基于Druid的Kylin存储引擎实践

美团点评

康凯森

大纲

- Kylin On HBase 问题
- Kylin 新存储引擎探索
- Kylin On Druid

Kylin 服务现状

973 Cube

380万 次查询/天

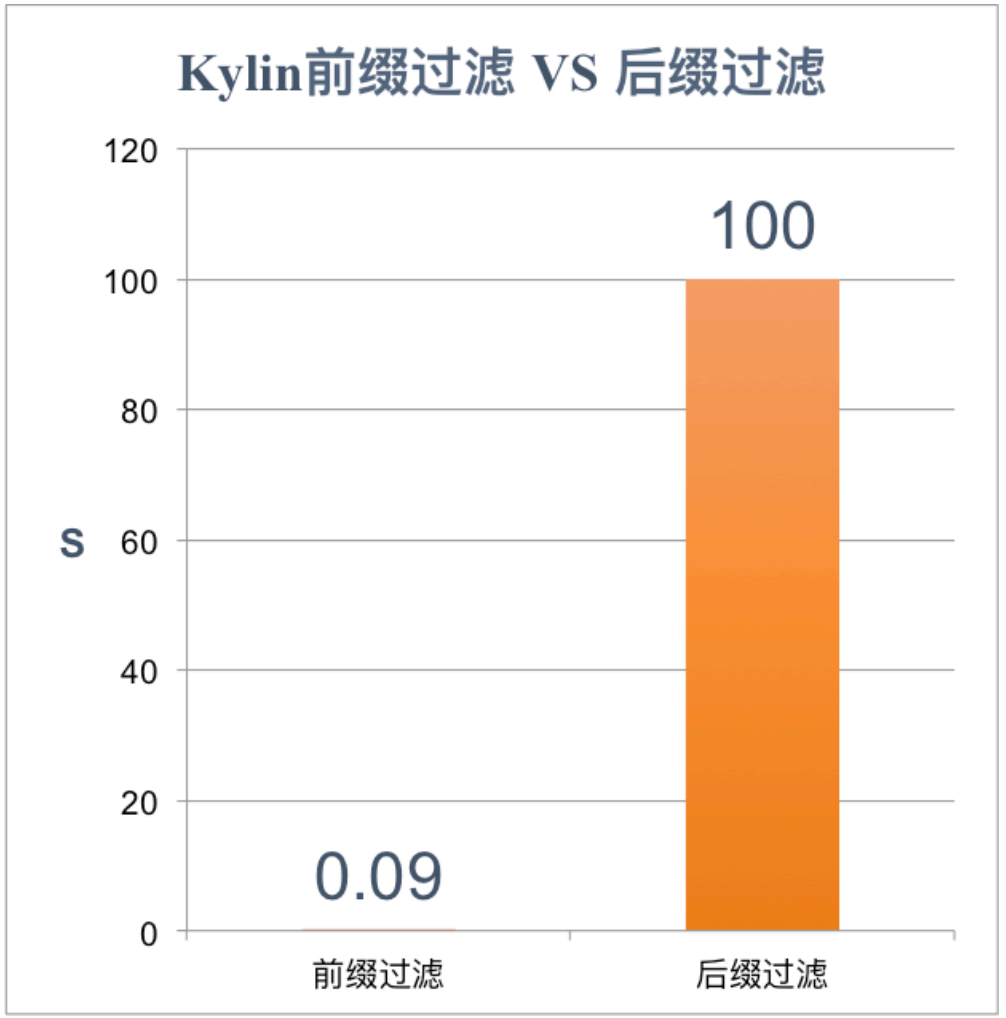
数据量	Cube存储	TP 50	TP 90
8.9万亿	971TB	200ms	1.2s

覆盖所有主要业务线

Kylin On HBase 问题

```
SELECT count(*) FROM T WHERE dt = x AND poi_id = x
```

- 同一SQL
- 同一Cube
- 同一集群



唯一不同点：Kylin维度在HBase的Rowkey中位置不同

Kylin On HBase 问题

- 每次读取所有指标列:

  Scan性能相比列存相差10倍多

- 单一Rowkey索引:

  前缀和后缀Filter性能相差上千倍

Kylin

Dimensions			Measures		
Year	City	Version	Price	Cost	Profit
2016	beijing	1	50	20	30
2016	shanghai	2	90	50	40
2017	beijing	1	80	50	30

HBase

KeyValue
KeyValue
KeyValue
KeyValue

Sort

RowKey	Value
00000111+2016beijing1	502030
00000111+2016shanghai2	905040
00000111+2017beijing1	805030

Kylin On HBase 问题的可能解

- 列存：更高效的Scan

- 索引：更高效的Filter

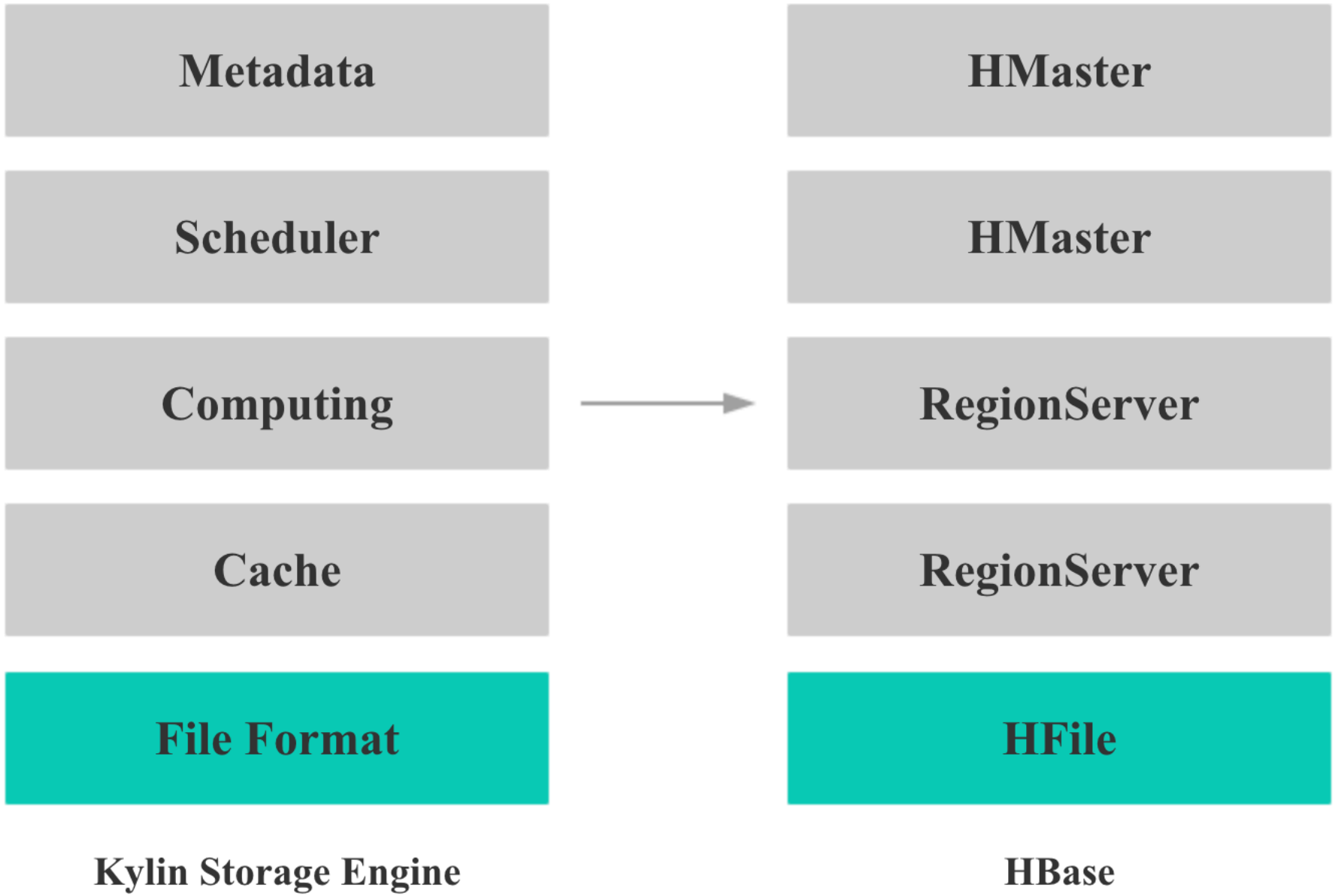
列存的优点

- 高效IO：只读取必需的列
- 高效编码和压缩：每列数据类型一致，格式一致
- 向量化执行

问题如何解：

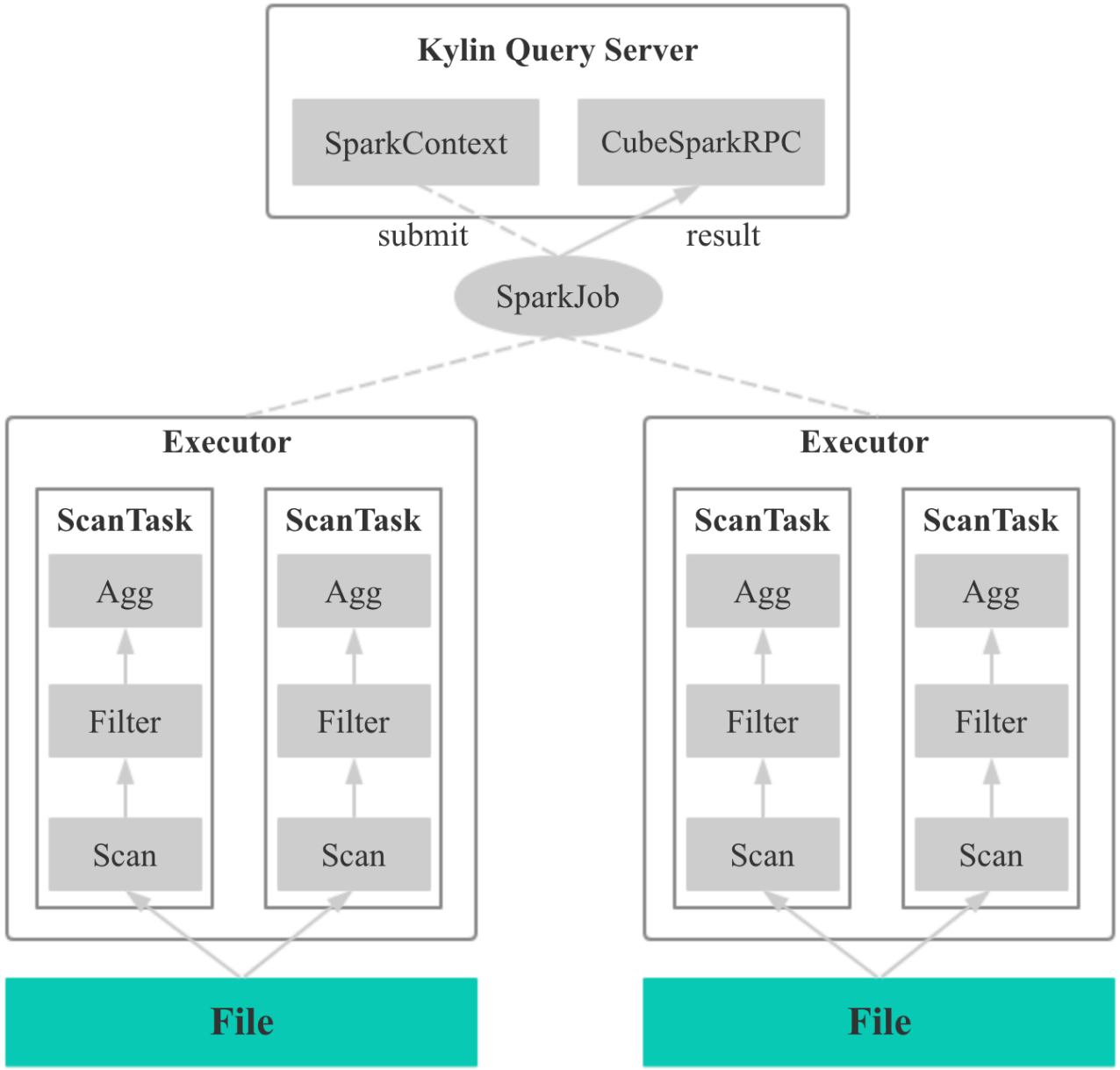
一个更高效的存储引擎

Kylin 存储引擎模块



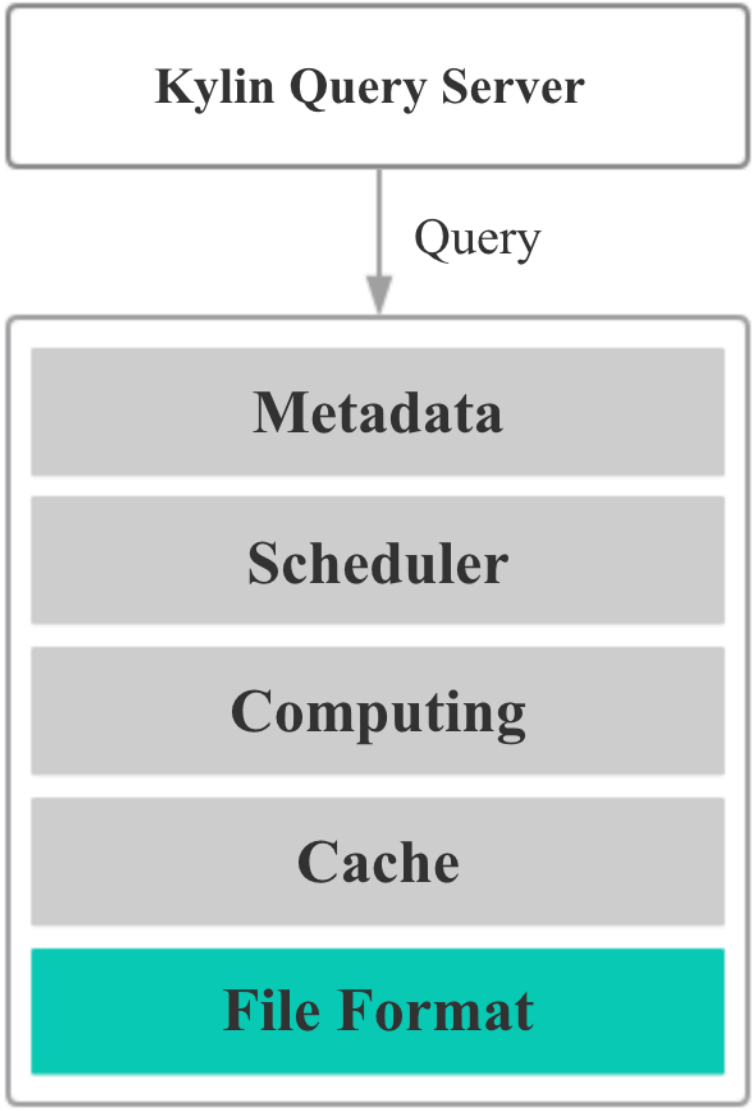
Kylin 新存储引擎思路一

基于Spark + File演进



Kylin 新存储引擎思路二

基于File Format演进新存储引擎

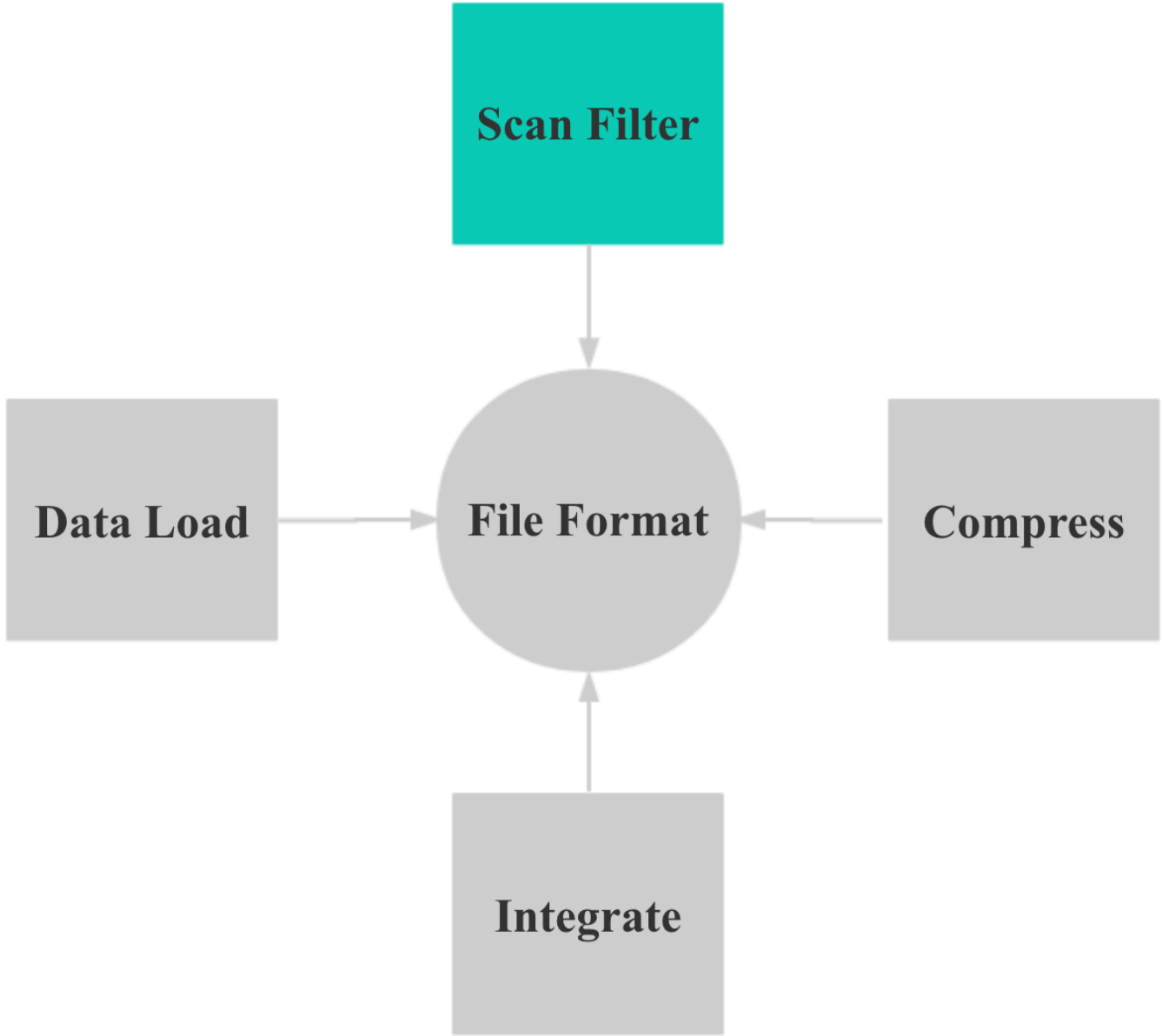


New Storage Engine

第一步

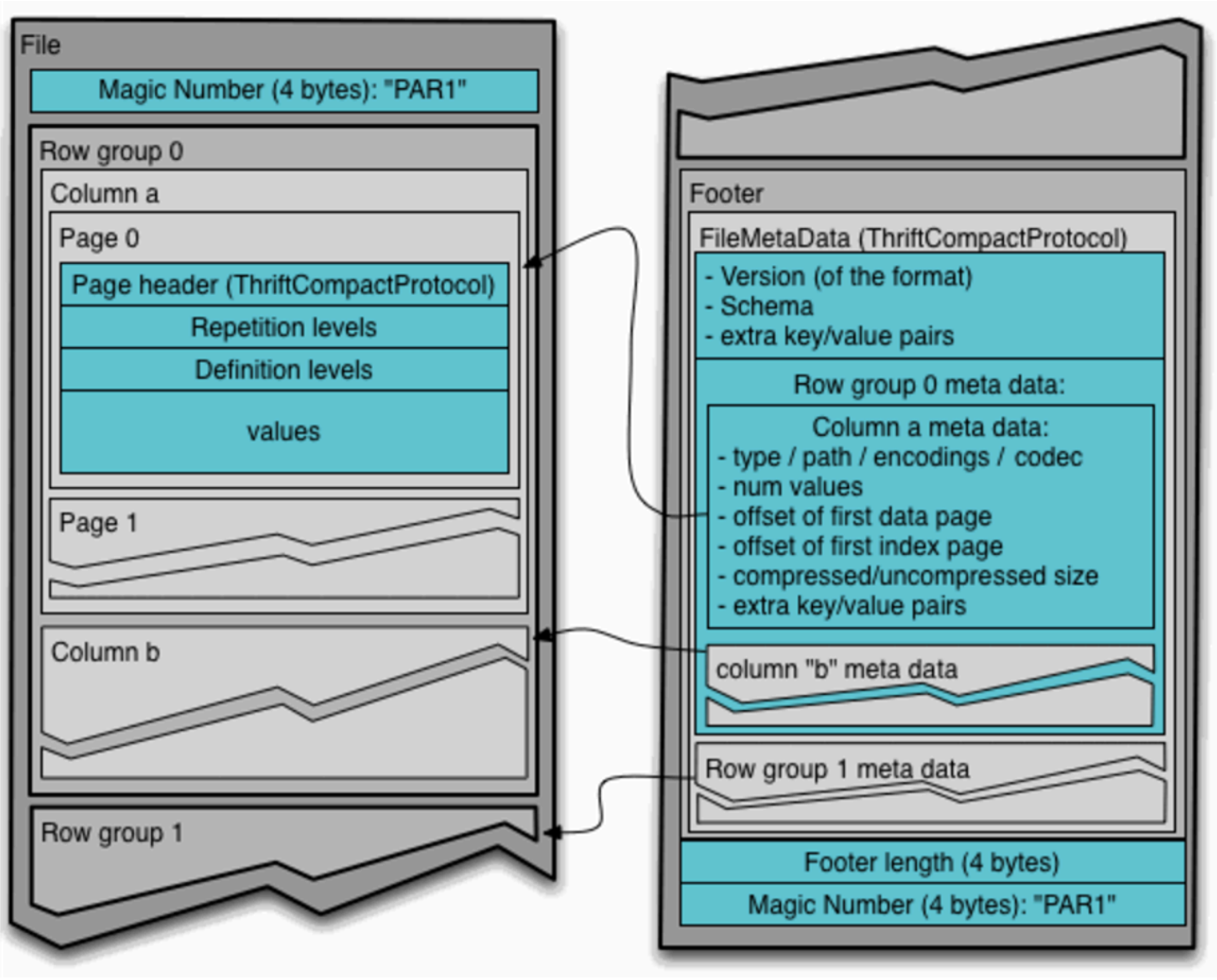
探索一个适合Kylin的存储格式

Kylin 存储格式评测标准



Parquet File Format 2.3.1

- 列存
- Min，Max索引
- Dictionary 过滤
- 无 PageIndex

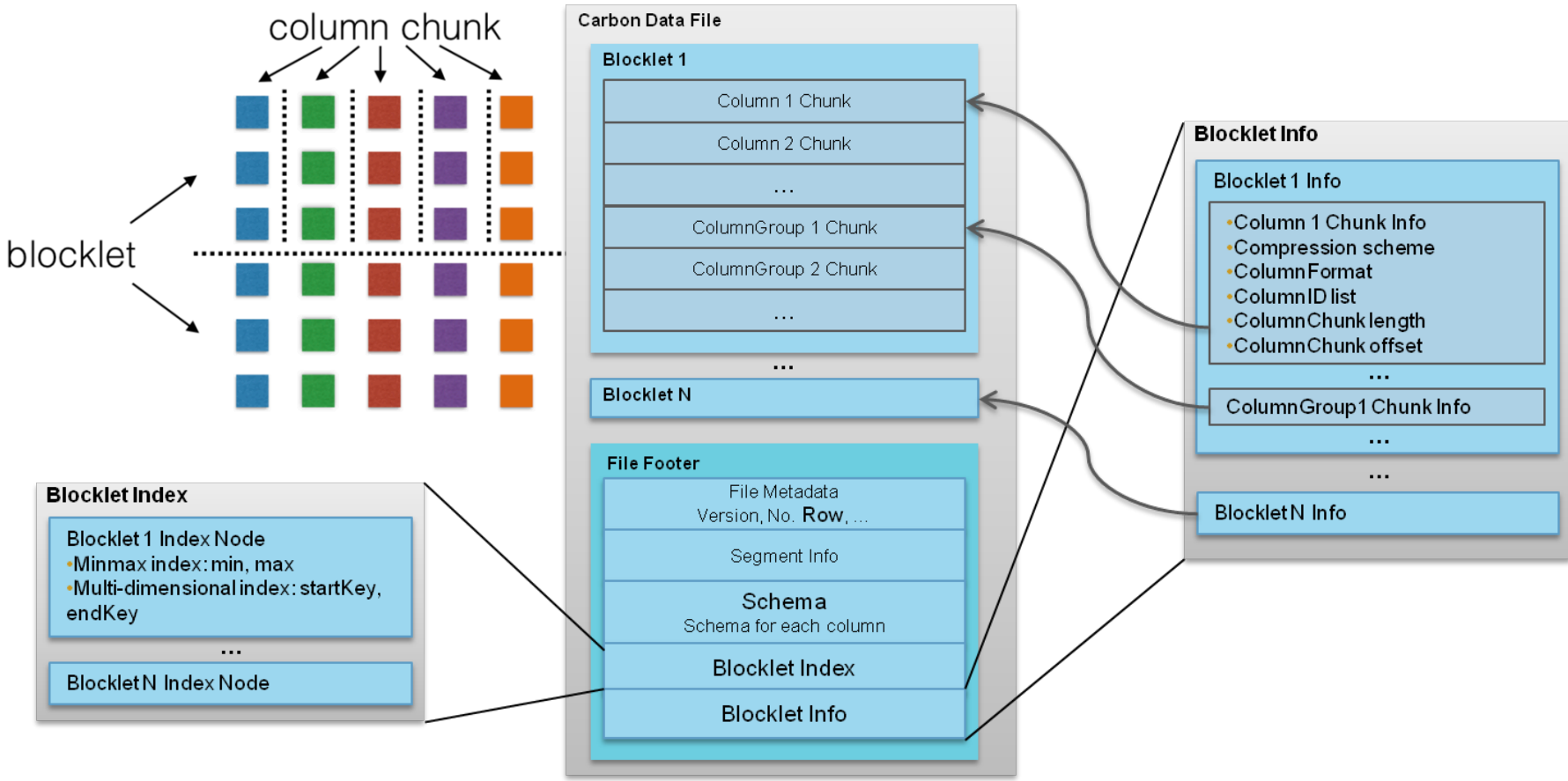


Kylin On Parquet POC

- 构建：单节点每分钟导入百万条数据
- 压缩：存储空间比HBase节省10%
- Scan 查询：部分列时是HBase的2~6倍；全部列时是HBase的80%
- Filter 查询：前缀查询明显不如HBase，后缀查询有一定优势

CarbonData File Format 1.0

- 列存
- MDK索引
- Min，Max索引
- 倒排索引

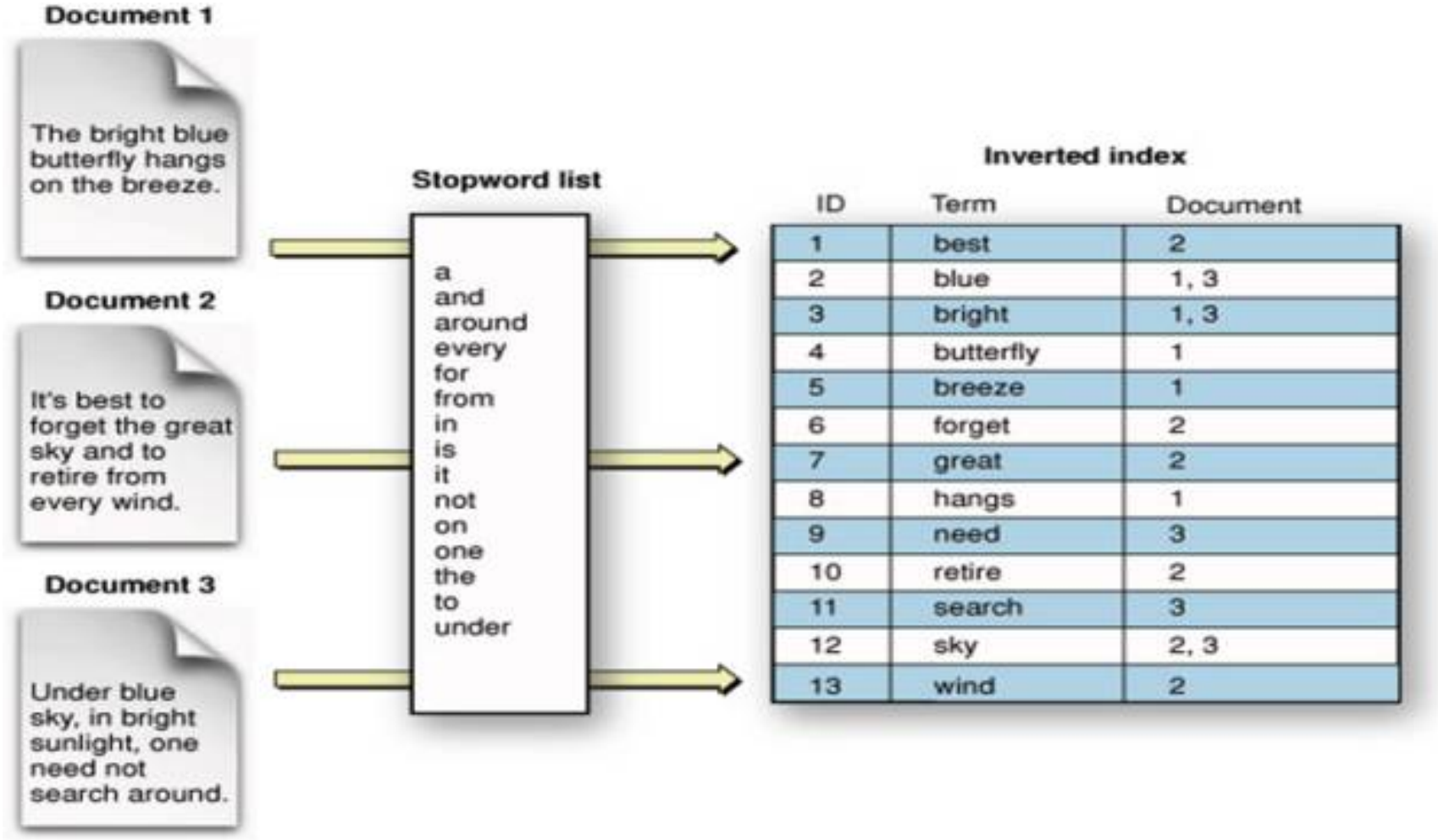


Why Not Kylin On Carbondata 1.0

- 没有OutputFormat
- 和 Spark 耦合较深
- 尚不成熟(2017-05)

Lucene 7.0

- 列存
- 倒排索引
- MMap



Kylin On Lucene POC

- 构建：导入速度是HBase的1/3
- 压缩：存储是HBase的4倍
- Scan查询：部分列时性能优于HBase
- Filter查询：过滤性能明显优于HBase, 不过点查询性能比HBase差

Druid Segment File Format

- 列式存储
- Bitmap倒排索引
- MMap

00000.smoosh

meta.smoosh

file	start_offset	end_offset
_time	0	100
M1	100	200
M2	200	300
D1	300	400
D2	400	500
index.drd	500	600
metadata.drd	600	700

Druid Bitmap 倒排索引

Raw Column

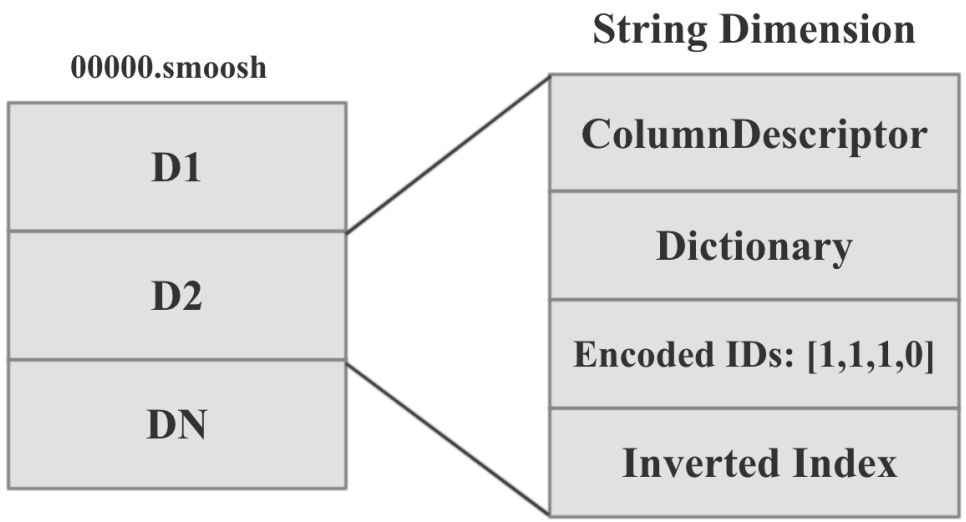
D2
meituan
meituan
meituan
dianping

Dictionary

Value	Id
meituan	0
dianping	1

Inverted Index

Id	Bitmap
0	[1,1,1,0]
1	[0,0,0,1]



Kylin On Druid POC

前缀过滤

用例	HBase耗时	Druid耗时	Druid访问记录数
Q1	42.5s	4.64s	10162248
Q2	15.3s	1.78s	1533488
Q3	0.95s	0.8s	56392
Q4	0.085s	0.15s	240

后缀过滤

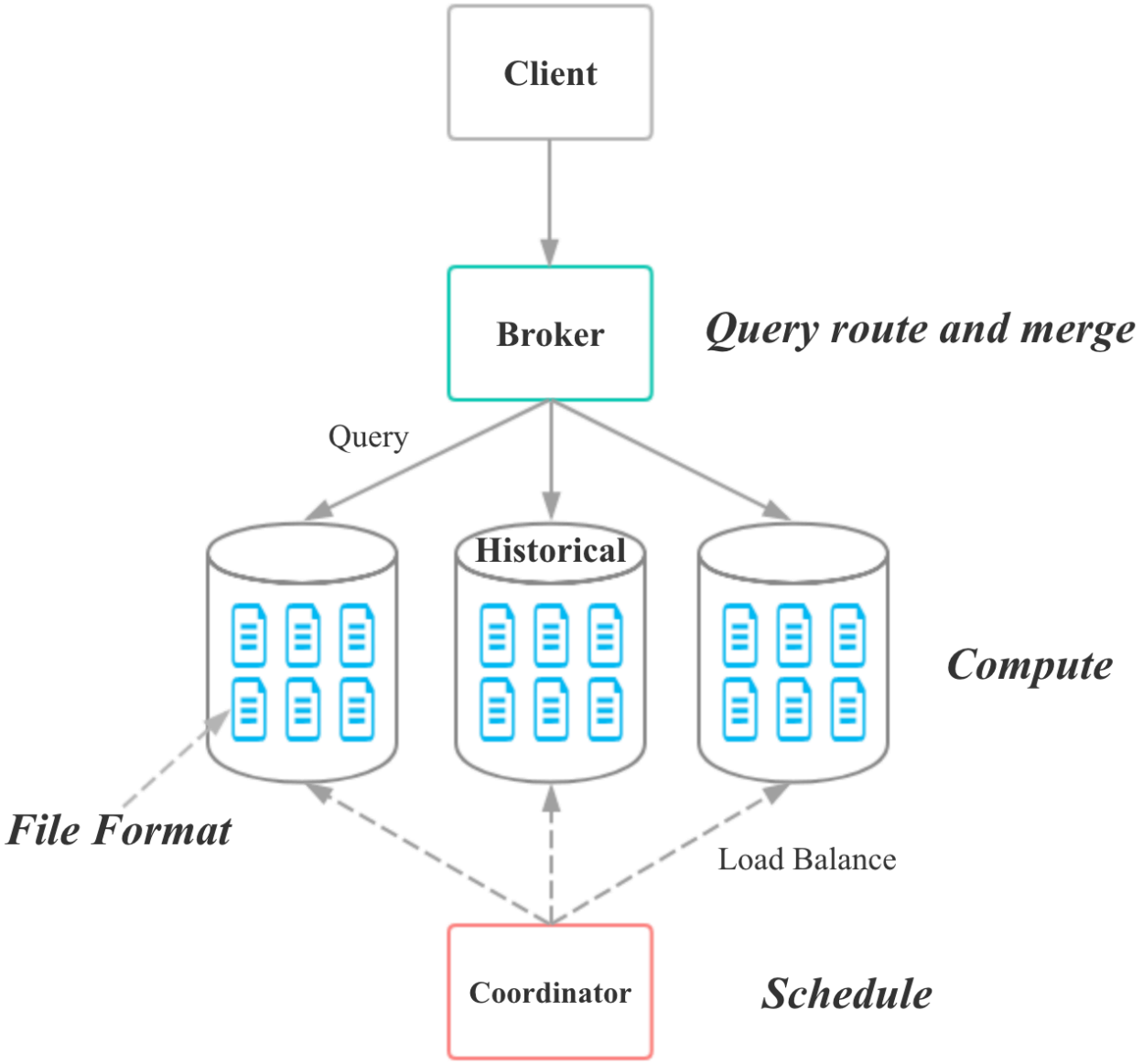
用例	HBase耗时	Druid耗时	Druid访问记录数
Q5	85s	0.55s	152416
Q6	120s	1.7s	1042910

Why Kylin On Druid

- Parquet：索引粒度较粗，Filter 性能不足
- CarbonData：与Spark耦合较深，集成难度大
- Lucene：存储膨胀率较高

Why Kylin On Druid

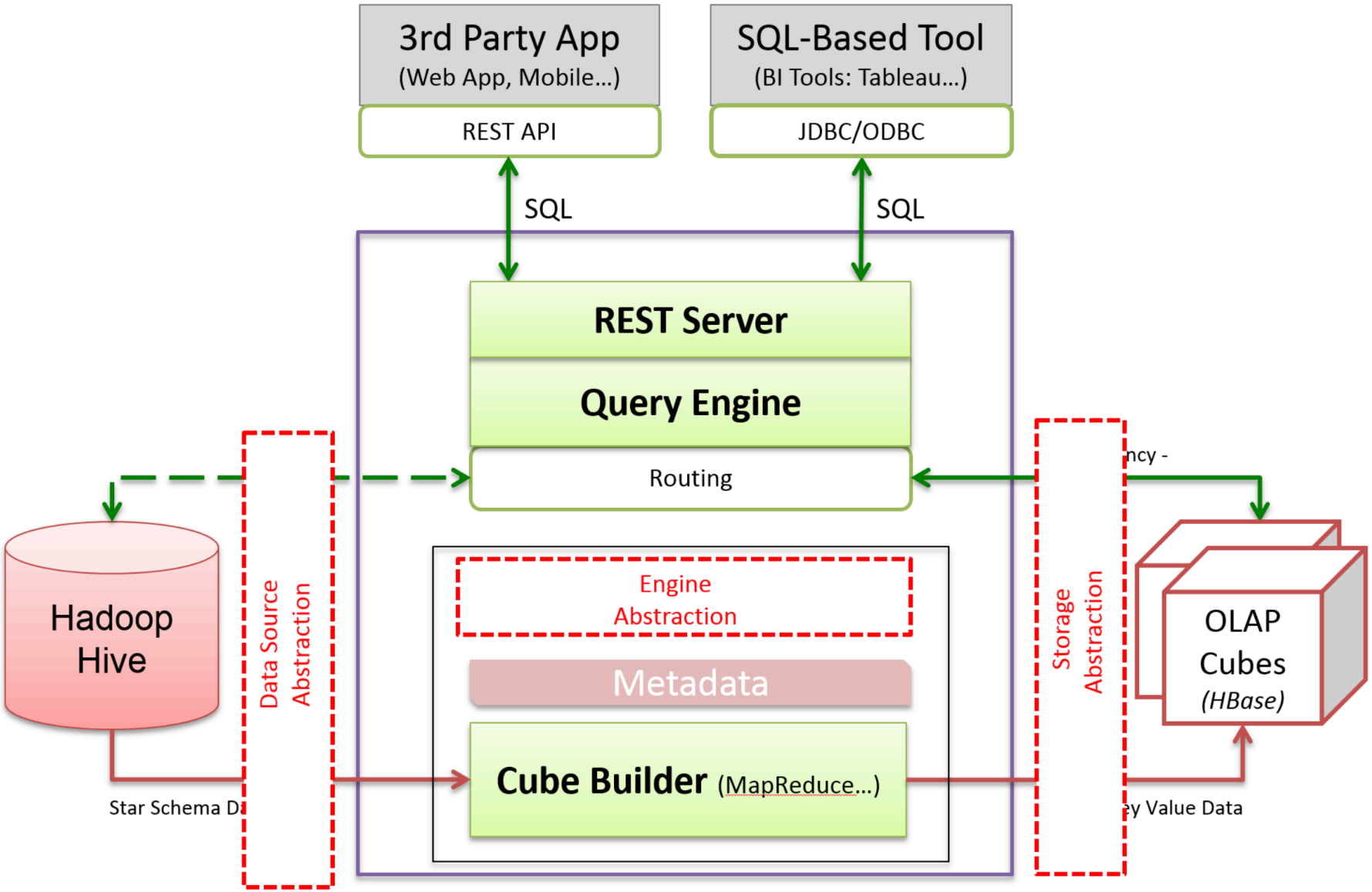
- Scan Filter 性能
- 开发效率
- 风险可控



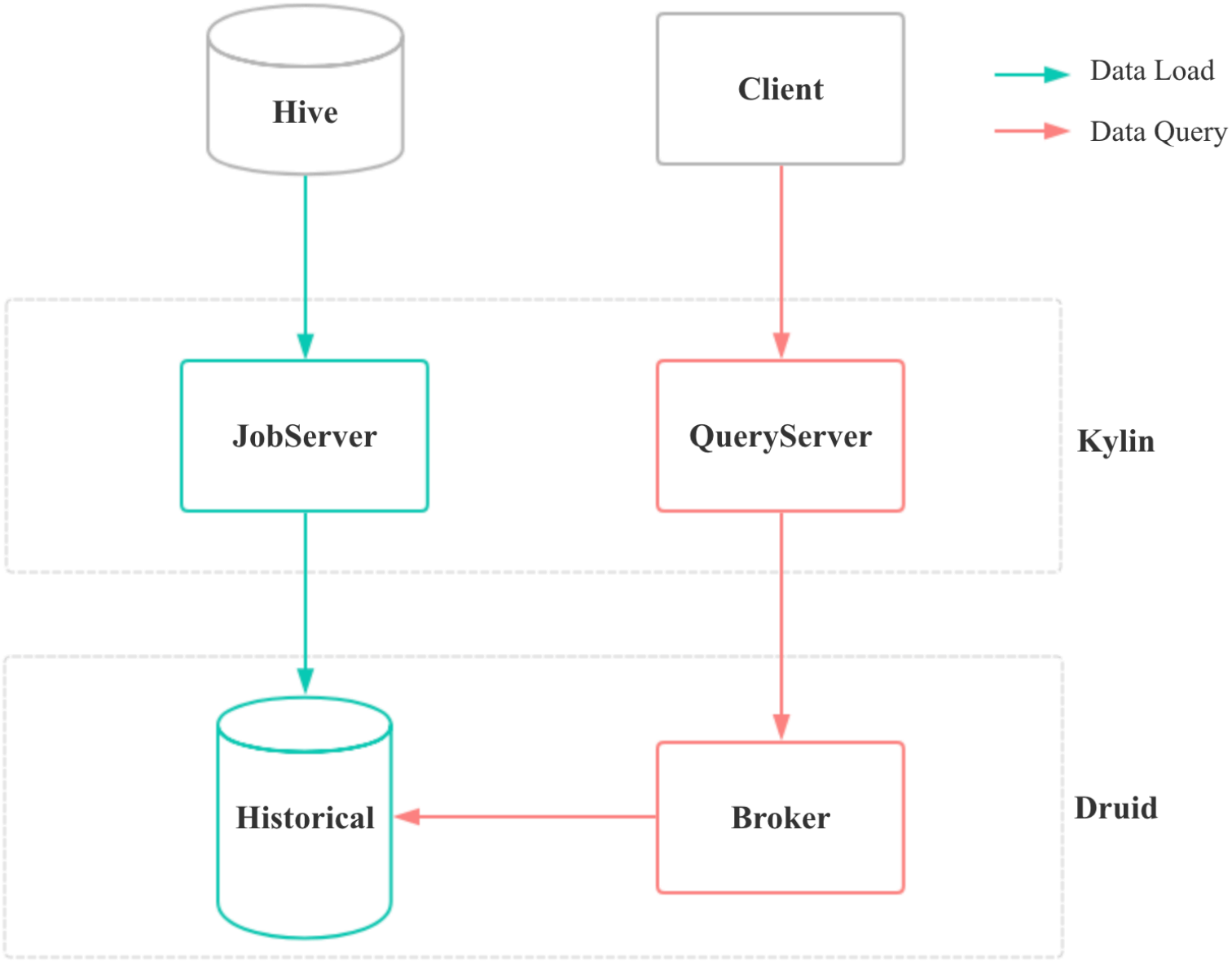
大纲

- Kylin On HBase 问题
- Kylin 新存储引擎探索
- Kylin On Druid

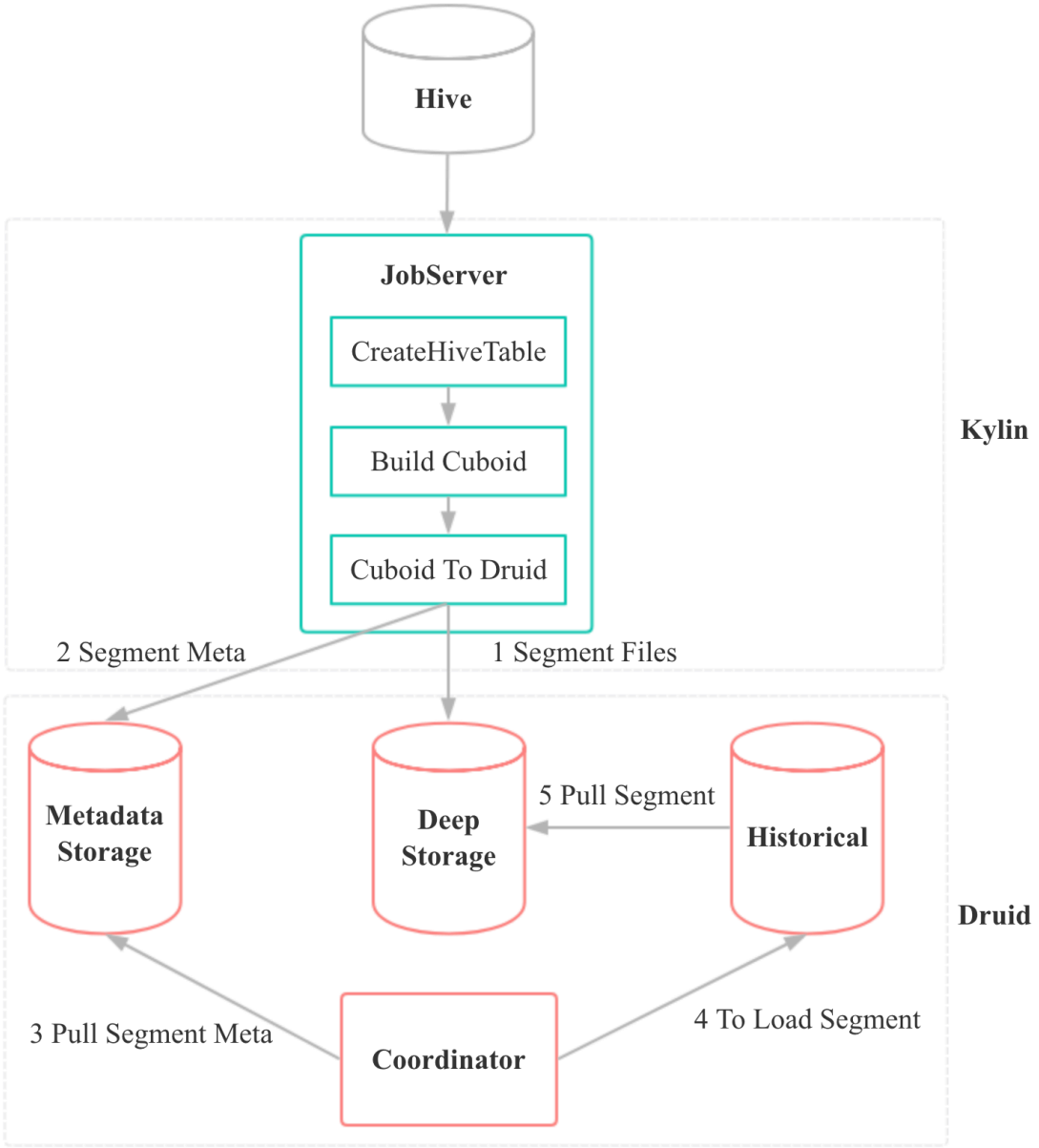
Kylin 可插拔架构


3rd Party App
(Web App, Mobile...)
SQL-Based Tool
(BI Tools: Tableau...)
REST API
JDBC/ODBC
SQL
SQL
REST Server
Query Engine
Routing
Hadoop
Hive
Data Source
Abstraction
Engine
Abstraction
Metadata
Cube Builder (MapReduce...)
Storage
Abstraction
OLAP
Cubes
(HBase)
Star Schema D
y Value Data

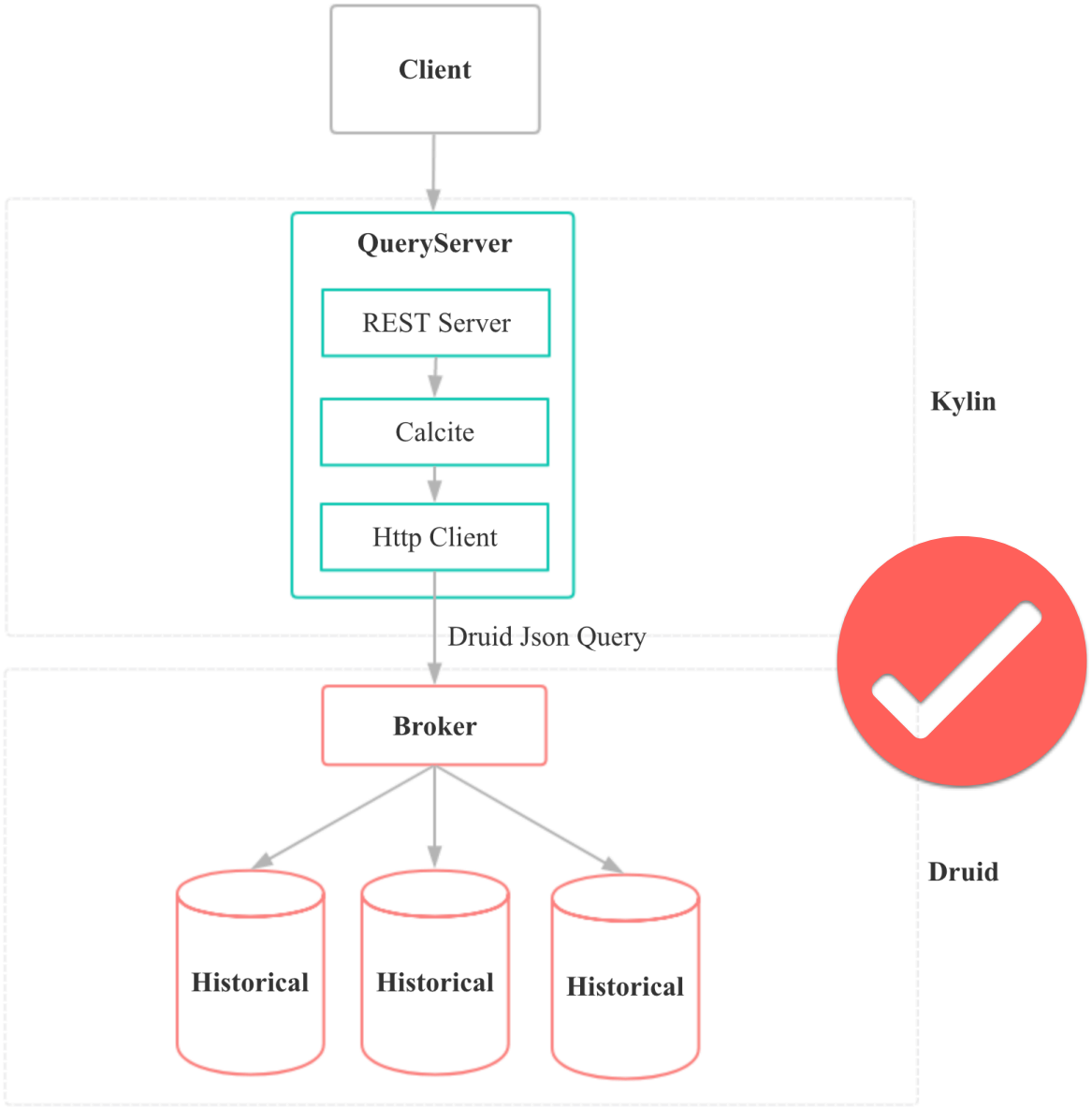
How KOD：KOD架构

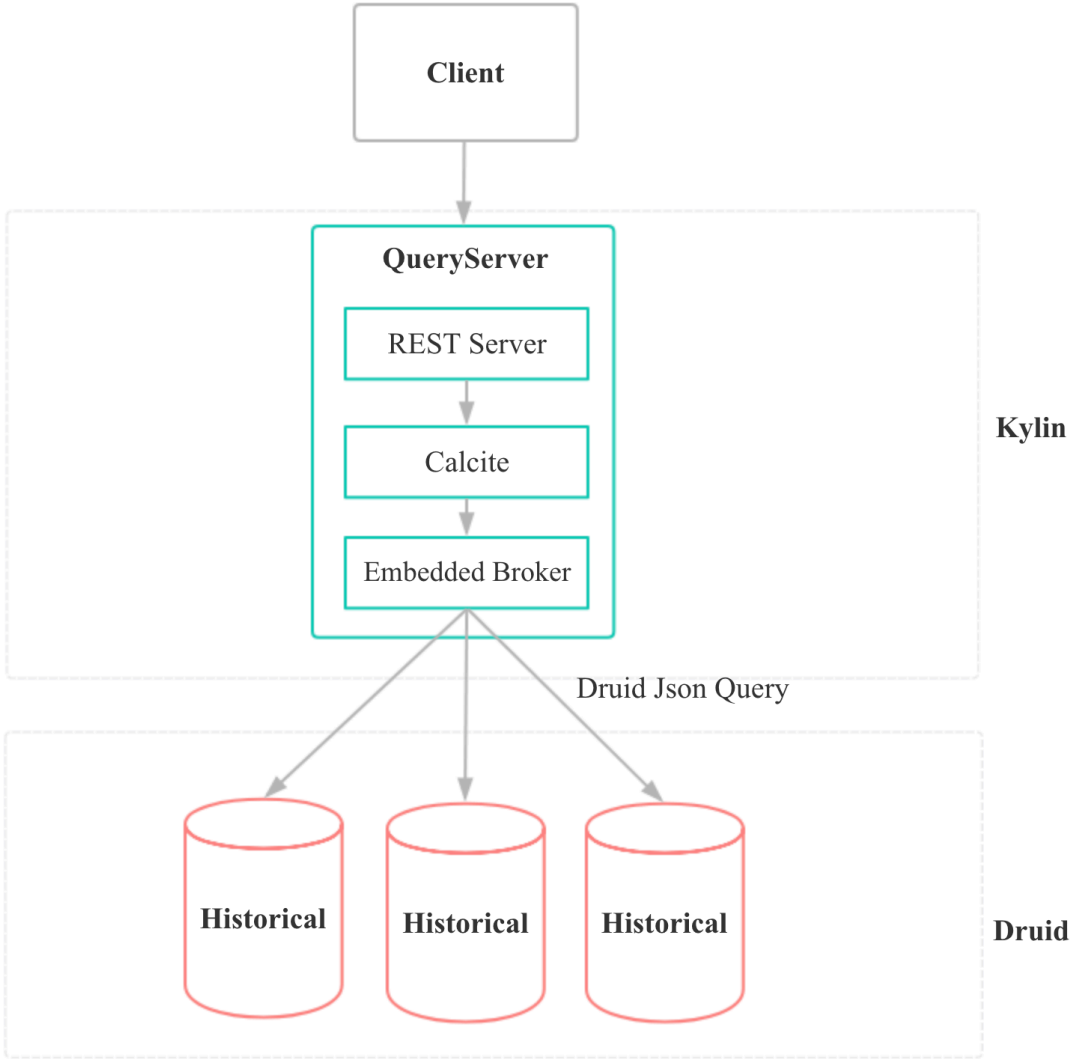


How KOD： Data Load



How KOD： Data Query





How KOD

- Schema映射
- Cube 构建侧适配
- Cube 查询侧适配
- 运维工具适配

如何为Kylin增加一个新的存储引擎

How KOD： Schema映射

Kylin

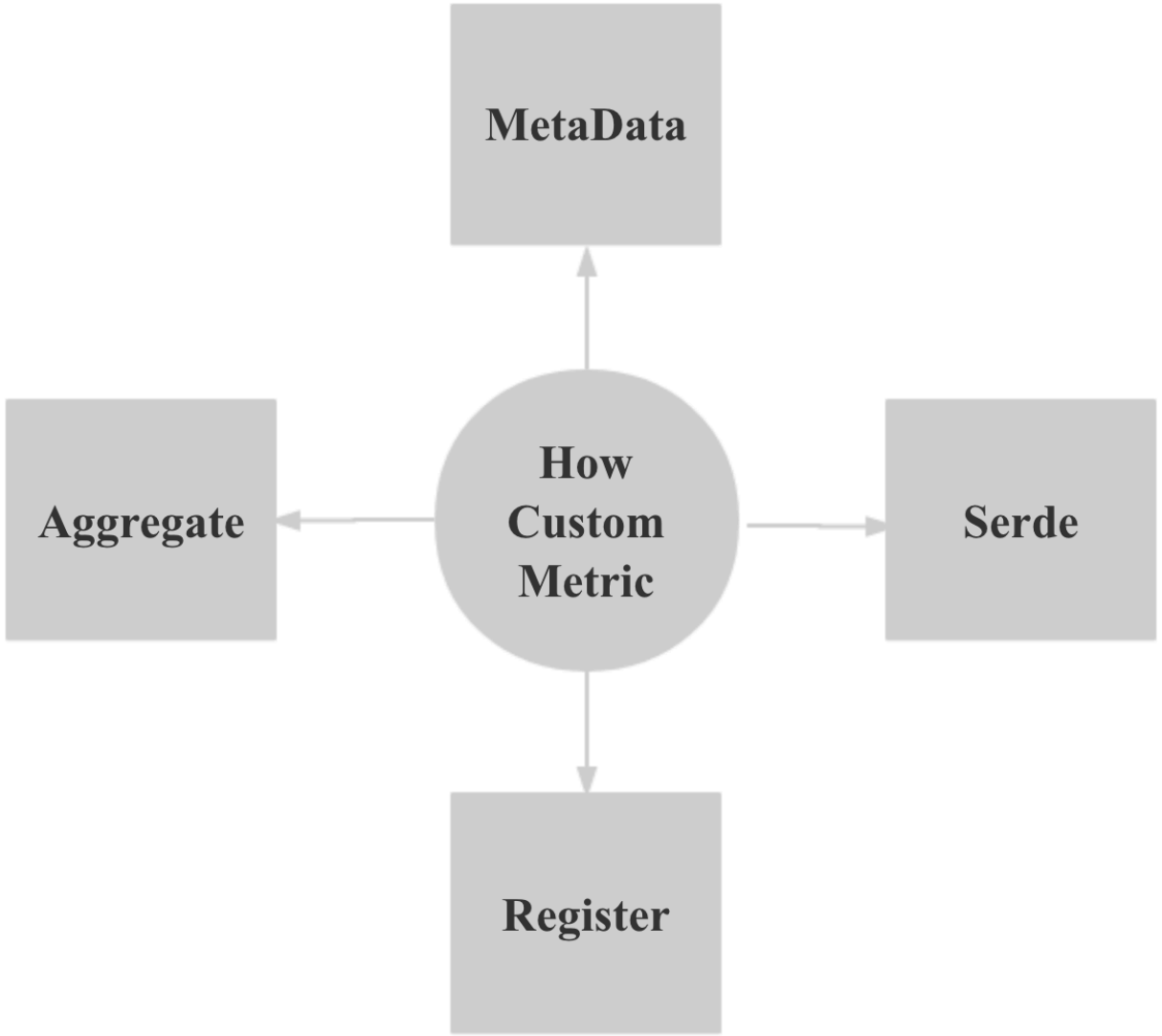
Dimensions			Measures		
Year	City	Version	Price	Cost	Profit
2016	beijing	1	50	20	30
2016	shanghai	2	90	50	40
2017	beijing	1	80	50	30

Druid

Cuboid	Year	City	Version	Price	Cost	Profit
00000111	2016	beijing	1	50	20	30
00000111	2016	shanghai	2	90	50	40
00000111	2017	beijing	1	80	50	30

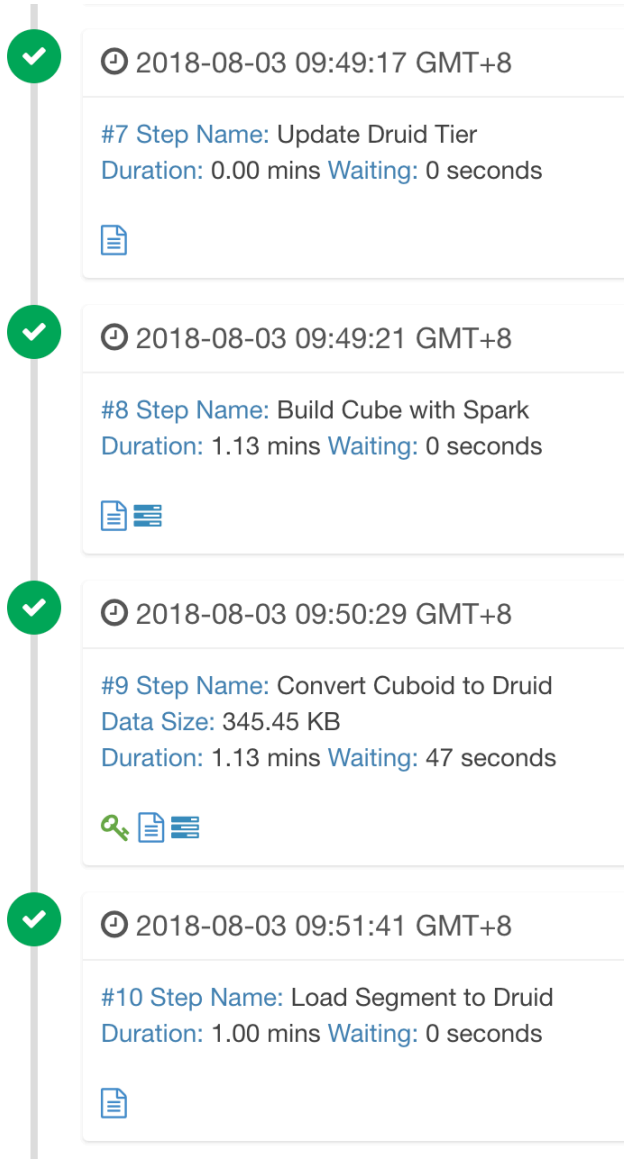
How KOD：Druid指标扩展

- Kylin 精确去重指标
- Kylin ExtendColumn指标
- Decimal指标

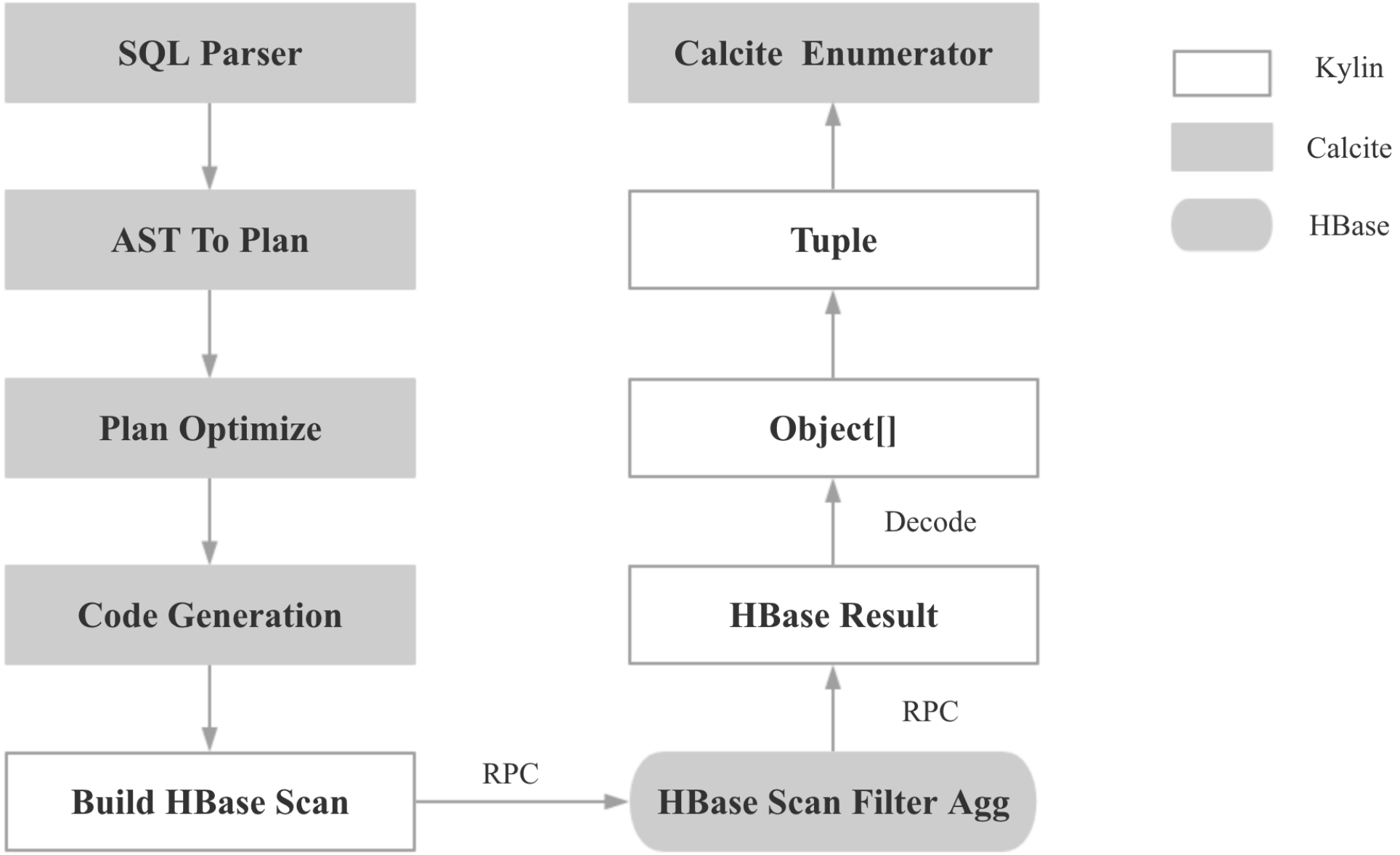


How KOD：Cube 构建侧适配

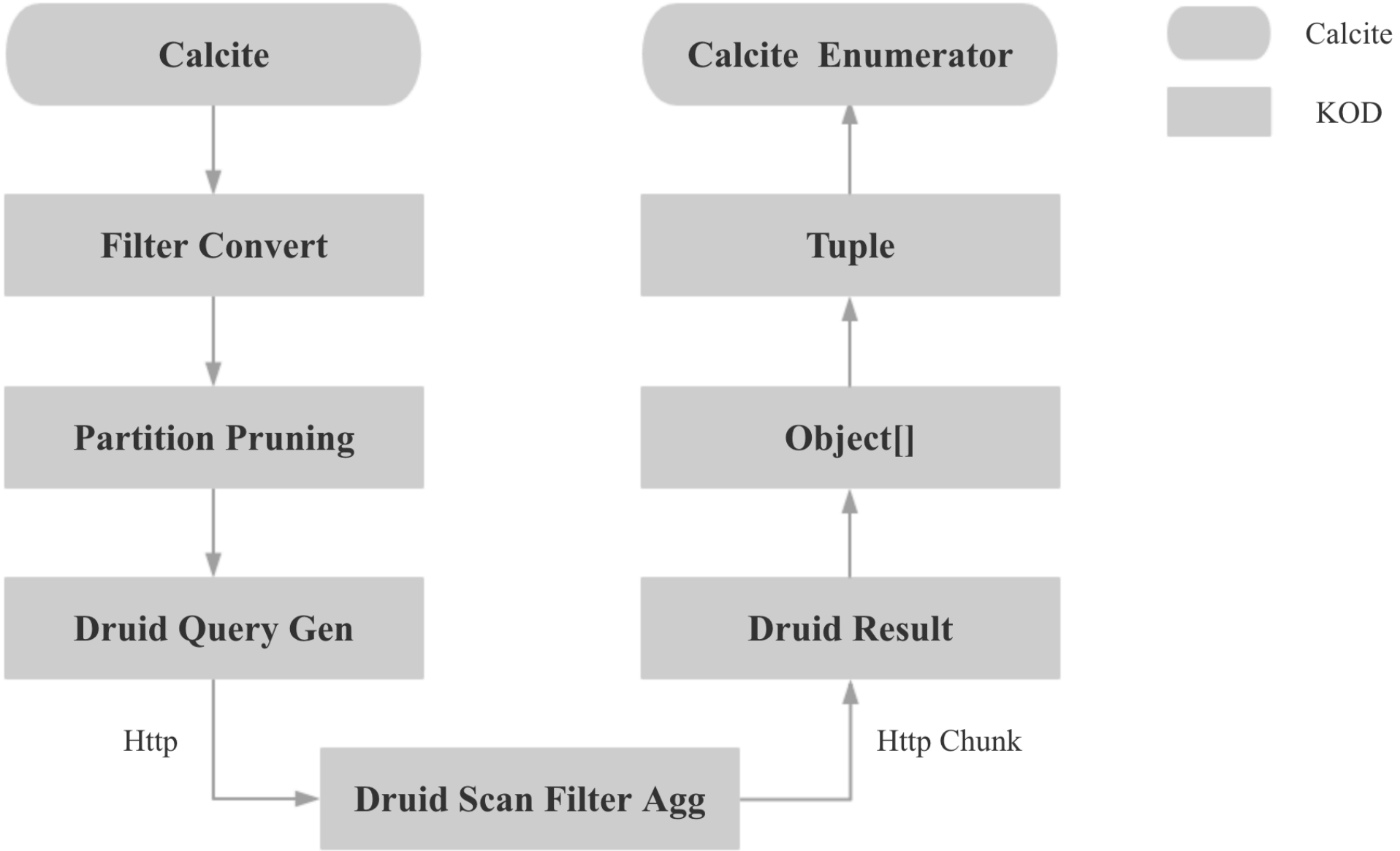
- Update Druid Tier: 支持业务隔离
- Convert Cuboid To Druid: **Reducer内存优化**
- Load Segment To Druid: **Load并行优化**



Kylin 查询流程


SQL Parser
AST To Plan
Plan Optimize
Code Generation
Build HBase Scan
RPC
HBase Scan Filter Agg
RPC
HBase Result
Decode
Object[]
Tuple
Calcite Enumerator
Kylin
Calcite
HBase

How KOD：Cube 查询侧适配



How KOD：运维工具适配

- Cube Migrate 工具
- Storage Cleanup 工具
- Cube Purge，Drop，Retention 操作

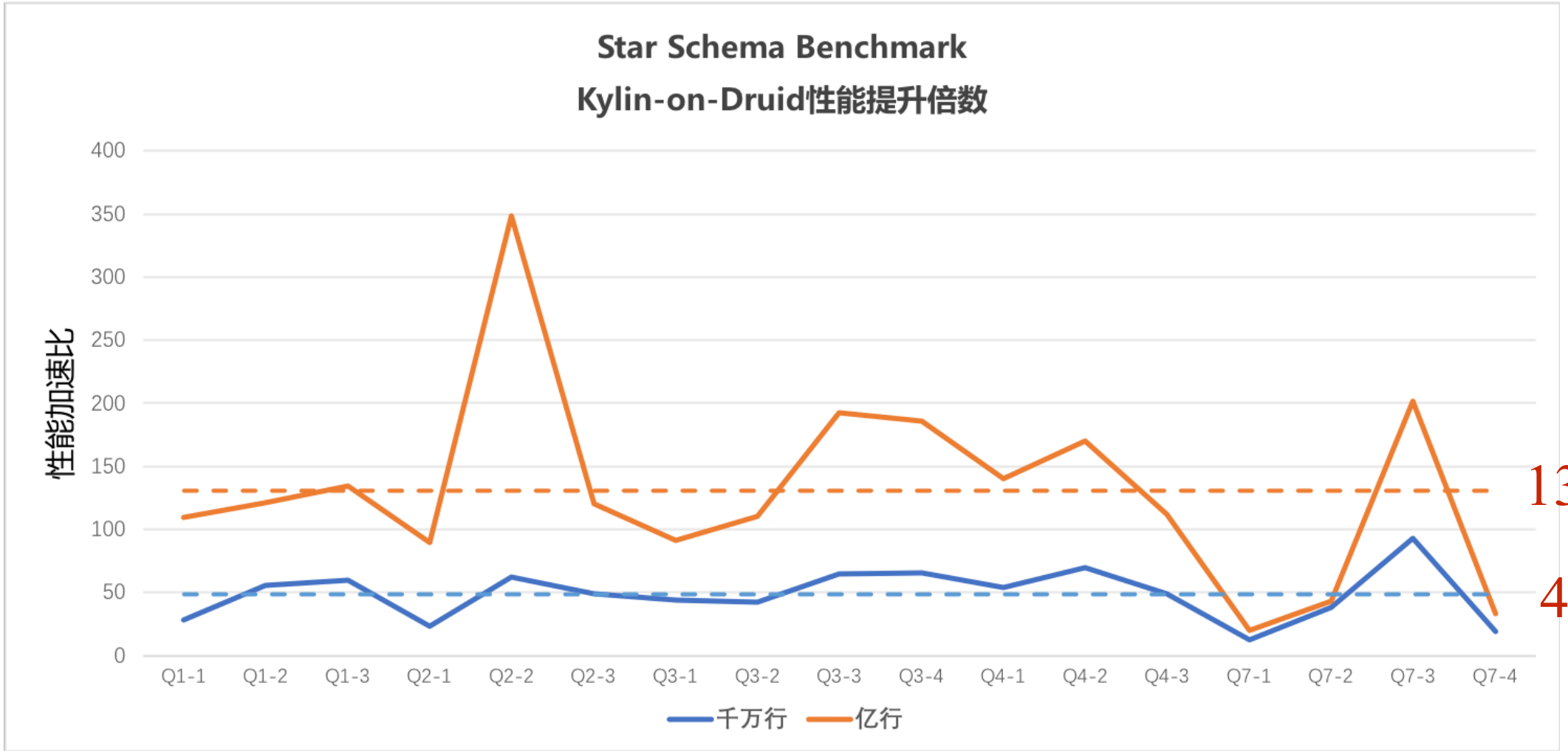
KOD Performance: SSB 测试

- SSB是一个标准的OLAP测试集
- Cube Schema: 28个维度，43个指标
- 数据规模：千万级，亿级
- 4台物理机：40CPU，128G MEM
- KOD和Kylin都只计算了Base Cuboid

```
SELECT sum(lo_revenue2) AS revenue
FROM v2_lineorder
JOIN "DATE" ON lo_orderdate = d_datekey
WHERE d_yearmonthnum = 199401
AND lo_discount BETWEEN 4 AND 6
AND lo_quantity BETWEEN 26 AND 35;
```

测试目的：对比KOD和Kylin的现场计算能力

KOD Performance: SSB 测试



KOD的现场计算能力相比Kylin有了两个数量级的提升

KOD Performance: 第一批上线

Cube总数	KOD总数	KOD覆盖率
707	24	3.4%

资源维度	上线前	上线后	变化
Cube存储	1.90 TB	406.72 GB	-79%
构建cpu消耗	4,244,946	3,449,028	-19%
构建mem消耗	14,192,470,039	10,186,894,652	-28%

性能指标	降低50%以上	降低10~50%	无明显变化	增加10~50%	增加50%以上
tp99	69%	15%	8%	8%	0%
tp999	62%	31%	0%	8%	0%

KOD在查询性能提升的同时，存储和计算资源也有了明显的下降。

KOD 易用性提升

Cube 优化复杂

- 聚集组
- 强制维度
- 联合维度
- 层次维度
- 衍生维度
- Extended Column
- HBase RowKey 顺序

Only Base Cuboid

（亿级及以下数据规模）

KOD 特性

- 和Kylin完全兼容：SQL，Web 等
- 分区预过滤
- 查询时无需加载字典
- 存储层支持业务隔离
- 亿级及以下数据只需构建Base Cuboid

KOD 未来规划

- 更高效，更精简的Cube构建流程
- 优化高基数列点查询的场景
- 支持在线Schema变更

总结

- Kylin On HBase 问题
- Kylin 新存储引擎探索
- Kylin On Druid

Parquet With PageIndex

Performance Gains

HDFS I/O

⇩ Bytes read

Point Queries : 98%

Range Scans: Depends on Range

Full Scans: -0.03%

Memory Utilization

⇩ Number of tuple materialized

Point Queries : 95%

Range Scans: Depends on Range

Full Scans: -1%

CPU Time

⇩ Number of scalar expressions evaluated

Point Queries : 98%

Range Scans: Depends on Range

Full Scans: -2%

Kylin On Parquet Maybe Still Worth Trying

Thanks