

Apache Kylin

OLAP on Hadoop

DTCC

2015中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2015

大数据技术探索和价值发现



Agenda

- What's Apache Kylin?
- Tech Highlights
- Performance
- Roadmap
- Q & A

What's Kylin



kylin / 'ki:lin / 麒麟

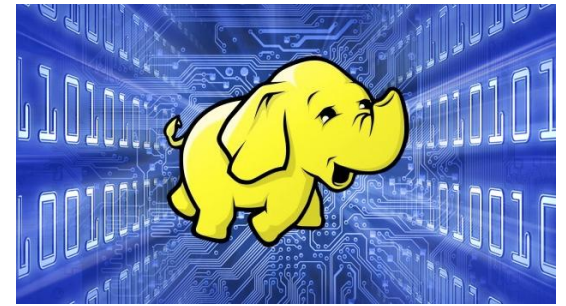
--n. (in Chinese art) a mythical animal of composite form

Extreme OLAP Engine for Big Data

Kylin is an open source Distributed Analytics Engine from eBay that provides SQL interface and multi-dimensional analysis (OLAP) on Hadoop supporting extremely large datasets

- Open Sourced on Oct 1st, 2014
- Be accepted as Apache Incubator Project on Nov 25th, 2014

Big Data Era



- More and more data becoming available on Hadoop
- Limitations in existing Business Intelligence (BI) Tools
 - Limited support for Hadoop
 - Data size growing exponentially
 - High latency of interactive queries
 - Scale-Up architecture
- Challenges to adopt Hadoop as interactive analysis system
 - Majority of analyst groups are SQL savvy
 - No mature SQL interface on Hadoop
 - OLAP capability on Hadoop ecosystem not ready yet



**Why not
Build an engine from scratch?**

Features Highlights

- **Extreme Scale OLAP Engine**

Kylin is designed to query 10+ billions of rows on Hadoop

- **ANSI SQL Interface on Hadoop**

Kylin offers ANSI SQL on Hadoop and supports most ANSI SQL query functions

- **Seamless Integration with BI Tools**

Kylin currently offers integration capability with BI Tools like Tableau.

- **Interactive Query Capability**

Users can interact with Hive tables at sub-second latency

- **MOLAP Cube**

Define a data model from Hive tables and pre-build in Kylin

- **Scale Out Architecture**

Query server cluster supports thousands concurrent users and provide high availability

Features Highlights...

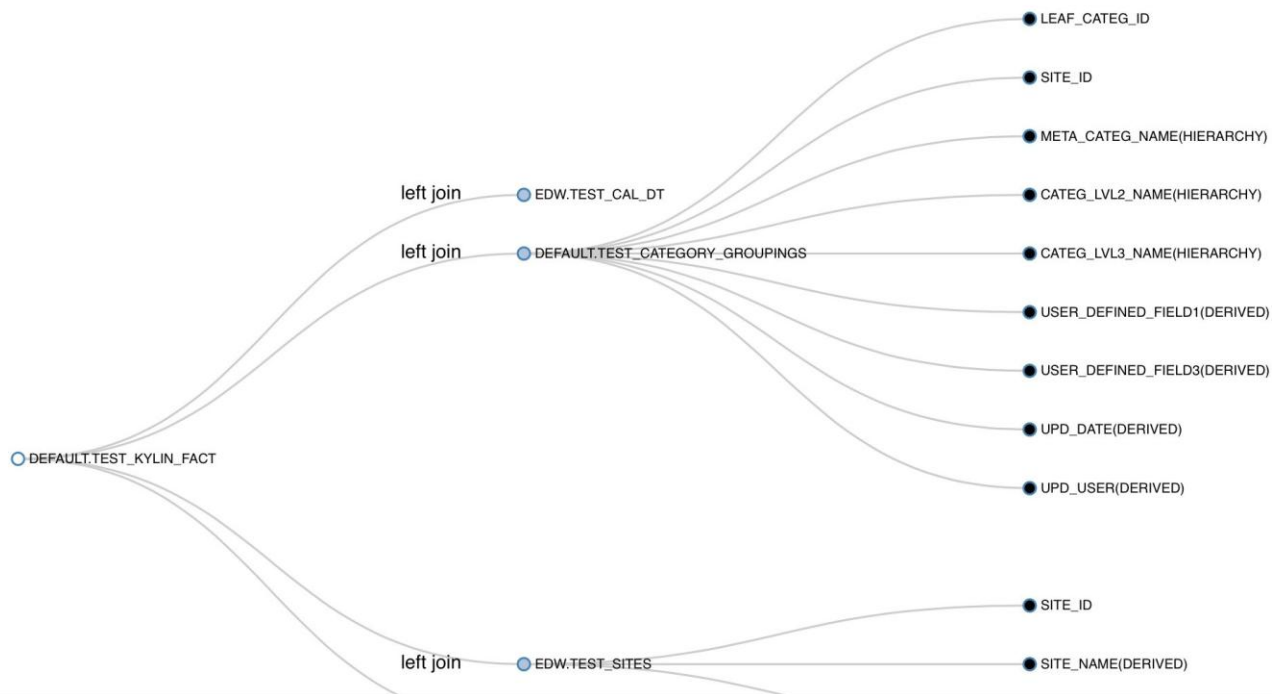
- Compression and Encoding Support
- Incremental Refresh of Cubes
- Approximate Query Capability for distinct count (HyperLogLog)
- Leverage HBase Coprocessor for query latency
- Job Management and Monitoring
- Easy Web interface to manage, build, monitor and query cubes
- Security capability to set ACL at Cube/Project Level
- Support LDAP Integration

Cube Designer

Cube Designer

Name ▾	Status ▾	Cube Size ▾	Source Records ▾	Last Build Time ▾	Owner ▾	Create Time ^	Actions	Admins
test_kylin_cube_with_slr_left_join_empty	DISABLED	0.00 KB	0				Action ▾	Action ▾

Grid Visualization SQL JSON(Cube) JSON(Model) Access Notification HBase



Job Management

Jobs <input type="checkbox"/> NEW <input type="checkbox"/> PENDING <input type="checkbox"/> RUNNING <input type="checkbox"/> FINISHED <input type="checkbox"/> ERROR <input type="checkbox"/> DISC					
Job Name ↕	Cube ↕	Progress ↕	Last Modified Time ^	Duration ↕	
PC_SESSION_COPY - 20150325000000_20150326000000 - BUILD - PDT 2015-04-12 16:56:38	PC_SESSION_COPY	100%	2015-04-12 18:20:41 PST	144.00 mins	N
PC_SESSION_COPY - 20150324000000_20150325000000 - BUILD - PDT 2015-04-12 06:49:57	PC_SESSION_COPY	100%	2015-04-12 09:07:41 PST	197.00 mins	N
PC_SESSION_COPY - 20150323000000_20150324000000 - BUILD - PDT 2015-04-09 22:34:08	PC_SESSION_COPY	100%	2015-04-09 23:57:09 PST	143.00 mins	N

Detail Information

Job Name

PC_SESSION_COPY - 20150325000000_20150326000000 - BUILD - PDT 2015-04-12 16:56:38

Job ID

bae5c8b9-78e0-4649-a579-042a3f05cab1

Status

FINISHED

Duration

144.00 mins

MapReduce Waiting

3.62 mins

Start 2015-04-12 15:56:40 PST

2015-04-12 15:56:40 PST

#1 Step Name: Create Intermediate Flat Hive Table

Data Size: 10.77 GB

Duration: 41.81 mins

Output

```

2015-04-12 18:17:52.434 - State of Hadoop Job: job_1427705526386_166781:FINISHED - SUCCEEDED
Counters: 52
  File System Counters
    FILE: Number of bytes read=27406033130
    FILE: Number of bytes written=56621871289
    FILE: Number of read operations=0
    FILE: Number of large read operations=0
    FILE: Number of write operations=0
    HDFS: Number of bytes read=11690233914
    HDFS: Number of bytes written=1143439
    HDFS: Number of read operations=25315
    HDFS: Number of large read operations=0
    HDFS: Number of write operations=17
  Job Counters
    Killed map tasks=73
    Launched map tasks=6401
    Launched reduce tasks=1
    Other local map tasks=109
  
```

hadoop

MapReduce Job job_1415168056392_0001

Application

Job

Overview

Counters

Configuration

Map tasks

Reduce tasks

Tools

Job Name: INSERT OVERWRITE TABLE DIM.SELLER_TYPE_CD(Stage=10)

User Name: root

Owner: default

State: SUCCEEDED

Uberized: false

Submitted: Wed Nov 05 01:31:34 PST 2014

Started: Wed Nov 05 01:32:19 PST 2014

Finished: Wed Nov 05 01:34:16 PST 2014

Elapsed: 1mins, 57sec

Diagnostics:

Average Map Time: 1mins, 51sec

ApplicationMaster

Attempt Number

Start Time

Node

Logs

1

Wed Nov 05 01:32:01 PST 2014

sandbox.hortonworks.com:8042

logs

Task Type

Total

Complete

Map

1

0

Reduce

0

0

Attempt Type

Failed

Killed

Successful

Maps

0

0

1

Reducers

0

0

0

About Apache Hadoop

Query and Visualization

Results (1215)

Visualization Export

Drag a column header here and drop it to group by that column.

LSTG_FORMAT...	WEEK_BEG_DT	META_CATEG...	PRICE
FP-GTC	2013-01-01	Sports MeCard...	93.268450871...
Auction	2013-01-01	Business & Ind...	55.635632631...
Others	2013-01-01	Sporting Goods	82.533676494...
Auction	2013-01-01	Real Estate	20.347844733...
Auction	2013-01-01	Toys & Hobbies	4.0906749147...

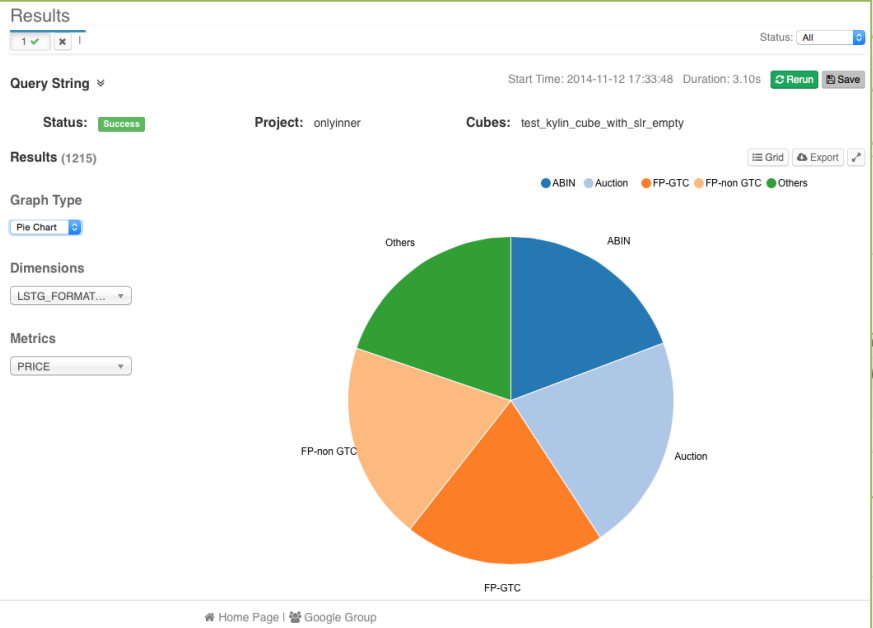
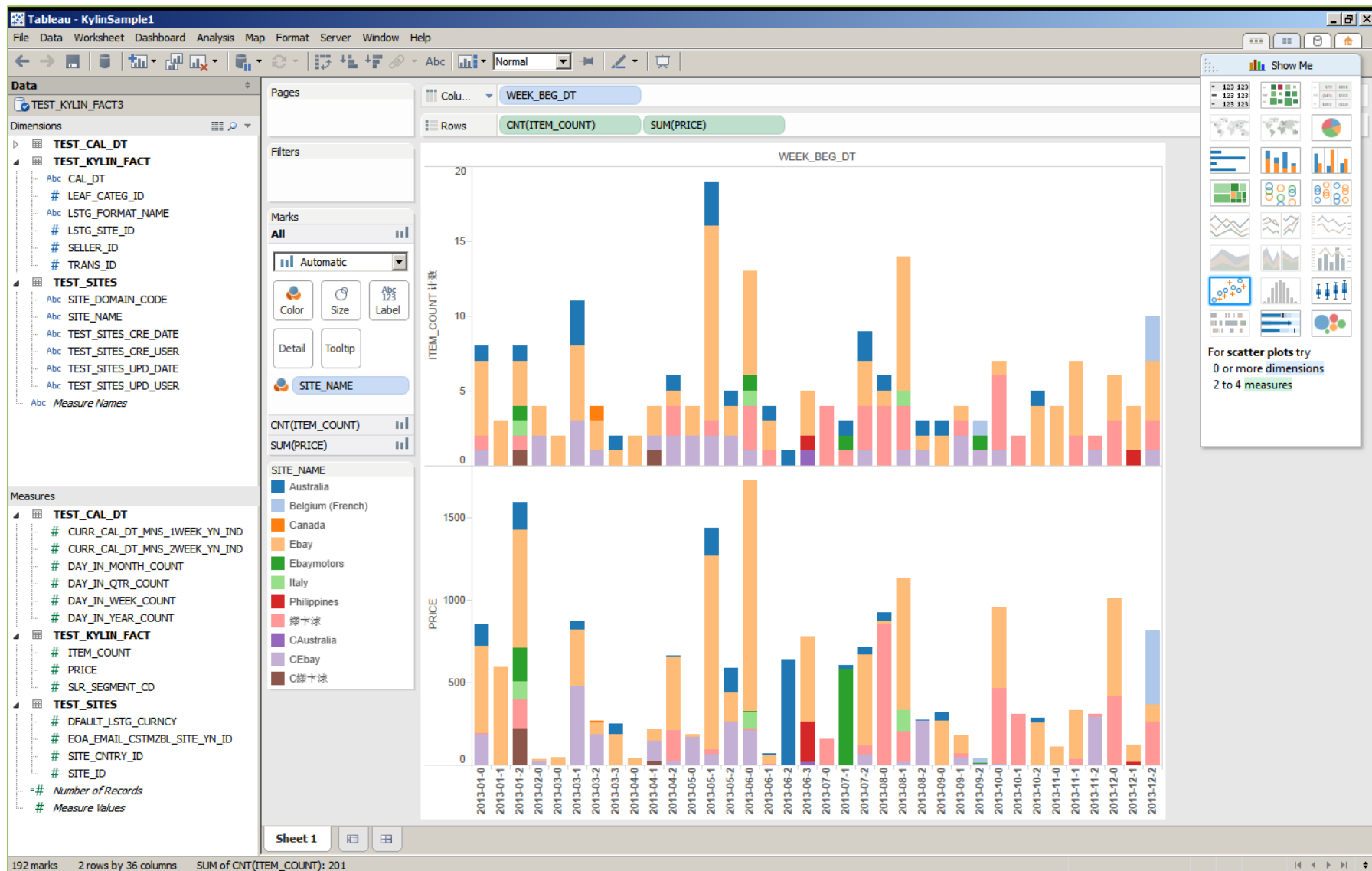


Tableau Integration



Who are using Kylin

- eBay
 - 90% query < 5 seconds

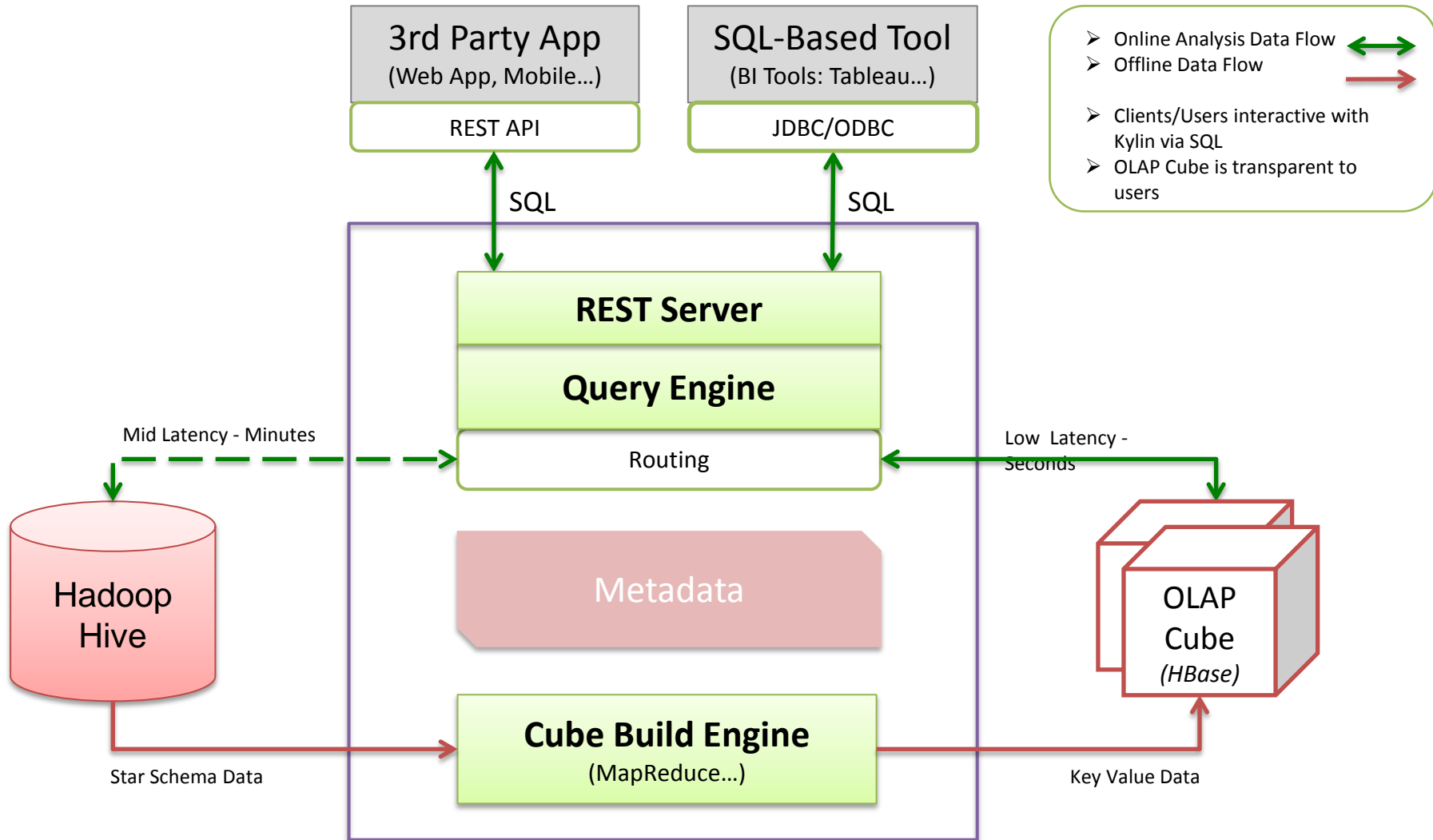
Case	Cube Size	Raw Records
User Session Analysis	26 TB	28+ billion rows
Classified Traffic Analysis	21 TB	20+ billion rows
GeoX Behavior Analysis	560 GB	1.2+ billion rows

- Baidu
 - Baidu Map internal analysis
- Many other Proof of Concepts
 - Bloomberg Law, British GAS, JD, Microsoft, StubHub, Tableau ...

Agenda

- What's Apache Kylin?
- Tech Highlights
- Performance
- Roadmap
- Q & A

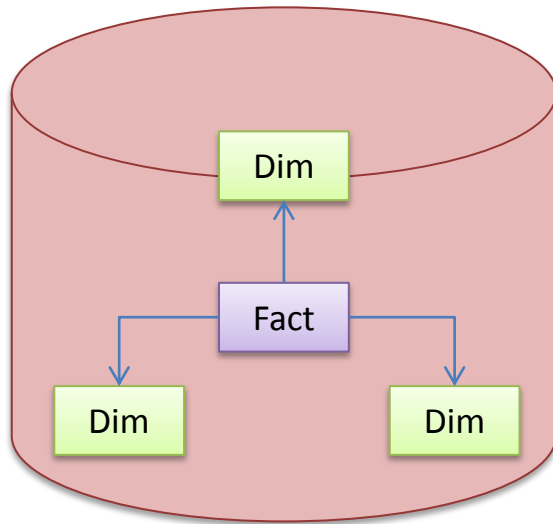
Kylin Architecture Overview



Data Modeling



End User



Source
Star Schema



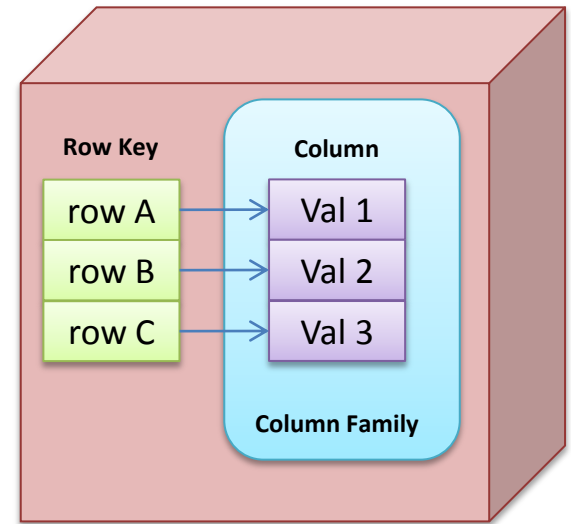
Cube Modeler

Cube: ...
Fact Table: ...
Dimensions: ...
Measures: ...
Storage(HBase): ...

Mapping
Cube Metadata



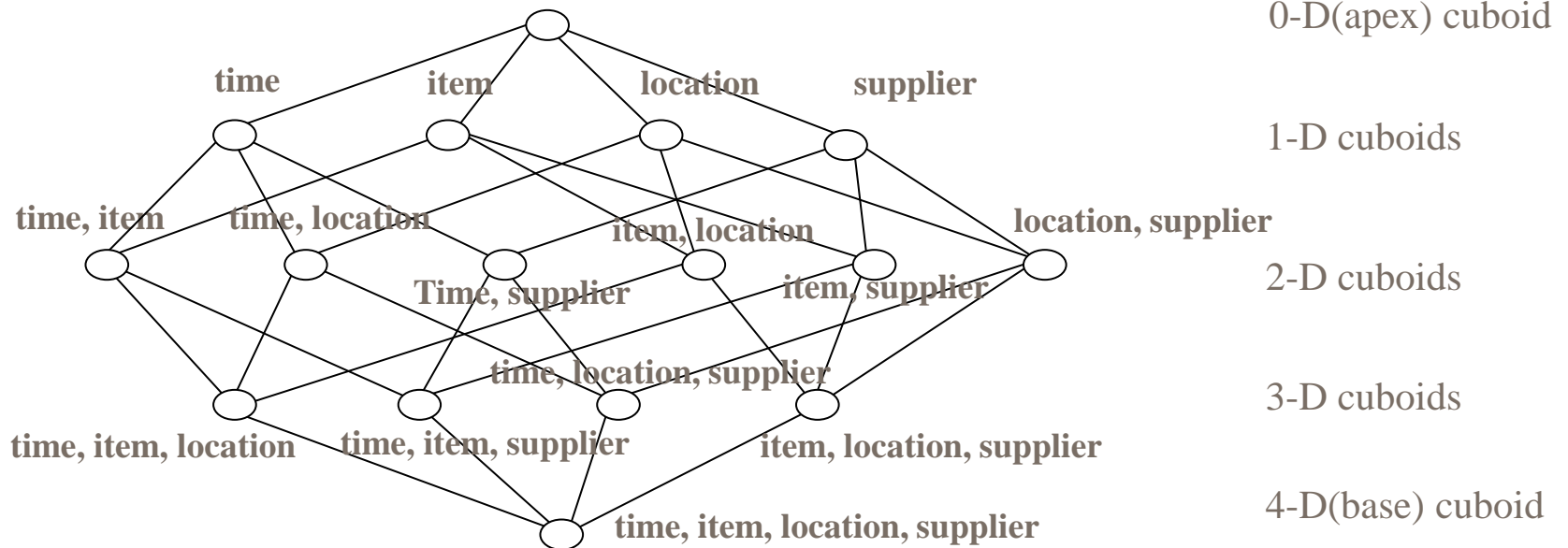
Admin



Target
HBase Storage

OLAP Cube – Balance between Space and Time

- Cuboid = one combination of dimensions
- Cube = all combination of dimensions (all cuboids)



- Base vs. aggregate cells; ancestor vs. descendant cells; parent vs. child cells
 1. (9/15, milk, Urbana, Dairy_land) - **<time, item, location, supplier>**
 2. (9/15, milk, Urbana, *) - **<time, item, location>**
 3. (*, milk, Urbana, *) - **<item, location>**
 4. (*, milk, Chicago, *) - **<item, location>**
 5. (*, milk, *, *) - **<item>**

Cube Build Job Flow

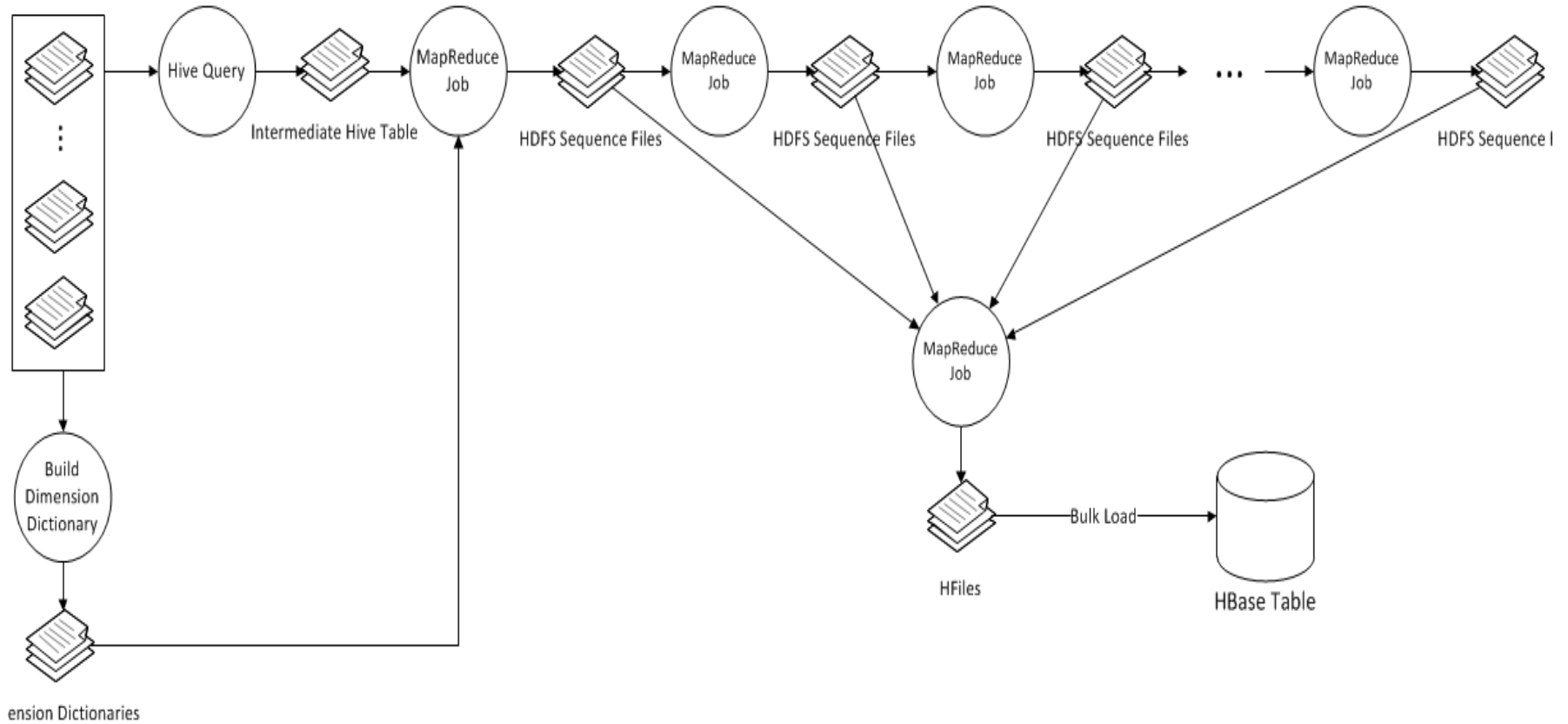
Source Hive Tables

N D (Base) Cuboid

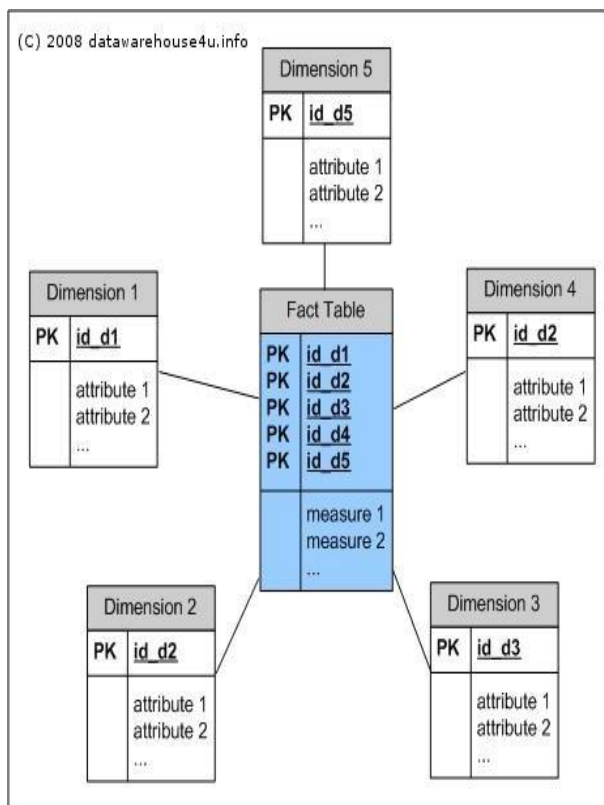
N-1 D Cuboids

N-2 D Cuboids

0 D (Apex) Cub



How To Store Cube? – HBase Schema



Pre-Joined Flat Table	
Dimensions	D1
	D2
	...
	DX
Measures	M1
	M2
	...
	MY

Pre-Aggregated Table	
Dimensions	D3
	D6
	D8
Measures	M1
	M2
	...
	MY

Cuboid ID	D1	D2	...	DX
-----------	----	----	-----	----

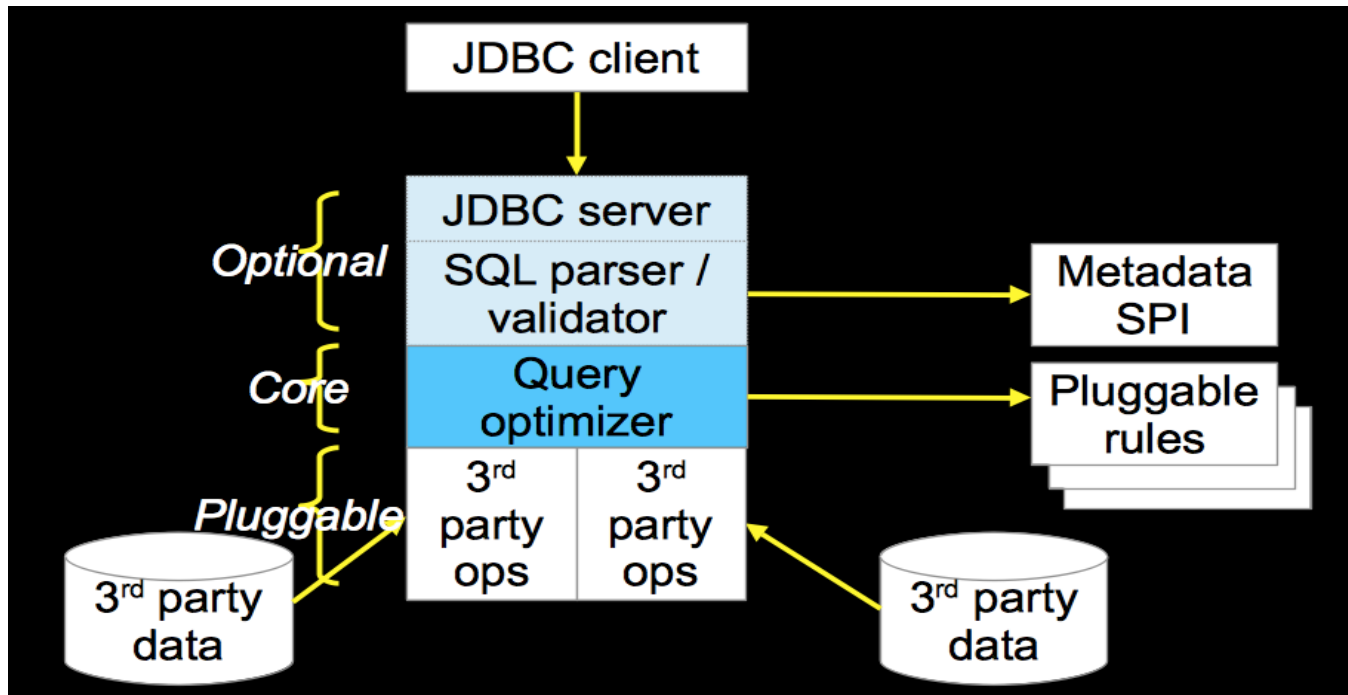
M1	M2	...	MY
----	----	-----	----

Row Key = Cuboid ID+Dimensions

Row Value = Measures

Query Engine – Calcite

- Dynamic data management framework.
- Formerly known as Optiq, Calcite is an Apache incubator project, used by Apache Drill and Apache Hive, among others.
- <http://optiq.incubator.apache.org>



Kylin Extensions on Calcite

- **Metadata SPI**
 - Provide table schema from Kylin metadata
- **Optimize Rule**
 - Translate the logic operator into Kylin operator
- **Relational Operator**
 - Find right cube
 - Translate SQL into storage engine API call
 - Generate physical execute plan by linq4j java implementation
- **Result Enumerator**
 - Translate storage engine result into java implementation result.
- **SQL Function**
 - Add HyperLogLog for distinct count
 - Implement date time related functions (i.e. Quarter)

Query Engine – Kylin Explain Plan

```
SELECT test_cal_dt.week_beg_dt, test_category.category_name, test_category.lvl2_name, test_category.lvl3_name,  
test_kylin_fact.lstg_format_name, test_sites.site_name, SUM(test_kylin_fact.price) AS GMV, COUNT(*) AS TRANS_CNT  
FROM test_kylin_fact  
  LEFT JOIN test_cal_dt ON test_kylin_fact.cal_dt = test_cal_dt.cal_dt  
  LEFT JOIN test_category ON test_kylin_fact.leaf_categ_id = test_category.leaf_categ_id AND test_kylin_fact.lstg_site_id =  
test_category.site_id  
  LEFT JOIN test_sites ON test_kylin_fact.lstg_site_id = test_sites.site_id  
WHERE test_kylin_fact.seller_id = 123456 OR test_kylin_fact.lstg_format_name = 'New'  
GROUP BY test_cal_dt.week_beg_dt, test_category.category_name, test_category.lvl2_name, test_category.lvl3_name,  
test_kylin_fact.lstg_format_name, test_sites.site_name
```

OLAPToEnumerableConverter

```
OLAPProjectRel(WEEK_BEG_DT=[0], category_name=[1], CATEG_LVL2_NAME=[2], CATEG_LVL3_NAME=[3],  
LSTG_FORMAT_NAME=[4], SITE_NAME=[5], GMV=[CASE(=($7, 0), null, $6)], TRANS_CNT=[8])  
  OLAPAggregateRel(group=[{0, 1, 2, 3, 4, 5}], agg#0=[SUM0($6)], agg#1=[COUNT($6)], TRANS_CNT=[COUNT()])  
    OLAPProjectRel(WEEK_BEG_DT=[13], category_name=[21], CATEG_LVL2_NAME=[15], CATEG_LVL3_NAME=[14],  
LSTG_FORMAT_NAME=[5], SITE_NAME=[23], PRICE=[0])  
      OLAPFilterRel(condition=[OR(=($3, 123456), =($5, 'New'))])  
        OLAPJoinRel(condition=[=($2, $25)], joinType=[left])  
          OLAPJoinRel(condition=[AND(=($6, $22), =($2, $17))], joinType=[left])  
            OLAPJoinRel(condition=[=($4, $12)], joinType=[left])  
              OLAPTableScan(table=[[DEFAULT, TEST_KYLIN_FACT]], fields=[[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11]])  
                OLAPTableScan(table=[[DEFAULT, TEST_CAL_DT]], fields=[[0, 1]])  
                  OLAPTableScan(table=[[DEFAULT, test_category]], fields=[[0, 1, 2, 3, 4, 5, 6, 7, 8]])  
                    OLAPTableScan(table=[[DEFAULT, TEST_SITES]], fields=[[0, 1, 2]])
```

Storage Engine

- **Plugin-able storage engine**
 - Common iterator interface for storage engine
 - Isolate query engine from underline storage
- **Translate cube query into HBase table scan**
 - Columns, Groups → Cuboid ID
 - Filters -> Scan Range (Row Key)
 - Aggregations -> Measure Columns (Row Values)
- **Scan HBase table and translate HBase result into cube result**
 - HBase Result (key + value) -> Cube Result (dimensions + measures)

Cube Optimization

- Curse of dimensionality: N dimension cube has 2^N cuboid
 - Full Cube vs. Partial Cube
- High data volume
 - Dictionary Encoding
 - Incremental Building

Full Cube vs. Partial Cube

■ Full Cube

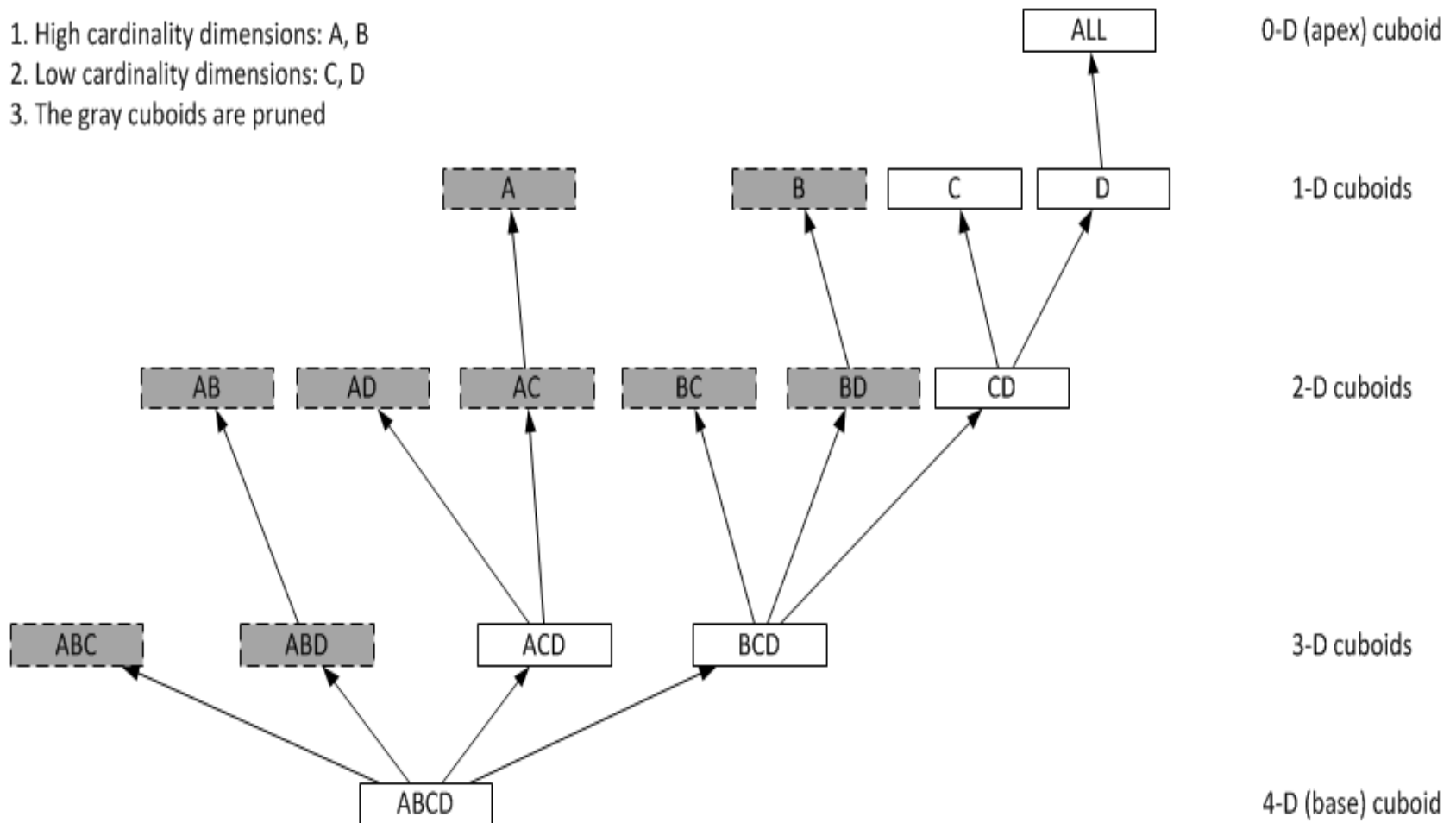
- Pre-aggregate all dimension combinations
- “Curse of dimensionality”: N dimension cube has 2^N cuboid.

■ Partial Cube

- To avoid dimension explosion, we divide the dimensions into different aggregation groups
 - $2^{N+M+L} \rightarrow 2^N + 2^M + 2^L$
- For cube with 30 dimensions, if we divide these dimensions into 3 group, the cuboid number will reduce from 1 Billion to 3 Thousands
 - $2^{30} \rightarrow 2^{10} + 2^{10} + 2^{10}$
- Tradeoff between online aggregation and offline pre-aggregation

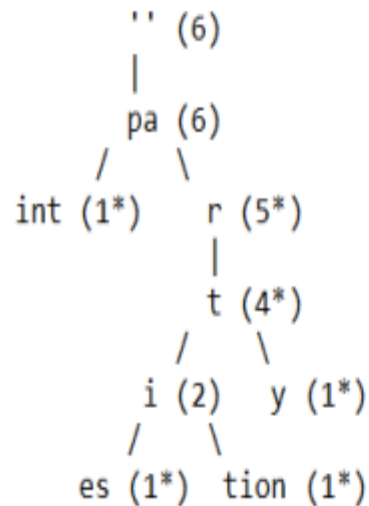
Partial Cube

1. High cardinality dimensions: A, B
2. Low cardinality dimensions: C, D
3. The gray cuboids are pruned



Dictionary Encoding

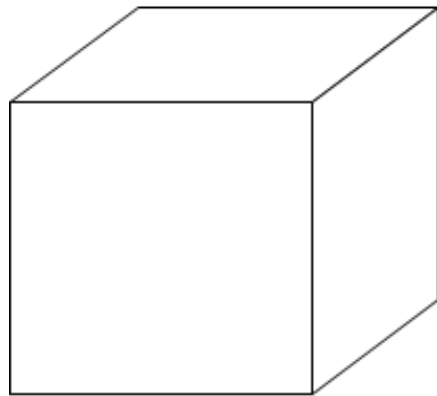
- Data cube has lots of duplicated dimension values
- Dictionary maps dimension values into IDs that will reduce the memory and storage footprint.
- Dictionary is based on Trie



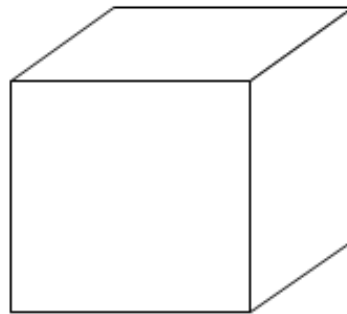
[value]	[seq no]
par	-> 1
part	-> 2
party	-> 5
parties	-> 3
partition	-> 4
paint	-> 0

Incremental Build

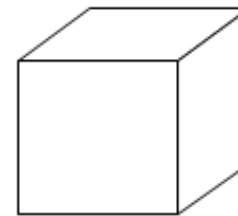
Aggregate the results when querying



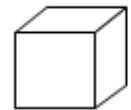
cube-Y-2011:2012



cube-M-2013-1:8



cube-D-2013-09-1:20



cube-D-2013-09-21

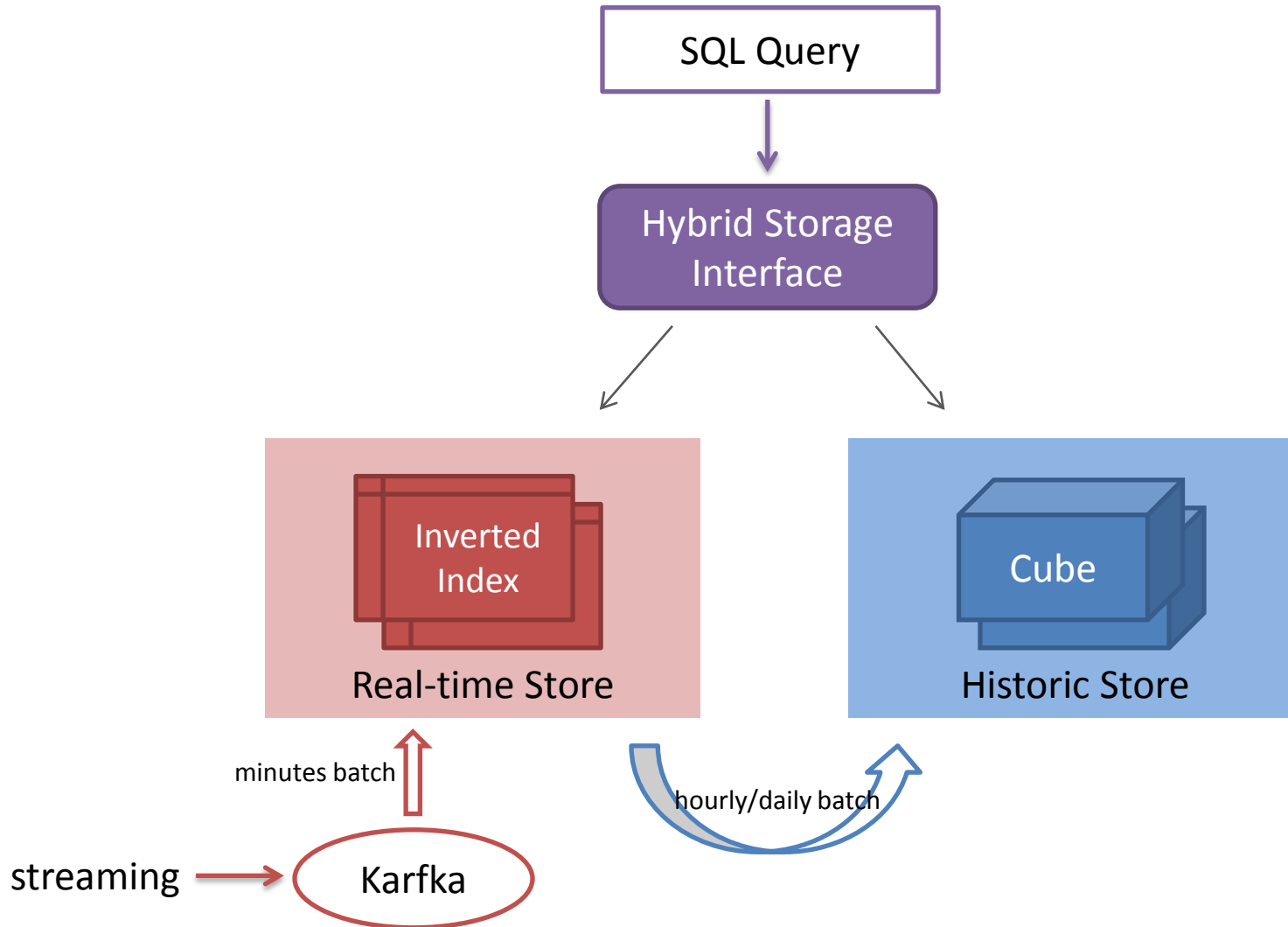
1. Cube is immutable
2. Merge small cubes into a larger one

Streaming, ongoing effort

- Cube is great, but...
 - Sometimes we want to drill down to row level information
 - Cube takes time to build, how about real-time analysis?
- Streaming with inverted index

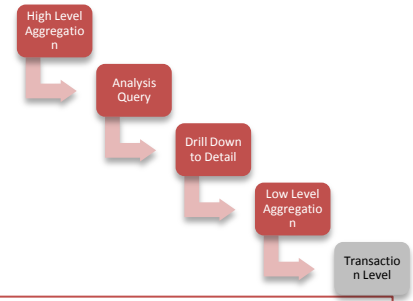
	Cube	Inverted Index
Storage format	Pre-aggregated cuboids	Sharding, columnar storage, with inverted index on row blocks
Query method	Cuboid scanning	Massive parallel processing
Strength	Pre-aggregate huge <u>historic</u> data to small summaries	Swift response to <u>real-time</u> data
Weakness	Take time to build	Slow at scanning large data volume

Kylin 0.8, Lambda Architecture

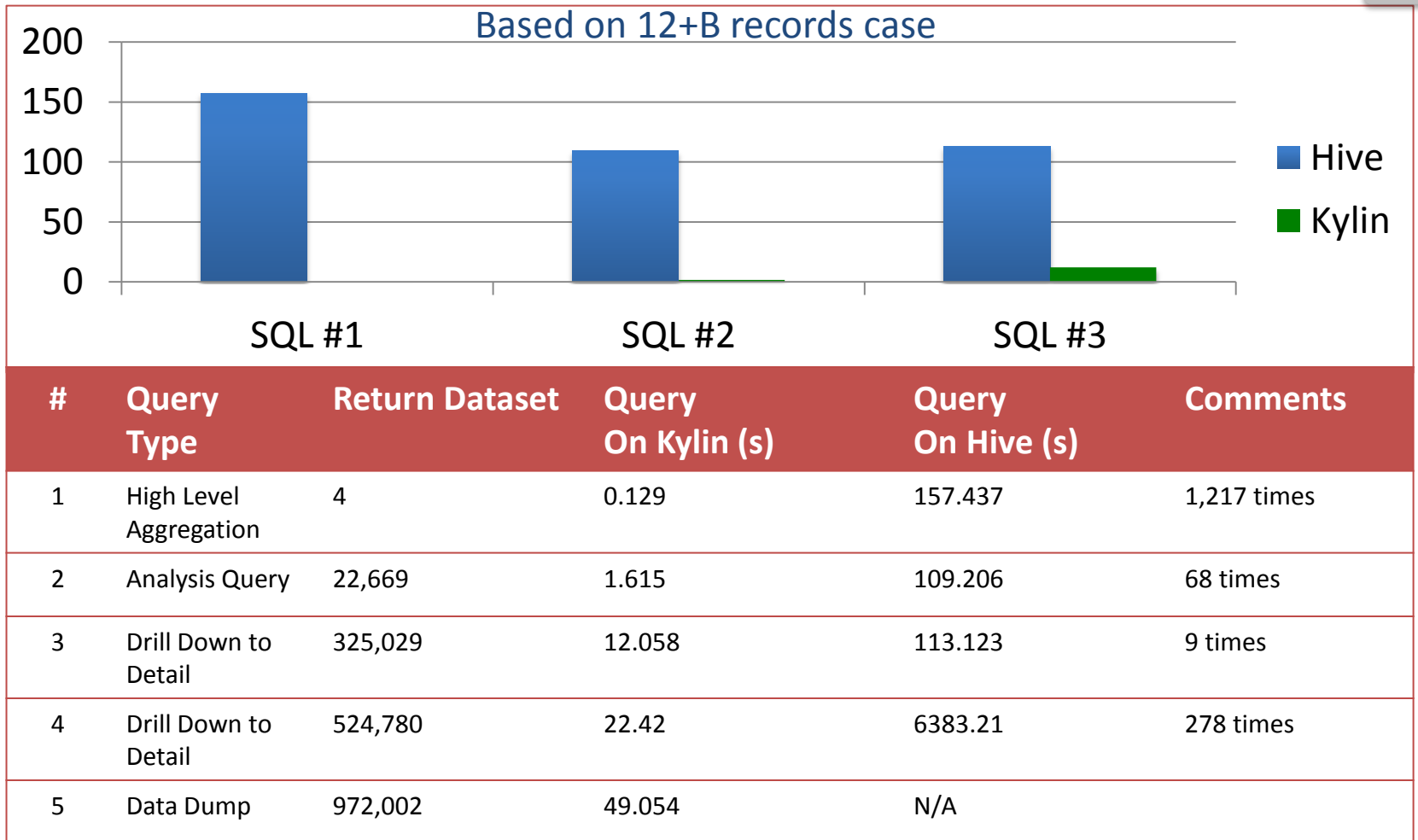


Agenda

- What's Apache Kylin?
- Tech Highlights
- Performance
- Roadmap
- Q & A



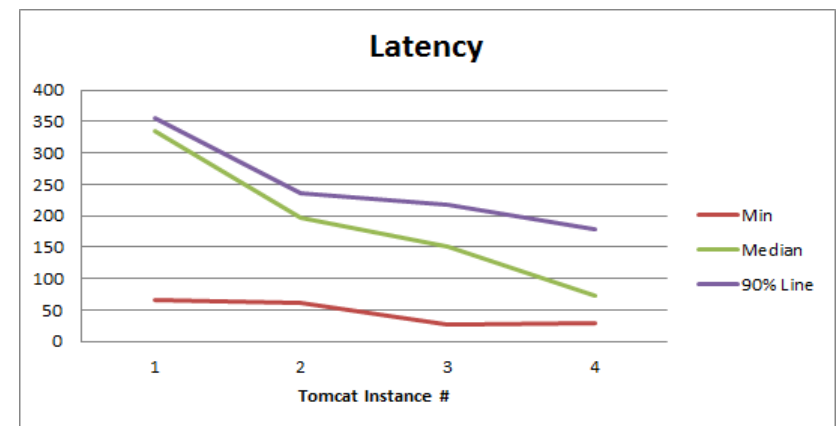
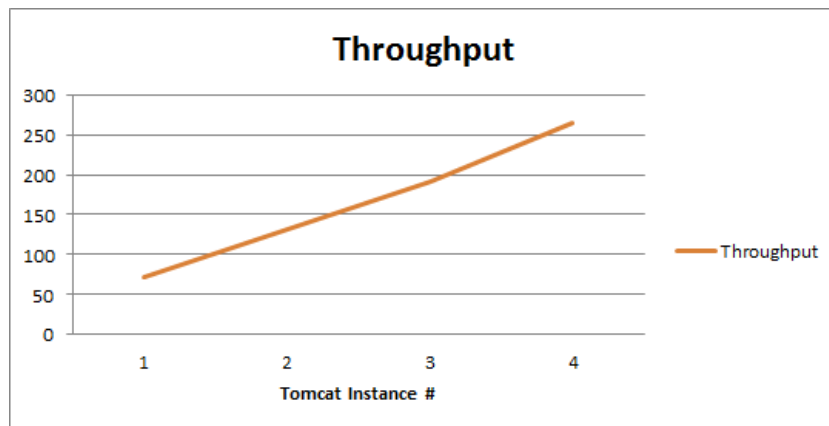
Kylin vs. Hive



Performance -- Concurrency

Single Tomcat Instance on a Single Machine

	Parallel Thread #	Data			Latency (ms)				Throughput
		Raw Recors	HBase Scan	Return	Min	Max	Median	90% Line	
High Level Aggregation Query	30	1,940,304,293	5	5	67	1809	334	355	72.5/sec
Detail Level Query (with Seller ID)	30	13,683,834,542	43934	7283	1758	4534	2182	3171	9.7/sec

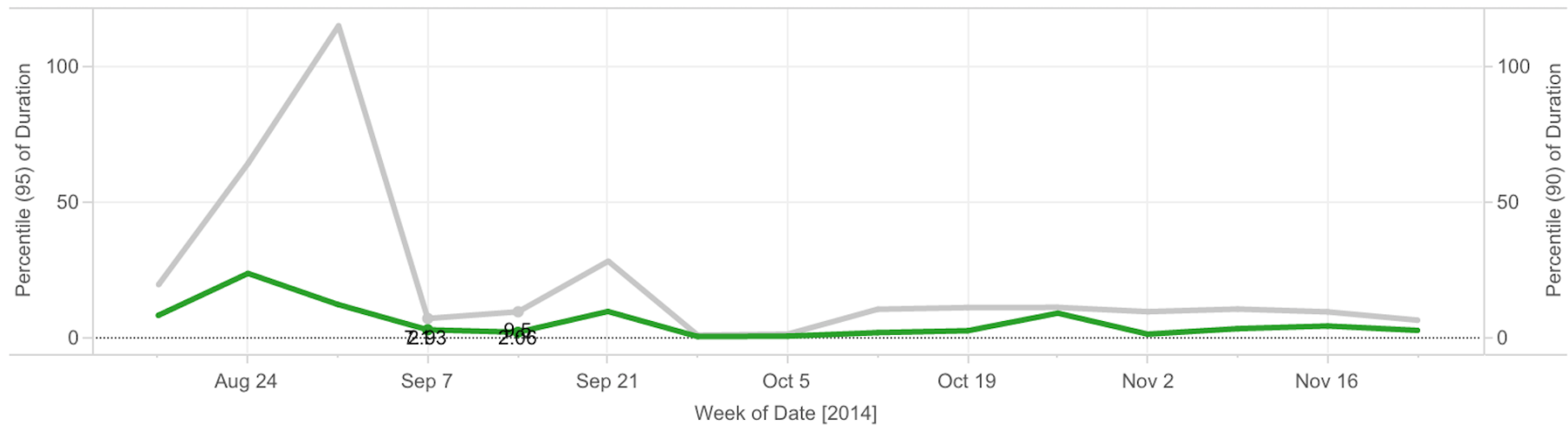


Linear scale out with more nodes

Performance - Query Latency

90% queries <5s

Kylin Query Latency (90% and 95%)

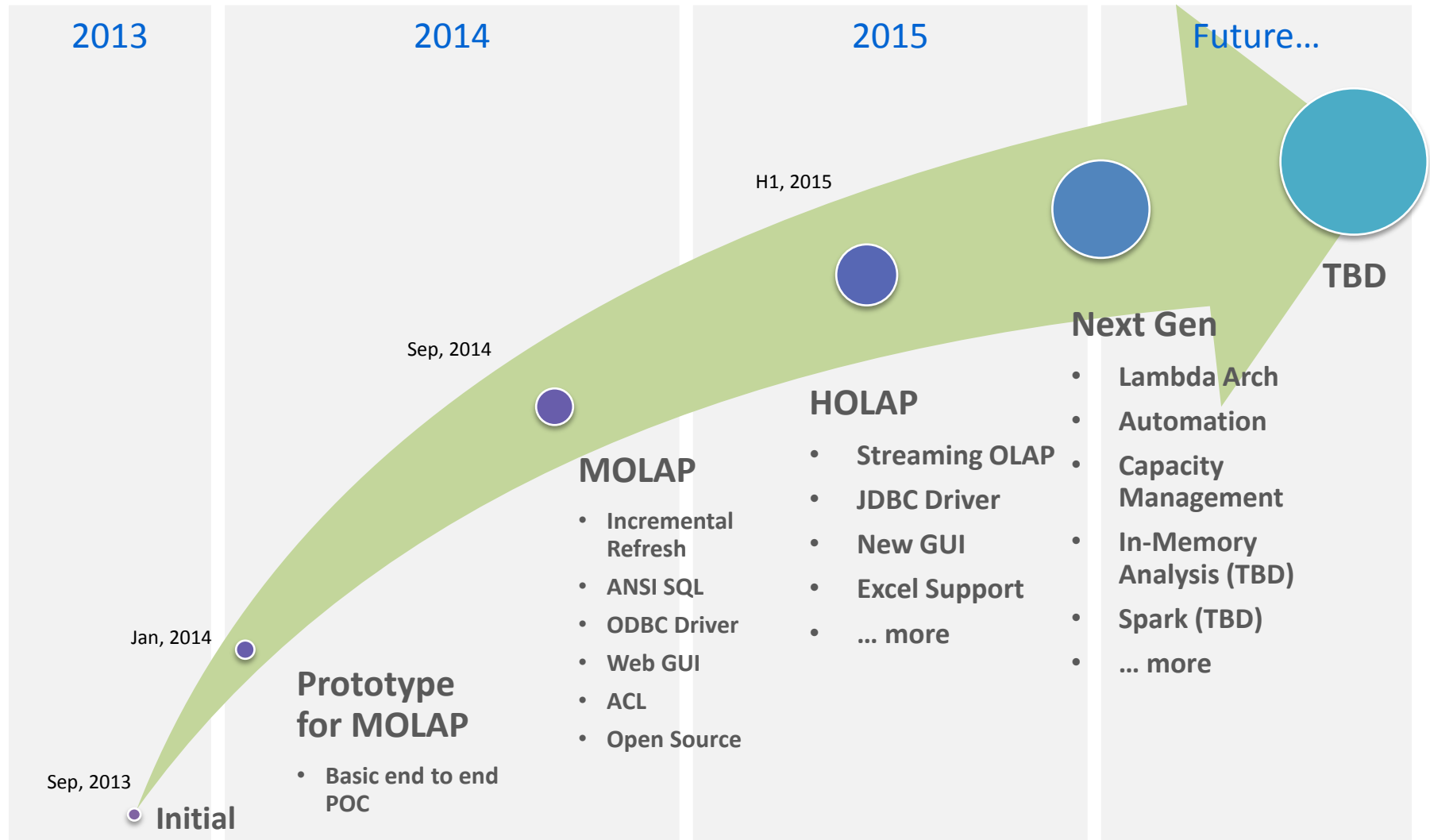


Green Line: 90%tile queries
Gray Line: 95%tile queries

Agenda

- What's Apache Kylin?
- Tech Highlights
- Performance
- Roadmap
- Q & A

Kylin Evolution Roadmap



Kylin Ecosystem

■ Kylin Core

- Fundamental framework of Kylin OLAP Engine

■ Extension

- Plugins to support for additional functions and features

■ Integration

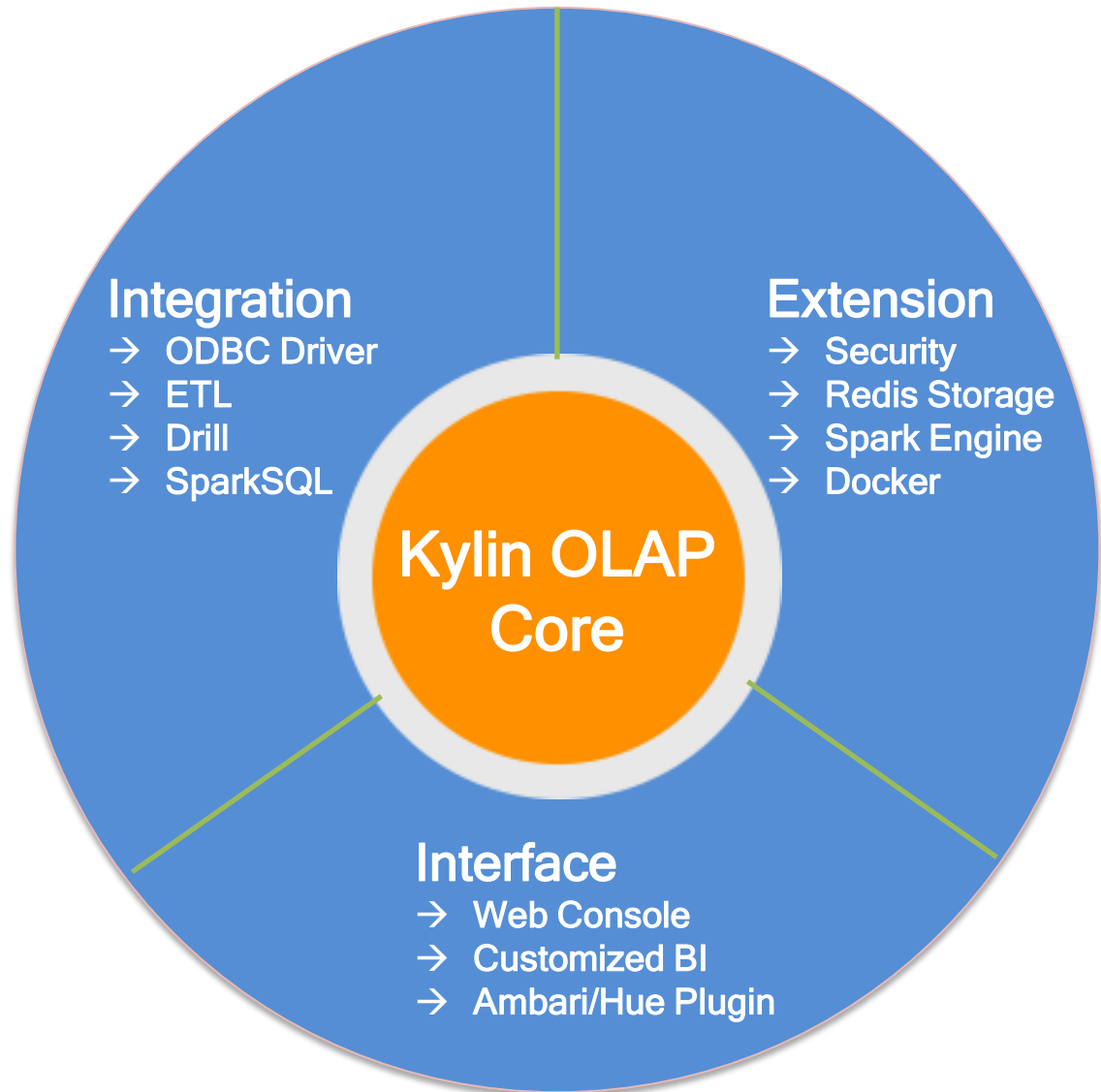
- Lifecycle Management Support to integrate with other applications

■ Interface

- Allows for third party users to build more features via user-interface atop Kylin core

■ Driver

- ODBC and JDBC Drivers



Apache Kylin

- Kylin Site:
 - <http://kylin.io>
- Twitter:
 - [@ApacheKylin](https://twitter.com/ApacheKylin)
- 微信
 - ApacheKylin





THANKS