ebay inc™

Apache Kylin Introduction

Dec 8, 2014 | @ApacheKylin

Luke Han Sr. Product Manager | lukhan@ebay.com | @lukehq
Yang Li Architect & Tech Leader | yangli9@ebay.com

Agenda

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

ebay inc

What's Kylin

kylin / ˈkiːˈlɪn / 麒麟

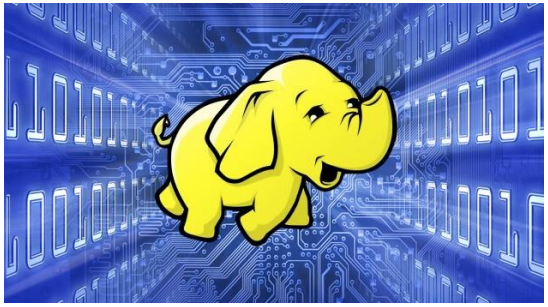
--n. (in Chinese art) a mythical animal of composite form

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

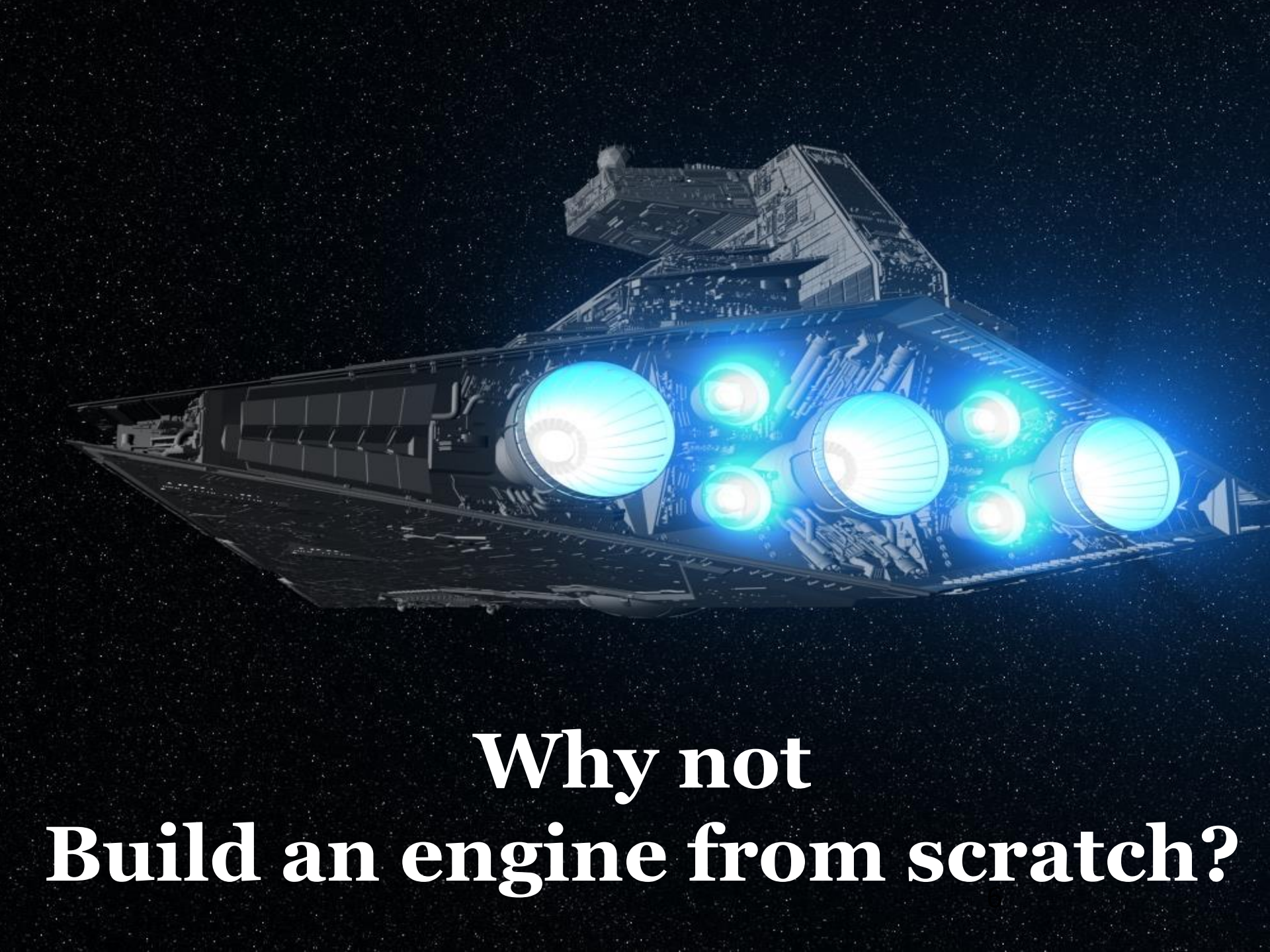


- More and more data becoming available on Hadoop
- Limitations in existing Business Intelligence (BI) Tools
  - Limited support for Hadoop
  - Data size growing exponentially
  - High latency of interactive queries
  - Scale-Up architecture
- Challenges to adopt Hadoop as interactive analysis system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

Business Needs for Big Data Analysis

- Sub-second query latency on billions of rows
- ANSI SQL for both analysts and engineers
- Full OLAP capability to offer advanced functionality
- Seamless Integration with BI Tools
- Support of high cardinality and high dimensions

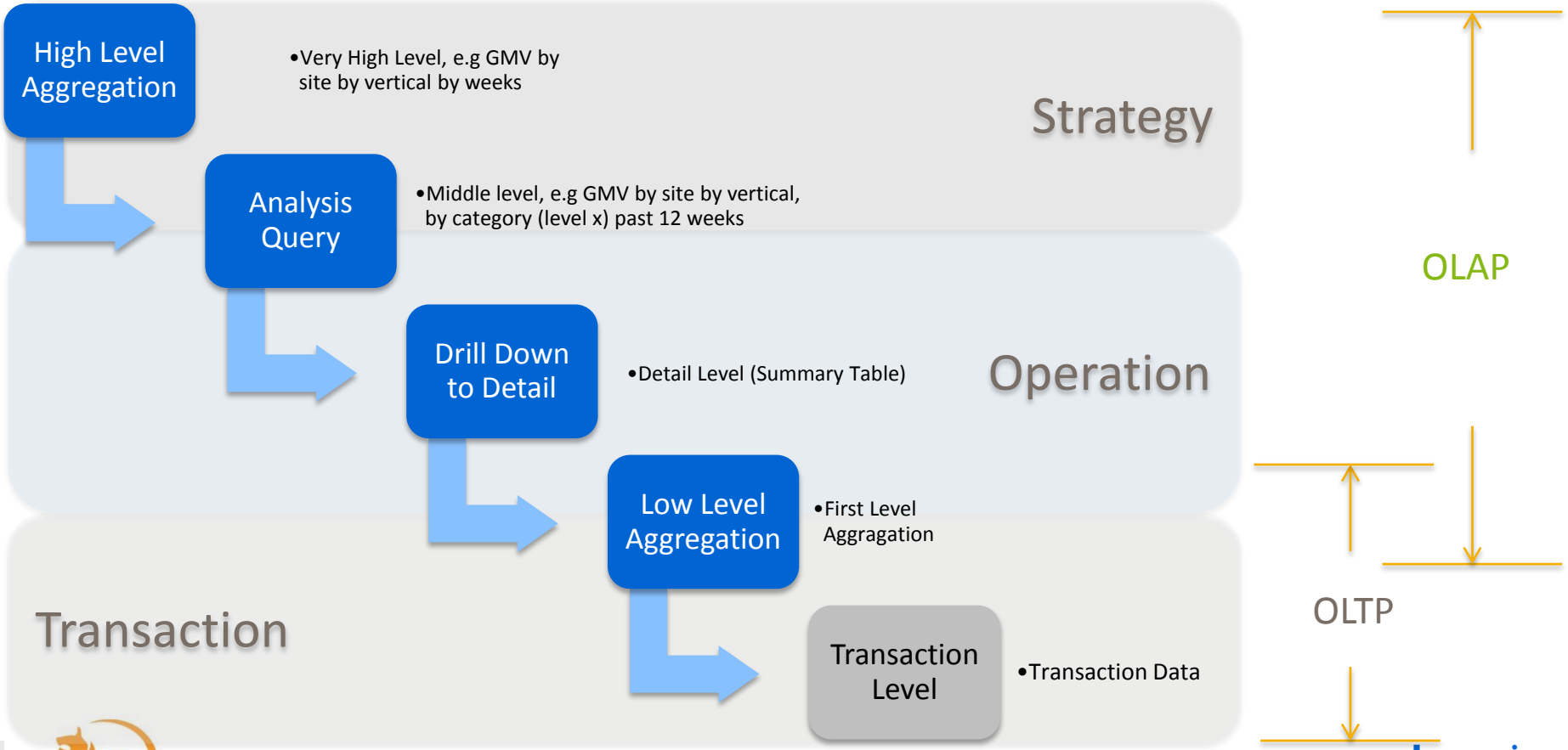
- High concurrency - thousands of end users
- Distributed and scale out architecture for large data volume
- Open source solution



Why not
Build an engine from scratch?

Analytics Query Taxonomy

Kylin is designed to accelerate 80+% analytics queries performance on Hadoop

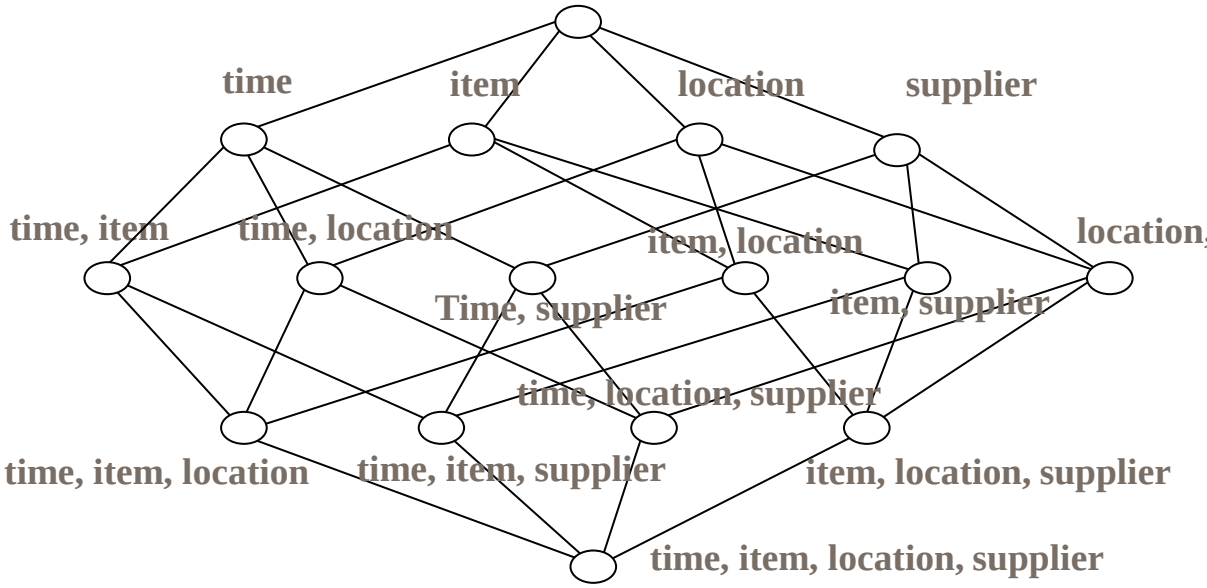


Technical Challenges

- Huge volume data
  - Table scan
- Big table joins
  - Data shuffling
- Analysis on different granularity
  - Runtime aggregation expensive
- Map Reduce job
  - Batch processing

OLAP Cube – Balance between Space and Time

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



0-D(apex) cuboid

1-D cuboids

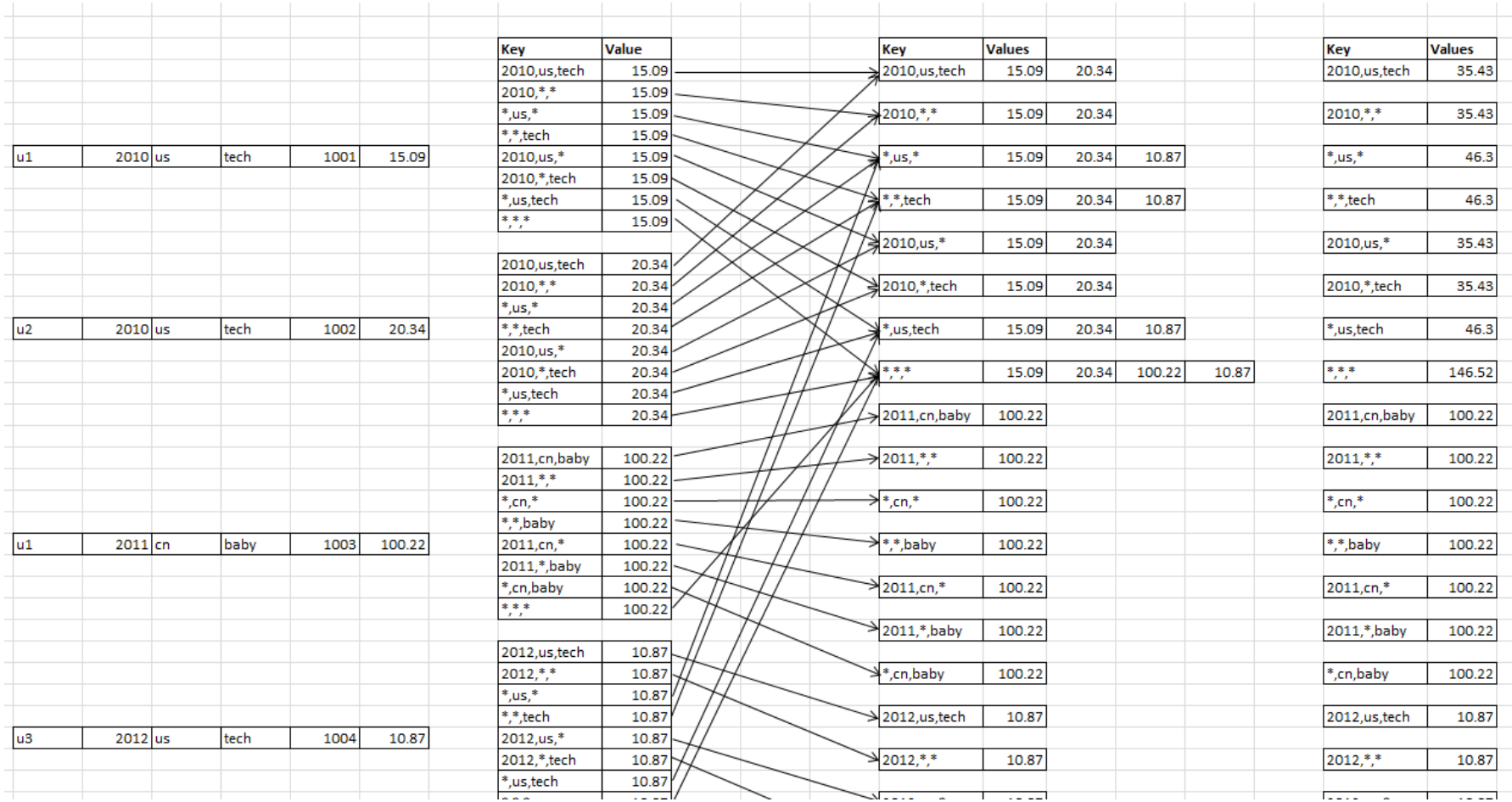
2-D cuboids

3-D cuboids

4-D(base) cuboid

- 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>**

From Relational to Key-Value



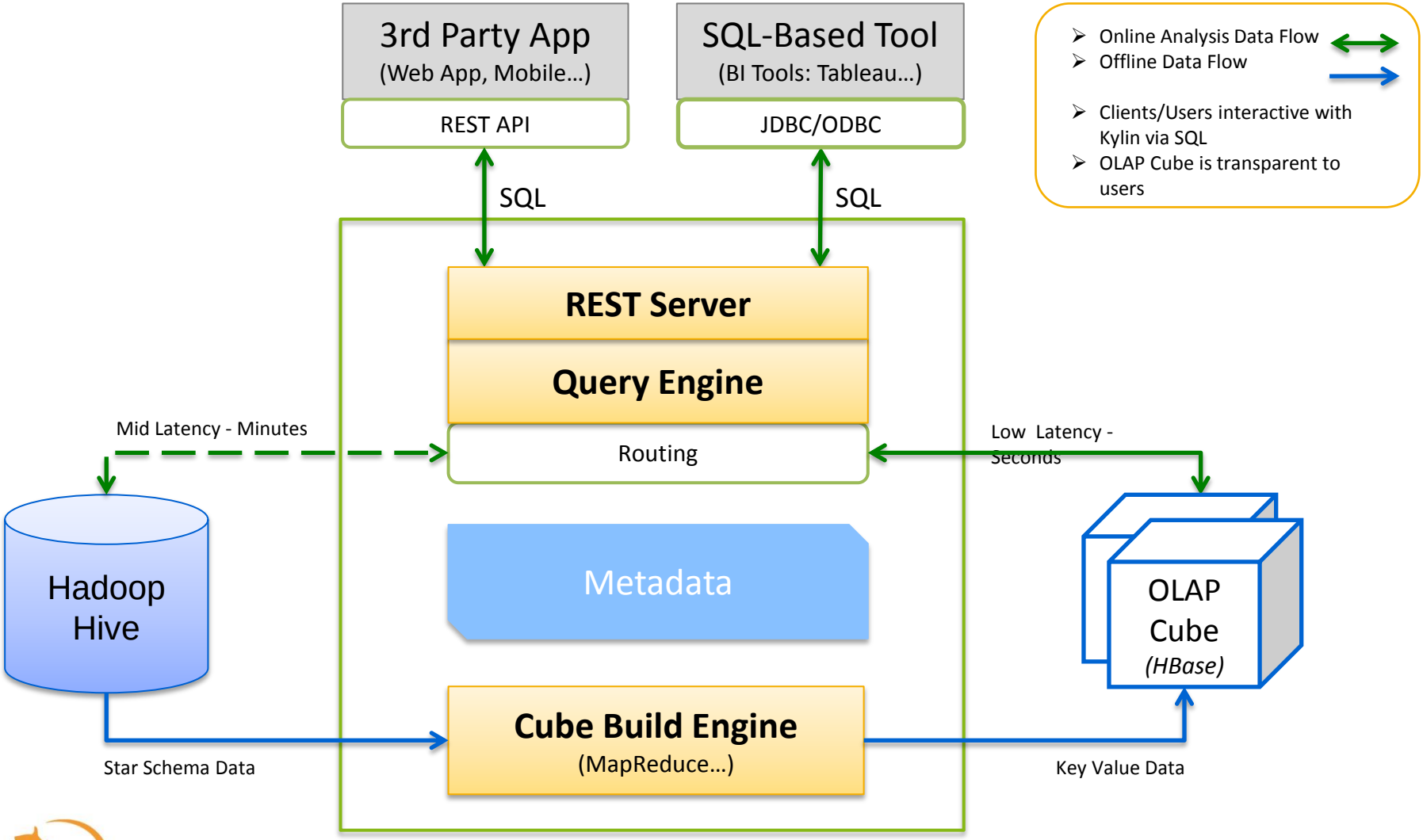
u1	2010	us	tech	1001	15.09
u2	2010	us	tech	1002	20.34
u1	2011	cn	baby	1003	100.22
u3	2012	us	tech	1004	10.87

Key	Value
2010,us,tech	15.09
2010,*,*	15.09
,us,	15.09
,,tech	15.09
2010,us,*	15.09
2010,*,tech	15.09
*,us,tech	15.09
,,*	15.09
2010,us,tech	20.34
2010,*,*	20.34
,us,	20.34
,,tech	20.34
2010,us,*	20.34
2010,*,tech	20.34
*,us,tech	20.34
,,*	20.34
2011,cn,baby	100.22
2011,*,*	100.22
,cn,	100.22
,,baby	100.22
2011,cn,*	100.22
2011,*,baby	100.22
*,cn,baby	100.22
,,*	100.22
2012,us,tech	10.87
2012,*,*	10.87
,us,	10.87
,,tech	10.87
2012,us,*	10.87
2012,*,tech	10.87
*,us,tech	10.87

Key	Values			
2010,us,tech	15.09	20.34		
2010,*,*	15.09	20.34		
,us,	15.09	20.34	10.87	
,,tech	15.09	20.34	10.87	
2010,us,*	15.09	20.34		
2010,*,tech	15.09	20.34		
*,us,tech	15.09	20.34	10.87	
,,*	15.09	20.34	100.22	10.87
2011,cn,baby	100.22			
2011,*,*	100.22			
,cn,	100.22			
,,baby	100.22			
2011,cn,*	100.22			
2011,*,baby	100.22			
*,cn,baby	100.22			
2012,us,tech	10.87			
2012,*,*	10.87			

Key	Values
2010,us,tech	35.43
2010,*,*	35.43
,us,	46.3
,,tech	46.3
2010,us,*	35.43
2010,*,tech	35.43
*,us,tech	46.3
,,*	146.52
2011,cn,baby	100.22
2011,*,*	100.22
,cn,	100.22
,,baby	100.22
2011,cn,*	100.22
2011,*,baby	100.22
*,cn,baby	100.22
2012,us,tech	10.87
2012,*,*	10.87

Kylin Architecture Overview



Features Highlights

- **Extremely Fast OLAP Engine at Scale**

Kylin is designed to reduce query latency on Hadoop for 10+ billions of rows of data

- **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 Hadoop data via Kylin at sub-second latency, better than Hive queries for the same dataset

- **MOLAP Cube**

User can define a data model and pre-build in Kylin with more than 10+ billions of raw data records

Features Highlights Cons

- 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

How Does Kylin Utilize Hadoop Components?

- Hive
  - Input source
  - Pre-join star schema during cube building
- MapReduce
  - Pre-aggregation metrics during cube building
- HDFS
  - Store intermediated files during cube building.
- HBase
  - Store data cube.
  - Serve query on data cube.
  - Coprocessor is used for query processing.

Why Kylin is Fast?

- Pre-built cube – query result already be calculated
- Leveraging distributed computing infrastructure
- No runtime Hive table scan and MapReduce job
- Compression and encoding
- Put “Computing” to “Data”
- Cached

Agenda

- What's Kylin
- Tech Highlights
- Performance
- Open Source
- Q & A

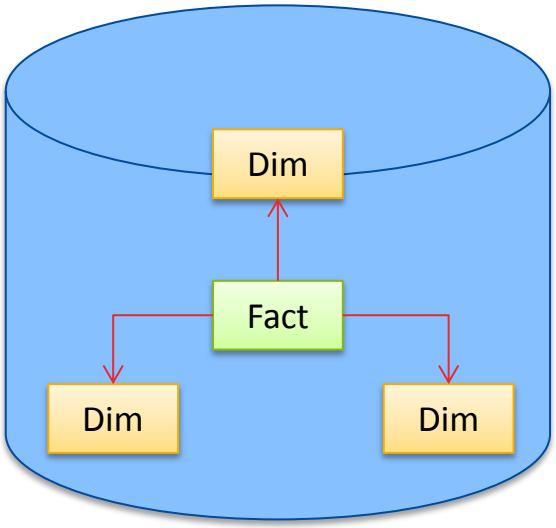
ebay inc

Data Modeling

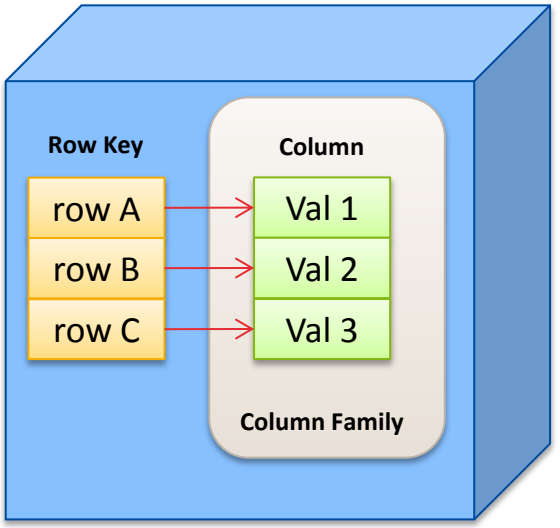
End User

Cube Modeler

Admin



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



Source
Star Schema

Mapping
Cube Metadata

Target
HBase Storage

Cube Metadata

- Dimension
  - Normal
  - Mandatory
  - Hierarchy
  - Derived
- Measure
  - Sum
  - Count
  - Max
  - Min
  - Average
  - Distinct Count (based on HyperLogLog)

Mandatory Dimension

- Dimension that must present on cuboid
  - E.g. Date

Normal

A	B	C
A	B	-
-	B	C
A	-	C
A	-	-
-	B	-
-	-	C
-	-	-

A is mandatory

A	B	C
A	B	-
A	-	C
A	-	-

Hierarchy Dimension

- Dimensions that form a “contains” relationship where parent level is required for child level to make sense.
  - E.g. Year -> Month -> Day; Country -> City

Normal

A	B	C
A	B	-
-	B	C
A	-	C
A	-	-
-	B	-
-	-	C
-	-	-

A -> B -> C is hierarchy

A	B	C
A	B	-
A	-	-
-	-	-

Derived Dimension

- Dimensions on lookup table that can be derived by PK
  - E.g. User ID -> [Name, Age, Gender]

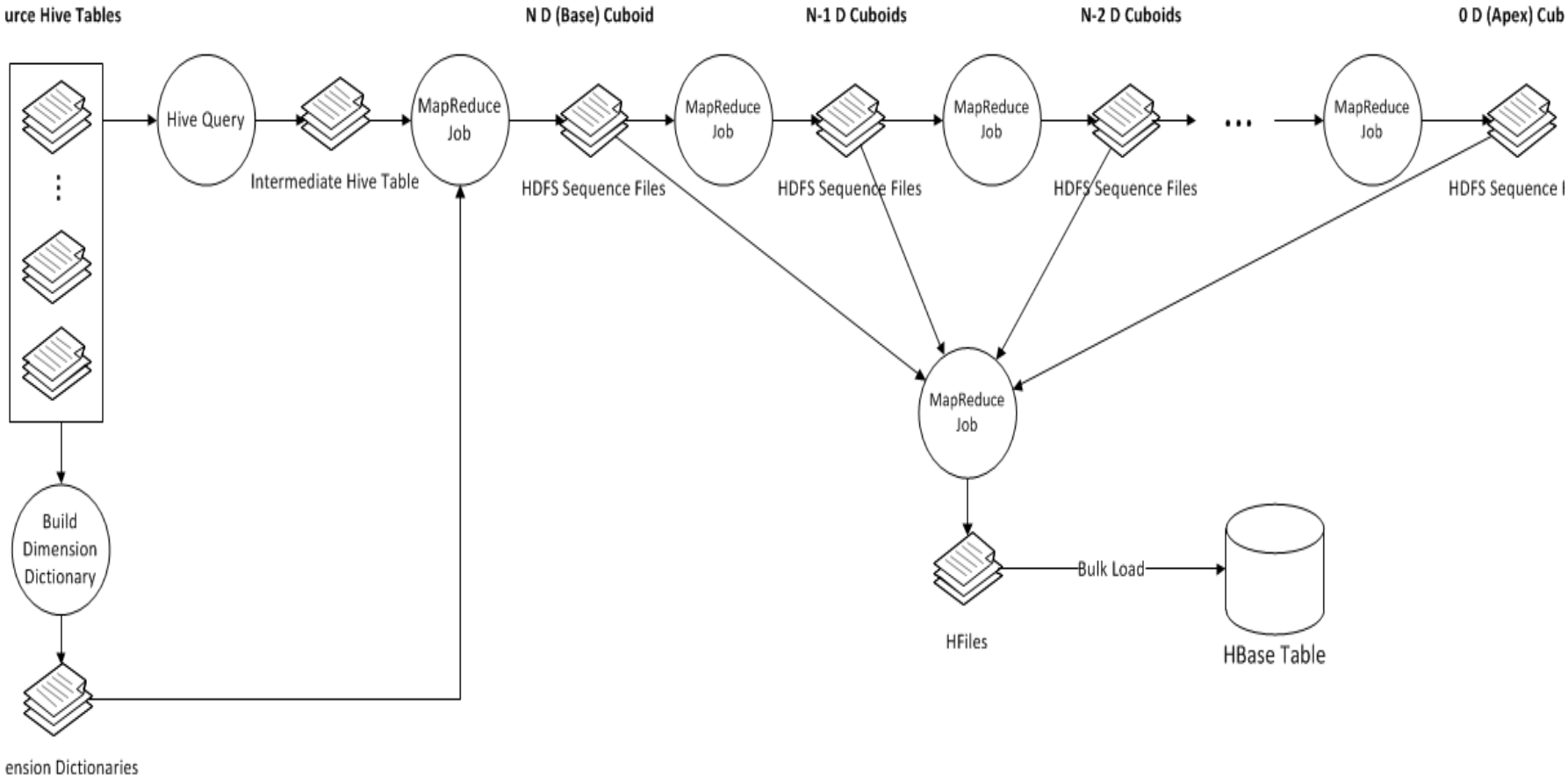
Normal

A	B	C
A	B	-
-	B	C
A	-	C
A	-	-
-	B	-
-	-	C
-	-	-

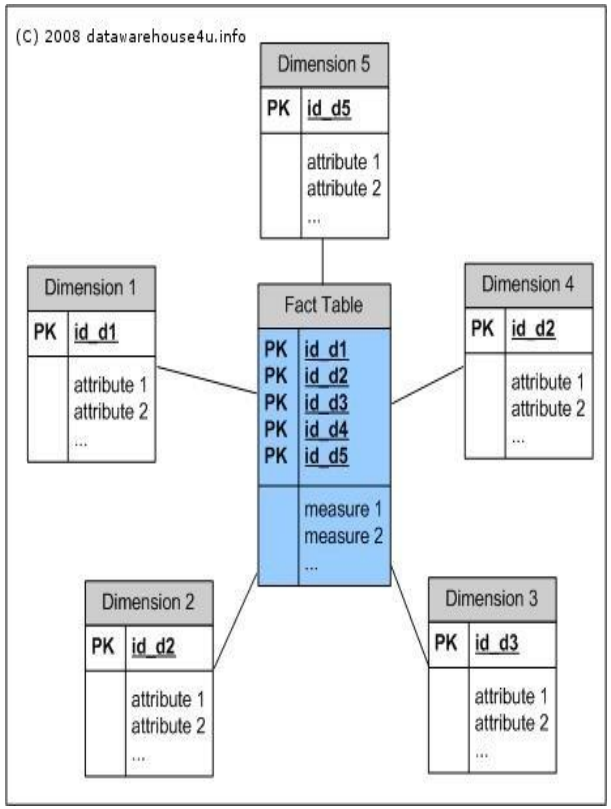
A, B, C is derived by ID

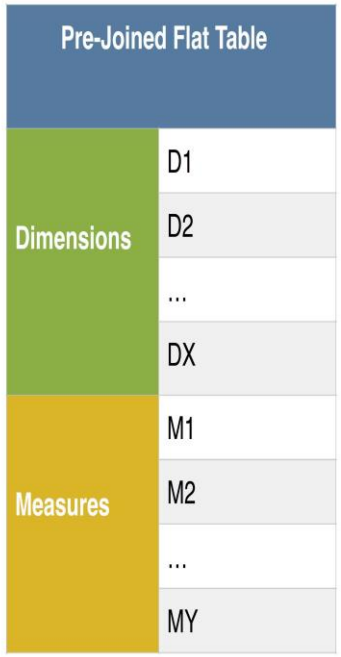
ID
-

Cube Build Job Flow



Cube Build Result



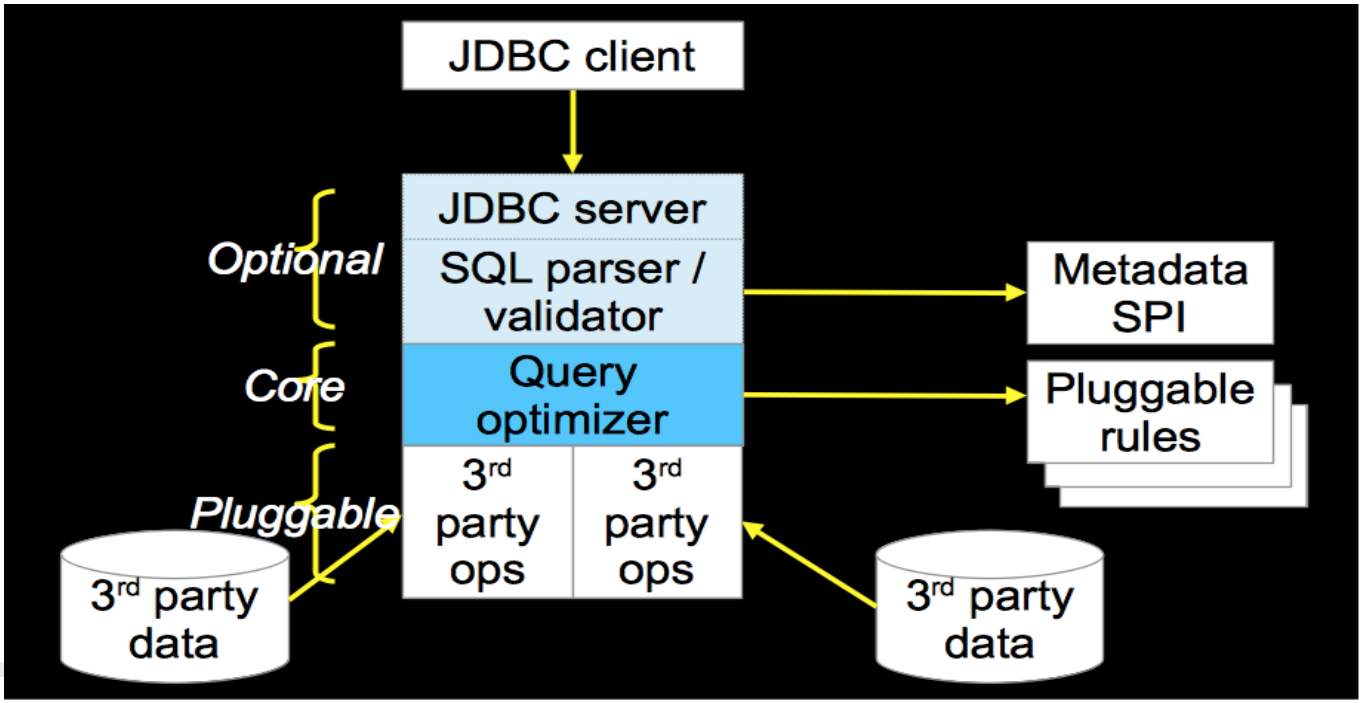


Row Key = Cuboid ID+Dimensions

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



Calcite Plugins

- **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)

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 = 123456OR 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 query 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
- Hugh data volume
  - Dictionary Encoding
  - Incremental Building

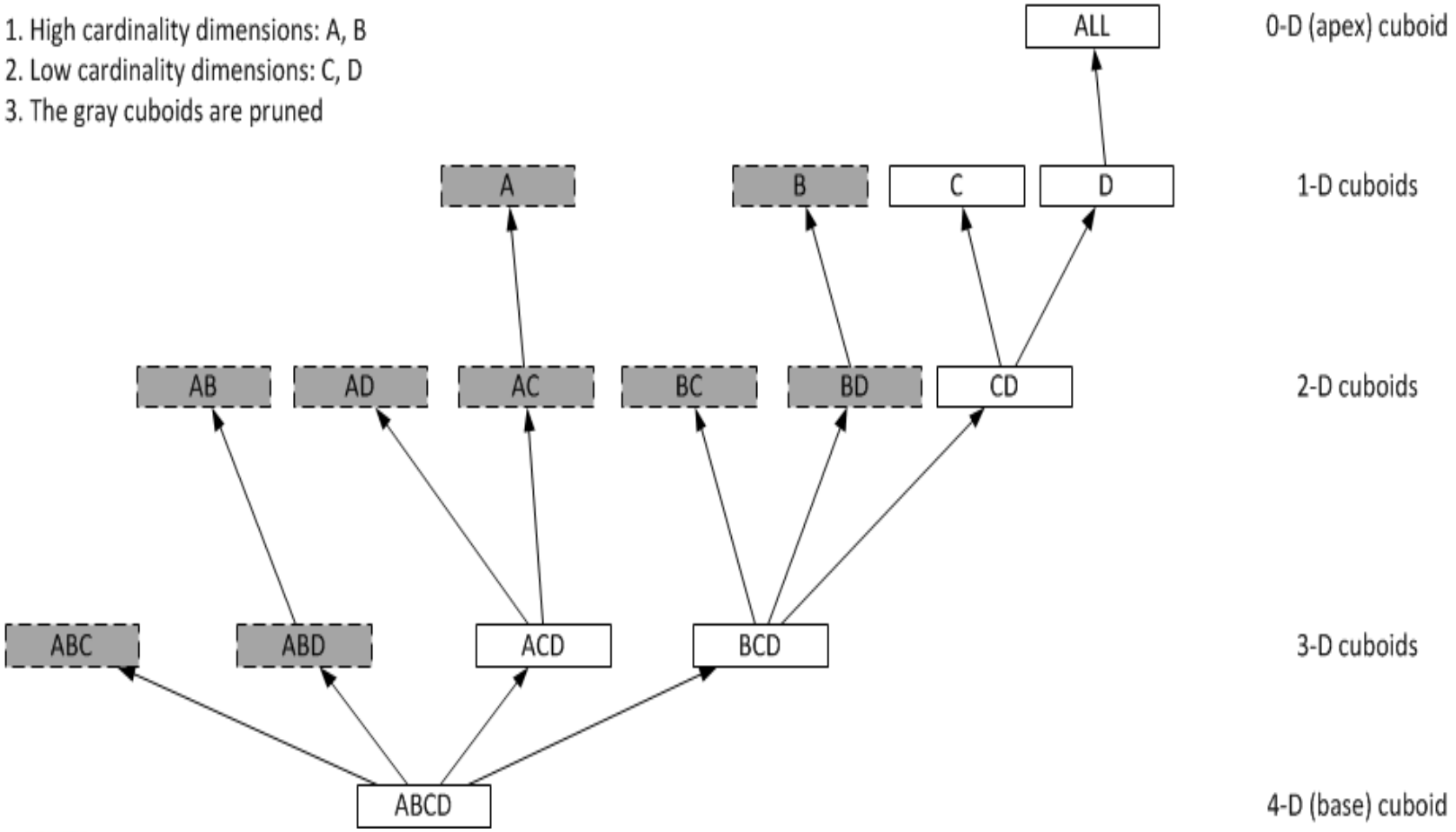
Full Cube vs. Partial Cube

- **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

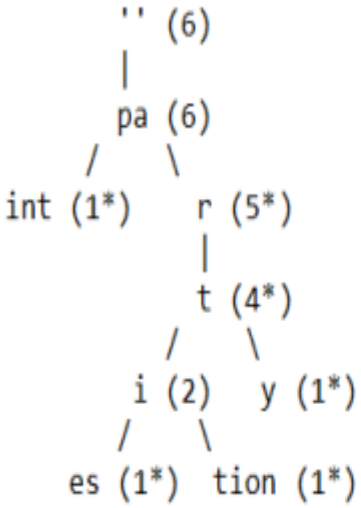
Partial Cube

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



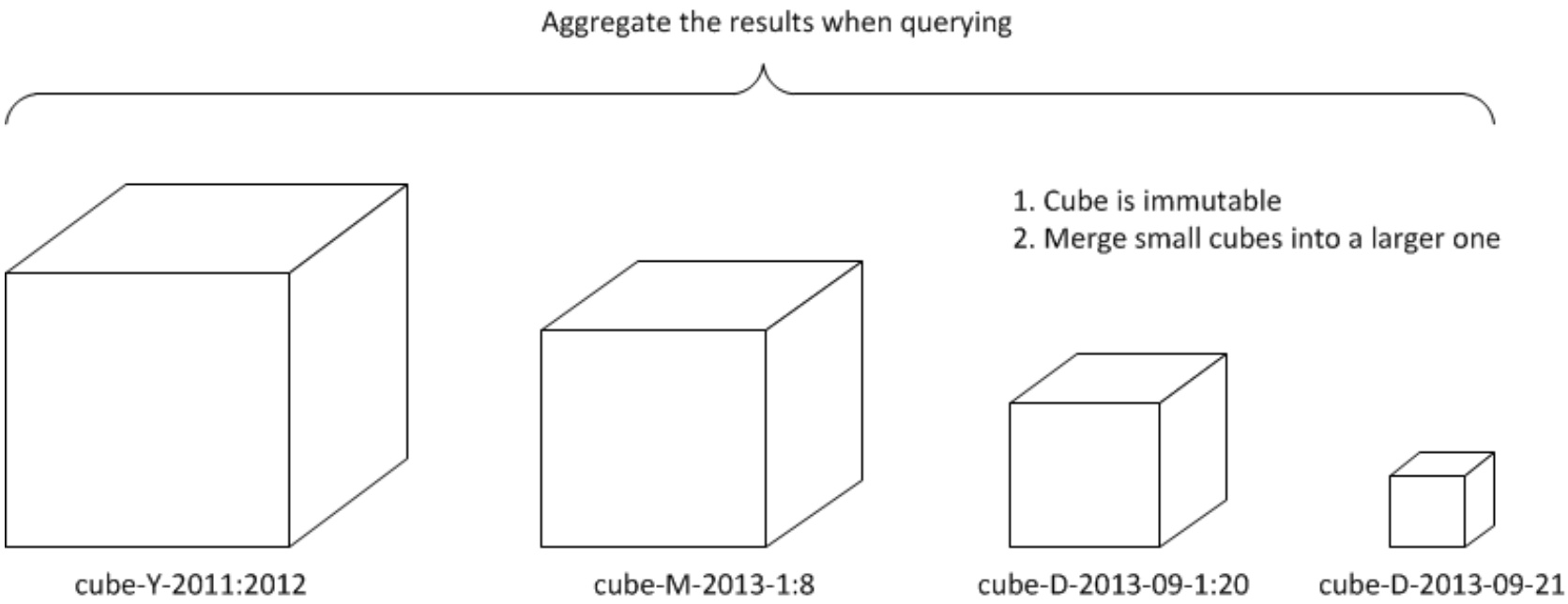
Dictionary Encoding

- Data cube has lost 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



Inverted Index

- Challenge
  - Has no raw data records
  - Slow table scan on high cardinality dimensions

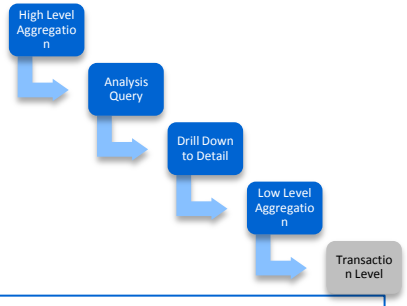
- Inverted Index Storage (an ongoing effort)
  - Persist the raw table
  - Bitmap inverted index
  - Time range partition
  - In-memory (block cache)
  - Parallel scan (endpoint coprocessor)

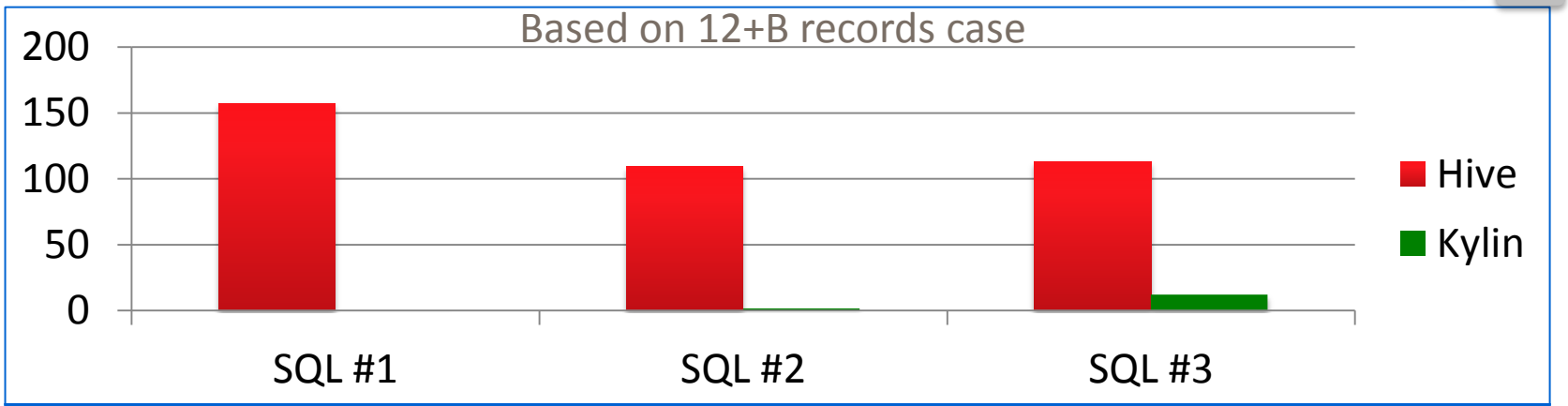
Agenda

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

ebay inc

Kylin vs. Hive



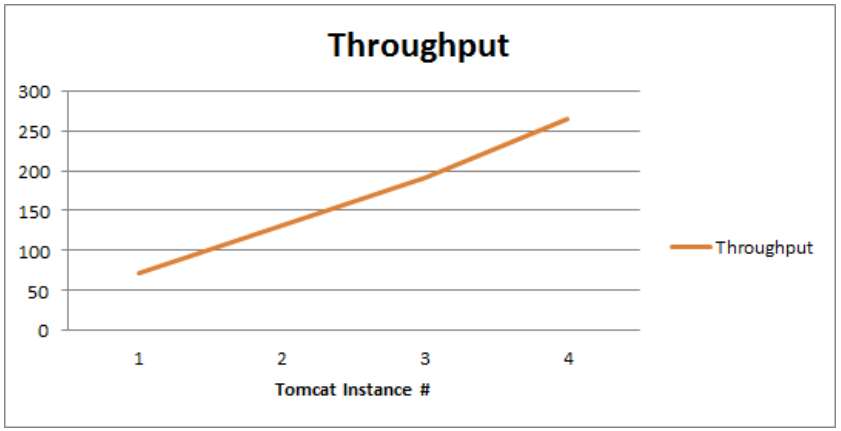


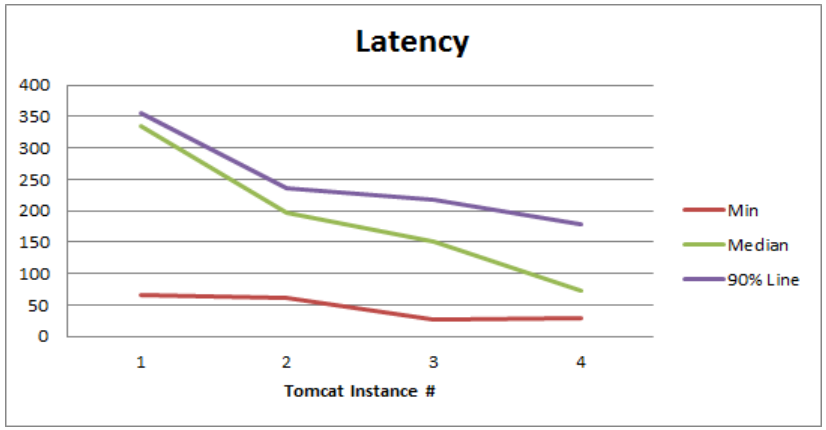
#	Query Type	Return Dataset	Query On Kylin (s)	Query On Hive (s)	Comments
1	High Level Aggregation	4	0.129	157.437	1,217 times
2	Analysis Query	22,669	1.615	109.206	68 times
3	Drill Down to Detail	325,029	12.058	113.123	9 times
4	Drill Down to Detail	524,780	22.42	6383.21	278 times
5	Data Dump	972,002	49.054	N/A	

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

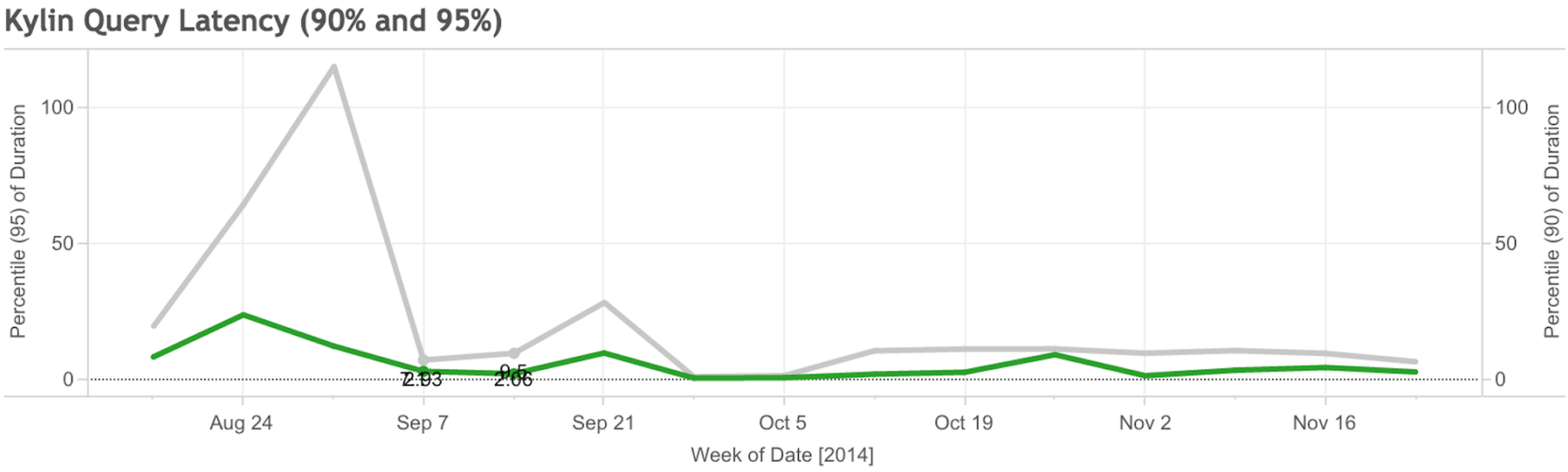




Linear scale out with more nodes

Performance - Query Latency

90%tile queries <5s



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

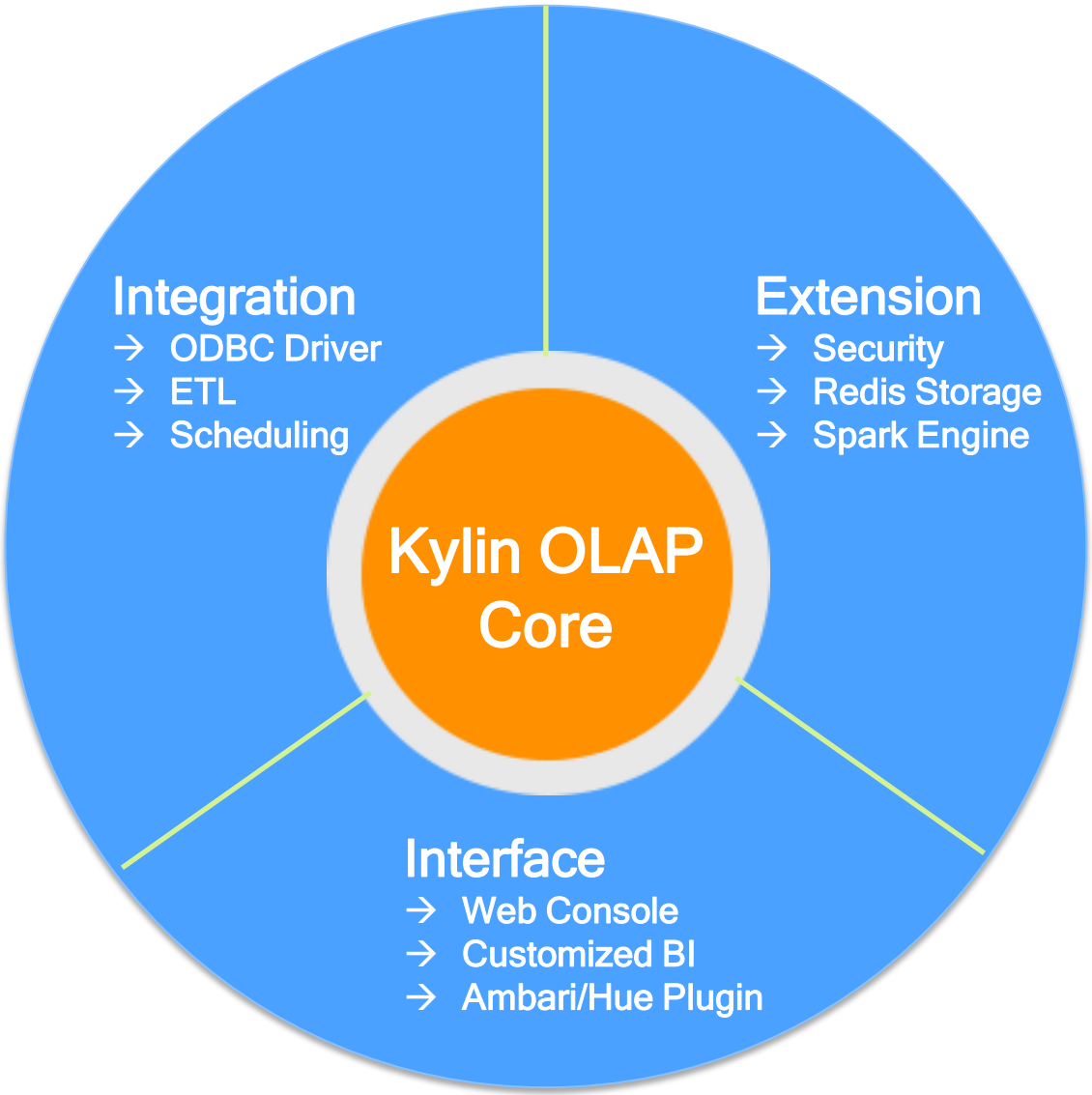
Agenda

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

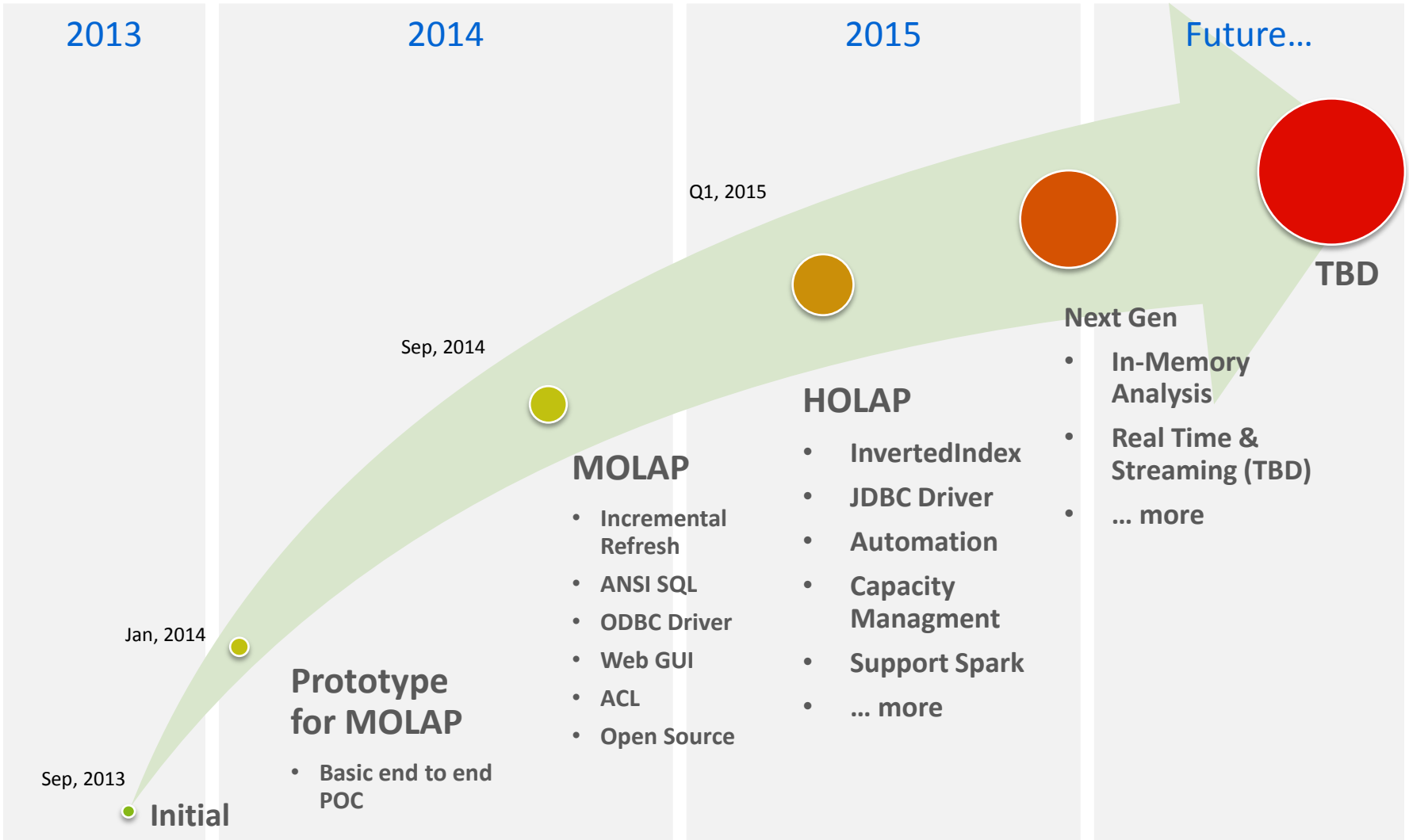
ebay inc

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



Kylin Evolution Roadmap



Open Source

- Kylin Site:
  - http://kylin.io
- Twitter:
  - @ApacheKylin
- Source Code Repo:
  - https://github.com/KylinOLAP
- Google Group:
  - Kylin OLAP

Thanks

http://kylin.io