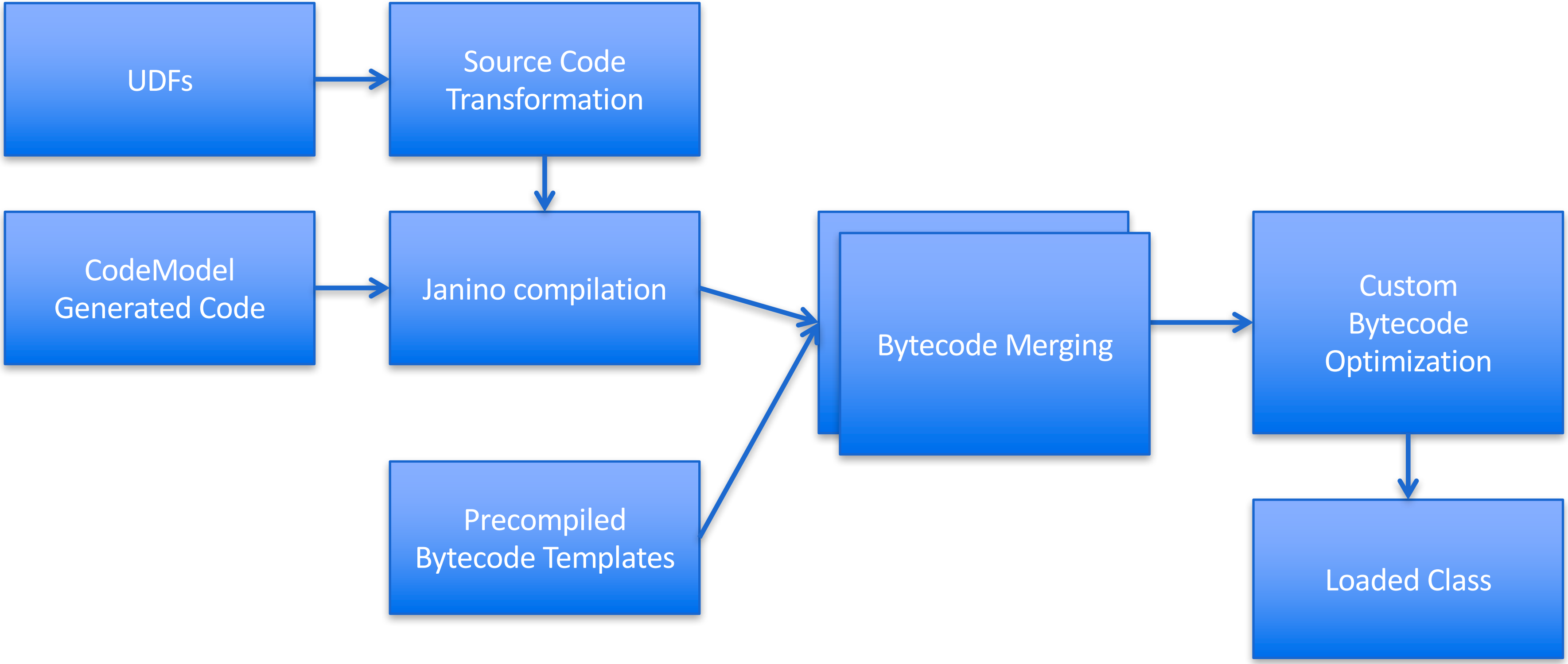
Using Arrow & Record Batches: Drill’s in-memory columnar work units

- Random access: sort without copy or restructuring
- Fully specified in memory shredded complex data structures
- Remove serialization or copy overhead at node boundaries
- Spool to disk as necessary
- Interact with data in Java or C without copy or overhead



Memory overflow uses disk

Runtime compilation pattern


UDFs
Source Code Transformation
CodeModel Generated Code
Janino compilation
Precompiled Bytecode Templates
Bytecode Merging
Custom Bytecode Optimization
Loaded Class

Advanced compilation techniques

- Optimization based on observation and assembly
- Drill does a number of pre-machine-code-compilation optimizations to ensure efficient execution
- Some examples:
  - Removal of type and bounds checking
  - Direct micro pointers for in-record-batch references
  - Little endian data formats
  - Bytecode-level scalar replacement

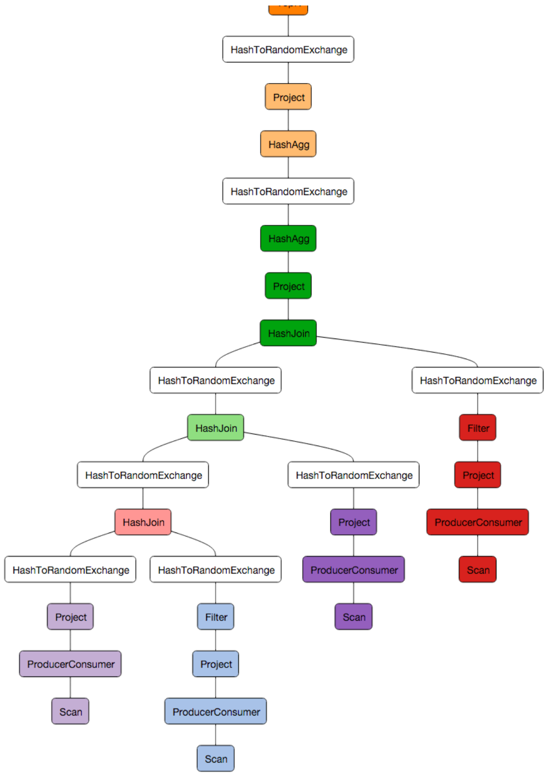
High-speed Storage and UDF APIs

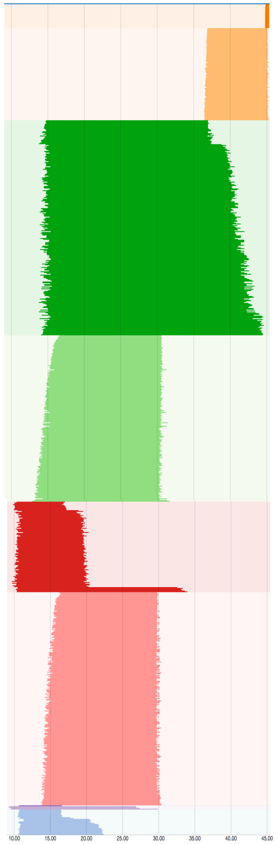
- All built-in functions in Drill are simply UDFs
- UDF interface works directly with
  - compilation engine
  - bytecode rewriting algorithms
- Storage Interface is a columnar vectorized transfer interface
  - Allows storage system to determine best transformation approach
  - Adding support for deferred materialization
  - Support for multiple target languages with minimal overhead

Drill provides advanced query telemetry

- What happened during query for all three levels of DAG execution
- Each profile is stored as JSON file for easy review, sharing and backup (at end of query execution)
- Profiles can be analyzed using Drill, allows:
  - easy longitudinal analysis of workload
  - multi-tenancy performance analysis
  - impact of configuration changes to benchmark workloads

A color-coded visual layout and Gant timing chart is provided





Questions

- arrow.apache.org / @ApacheArrow
- drill.apache.org / @ApacheDrill
- dremio.com / @DremioHQ
  - – exciting information soon