# Architecture of Kylin 4.X


- Part I . Why Kylin on Parquet
  - Abstraction
  - Storage Engine Highlight
  - Query Engine Highlight
    - Kylin 3.X or lower version
    - Kylin 4.X
- Part II . How Kylin on Parquet
  - Cubing Step : Resources detect
    - Collect and dump the following three source info
    - Adaptively adjust spark parameters
  - Cubing Step : Build by layer
  - Cubiod Storage
    - Parquet file schema
- Part III . Reference


## Part I . Why Kylin on Parquet

### Abstraction

Compare to kylin architechture, the main changes include the following:

- Query Engine

  Fully distributed query engine. Query task will be sumbit to spark.

- Build Engine

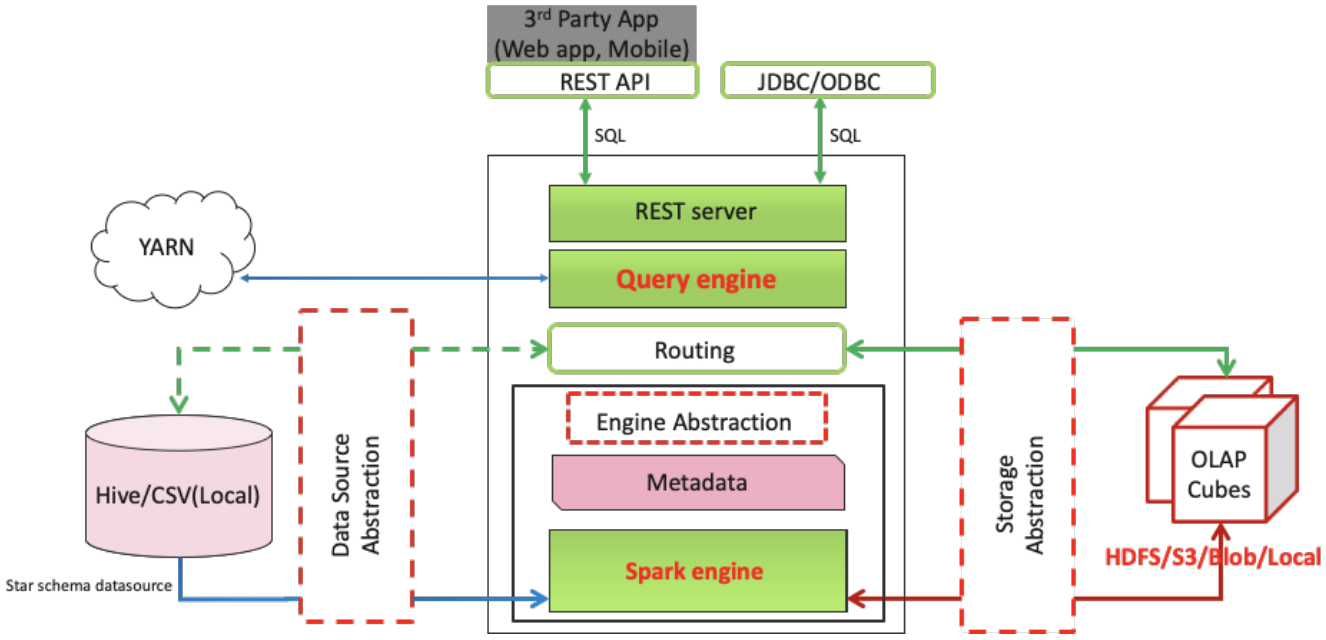
  Spark as the only build engine.

- Metadata

  Metadata still can be saved into HBase, RDBMS. *There's a little difference with kylin metadata, see more from MetadataConverter.scala.*

- Storage Engine

  Cuboids are saved into HDFS as parquet format(or other file system, no longer need HBase)




### Storage Engine Highlight

Currently(before Kylin 4.0), Kylin uses Apache HBase as the default storage. HBase Storage is very fast, while it also has some drawbacks:

- HBase is not real columnar storage;

- HBase has no secondary index; Rowkey is the only index;
- HBase has no encoding, Kylin has to do the encoding by itself;
- HBase does not fit for cloud deployment and auto-scaling;
- HBase has different API versions and has compatible issues (e.g, 0.98, 1.0, 1.1, 2.0);
- HBase has different vendor releases and has compatible issues (e.g, Cloudera's is not compatible with others);

Kylin 4.X is going to use Apache Parquet(with Spark) to replace HBase, because:

- Parquet is an open source columnar file format;
- Parquet is more cloud-friendly, can work with most FS including HDFS, S3, Azure Blob store, Ali OSS, etc;
- Parquet can integrate very well with Hadoop, Hive, Spark, Impala, and others;
- Support custom index;
- It is mature and stable;

~~The new build engine is faster and cost less storage space in file system. And the query engine also has a very good performance. See more with the performance from the following.~~

Benchmark Report for Parquet Storage

## Query Engine Highlight

### Kylin 3.X or lower version

- Query node calculate pressure, single bottleneck
- Hard to debug the code generated by Calcite

### Kylin 4.X

- Fully distributed
- Easy to debug and add breakpoint in each DataFrame

blocked URL

# Part II . How Kylin on Parquet

## Cubing Step : Resources detect

### Collect and dump the following three source info

If contains COUNT_DISTINCT measure(Boolean)

Resource paths(Array) we can using ResourceDetectUtils to Get source table infor(like source size, etc).

Table RDD leaf task numbers(Map). It's used for the next step -- Adaptively adjust spark parameters

### Adaptively adjust spark parameters

Turned on by default

Cluster mode only

Affect spark configuration property

```
kylin.engine.spark-conf.spark.executor.instances
kylin.engine.spark-conf.spark.executor.cores
kylin.engine.spark-conf.spark.executor.memory
kylin.engine.spark-conf.spark.executor.memoryOverhead
kylin.engine.spark-conf.spark.sql.shuffle.partitions
kylin.engine.spark-conf.spark.driver.memory
kylin.engine.spark-conf.spark.driver.memoryOverhead
kylin.engine.spark-conf.spark.driver.cores
```

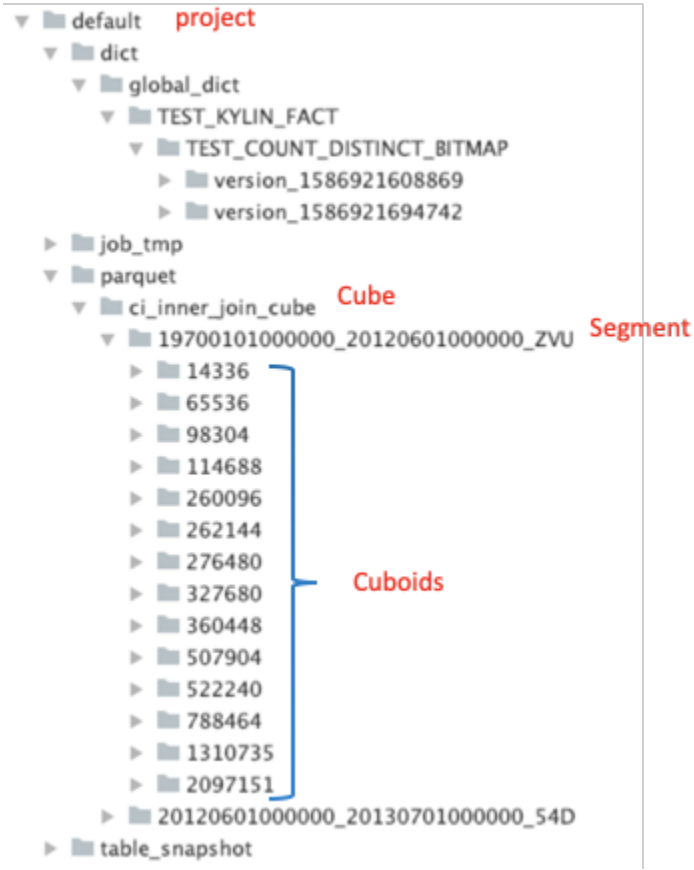
## Cubing Step : Build by layer

- Reduced build steps
  - From ten-twenty steps to only two steps
- Build Engine

- Simple and clear architecture
- Spark as the only build engine
- All builds are done via spark
- Adaptively adjust spark parameters
- Dictionary of dimensions no longer needed
- Supported measures
    - Sum
    - Count
    - Min
    - Max
    - TopN
    - CountDictinct(Bitmap, HyperLogLog)

## Cubiod Storage

The following is the tree of parquet storage directory in FileSystem. As we can see, cuboids are saved into path specified by Cube Name, Segment Name and Cuboid Id.




### Parquet file schema

If there is a dimension combination of columns[id, name, price] and measures[COUNT, SUM], then a parquet file will be generated:

Columns[id, name, age] correspond to Dimension[2, 1, 0], measures[COUNT, SUM] correspond to [3, 4]

# Part III . Reference

- Code Design Diagram