cloudera

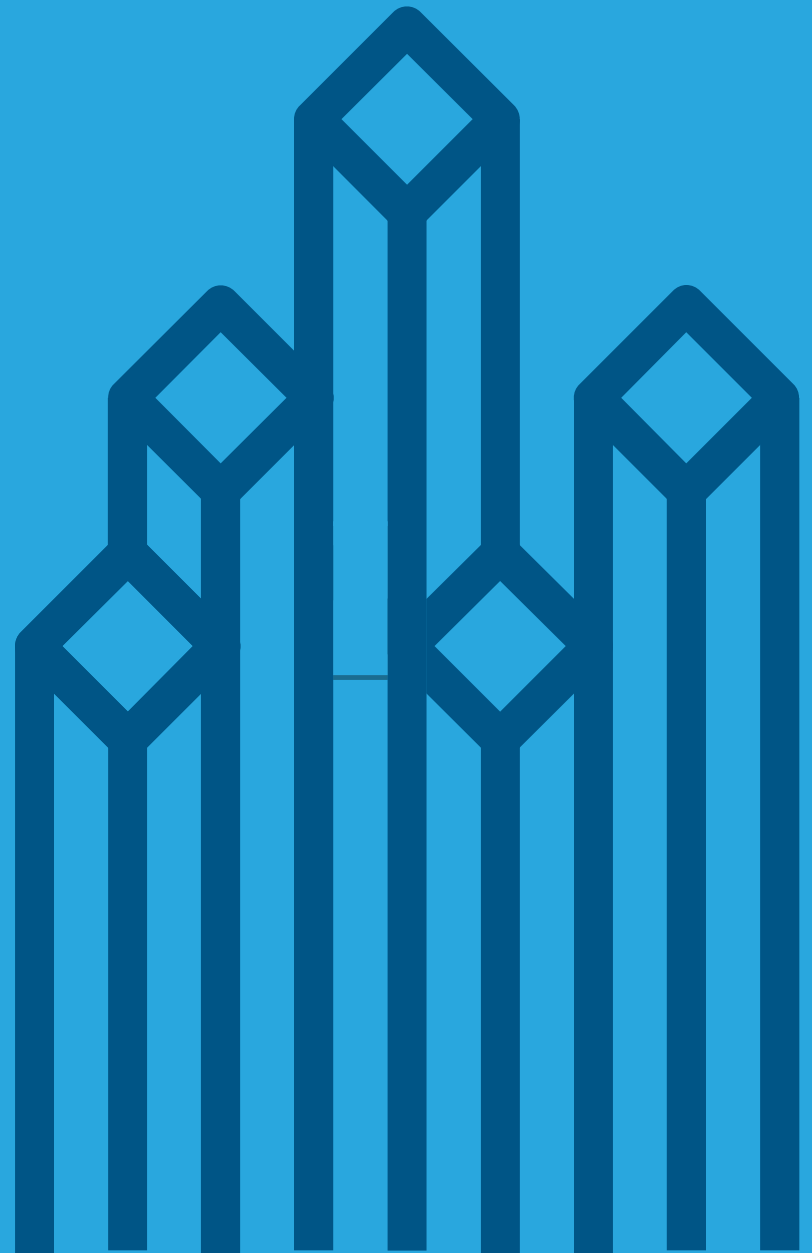
Apache Kudu*: Fast Analytics on Fast Data

Todd Lipcon (Kudu team lead) – todd@cloudera.com

@tlipcon

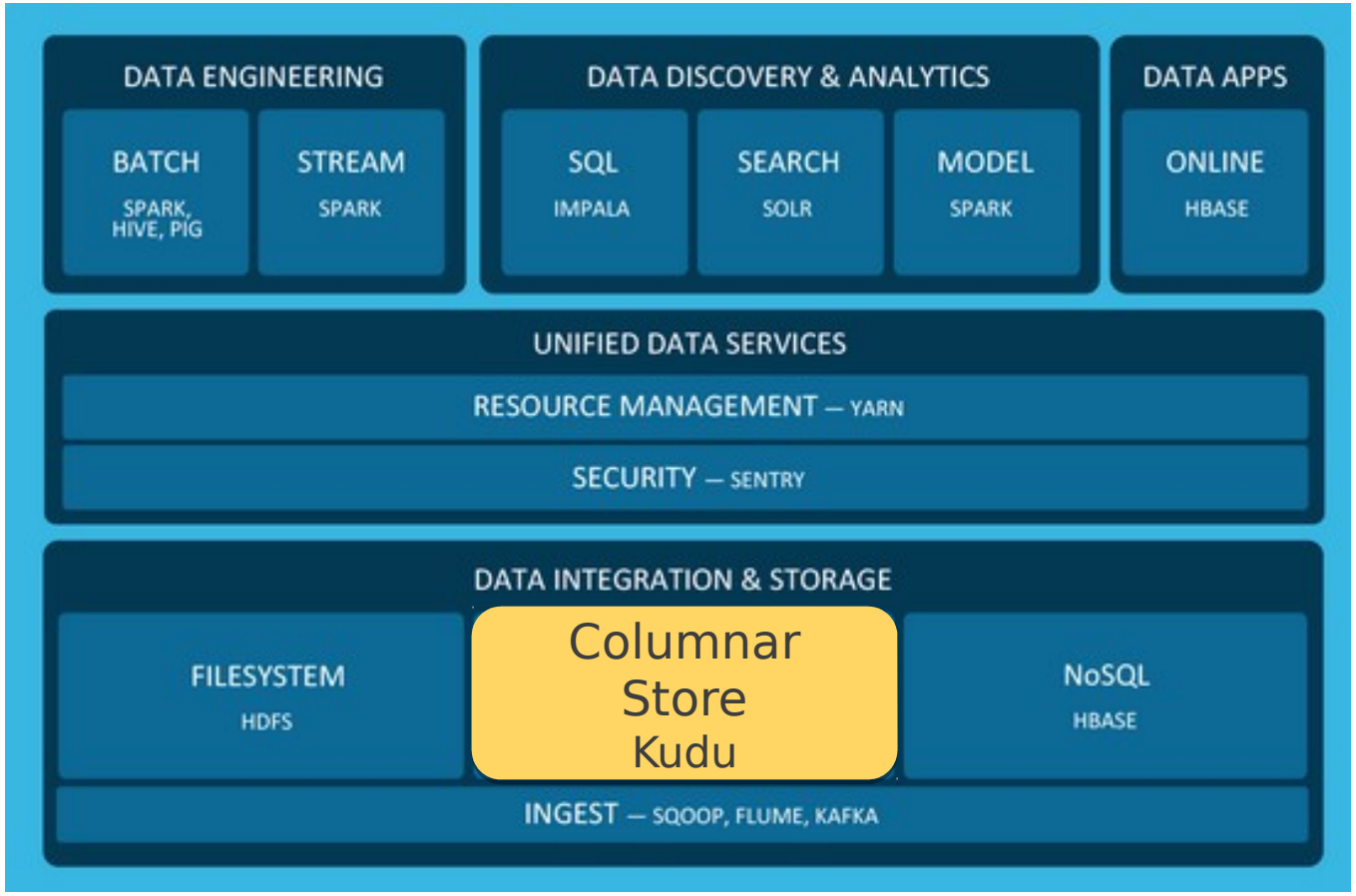
Tweet about this talk: @ApacheKudu or #kudu

* Incubating at the Apache Software



Apache Kudu

Storage for Fast Analytics on Fast Data



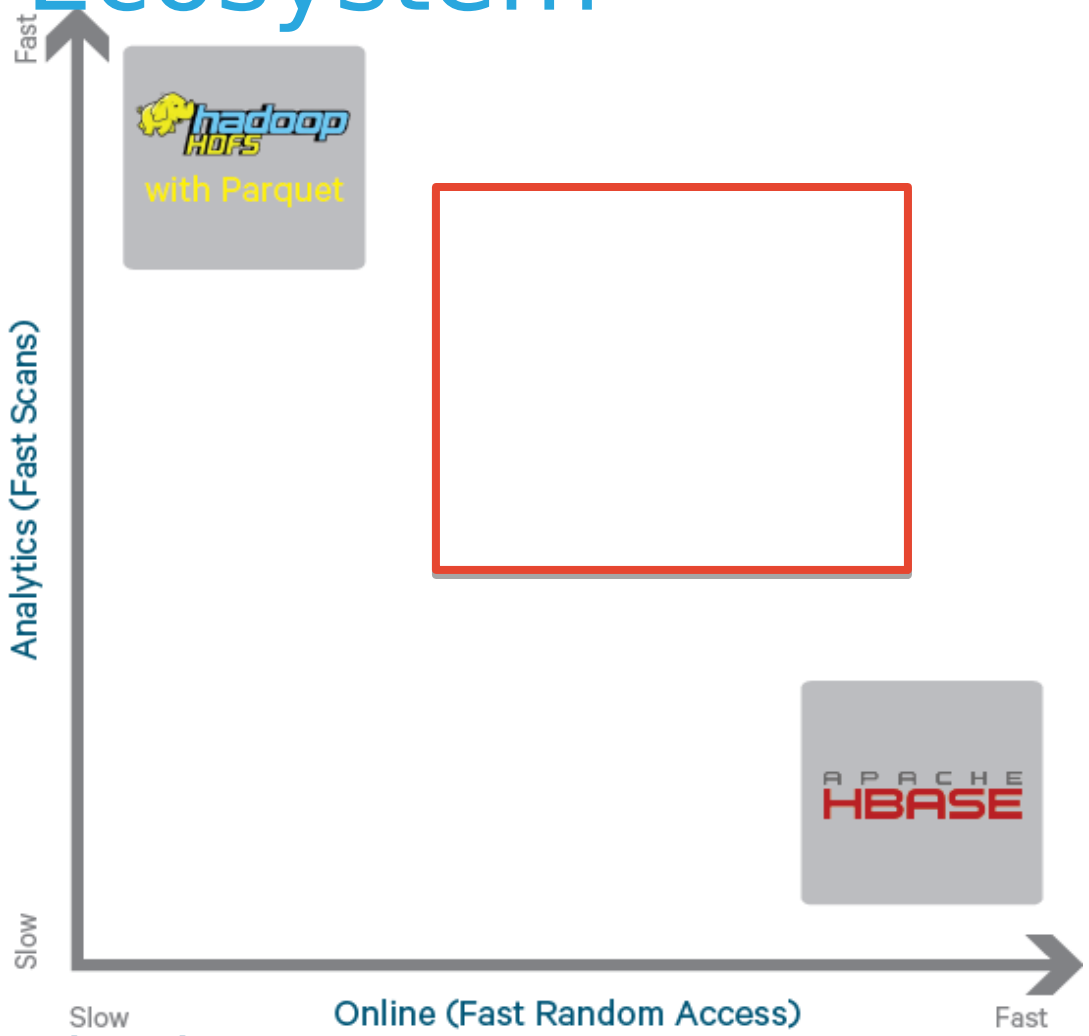
- New updatable column store for Hadoop
- Apache-licensed open source
- Beta now available

cloudera

Why Kudu?

cloudera

Current Storage Landscape in Hadoop Ecosystem



HDFS (GFS) excels at:

- Batch ingest only (eg hourly)
- Efficiently scanning large amounts of data (analytics)

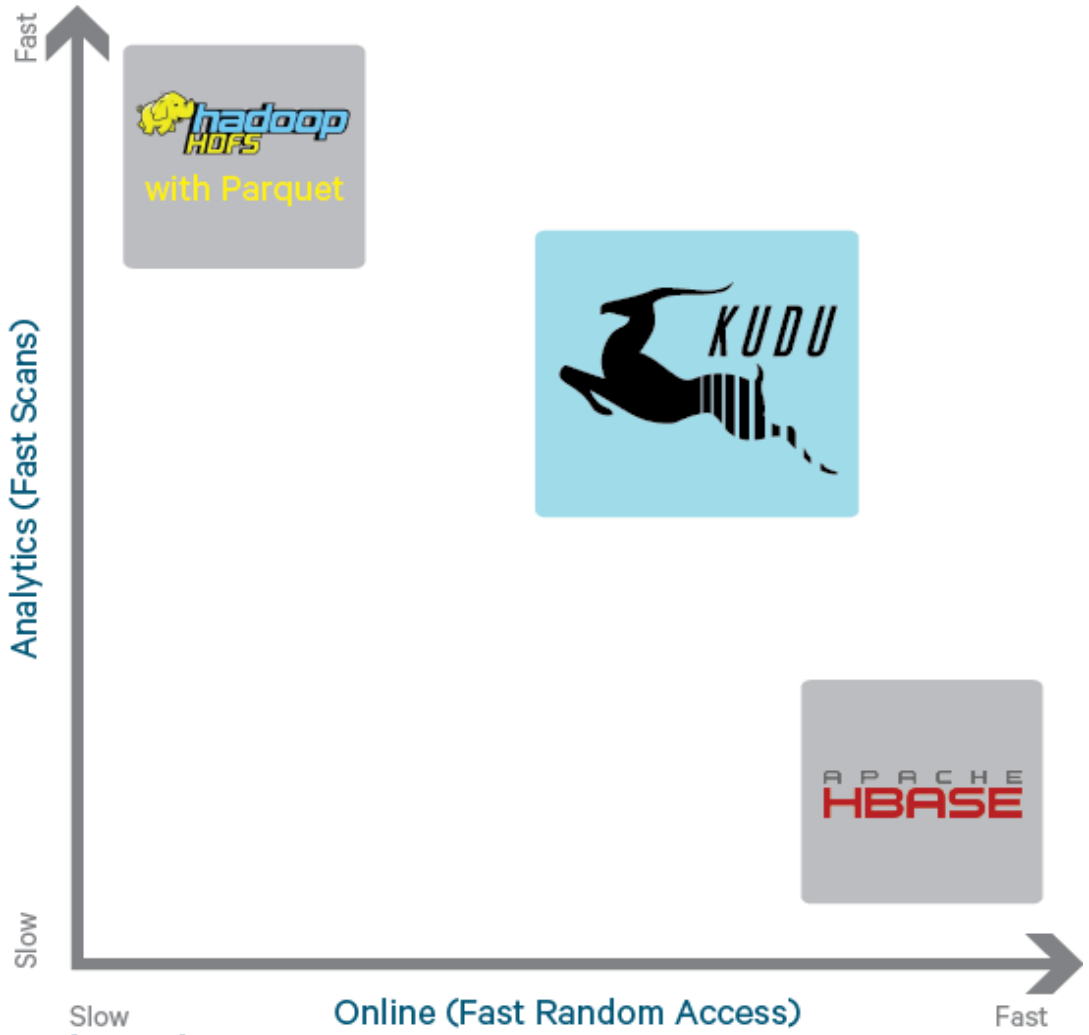
HBase (BigTable) excels at:

- Efficiently finding and writing individual rows
- Making data mutable

Gaps exist when these properties are needed *simultaneously*

cloudera

Kudu Design Goals



- **High throughput** for big scans

  Goal: Within 2x of Parquet

- **Low-latency** for short accesses

  Goal: 1ms read/write on SSD

cloudera

Changing Hardware landscape

- **Spinning disk** -> **solid state storage**
  - **NAND flash**: Up to 450k read 250k write iops, about 2GB/sec read and 1.5GB/sec write throughput, at a price of less than $3/GB and dropping
  - **3D XPoint memory** (1000x faster than NAND, cheaper than RAM)

- **RAM** is cheaper and more abundant:
  - 64->128->256GB over last few years

- ***Takeaway***: The **next bottleneck is CPU**, and current storage systems weren't designed with CPU efficiency in mind.

What’s Kudu?

Scalable and fast tabular storage

- **Scalable**
  - Tested up to 275 nodes (~3PB cluster)
  - Designed to scale to **1000s of nodes, tens of PBs**
- **Fast**
  - **Millions** of read/write operations per second across cluster
  - **Multiple GB/second** read throughput per node
- **Tabular**
  - **SQL-like** schema**: finite number** of **typed** columns (unlike HBase/Cassandra)
  - **Fast ALTER TABLE**
  - **"NoSQL"** APIs: Java/C++/Python **or SQL** (Impala/Spark/etc)

Use cases and architectures

Kudu Use Cases

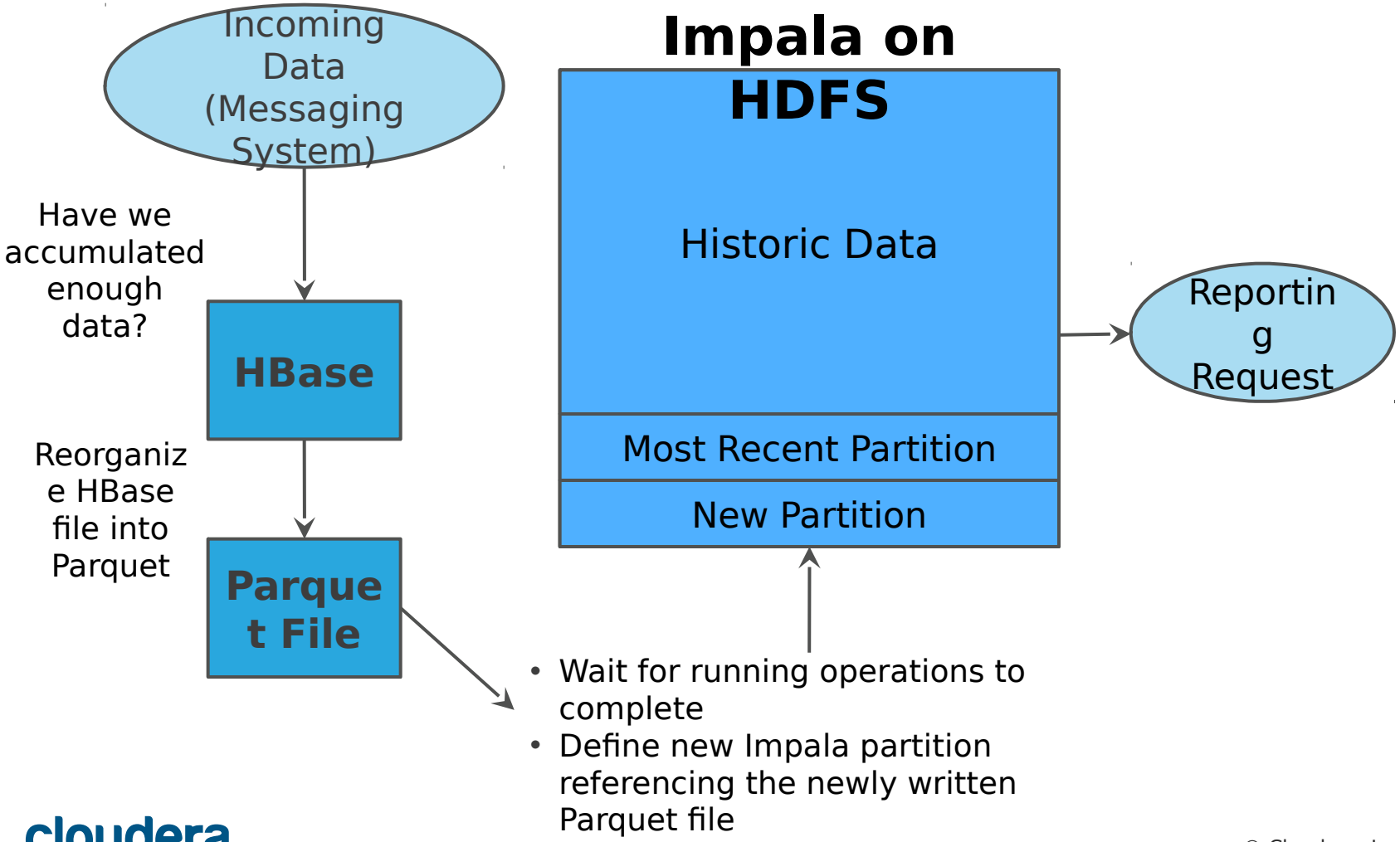
Kudu is best for use cases requiring a simultaneous combination of sequential and random reads and writes

- **Time Series**
  - Examples: Stream market data; fraud detection & prevention; network monitoring
  - Workload: Insert, updates, scans, lookups

- **Online Reporting**
  - Examples: ODS
  - Workload: Inserts, updates, scans, lookups

Real-Time Analytics in Hadoop Today

Fraud Detection in the Real World = Storage Complexity

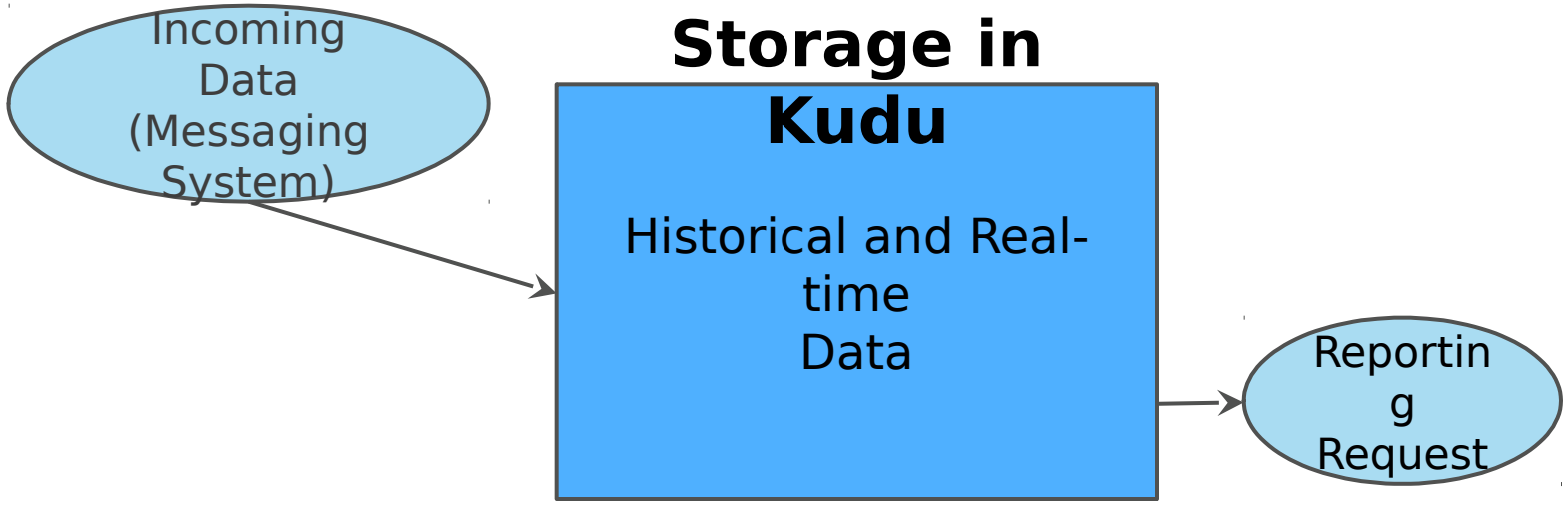


Considerations:

- How do I handle failure during this process?
- How often do I reorganize data streaming in into a format appropriate for reporting?
- When reporting, how do I see data that has not yet been reorganized?
- How do I ensure that important jobs aren't interrupted by maintenance?

cloudera

Real-Time Analytics in Hadoop with Kudu



Improvements:

- **One system** to operate
- **No cron jobs or background processes**
- **Handle late arrivals or data corrections with ease**
- **New data available immediately for analytics or operations**

Xiaomi use case

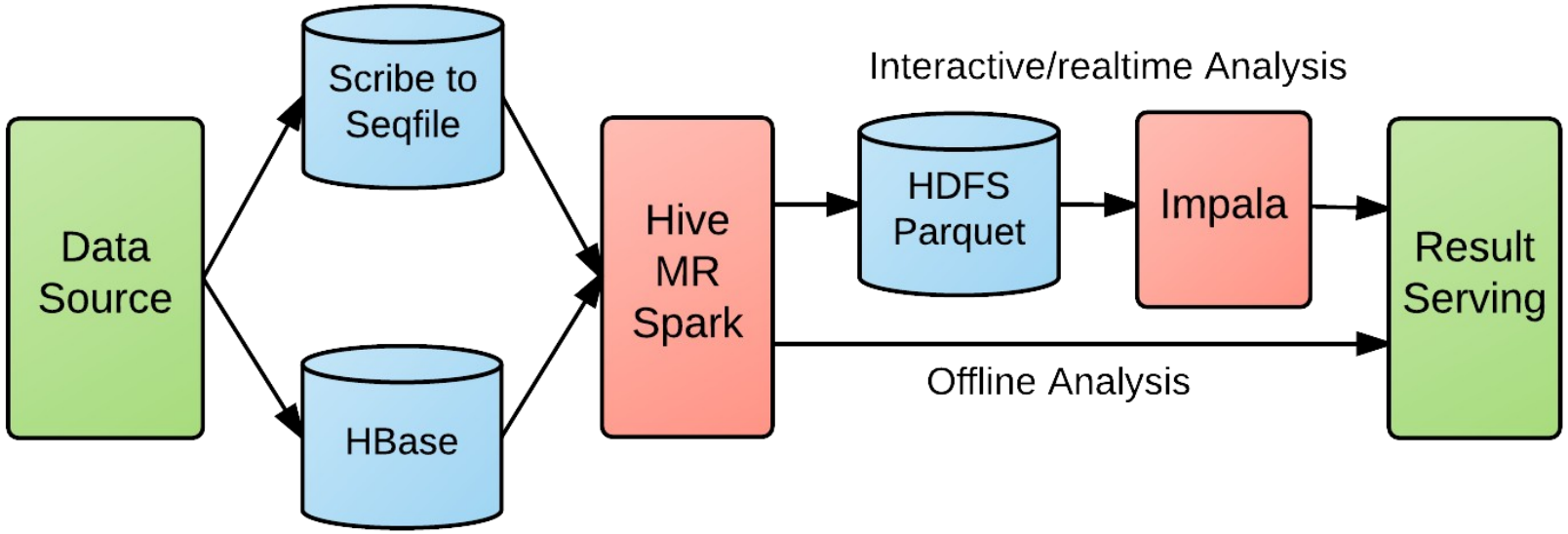
- World's 4th largest smart-phone maker (most popular in China)
- Gather important RPC tracing events from mobile app and backend service.
- Service monitoring & troubleshooting tool.

- **High write throughput**
  - >20 Billion records/day and growing
- **Query latest data and quick response**
  - Identify and resolve issues quickly
- **Can search for individual records**
  - Easy for troubleshooting

cloudera

Xiaomi Big Data Analytics Pipeline

Before Kudu

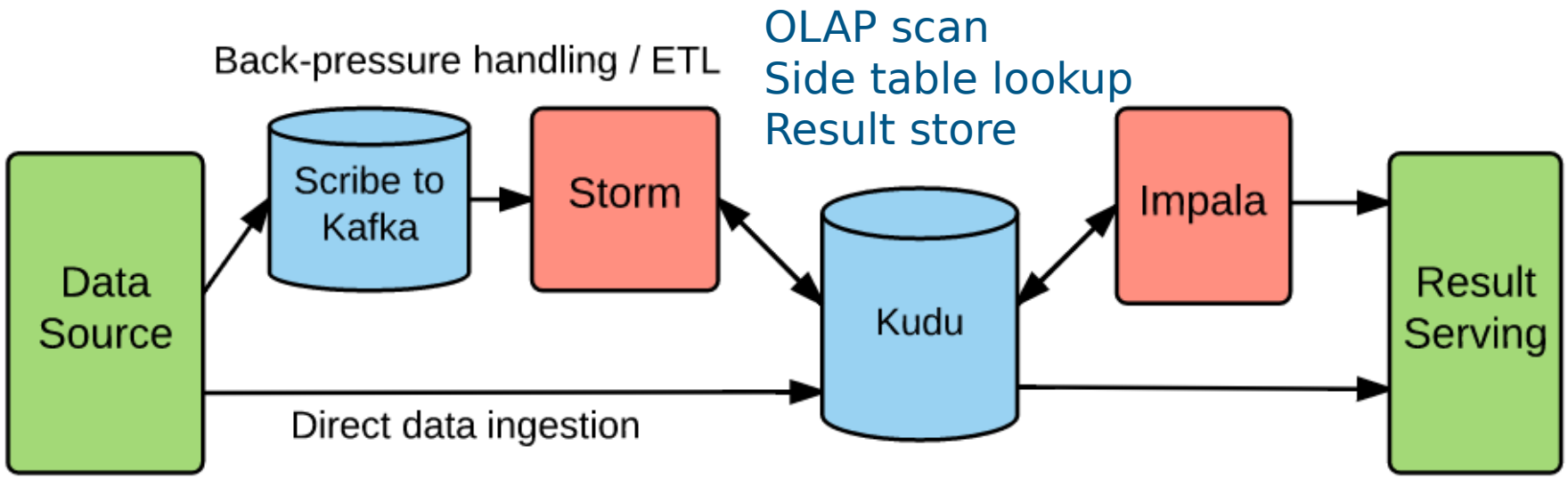


- **Long pipeline**
  high latency(1 hour ~ 1 day), data conversion pains
- **No ordering**
  Log arrival(storage) order not exactly logical order

cloudera

Xiaomi Big Data Analysis Pipeline

Simplified With Kudu



- ETL Pipeline(0~10s latency)
  Apps that need to prevent backpressure or require ETL
- Direct Pipeline(no latency)
  Apps that don’t require ETL and no backpressure issues

cloudera

How it works

Replication and fault tolerance

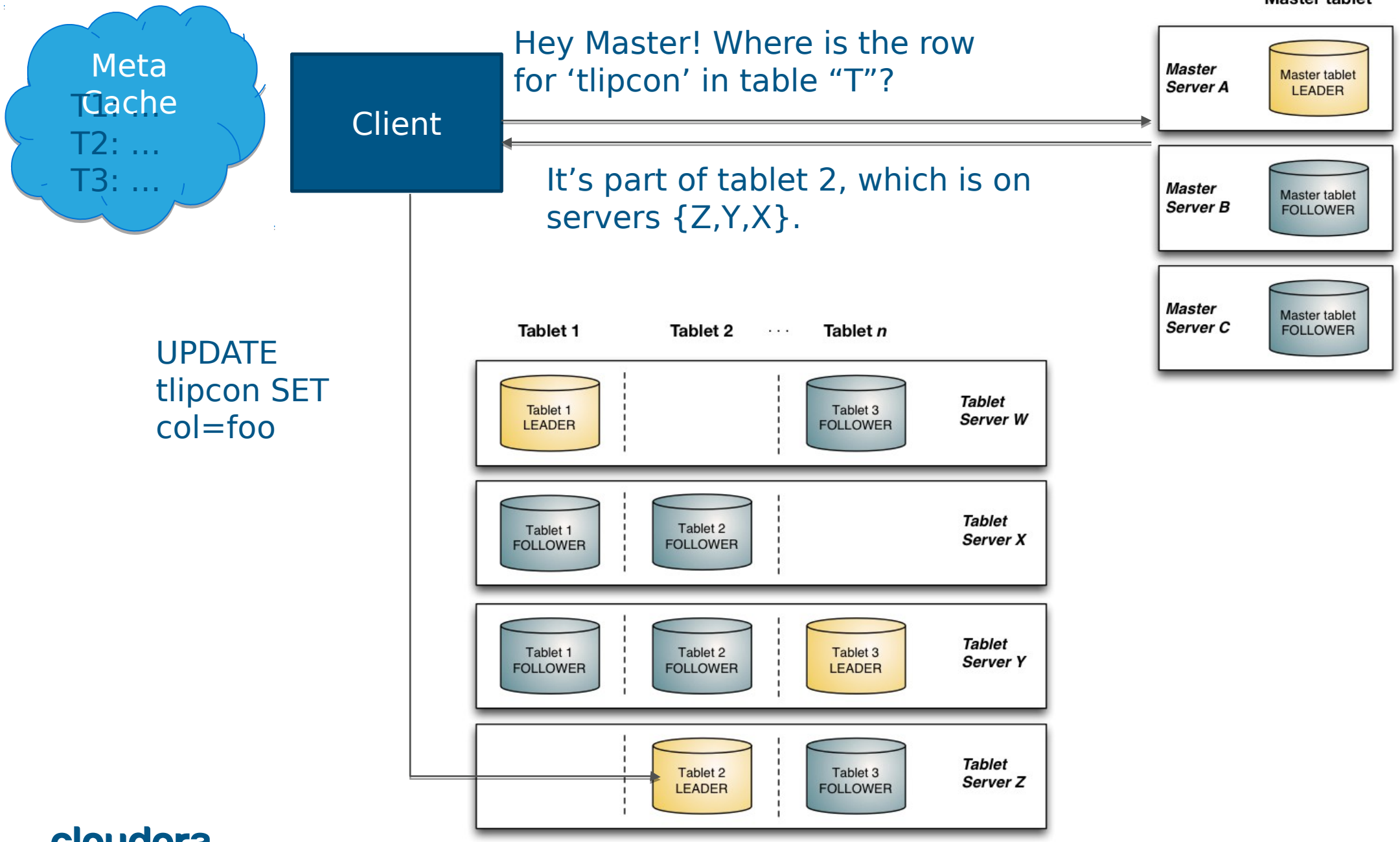
cloudera

Tables, Tablets, and Tablet Servers

- Table is **horizontally partitioned into *tablets***
  - *Range* or *hash* partitioning
  - `PRIMARY KEY (host, metric, timestamp) DISTRIBUTE BY HASH(timestamp) INTO 100 BUCKETS`
- Each tablet has N **replicas** (3 or 5), with **Raft consensus**
  - Automatic **fault tolerance**
  - MTTR: ~5 seconds
- **Tablet servers** host tablets on local disk drives

Metadata and the Master

- **Replicated master**
  - Acts as a tablet directory
  - Acts as a catalog (which tables exist, etc)
  - Acts as a load balancer (tracks TS liveness, re-replicates under-replicated tablets)
- **Not a bottleneck**
  - super fast in-memory lookups



cloudera

How it works

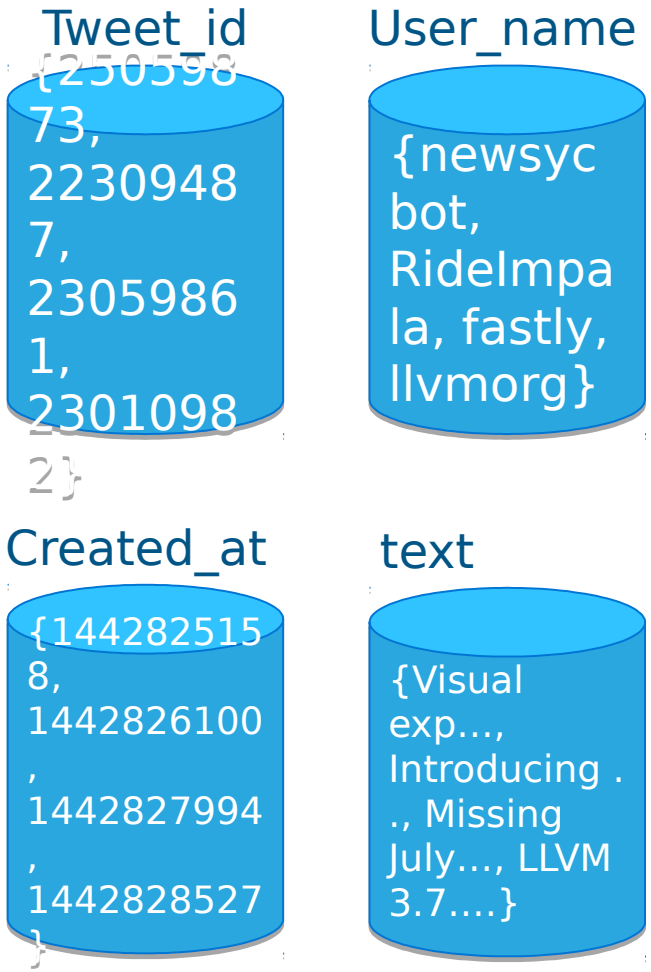
Columnar storage

cloudera

Columnar storage

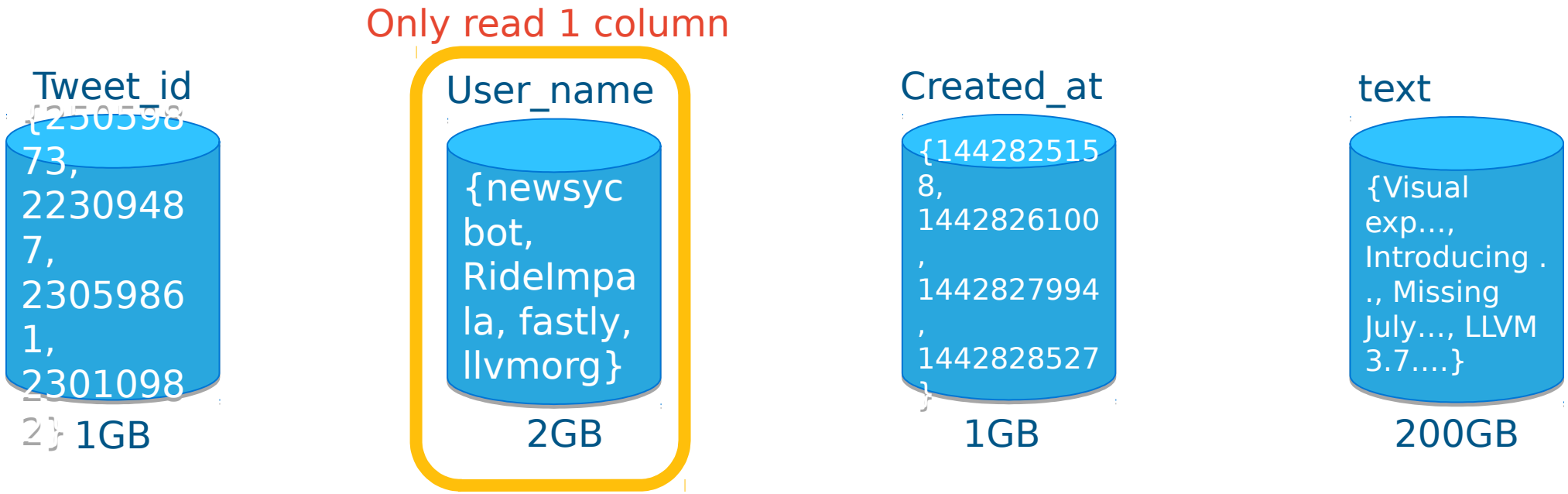
Twitter Firehose Table

tweet_id	user_name	created_at	text
INT64	STRING	TIMESTAMP	STRING
23059873	newsycbot	1442825158	Visual Explanation of the Raft Consensus Algorithm http://bit.ly/1DOUac0 (cmts http://bit.ly/1HKmjfc)
22309487	RideImpala	1442826100	Introducing the Ibis project: for the Python experience at Hadoop Scale
23059861	fastly	1442827994	Missed July's SF @papers_we_love? You can now watch @el_bhs talk about @google's globally-distributed database: http://fastly.us/1eVz8MM
23010982	llvmorg	1442828527	LLVM 3.7 is out! Get it while it's HOT! http://llvm.org/releases/download.html#3.7.0



cloudera

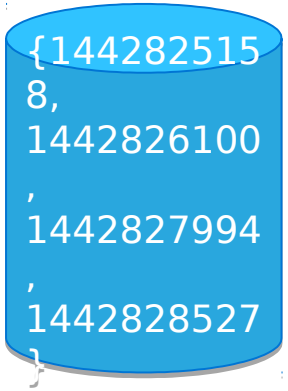
Columnar storage



LECT COUNT(*) FROM tweets WHERE user_name = ‘newsycbot’;

Columnar compression

Created_at



Created_at	Diff(created_at)
1442825158	n/a
1442826100	942
1442827994	1894
1442828527	533
64 bits each	**11 bits each**

- **Many columns can compr to a few bits per row!**
- Especially:
  - Timestamps
  - Time series values
  - Low-cardinality strings
- **Massive space savings an throughput increase!**

cloudera

Integrations

cloudera

Spark DataSource integration (WIP)

```
sqlContext.load("org.kududb.spark",
     Map("kudu.table" -> “foo”,
         "kudu.master" -> “master.example.com”))
   .registerTempTable(“mytable”)
df = sqlContext.sql(
   “select col_a, col_b from mytable “ +
   “where col_c = 123”)
```

vailable in Kudu 0.8.0, but more stable in 0.9.0+

Impala integration

- `CREATE TABLE … DISTRIBUTE BY HASH(col1) INTO 16 BUCKETS AS SELECT … FROM …`
- `INSERT/UPDATE/DELETE`

- Not an Impala user? Community working on other integrations (Hive, Drill, Presto, etc)

cloudera

MapReduce integration

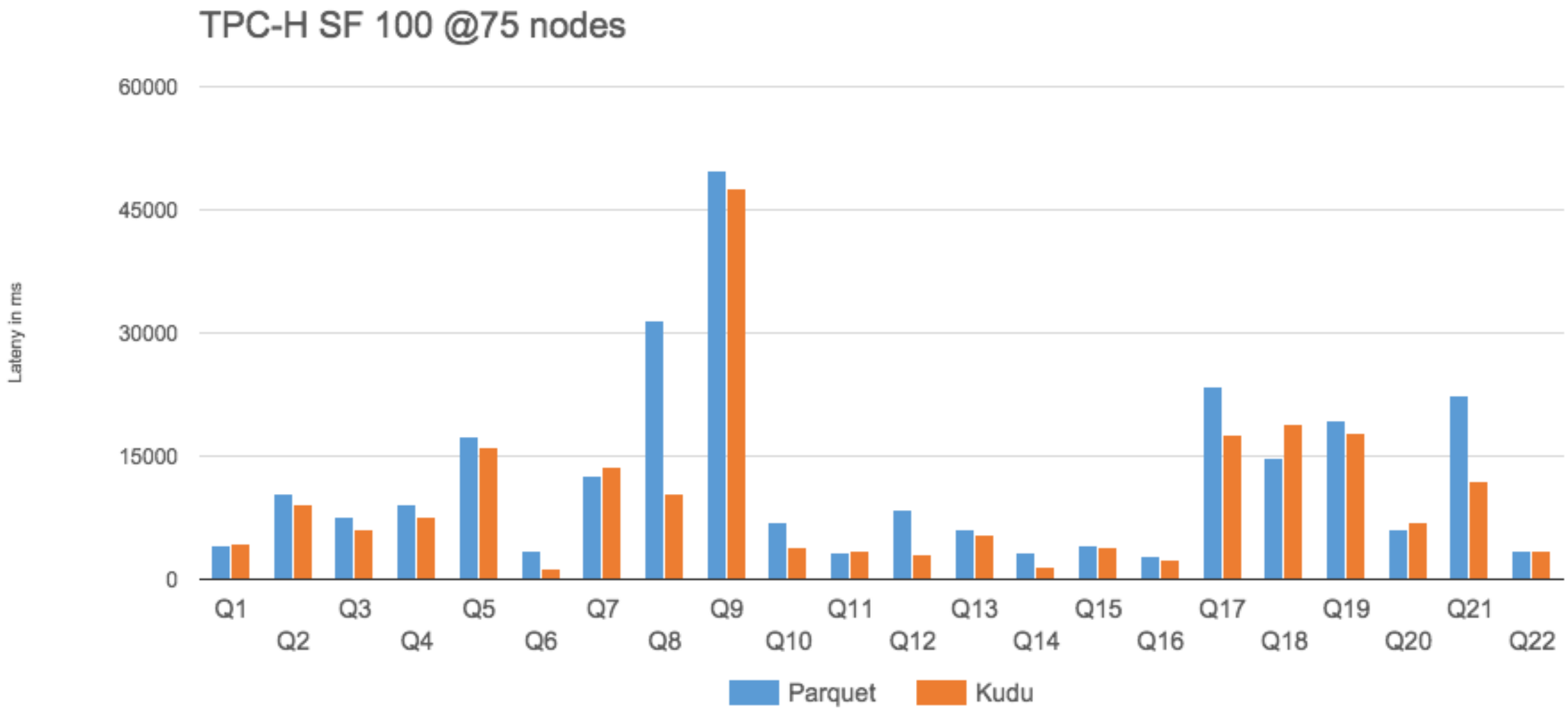
- Multi-framework cluster (MR + HDFS + Kudu on the same disks)
- **KuduTableInputFormat / KuduTableOutputFormat**
  - Support for pushing predicates, column projections, etc

Performance

cloudera

TPC-H (Analytics benchmark)

- 75 server cluster
  - 12 (spinning) disk each, enough RAM to fit dataset
  - TPC-H Scale Factor 100 (100GB)
- Example query:
  - ```
    SELECT n_name, sum(l_extendedprice * (1 - l_discount)) as revenue FROM customer,
    orders, lineitem, supplier, nation, region WHERE c_custkey = o_custkey AND
    l_orderkey = o_orderkey AND l_suppkey = s_suppkey AND c_nationkey = s_nationkey
    AND s_nationkey = n_nationkey AND n_regionkey = r_regionkey AND r_name = 'ASIA'
    AND o_orderdate >= date '1994-01-01' AND o_orderdate < '1995-01-01’ GROUP BY
    n_name ORDER BY revenue desc;
    ```
    ```

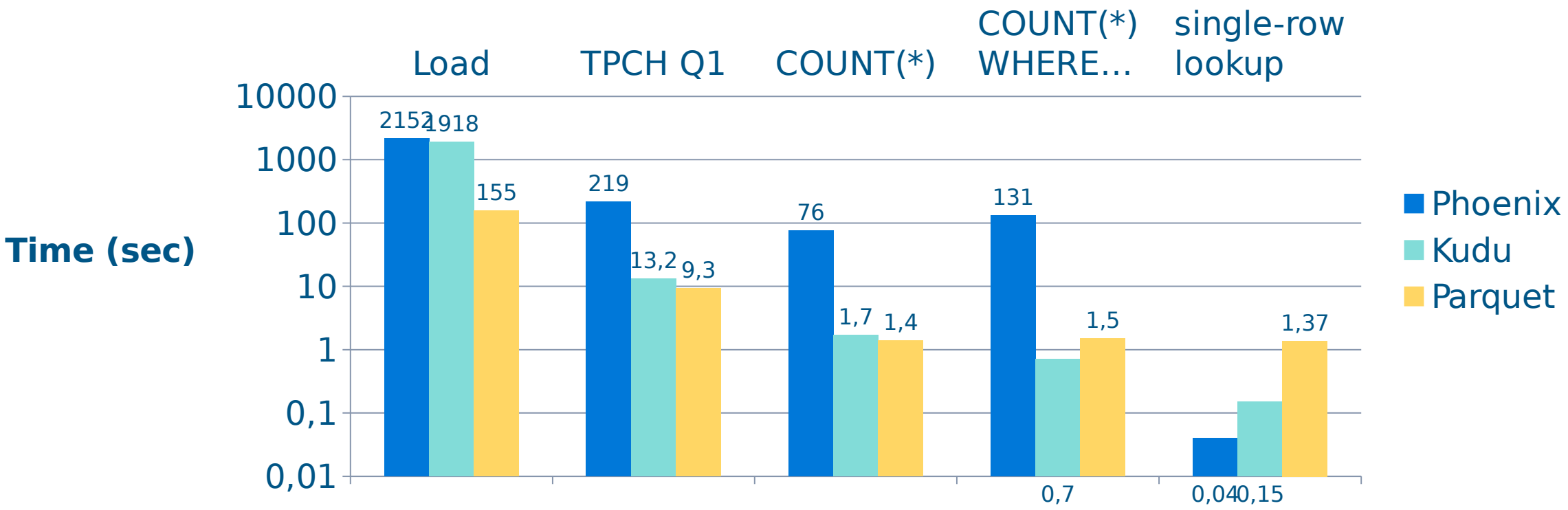


- Kudu outperforms Parquet by 31% (geometric mean) for RAM-resident data

cloudera

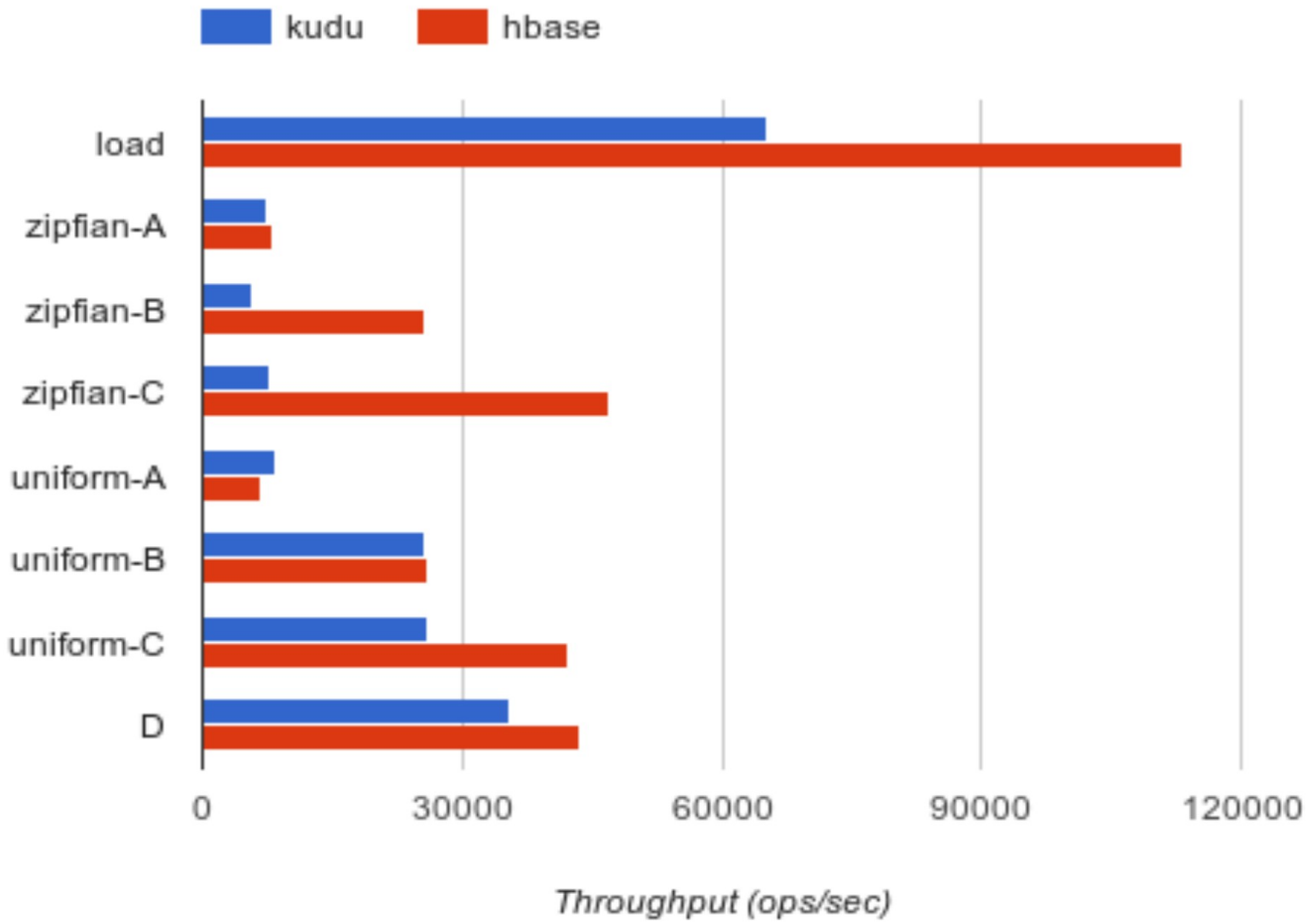
Versus other NoSQL storage

- **Phoenix: SQL layer on HBase**
- 10 node cluster (9 worker, 1 master)
- TPC-H LINEITEM table only (6B rows)



What about NoSQL-style random access? (YCSB)

- **YCSB** 0.5.0-snapshot
- 10 node cluster (9 worker, 1 master)
- 100M row data set
- 10M operations each workload



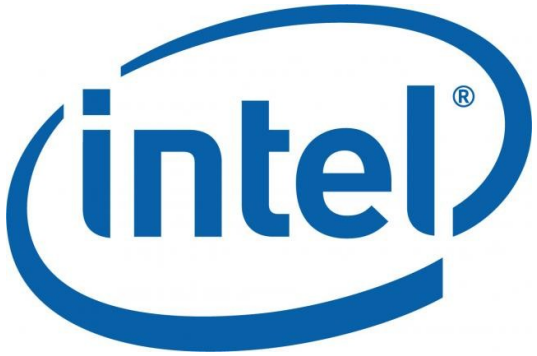
Getting started

cloudera

Project status

- Open source beta released in September
- Latest release 0.9.0 being voted upon now
  - Usable for many applications (Xiaomi in production)
  - Have not experienced unrecoverable data loss, reasonably stable (almost no crashes reported). Users testing up to 200 nodes so far.
  - Still requires some expert assistance, and you'll probably find some bugs
  - Aiming for 1.0 later this summer
- Part of the **Apache Software Foundation** Incubator
  - Community-driven open source process

Apache Kudu (incubating) Community



Getting started as a user

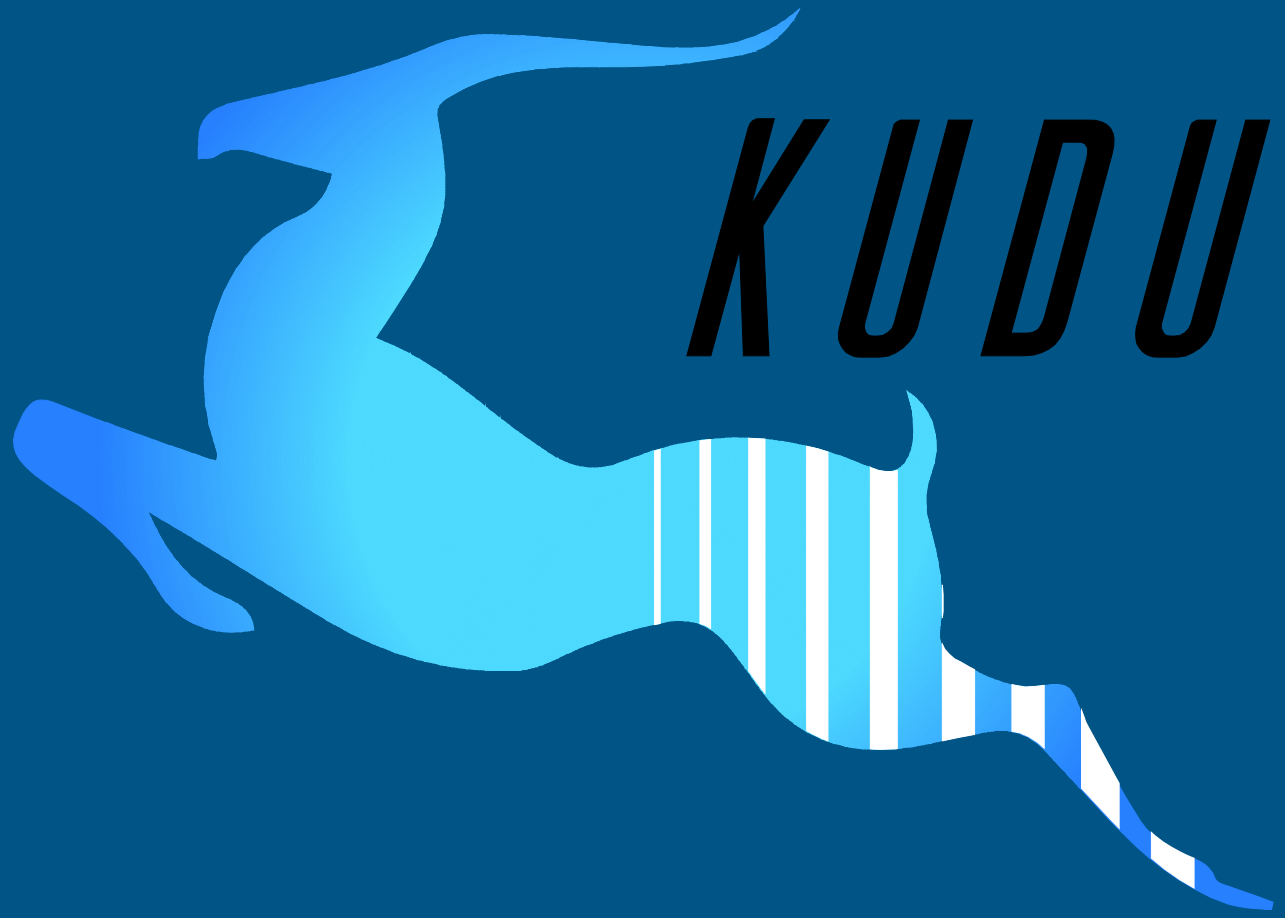
- http://getkudu.io
- user@kudu.incubator.apache.org
- http://getkudu-slack.herokuapp.com/

- Quickstart VM
  - Easiest way to get started
  - Impala and Kudu in an easy-to-install VM
- CSD and Parcels
  - For installation on a Cloudera Manager-managed cluster

Getting started as a developer

- http://github.com/apache/incubator-kudu
- Code reviews: http://gerrit.cloudera.org
- Public JIRA: http://issues.apache.org/jira/browse/KUDU
  - Includes bugs going back to 2013. Come see our dirty laundry!
- Mailing list: dev@kudu.incubator.apache.org

- Apache 2.0 license open source
- Contributions are welcome and encouraged!



http://getkudu.io/

@ApacheKudu

cloudera