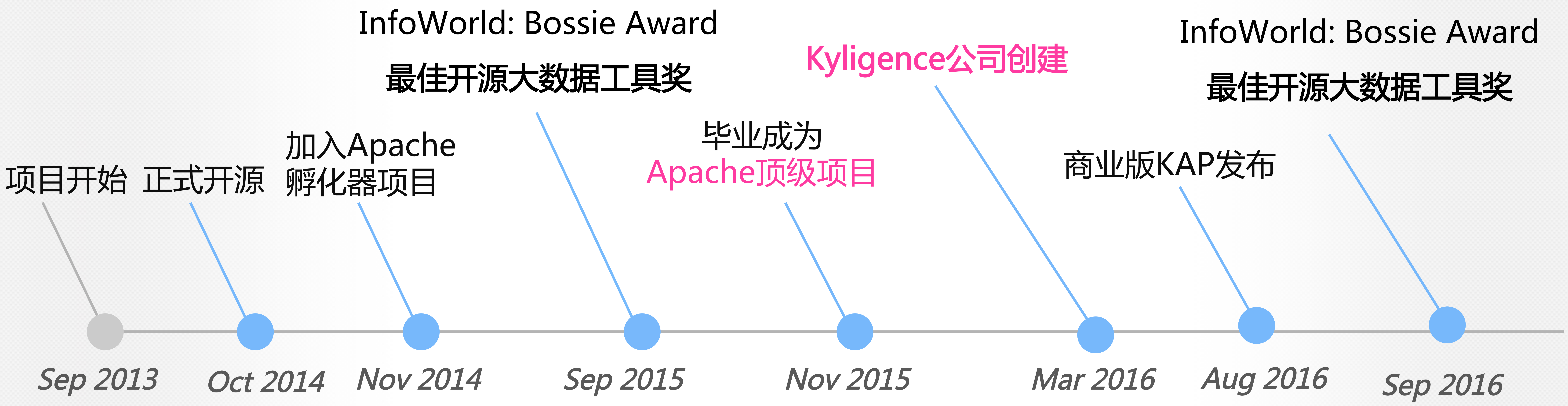
Apache Kylin 2.0
技术解密之 Spark Cubing

马洪宾 | ma@kyligence.io

PMC member of Apache Kylin
Kyligence Inc. 技术合伙人 & 高级架构师

Apache Kylin 历史



项目开始

Sep 2013

正式开源

Oct 2014

加入Apache
孵化器项目

Nov 2014

InfoWorld: Bossie Award
最佳开源大数据工具奖

Sep 2015

毕业成为
Apache顶级项目

Nov 2015

Kyligence公司创建

Mar 2016

商业版KAP发布

Aug 2016

InfoWorld: Bossie Award
最佳开源大数据工具奖

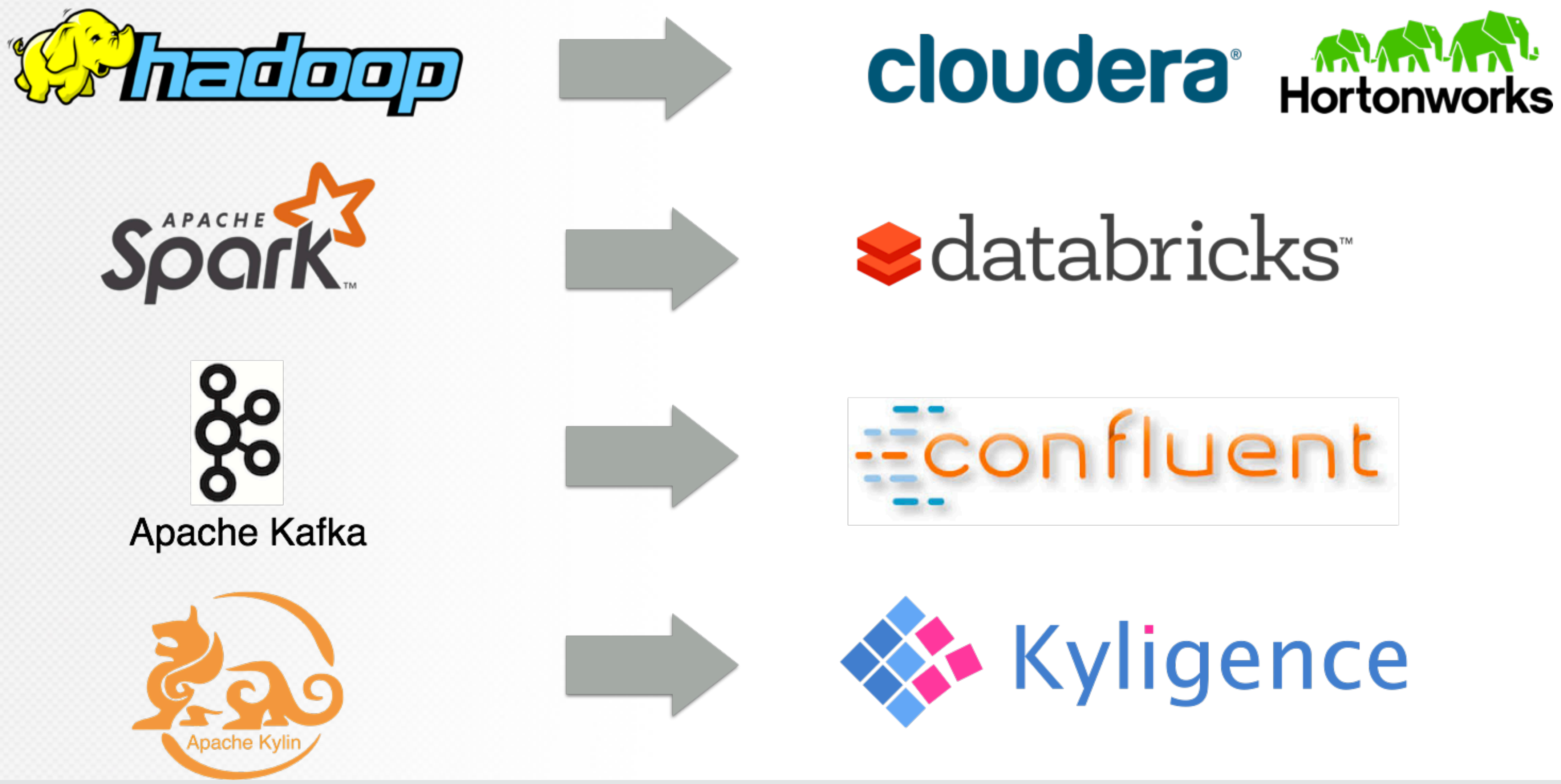
Sep 2016

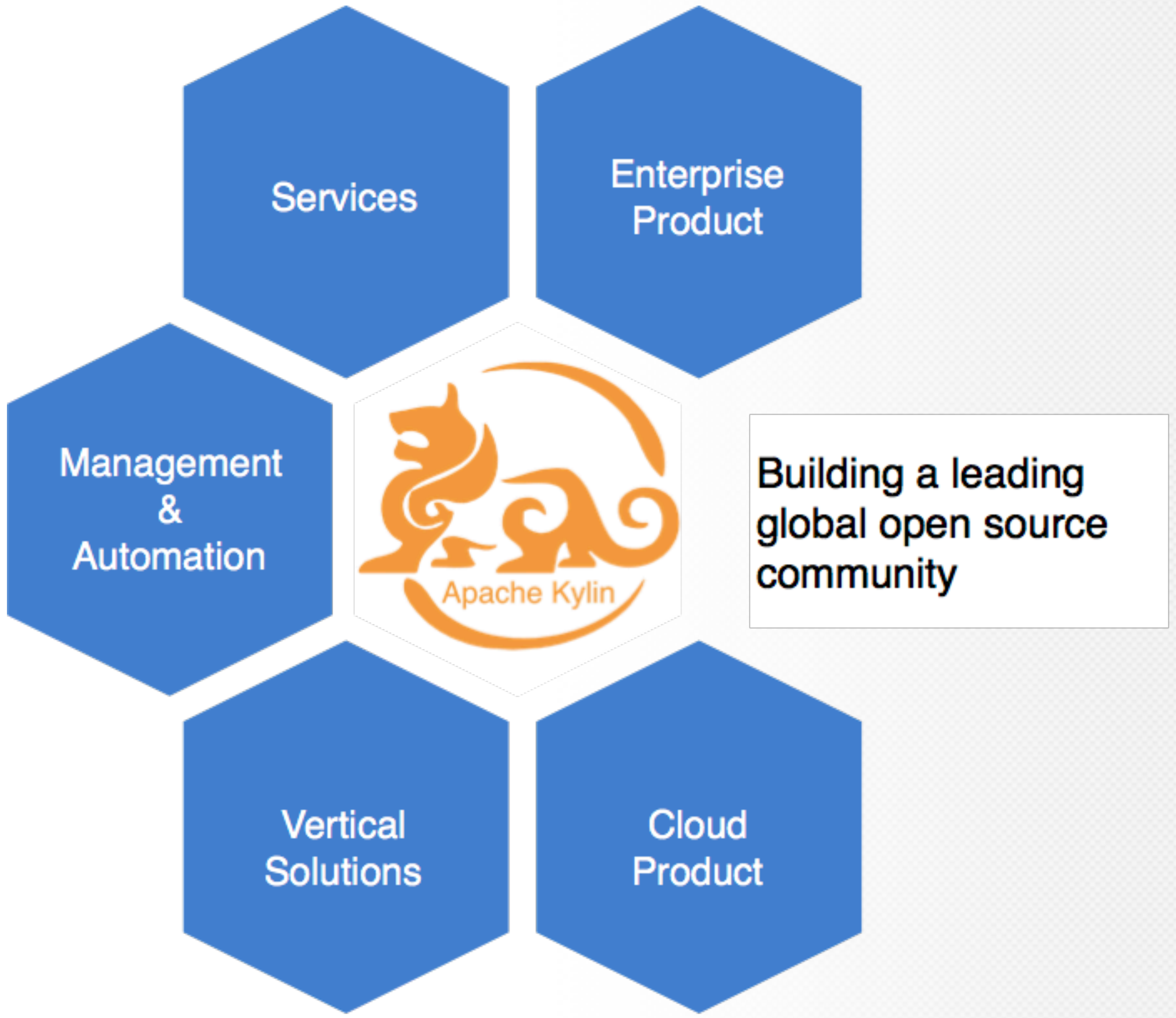
关于Kyligence

- Kyligence’s vision is to unleash big data productivity for everyone's analytics needs.

- The company was founded by the team who created Apache Kylin™, a top open source OLAP engine built for interactive analytics at petabyte-scale data on Hadoop. Kyligence is the primary contributor to the open source Kylin project globally.

- Kyligence provides a leading intelligent data platform to simplify big data analytics from on-premises to cloud.

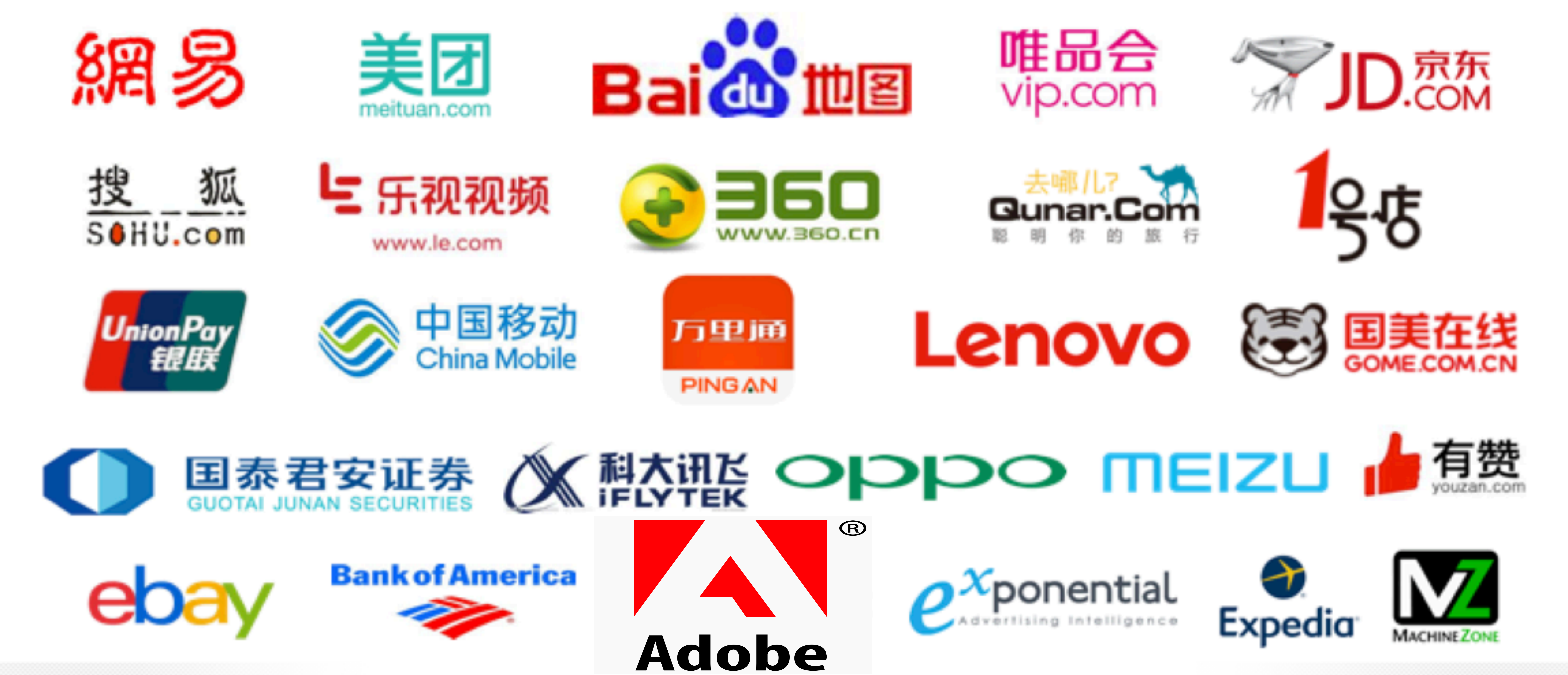




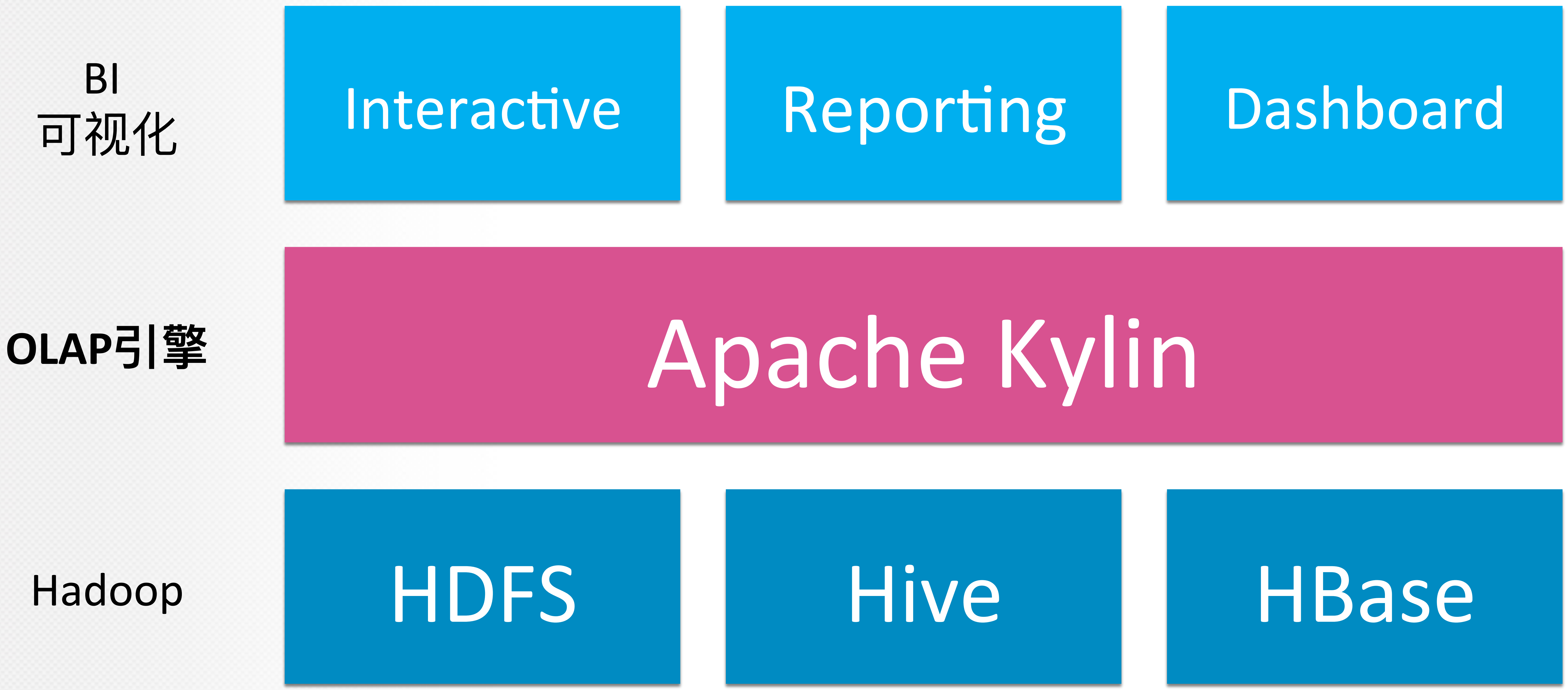
Building a leading global open source community

Kyligence

Apache Kylin全球案例



Apache Kylin是什么



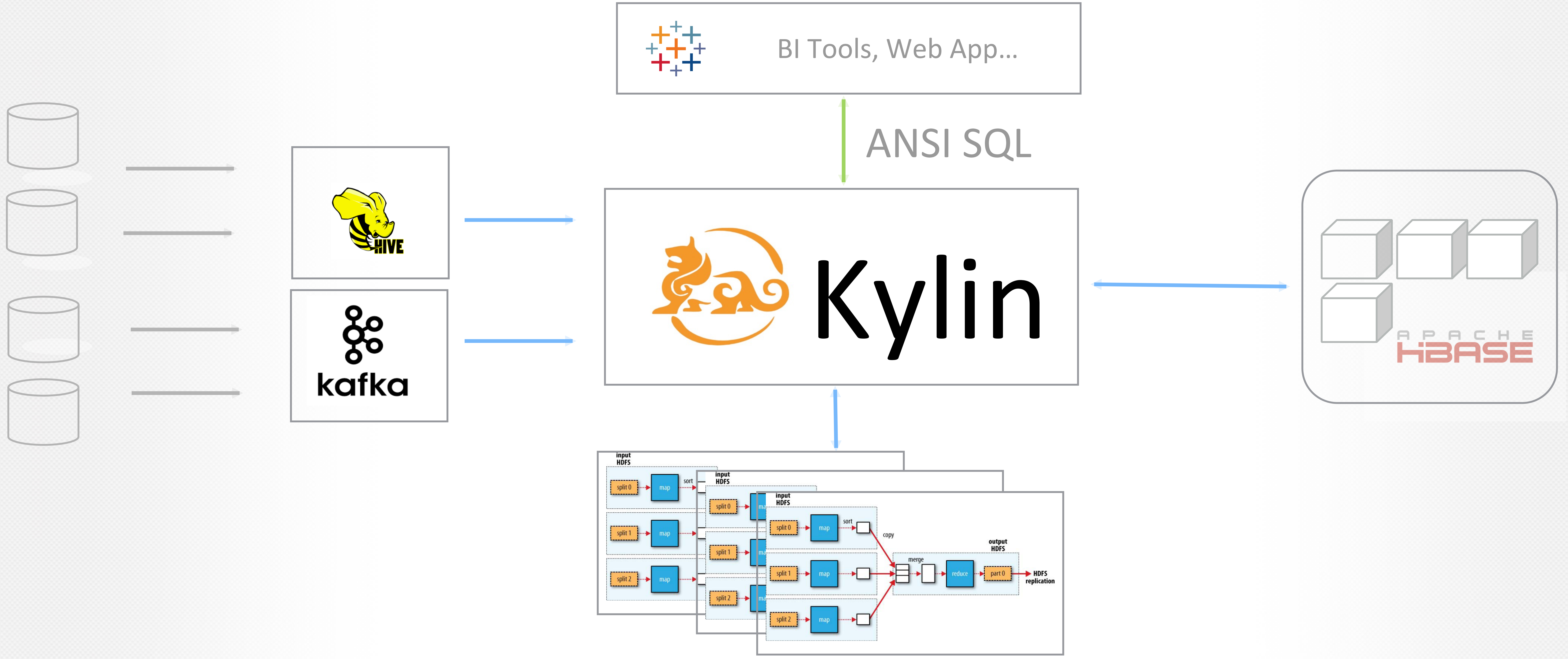
BI
可视化: Interactive | Reporting | Dashboard

OLAP引擎: Apache Kylin

Hadoop: HDFS | Hive | HBase

- 3 万亿条数据,
  < 1 秒 查询延迟
  @头条, 国内第一新闻资讯app
- 60+ 维度的cube
  @太平洋保险, 中国三大保险公司之一
- JDBC / ODBC / RestAPI
- BI 集成

Apache Kylin in the Zoo



BI Tools, Web App…

ANSI SQL

Kylin

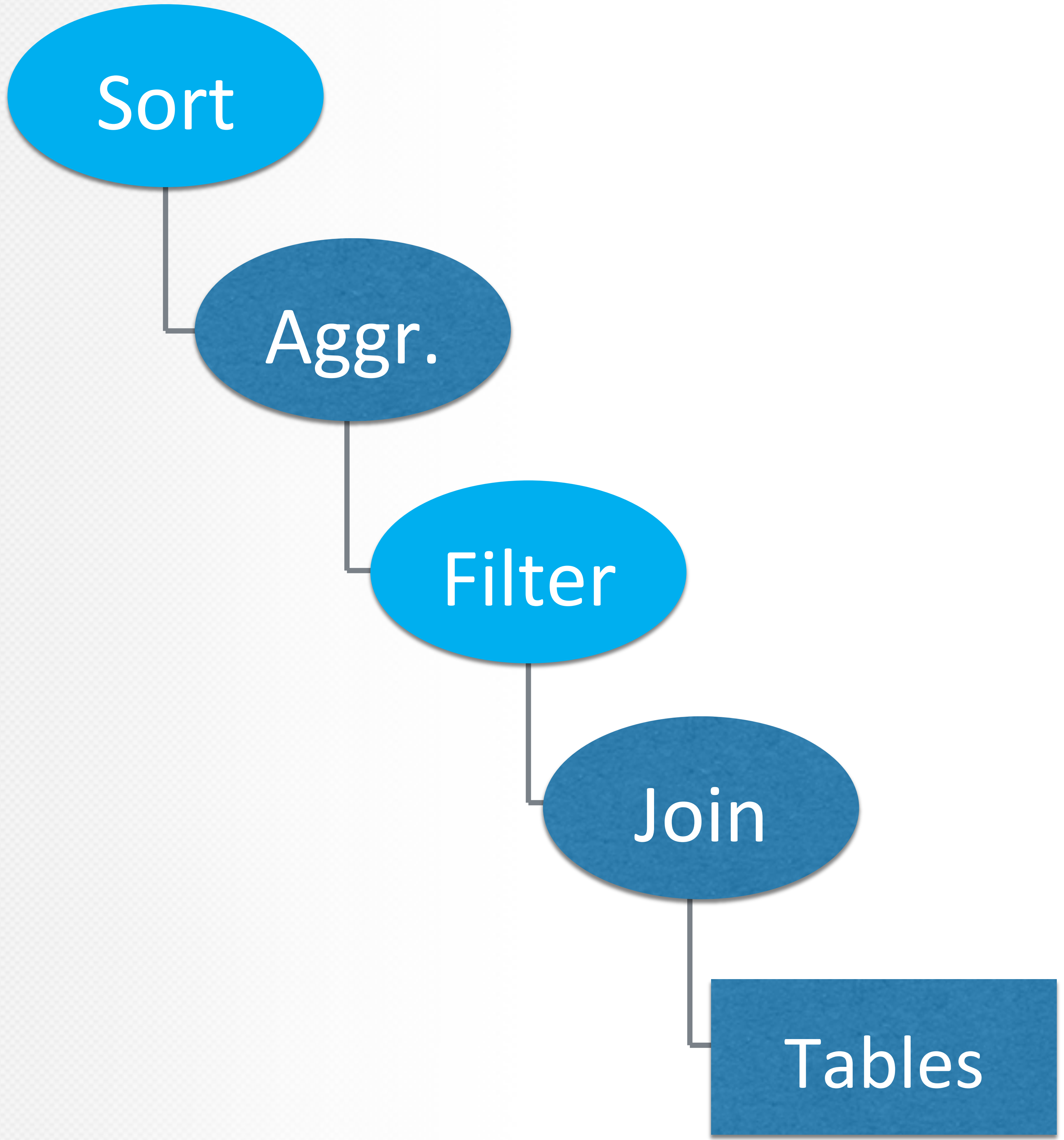
HIVE

kafka

APACHE HBASE

Offline Cubing

Kylin为什么快



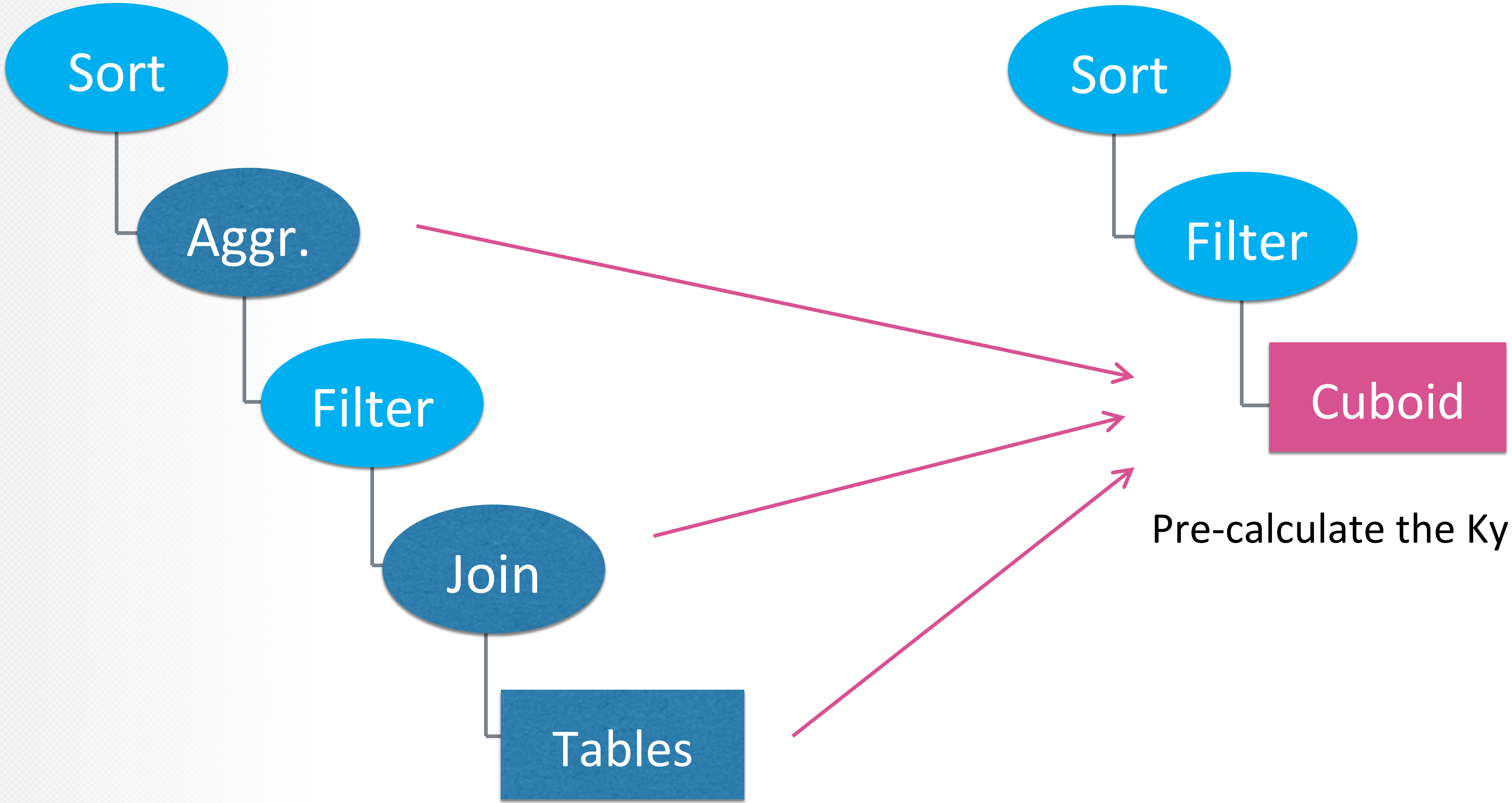
O(N)

A sample query:

Report revenue by “returnflag” and “orderstatus”

```
select
   l_returnflag,
   o_orderstatus,
   sum(l_quantity) as sum_qty,
   sum(l_extendedprice) as sum_base_price
   …
from
   v_lineitem
   inner join v_orders on l_orderkey = o_orderkey
where
   l_shipdate <= '1998-09-16'
group by
   l_returnflag,
   o_orderstatus
order by
   l_returnflag,
   o_orderstatus;
```

Kylin为什么快



Sort

Aggr.

Filter

Join

Tables

Sort

Filter

Cuboid

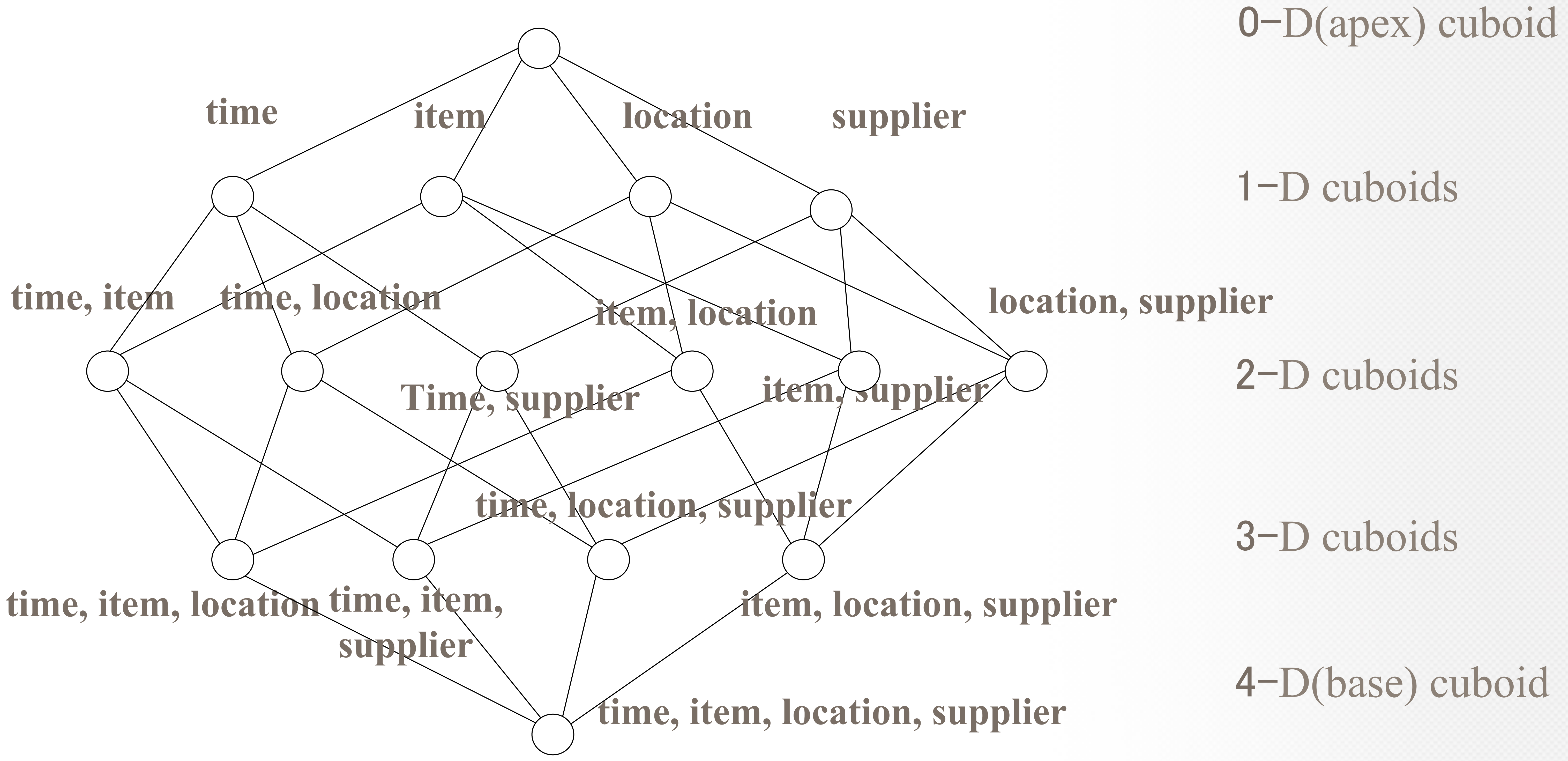
Pre-calculate the Kylin Cube

O(N)

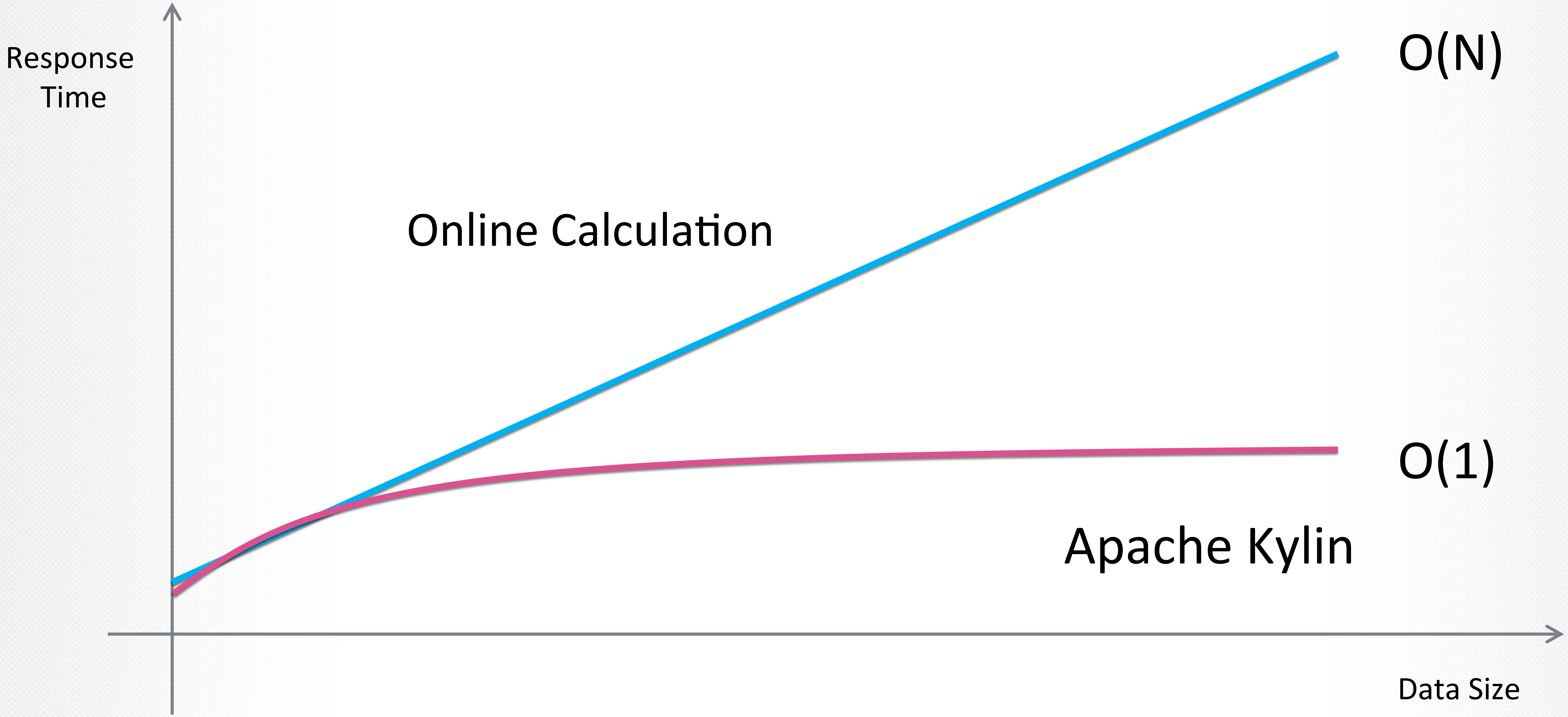
O(flag x status x days) = O(1)

Kylin 关键在于预计算

- 基于cube理论
- Model 和 Cube 定义了预计算的范围
- Build Engine 执行构建任务
- Query Engine 在预计算的结果之上完成查询



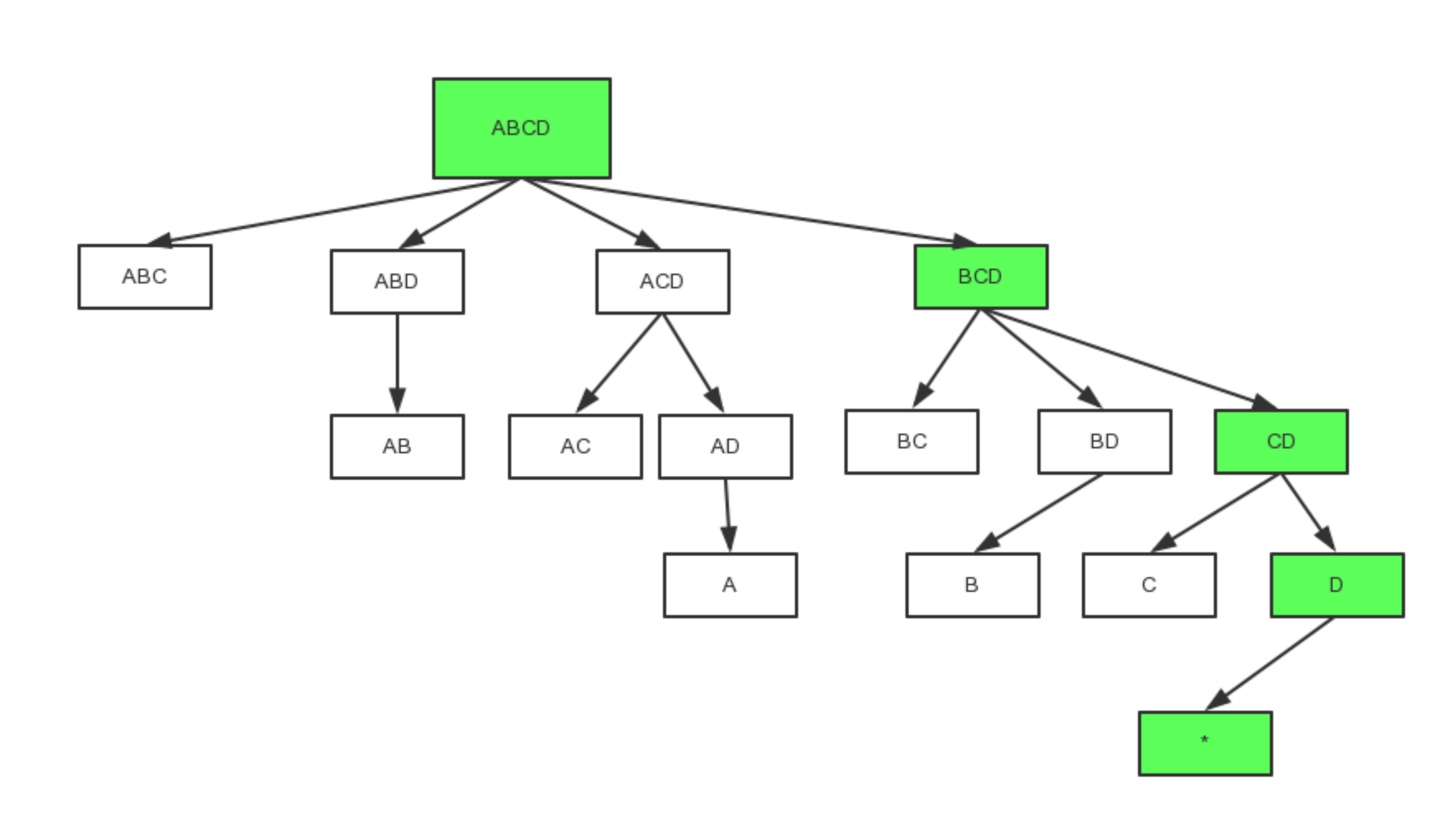
O(1)复杂度


Response Time
O(N)
Online Calculation
O(1)
Apache Kylin
Data Size

Spark Cubing

减少一半的构建时间

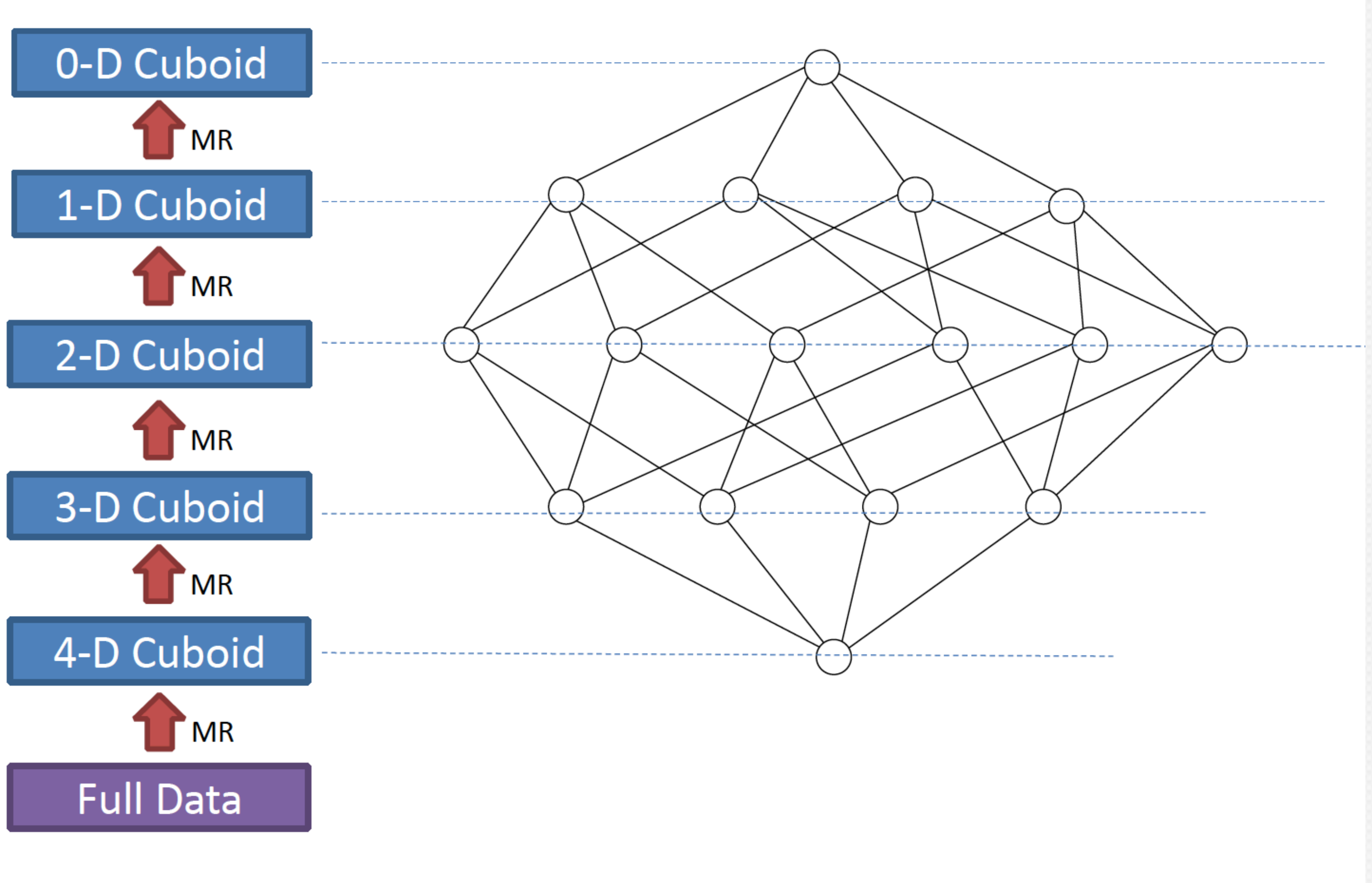
Cuboid Spanning Tree



Layered Cubing

标准的构建算法：Layered Cubing

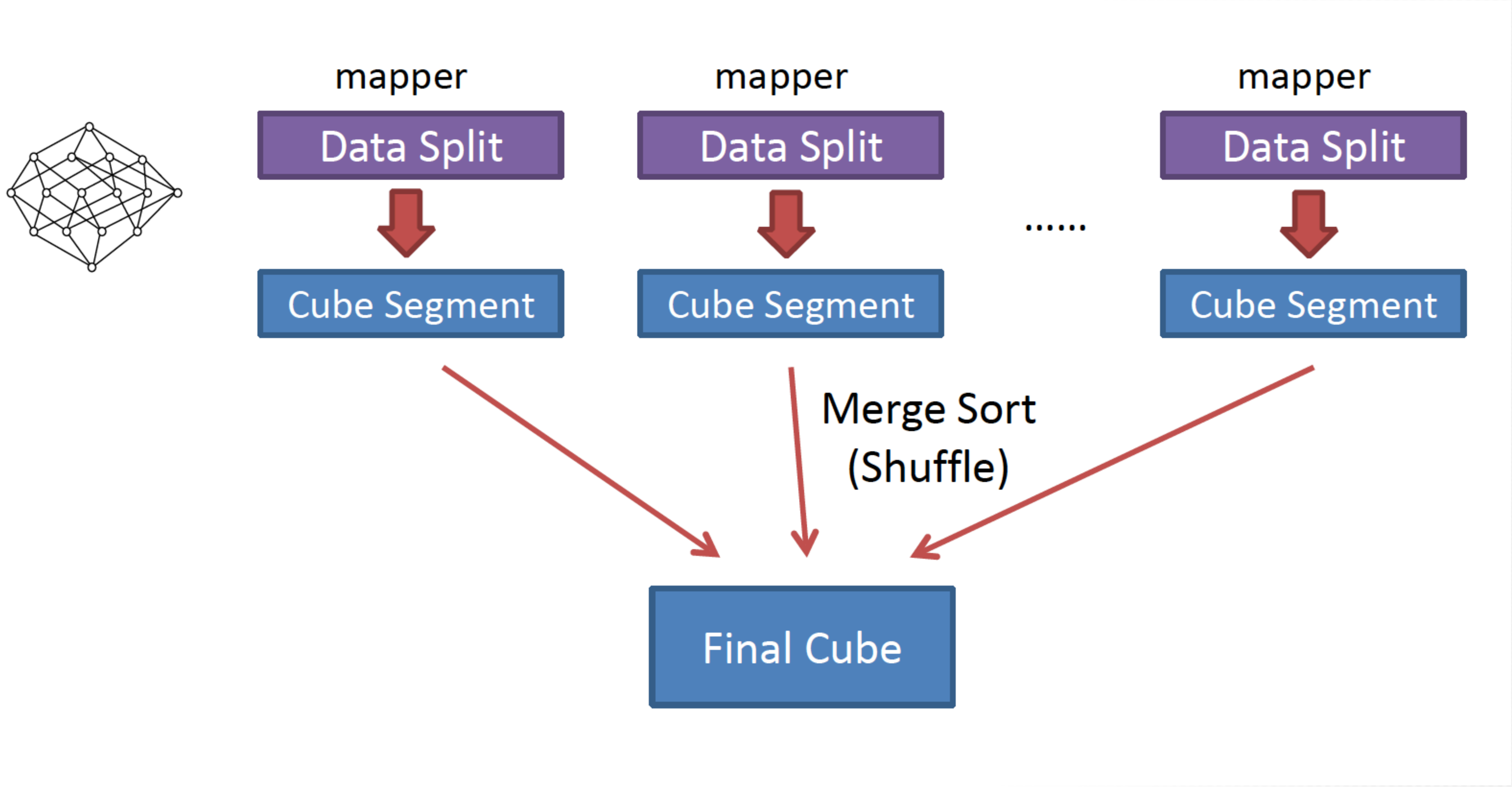
- 启动多轮MR任务
- 将大型shuffle切分到多个stage
- 稳定，但是在构建时间上并不是最优的



In-memory Cubing

In-memory Cubing是对Layered Cubing的强力补充

- 在某些条件下触发
- 并不适用于所有场景
- 一旦被触发，往往拥有更好性能



Cubing with MR 总结

比较稳定

Layered Cubing在某些场景下性能都有待提高

In-mem Cubing适用场景有限

社区迫不及待地想要尝试使用其他技术来加速cubing

Apache Spark介绍

Apache Spark is an open-source cluster-computing framework, which provides programmers with an application programming interface centered on a data structure called RDD.

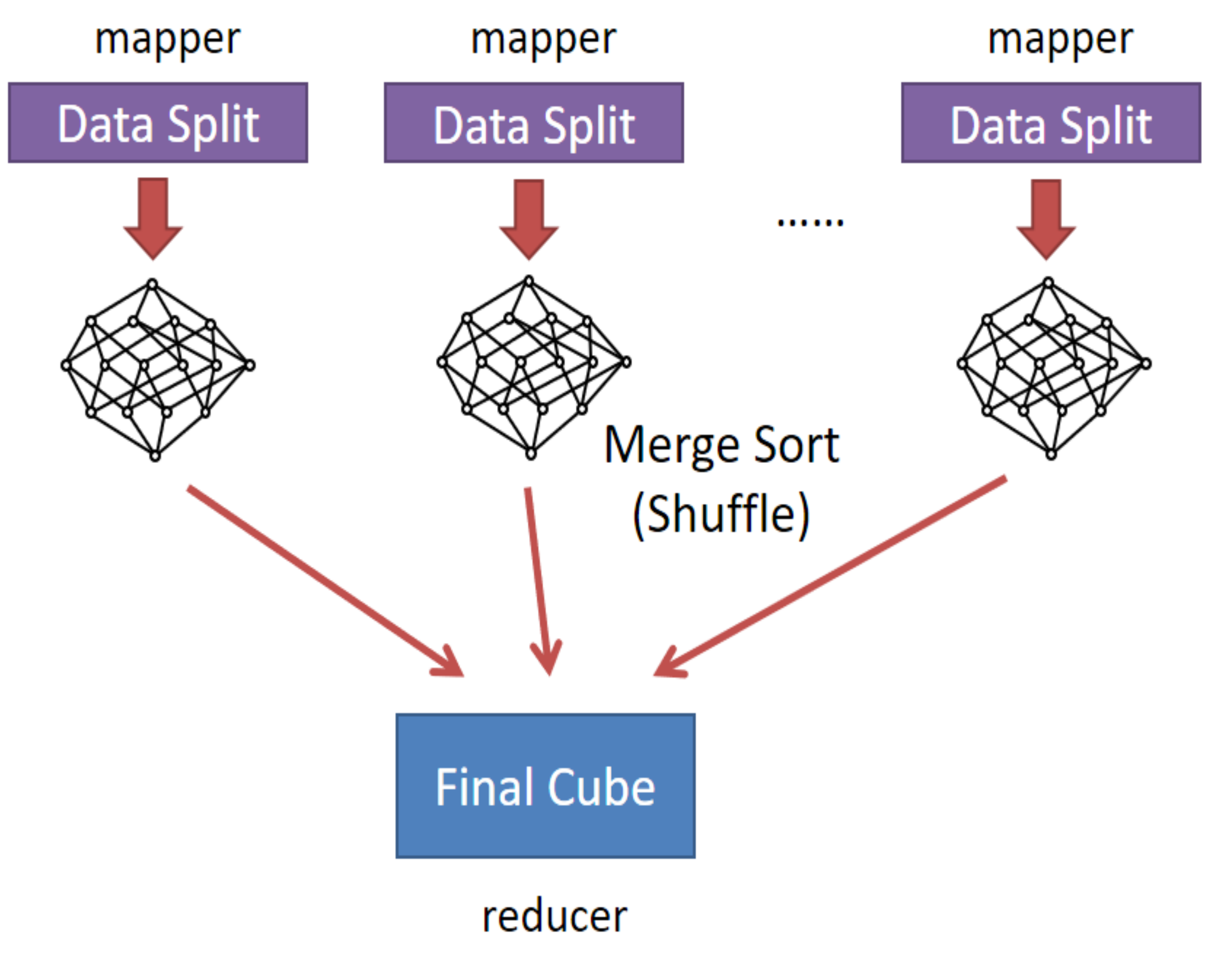
Spark was developed in response to limitations in the MapReduce cluster computing paradigm.

Spark runs on Hadoop, Mesos, standalone, or in the cloud. It can access diverse data sources including HDFS, Cassandra, HBase, and S3.

Spark Cubing:一次失败的尝试

Kylin 1.5曾经尝试使用过Spark Cubing，但是从未正式发布

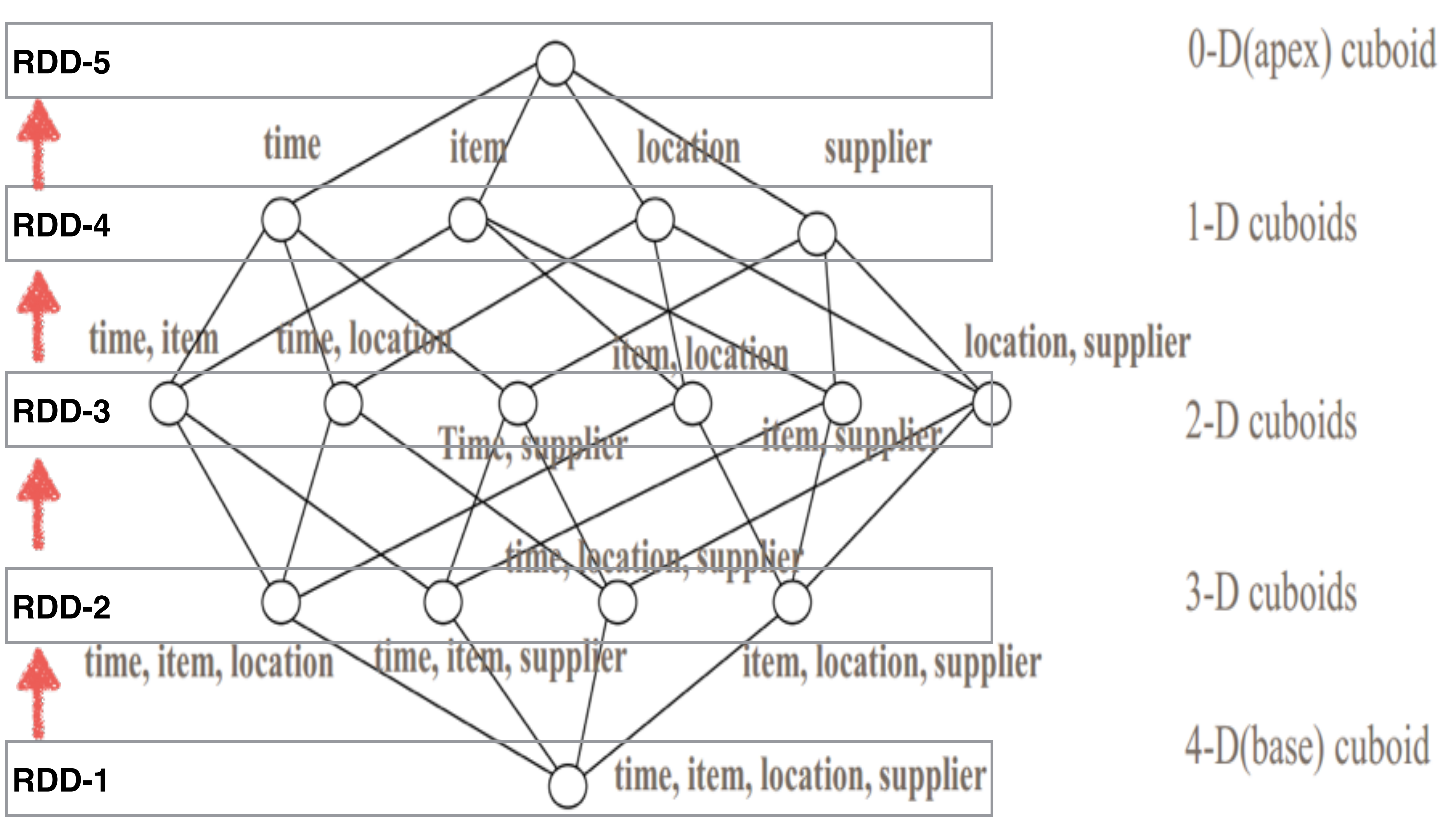
- 它只是简单地将In-memory Cubing移植到Spark上
- 使用一轮RDD转换计算整个cube
- 并未观察到明显改进
- Spark 计算方式与MR并无明显区别



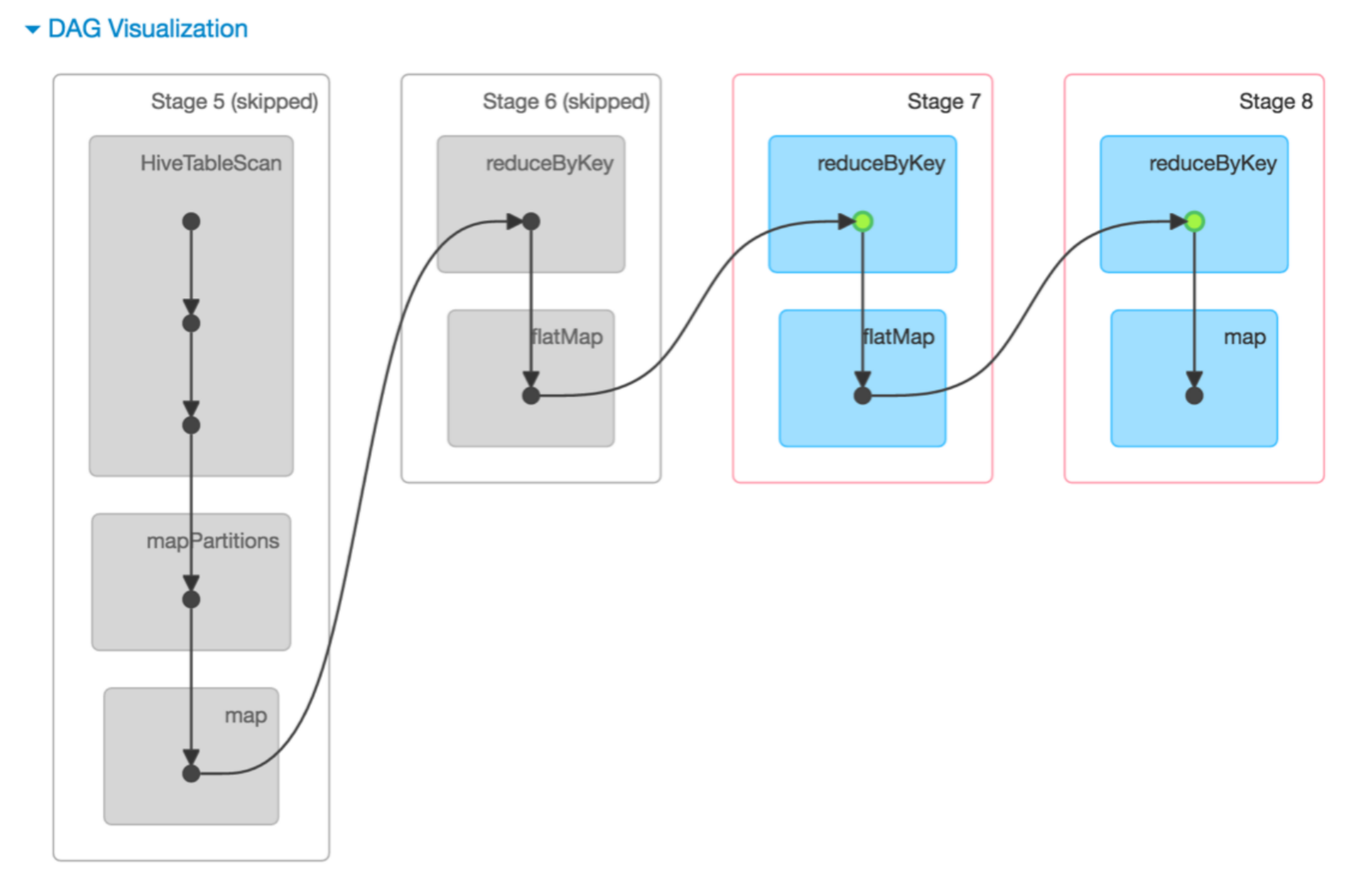
Spark Cubing in 2.0

Kylin 2.0 基于Layered Cubing 算法重新打造了Spark Cubing

- 每一层的cuboid视作一个RDD
- 父亲RDD被尽可能cache到内存
- RDD 被导出到sequence file,
- 通过将 “map” 替换为“flatMap”,
  以及把“reduce” 替换为 “reduceByKey”,
  可以复用大部分代码



计算第三层cuboid的DAG



Performance Test

- Environment

  4 nodes Hadoop cluster; each node has 28 GB RAM and 12 cores

  YRAN has 48GB RAM and 30 cores in total

  CDH 5.8, Apache Kylin 2.0 beta
- Spark

  Spark 1.6.3 on YARN

  6 executors, each has 4 cores, 4GB +1GB (overhead) memory
- Test Data

  Airline data, total160 million rows

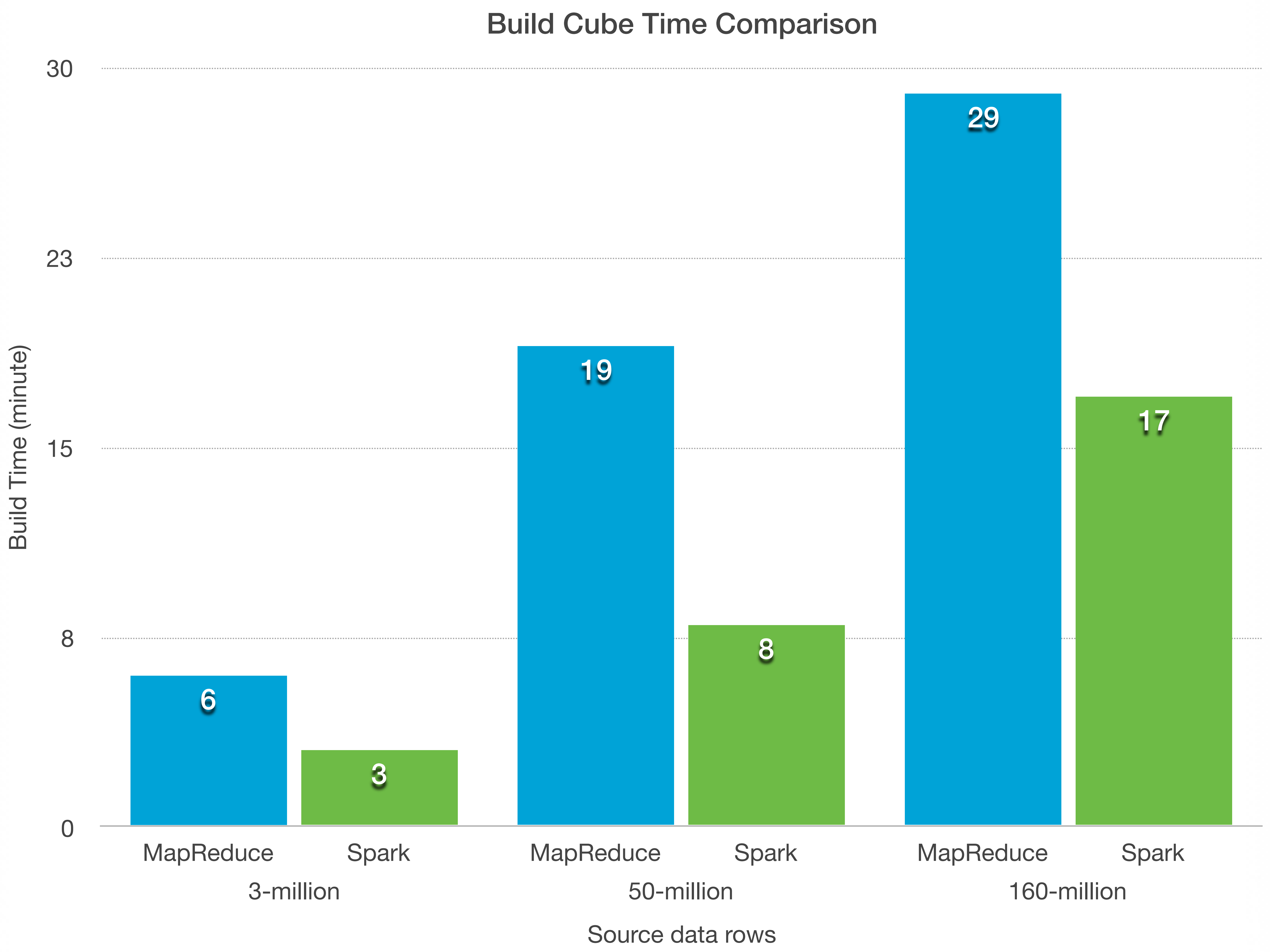
  Cube:10 dimensions, 5 measures (SUM)
- Test Scenarios

  Build the cube at different source data scale: 3 million, 50 million and 160 million source rows

  Compare the build time with MapReduce (both by layer and in-mem) and Spark. No compression

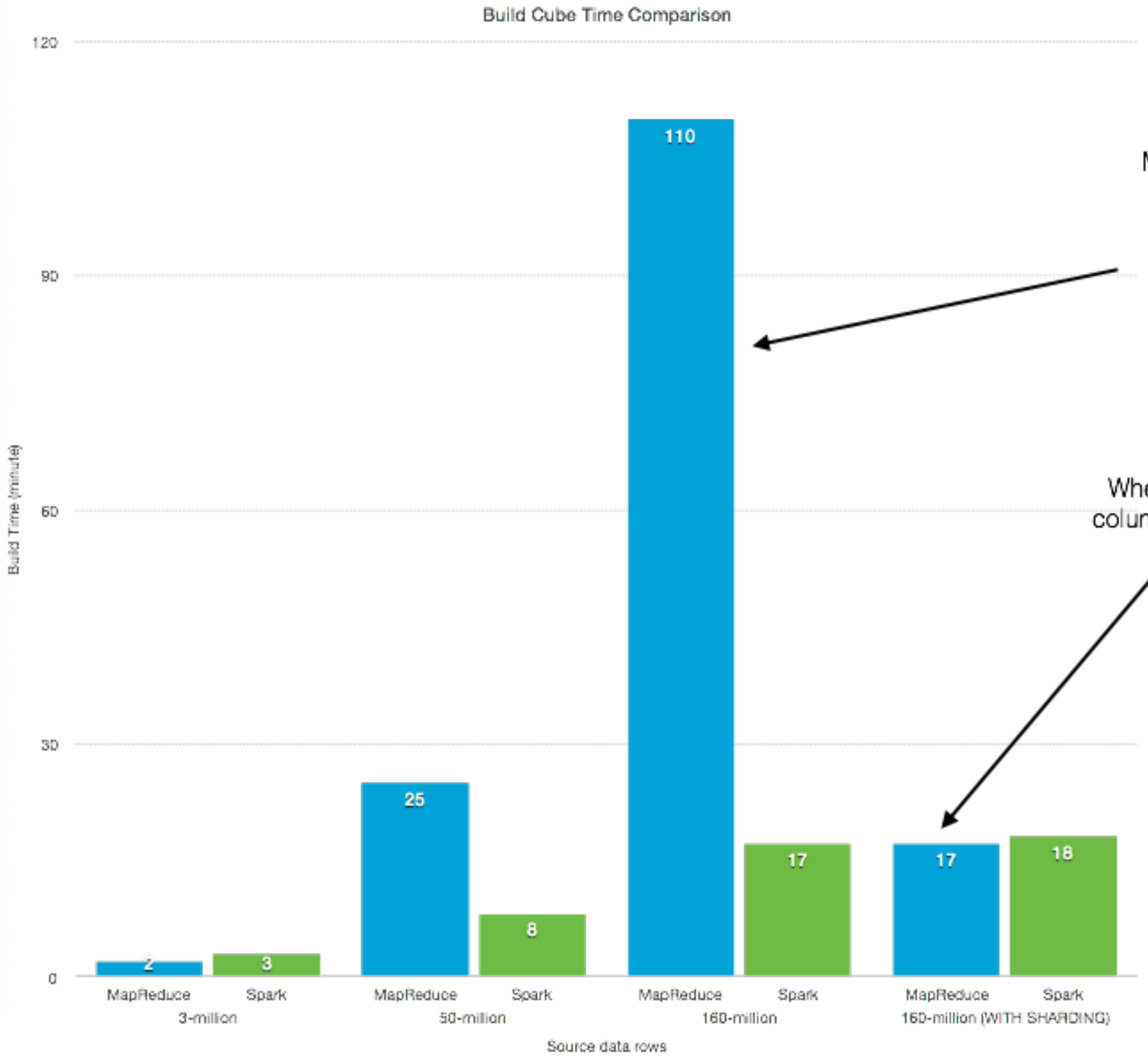
  The time only cover the building cube step, not including preparations and subsequent steps

Spark Cubing vs. MR Layered Cubing



70% to 130% improvement on Spark

Spark Cubing vs. MR In-mem Cubing



MR(in-mem) is unstable when source data is random distributed (cause too many shuffles)

When data is distributed by a column, MR(in-mem) is very fast

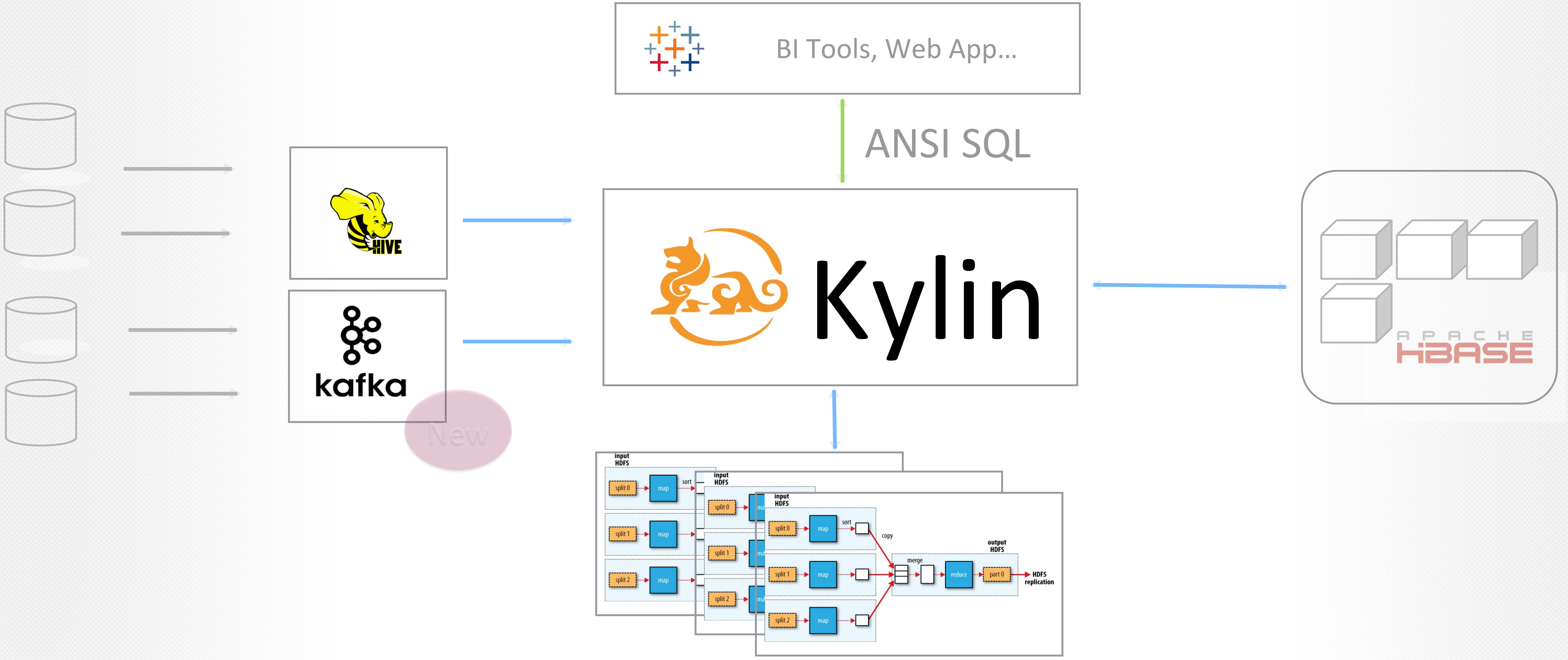
结论

- Layered cubing algorithm is stable on both MR and Spark Spark
- Compared with MR layered Cubing, 70% to 130% performance improvement is observed on Spark Cubing
- When source data is sharded, Spark still keeps close performance with MR in-mem cubing
- There’s still room for improvement

近实时流数据处理

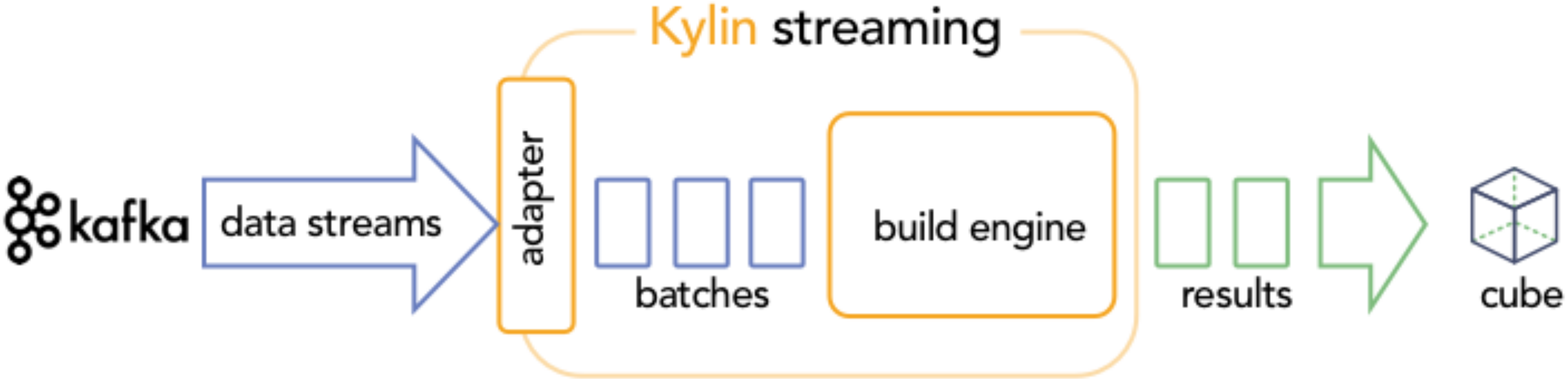
构建分钟级别延迟的cube

New in Kylin 1.6



Offline Cubing

kafka ♥ Apache Kylin



Kylin streaming

kafka

data streams

adapter

batches

build engine

results

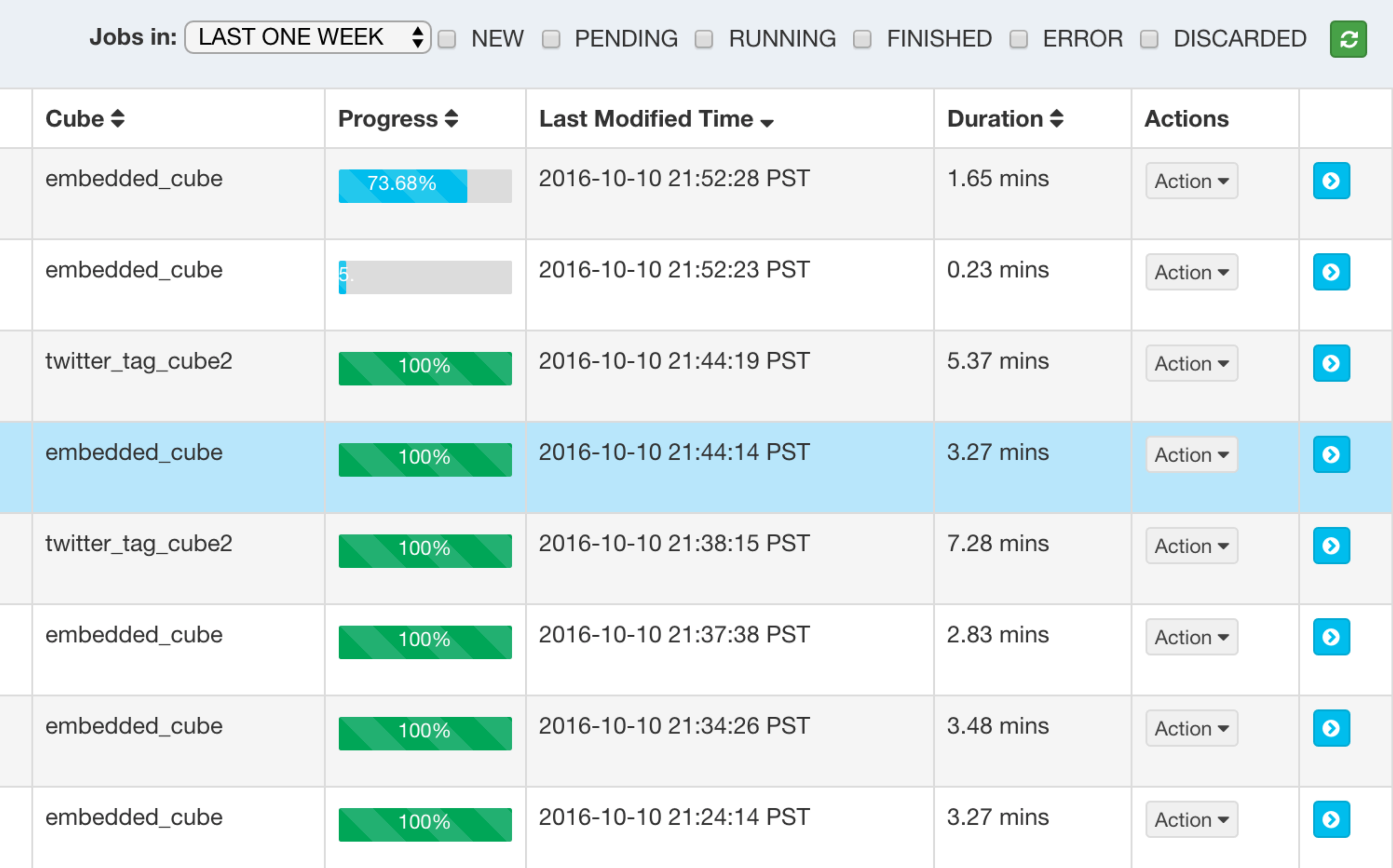
cube

Demo of Twitter Analysis

http://hub.kyligence.io

Incremental build triggers every 2 minutes, build finishes in 3 minutes.

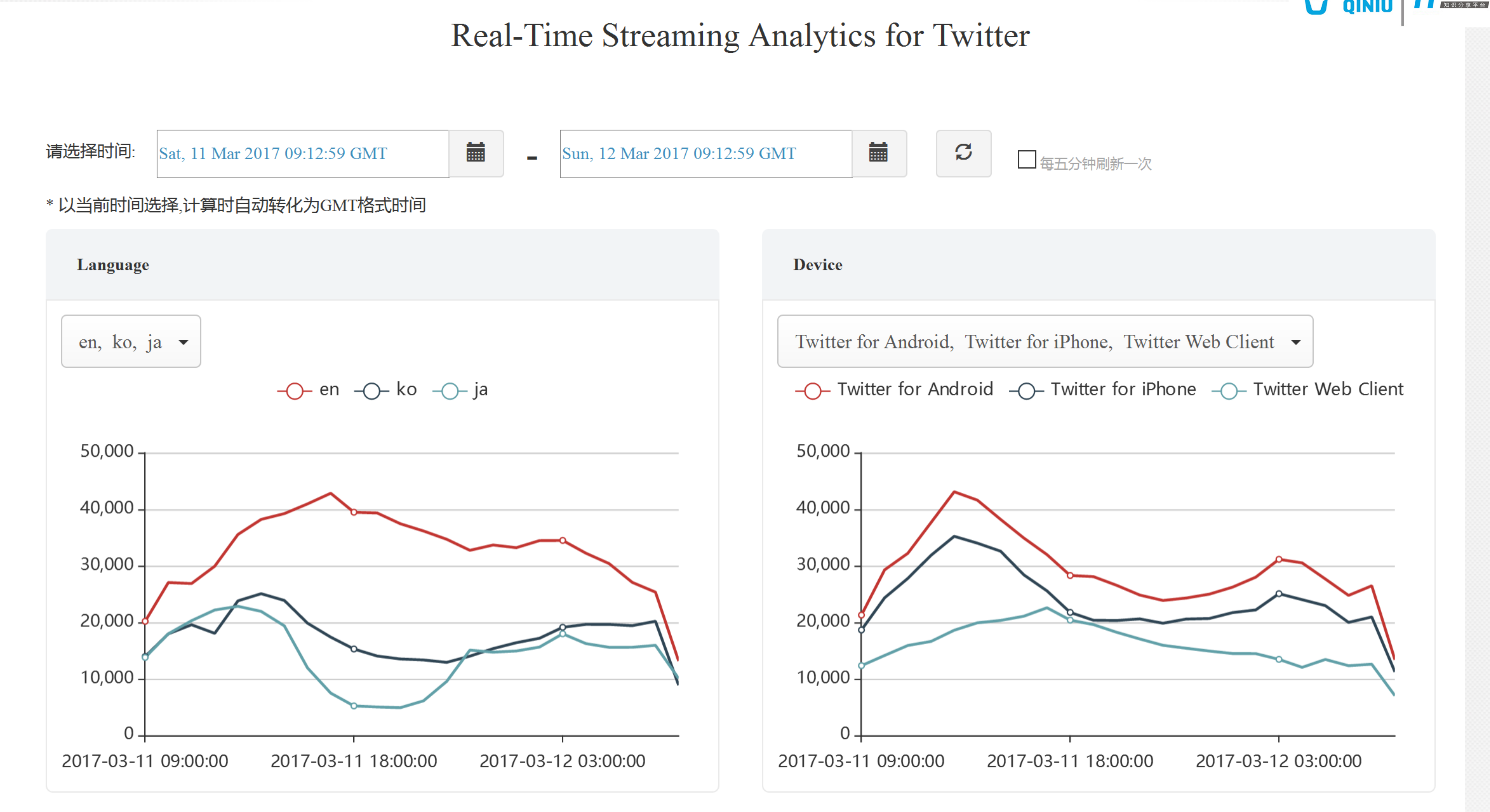
- 8-node cluster on AWS, 3 Kafka brokers
- Twitter sample feed, 10+ K messages per second
- Cube has 9 dimensions and 3 measures
- 2 jobs running at the same time



Jobs in: LAST ONE WEEK □ NEW □ PENDING □ RUNNING □ FINISHED □ ERROR □ DISCARDED

Cube	Progress	Last Modified Time	Duration	Actions	
embedded_cube	73.68%	2016-10-10 21:52:28 PST	1.65 mins	Action	
embedded_cube	5.	2016-10-10 21:52:23 PST	0.23 mins	Action	
twitter_tag_cube2	100%	2016-10-10 21:44:19 PST	5.37 mins	Action	
embedded_cube	100%	2016-10-10 21:44:14 PST	3.27 mins	Action	
twitter_tag_cube2	100%	2016-10-10 21:38:15 PST	7.28 mins	Action	
embedded_cube	100%	2016-10-10 21:37:38 PST	2.83 mins	Action	
embedded_cube	100%	2016-10-10 21:34:26 PST	3.48 mins	Action	
embedded_cube	100%	2016-10-10 21:24:14 PST	3.27 mins	Action	

Kyligence



Real-Time Streaming Analytics for Twitter

请选择时间: Sat, 11 Mar 2017 09:12:59 GMT - Sun, 12 Mar 2017 09:12:59 GMT

□ 每五分钟刷新一次

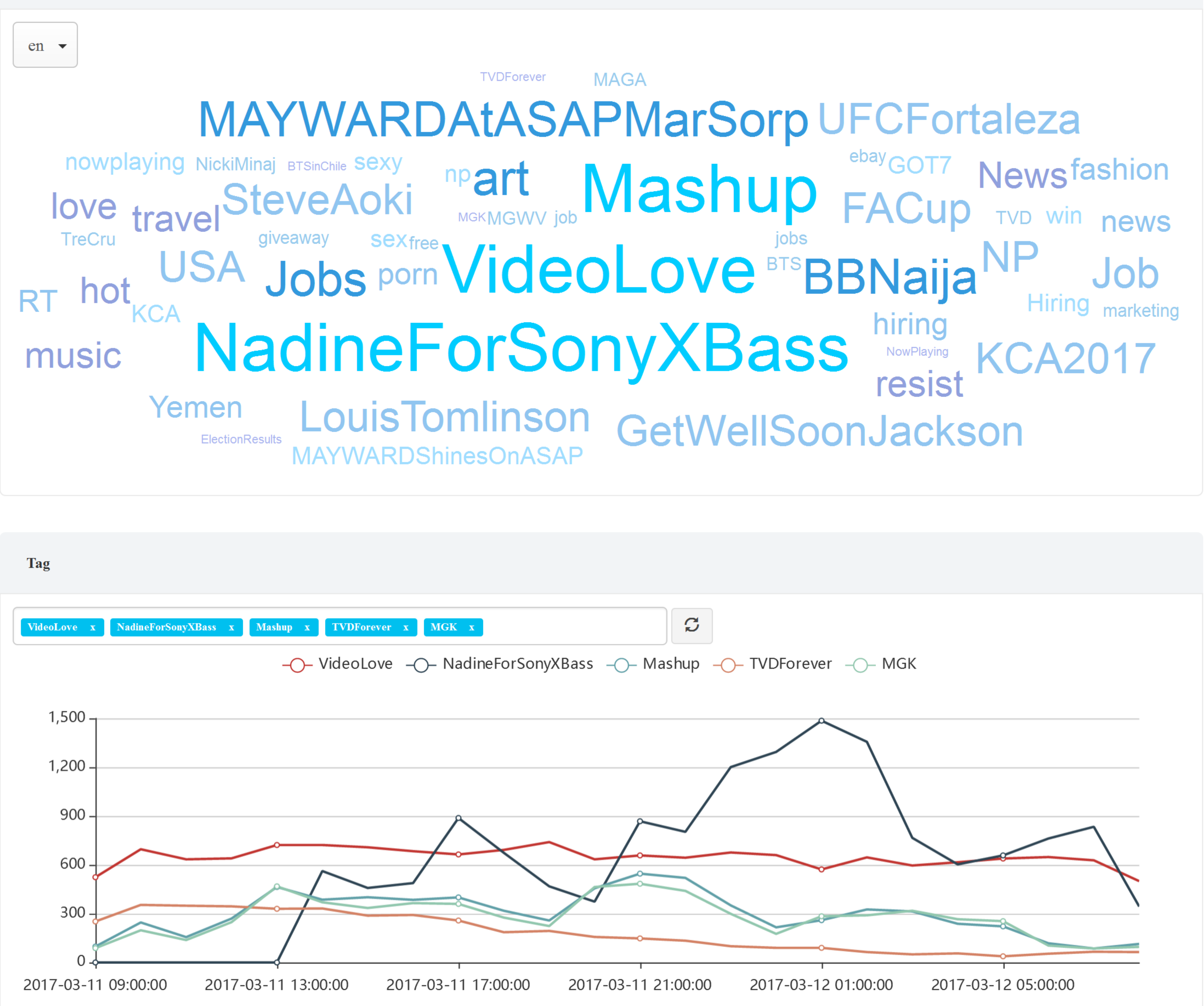
* 以当前时间选择,计算时自动转化为GMT格式时间

Language

en, ko, ja

Device

Twitter for Android, Twitter for iPhone, Twitter Web Client



en

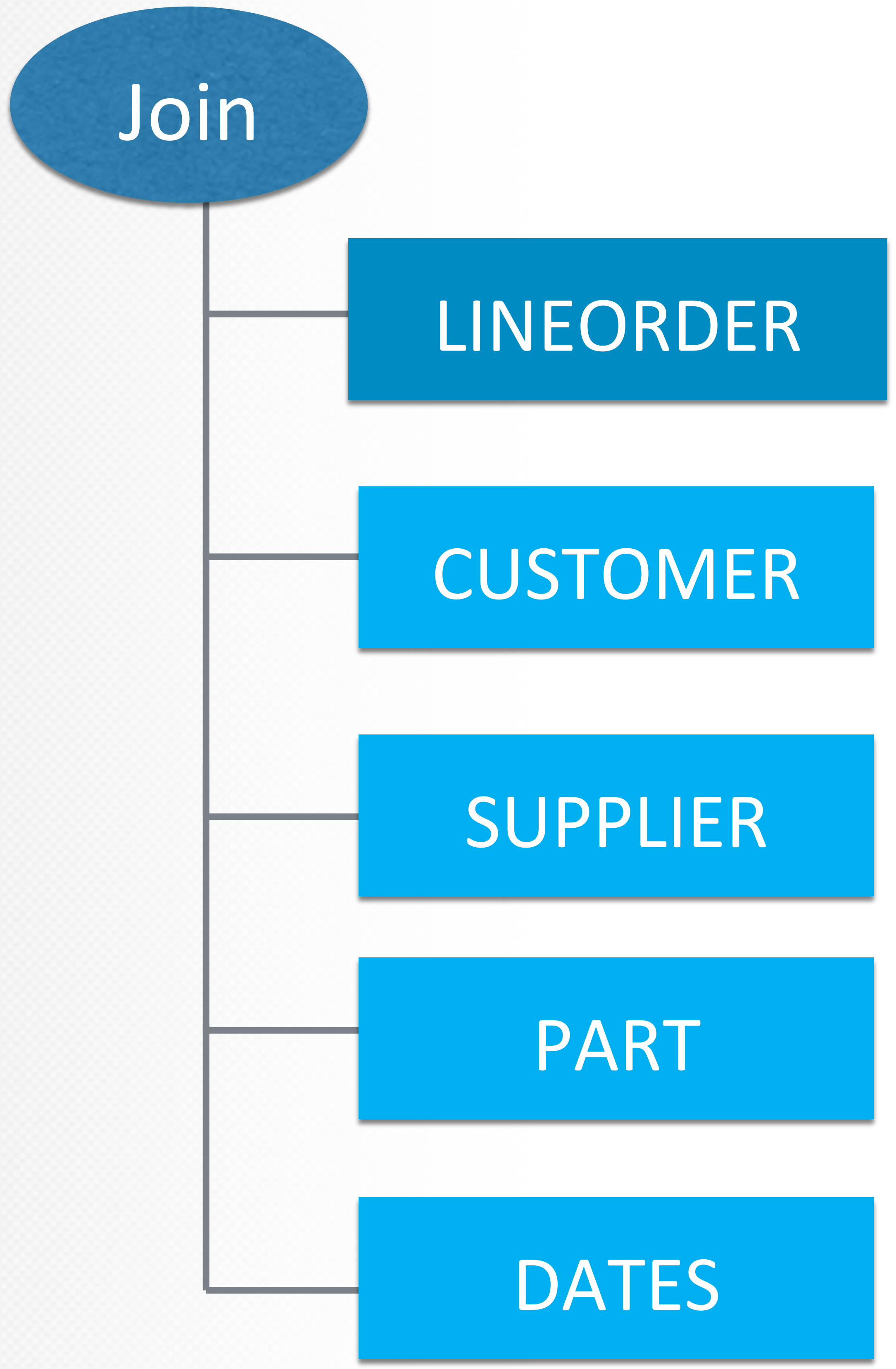
Tag

VideoLove x | NadineForSonyXBass x | Mashup x | TVDForever x | MGK x

雪花模型的支持

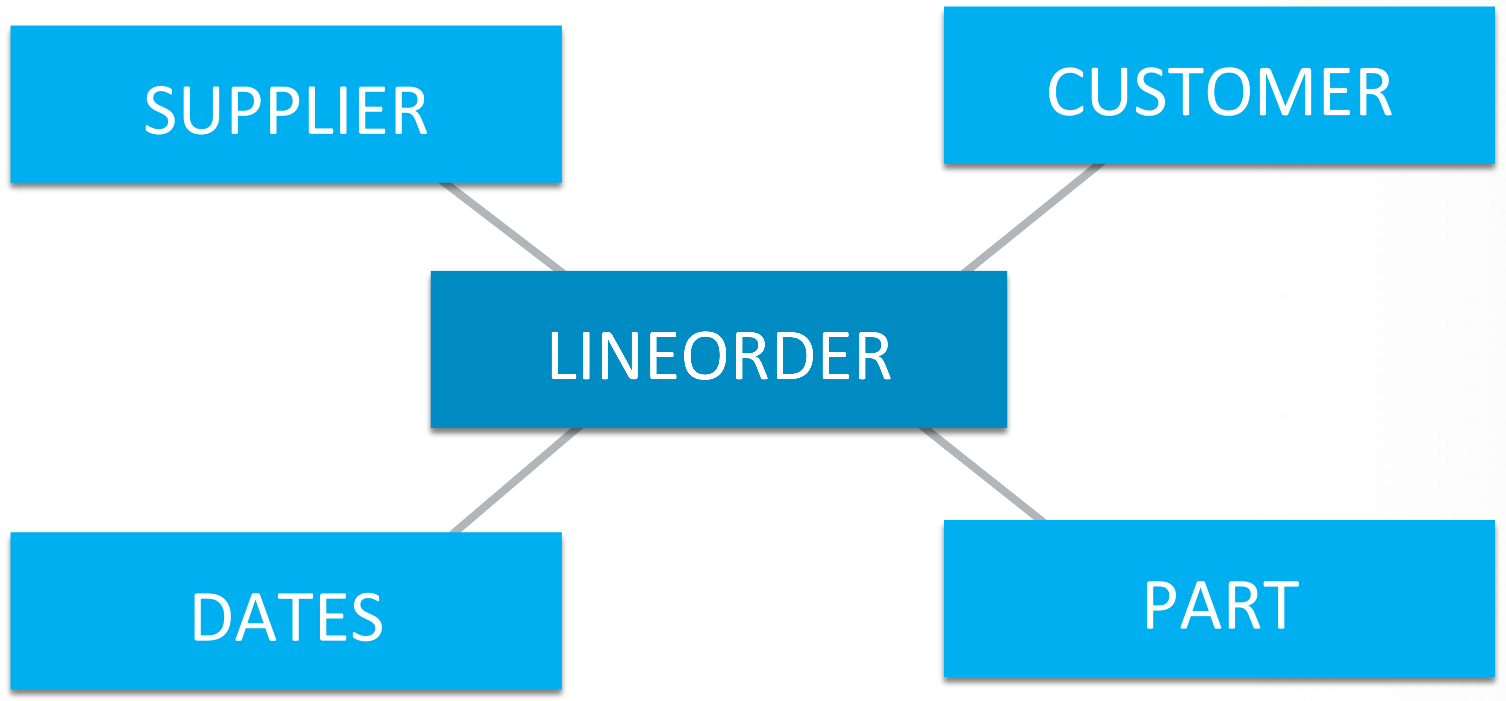
运行TPC-H benchmark

Kylin 1.0 Star Schema Limitation



Pre-calculate the join of 1 level of lookups

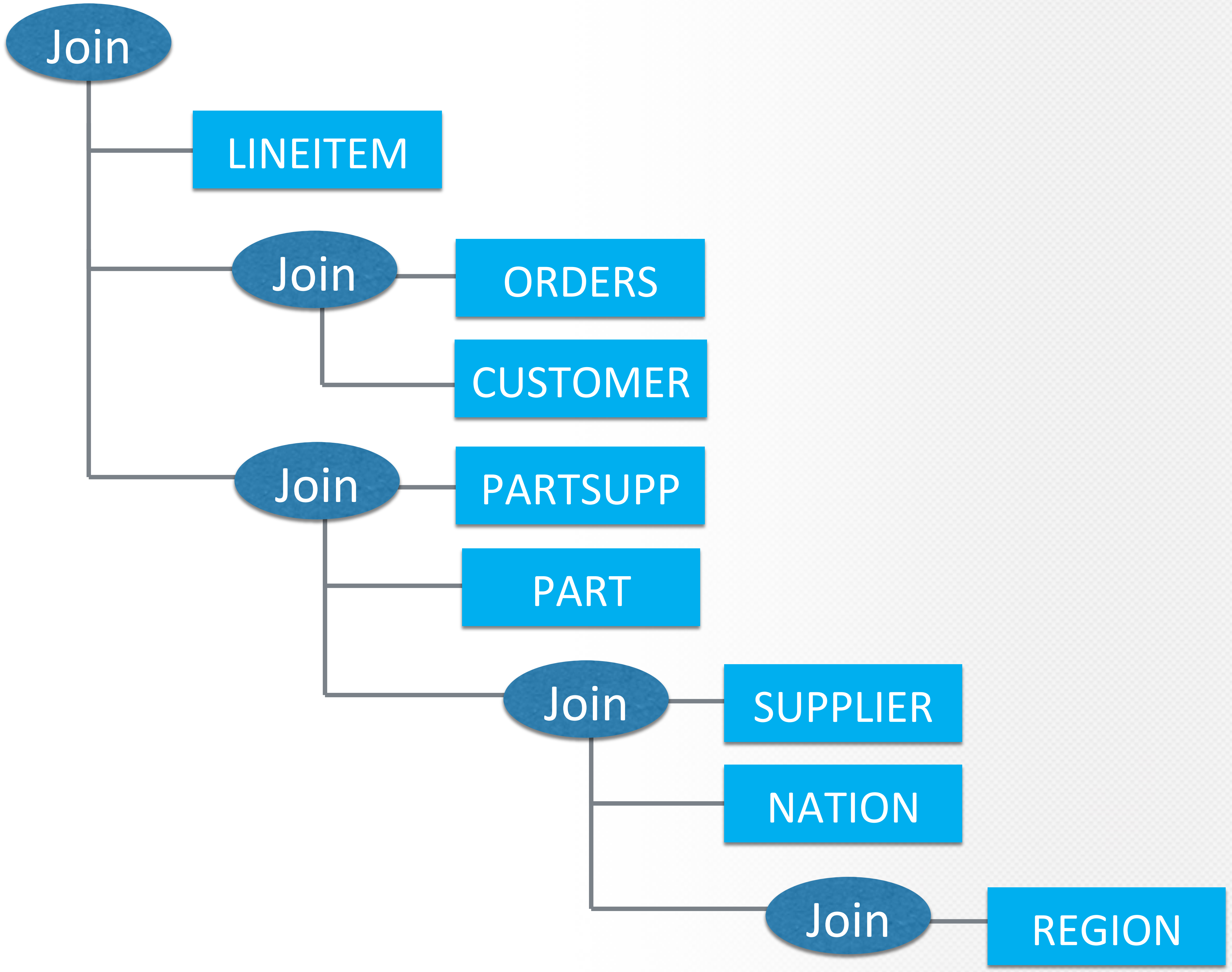
- Support star schema only
- Not allow same name columns from different tables
- Not allow table joining itself
- Difficult to support real world business cases



Kylin 2.0 Snowflake Schema

Pre-calculate unlimited levels of lookups

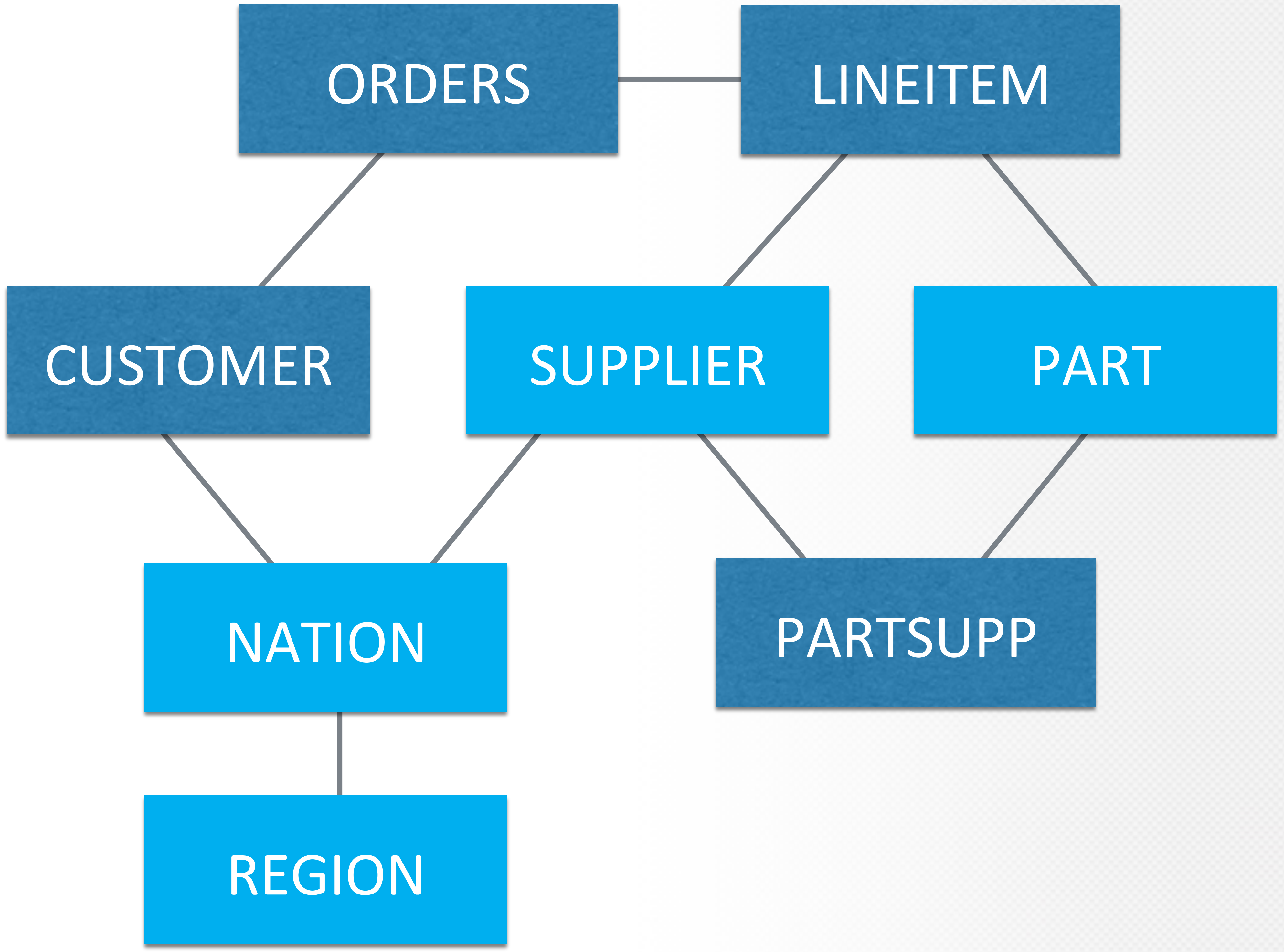
- Snowflake schema support (KYLIN-1875)
- Allow table be joined multiple times
- Big metadata change at Model level
- Many bug fixes regarding joins and sub-queries
- Support complex models of any kind, support flexible queries on the models



TPC-H on Kylin 2.0

TPC-H is a benchmark for decision support system.

- Popular among commercial RDBMS & DW solutions
- Queries and data have broad industry-wide relevance
- Examine large volumes of data
- Execute queries with a high degree of complexity
- Give answers to critical business questions



Kylin 2.0 runs all the 22 TPC-H queries. (KYLIN-2467)

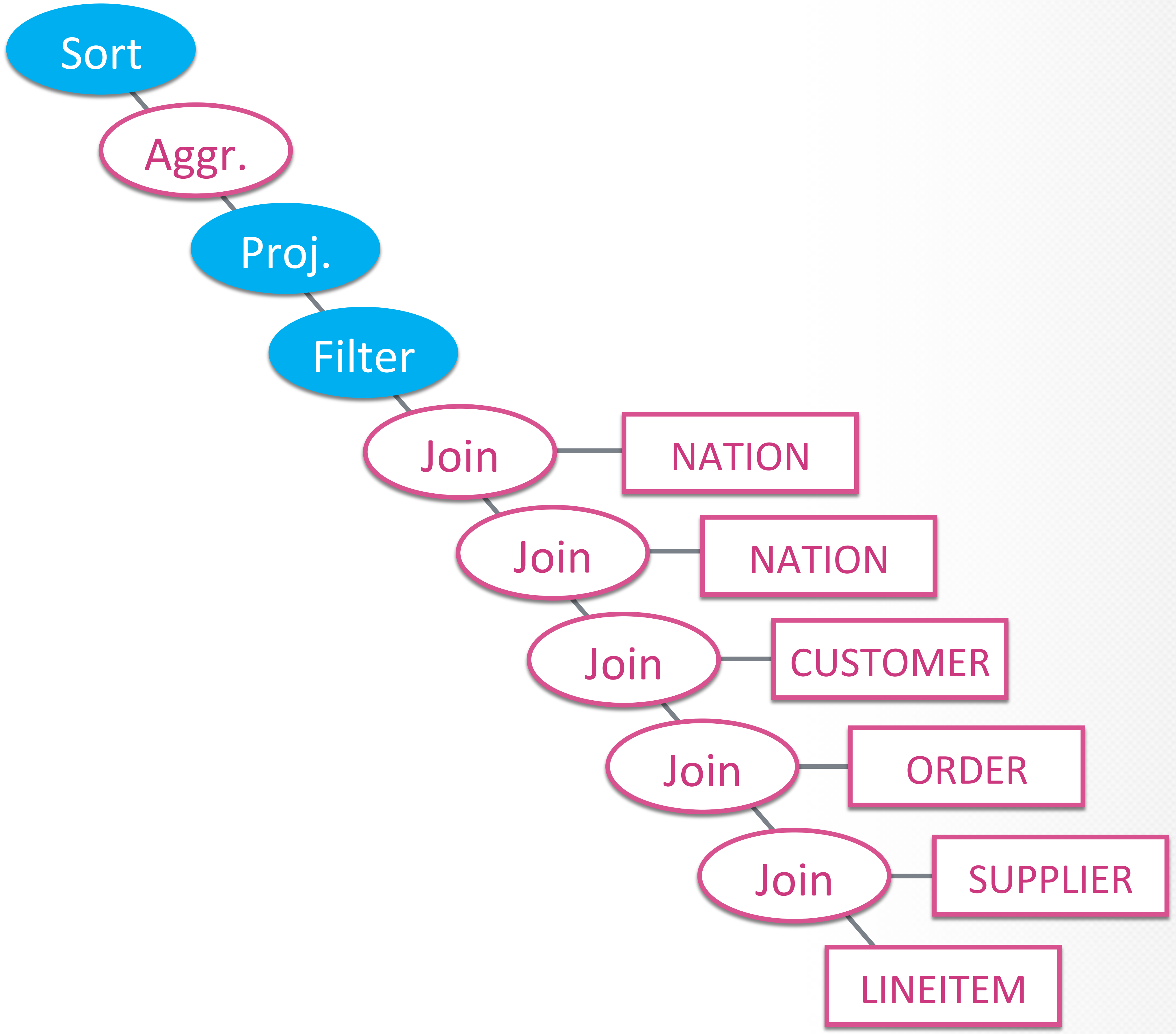
- Pre-calculation can answer very complex queries
- Goal is functionality at this stage
- Try it: https://github.com/Kyligence/kylin-tpch

Complex Query 1

TPC-H query 07

- 0.17 sec (Hive+Tez 35.23 sec)
- 2 sub-queries

```
select
  supp_nation,
  cust_nation,
  l_year,
  sum(volume) as revenue
from
  (
    select
      n1.n_name as supp_nation,
      n2.n_name as cust_nation,
      l_shipyear as l_year,
      l_saleprice as volume
    from
      v_lineitem
      inner join supplier on s_suppkey = l_suppkey
      inner join v_orders on l_orderkey = o_orderkey
      inner join customer on o_custkey = c_custkey
      inner join nation n1 on s_nationkey = n1.n_nationkey
      inner join nation n2 on c_nationkey = n2.n_nationkey
    where
      (
        (n1.n_name = 'KENYA' and n2.n_name = 'PERU')
        or (n1.n_name = 'PERU' and n2.n_name = 'KENYA')
      )
      and l_shipdate between '1995-01-01' and '1996-12-31'
  ) as shipping
group by
  supp_nation,
  cust_nation,
  l_year
order by
  supp_nation,
  cust_nation,
  l_year
```



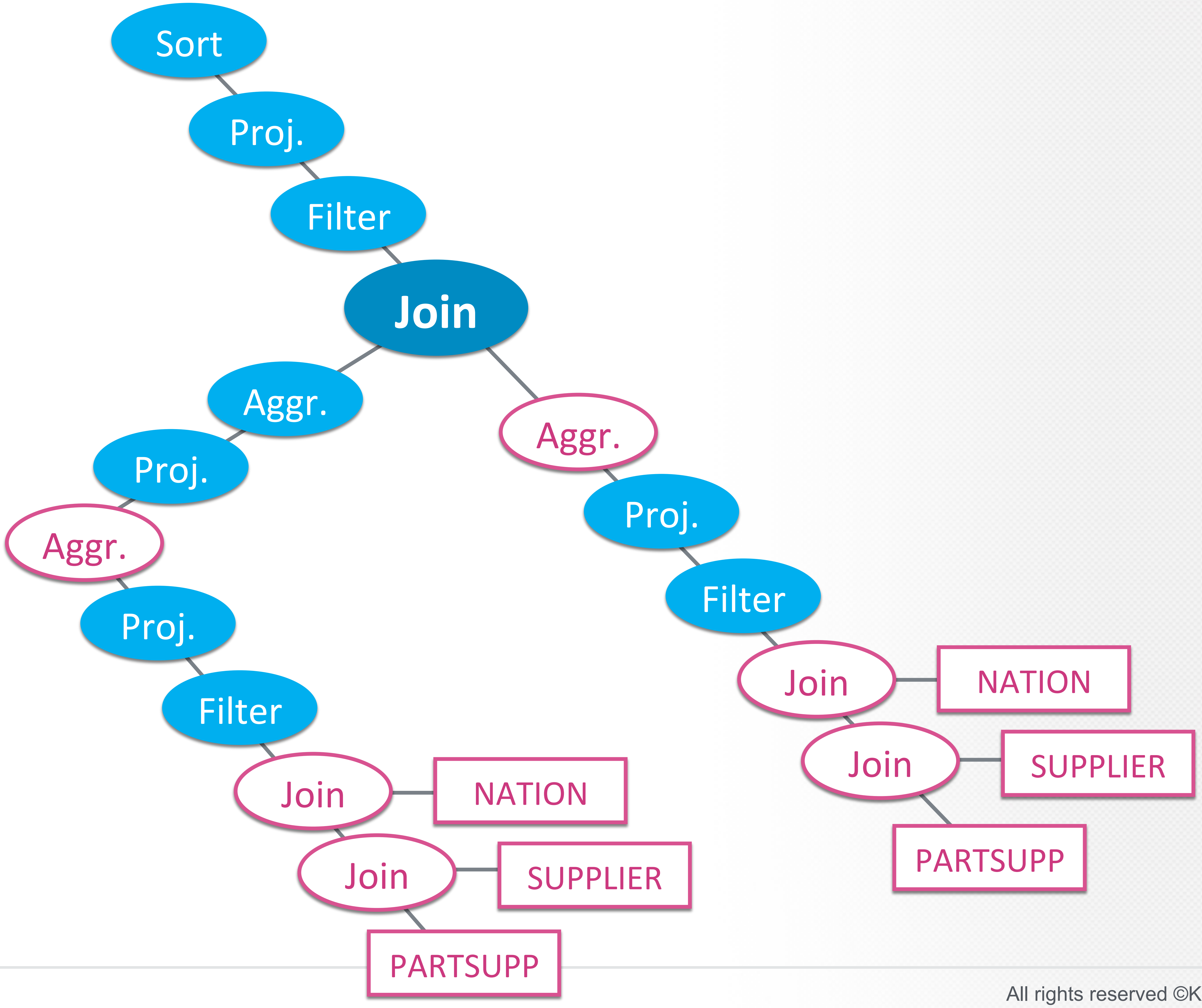
Complex Query 2

TPC-H query 11

- 3.42 sec (Hive+Tez 15.87 sec)
- 4 sub-queries, 1 online join

```
with q11_part_tmp_cached as (
  select
    ps_partkey,
    sum(ps_partvalue) as part_value
  from
    v_partsupp
    inner join supplier on ps_suppkey = s_suppkey
    inner join nation on s_nationkey = n_nationkey
  where
    n_name = 'GERMANY'
  group by ps_partkey
),
q11_sum_tmp_cached as (
  select
    sum(part_value) as total_value
  from
    q11_part_tmp_cached
)

select
  ps_partkey,
  part_value
from (
  select
    ps_partkey,
    part_value,
    total_value
  from
    q11_part_tmp_cached, q11_sum_tmp_cached
) a
where
  part_value > total_value * 0.0001
order by
  part_value desc;
```

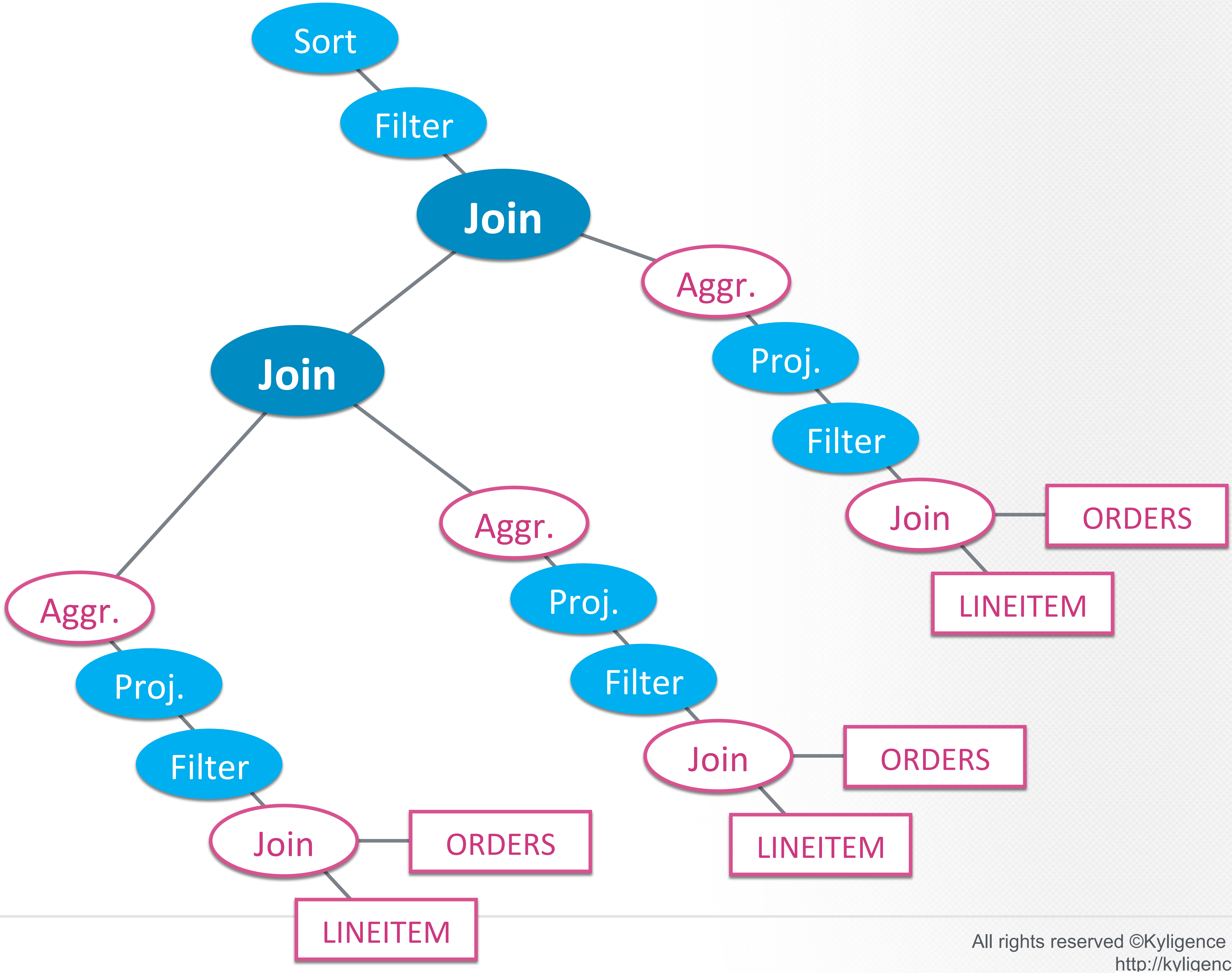


Complex Query 3

TPC-H query 12

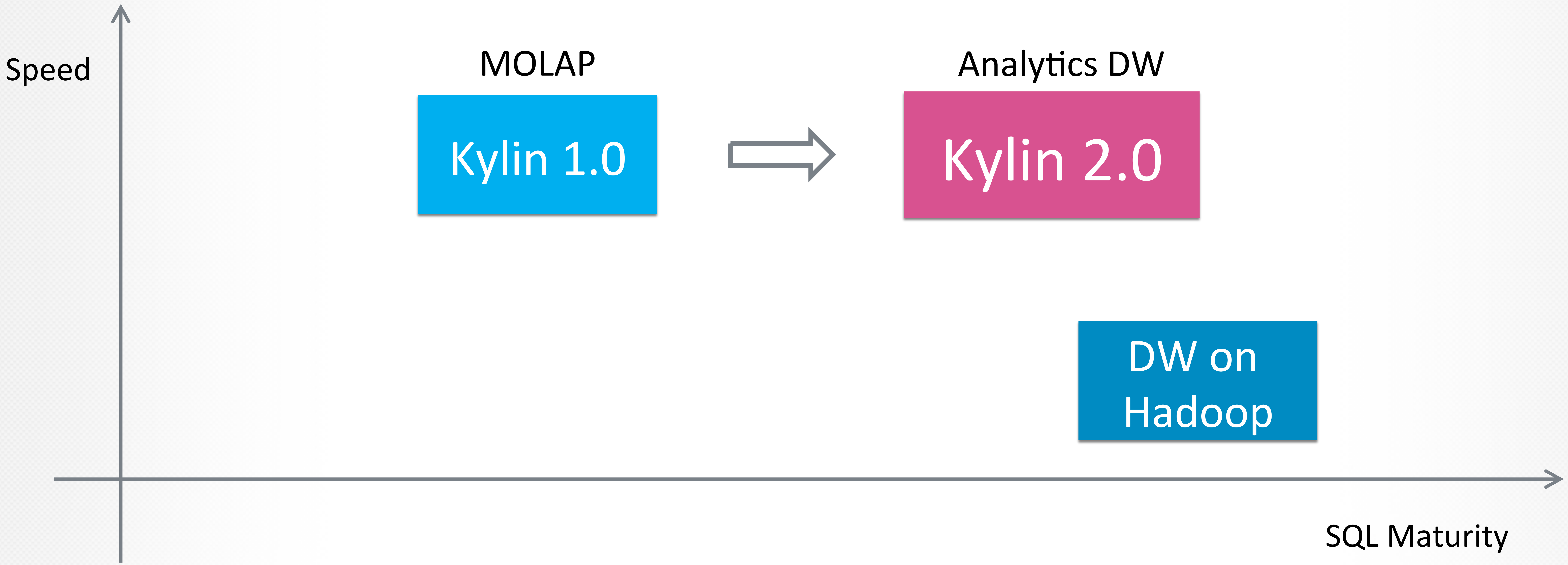
- 7.66 sec (Hive+Tez 12.64 sec)
- 5 sub-queries, 2 online joins

```
with in_scope_data as(
   select
      l_shipmode,
      o_orderpriority
   from
      v_lineitem inner join v_orders on l_orderkey = o_orderkey
   where
      l_shipmode in ('REG AIR', 'MAIL')
      and l_receiptdelayed = 1
      and l_shipdelayed = 0
      and l_receiptdate >= '1995-01-01'
      and l_receiptdate < '1996-01-01'
), all_l_shipmode as(
   select
      distinct l_shipmode
   from
      in_scope_data
), high_line as(
   select
      l_shipmode,
      count(*) as high_line_count
   from
      in_scope_data
   where
      o_orderpriority = '1-URGENT' or o_orderpriority = '2-HIGH'
   group by l_shipmode
), low_line as(
   select
      l_shipmode,
      count(*) as low_line_count
   from
      in_scope_data
   where
      o_orderpriority <> '1-URGENT' and o_orderpriority <> '2-HIGH'
   group by l_shipmode
)
select
   al.l_shipmode, hl.high_line_count, ll.low_line_count
from
   all_l_shipmode al
   left join high_line hl on al.l_shipmode = hl.l_shipmode
   left join low_line ll on al.l_shipmode = ll.l_shipmode
order by
   al.l_shipmode
```



More than MOLAP

- Supports complex data models and sub-queries; Runs TPC-H
- Percentile / Window / Time functions



总结

Apache Kylin 2.0

- Kylin 2.0 Beta 可供下载.

- Spark cubing

- 雪花模型的支持
- 可尝试的TPC-H benchmark
- 时间函数/窗口函数/百分比函数

- 近实时流式处理

What is next

- Hadoop 3.0 支持(Erasure Coding)

- 完善Spark Cubing

- 连接更多数据源(JDBC, SparkSQL)

- 替换存储层(Kudu?)

- 支持真正实时 lambda architecture