OSC源创会 Opensource Innovation Meetup | IT大咖说 知识分享平台

Kyligence

Apache Kylin 2.0

Spark构建引擎

李栋 | Dong Li

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

构建领先的
全球开源社区

hadoop → cloudera, Hortonworks

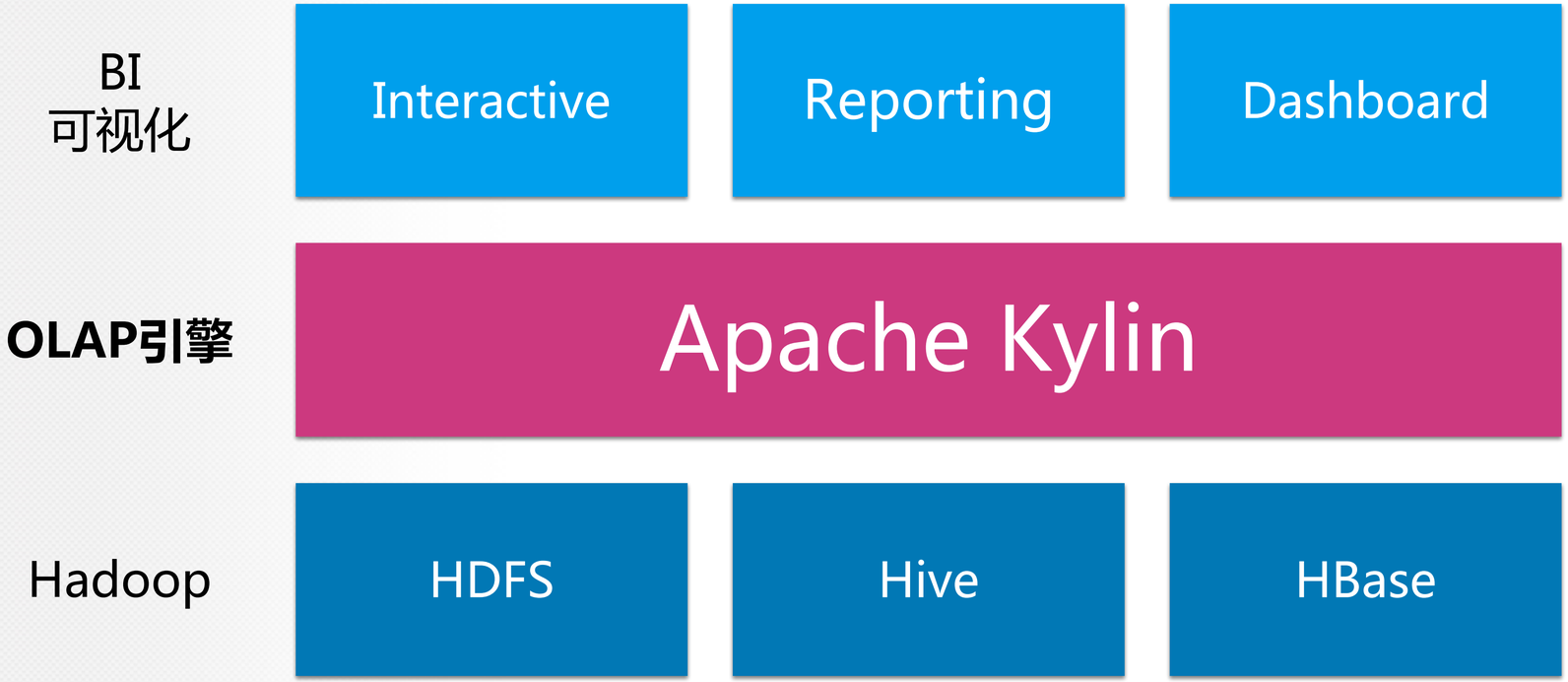
Apache Spark → databricks

Apache Kafka → confluent

Apache Kylin → Kyligence

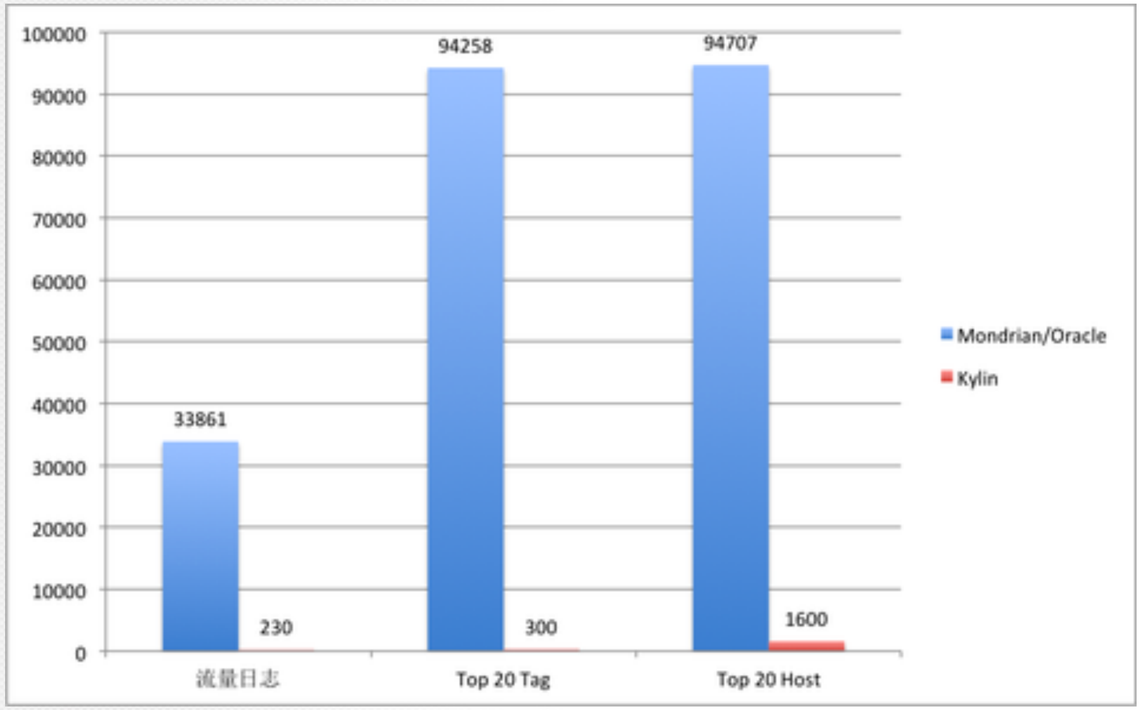
Apache Kylin全球案例

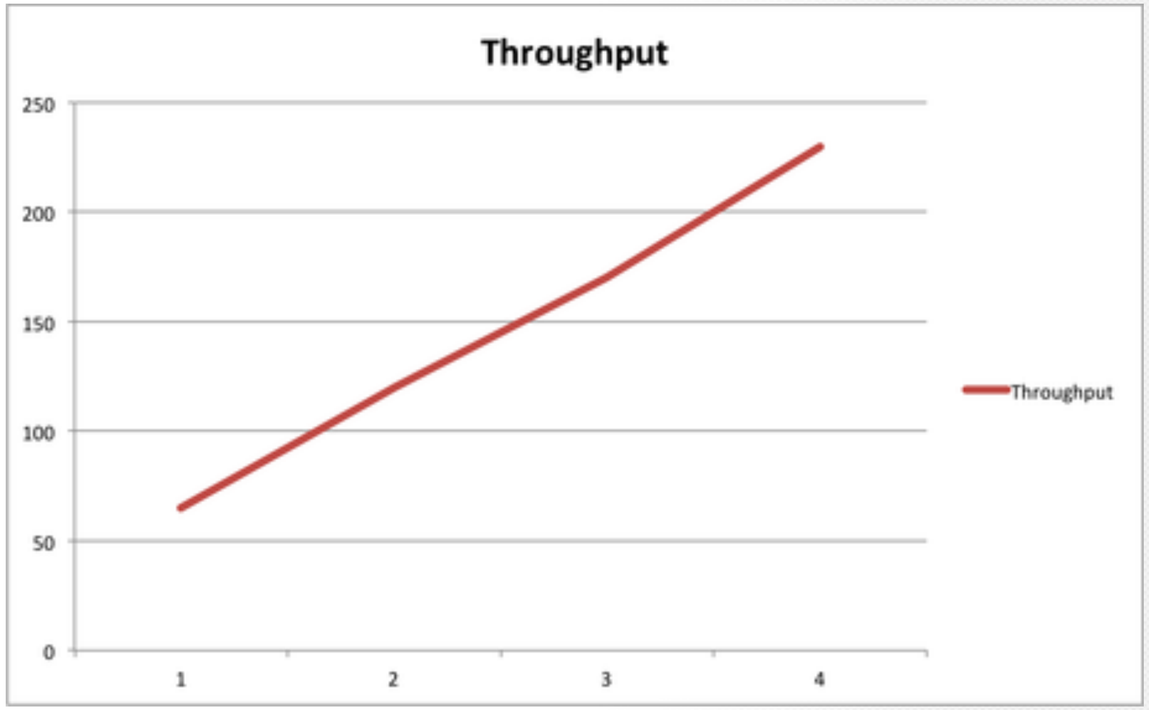
Apache Kylin是什么？



- 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

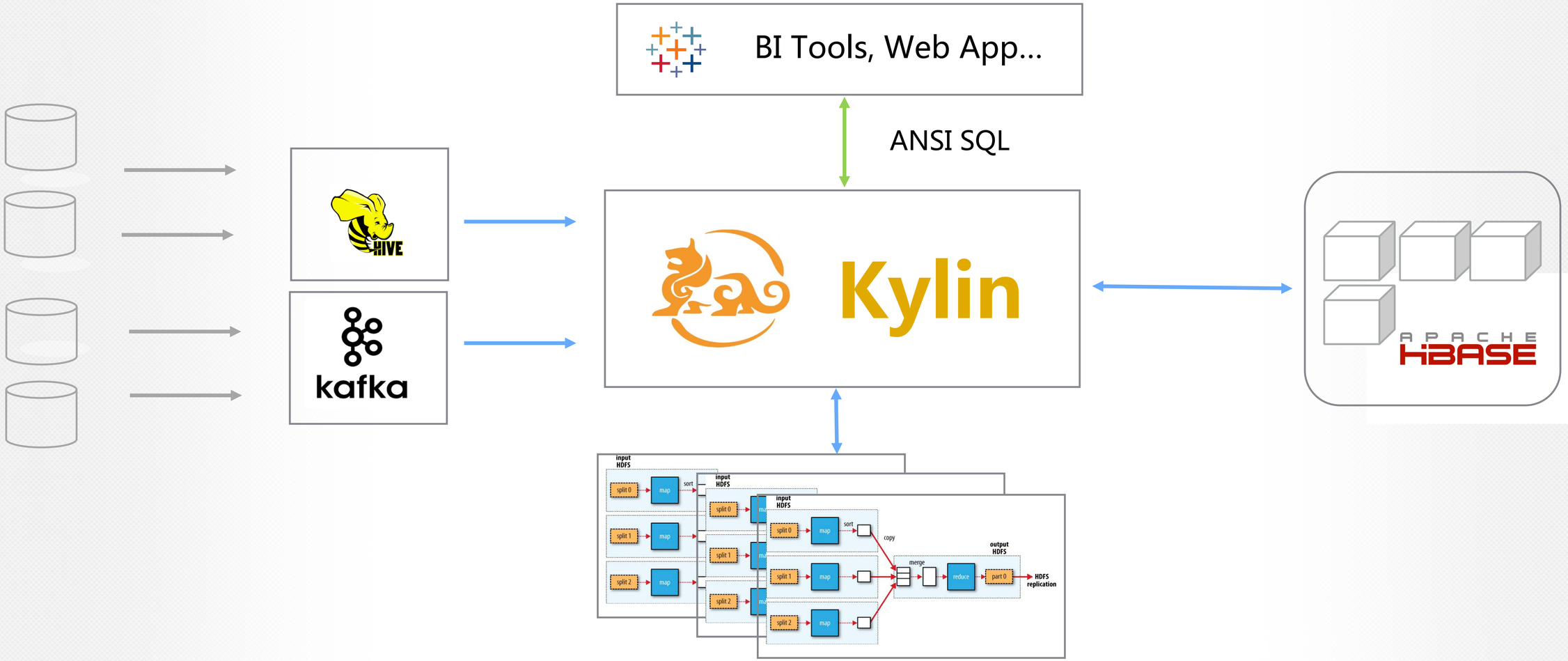
Apache Kylin是什么？





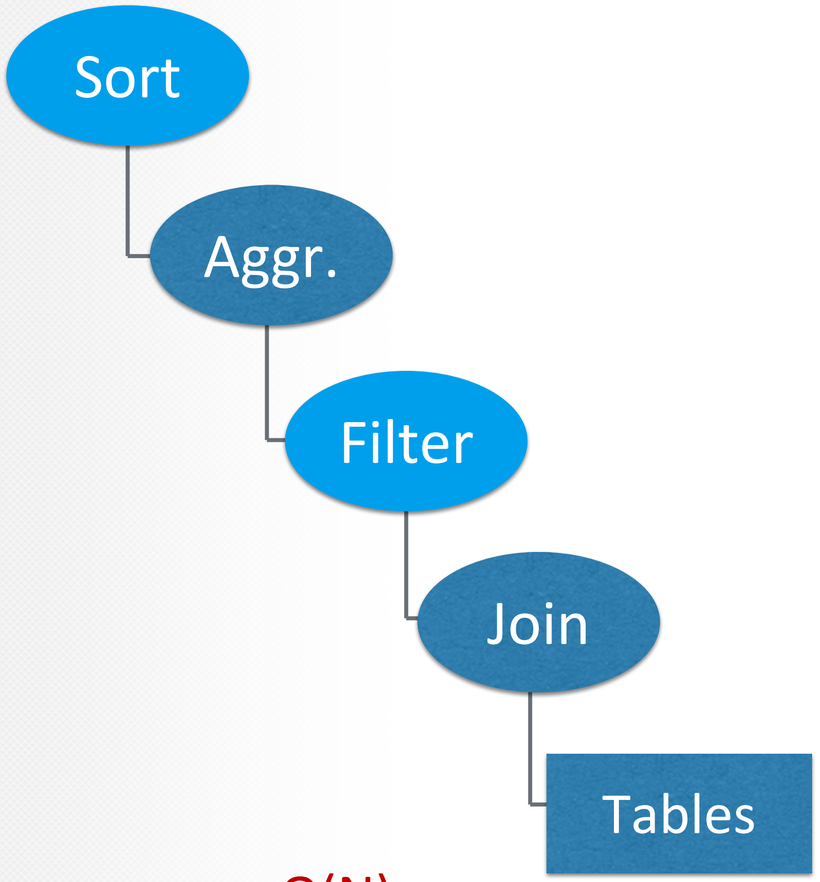
By 网易:
http://www.bitstech.net/2016/01/04/kylin-olap/

Apache Kylin in the Zoo



Offline Cubing

Apache Kylin为什么快？

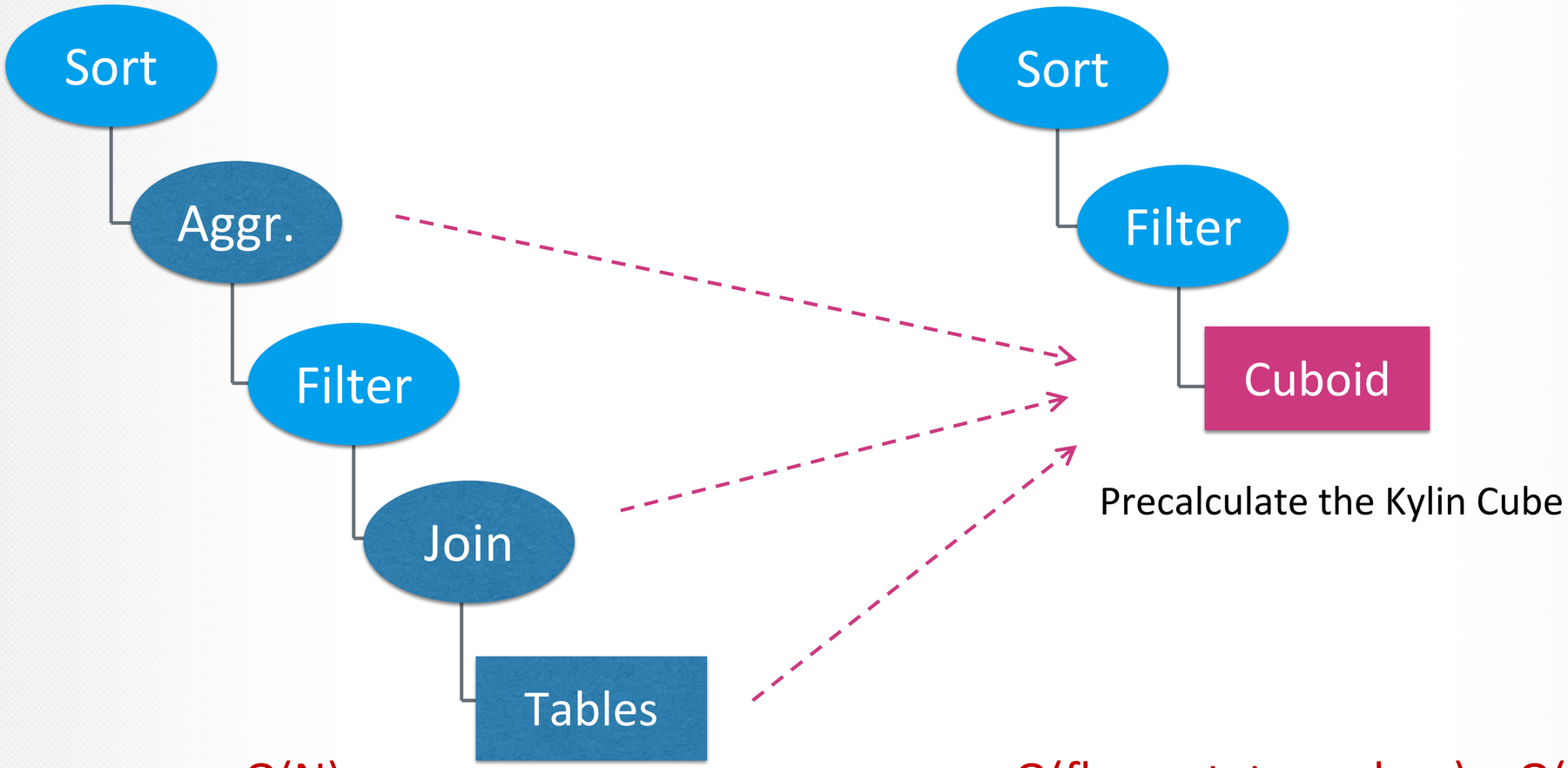


A sample query:

Report revenue by “returnflag” and “orderstatus”

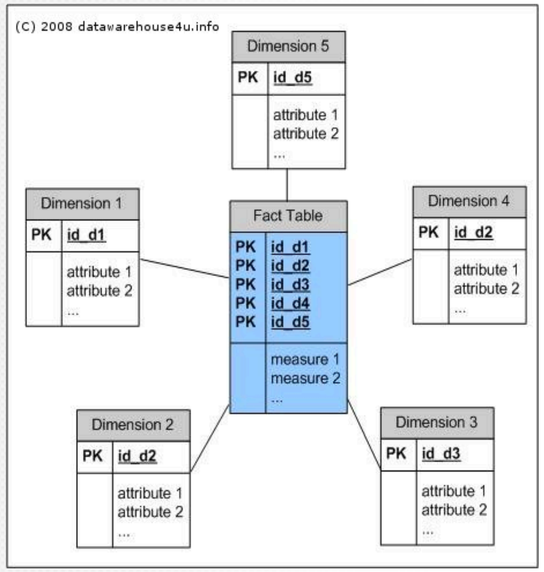
```
select
   l_returnflag,
   o_orderstatus,
   sum(l_quantity) as sum_qty,
   sum(l_extendedprice) as sum_base_price
   …
from
   v_lineitem
   inner join v_orders on l_orderkey = o_orderkey
where
   l_shipdate <= '1998-09-16'
group by
   l_returnflag,
   o_orderstatus
order by
   l_returnflag,
  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

Apache Kylin为什么快？

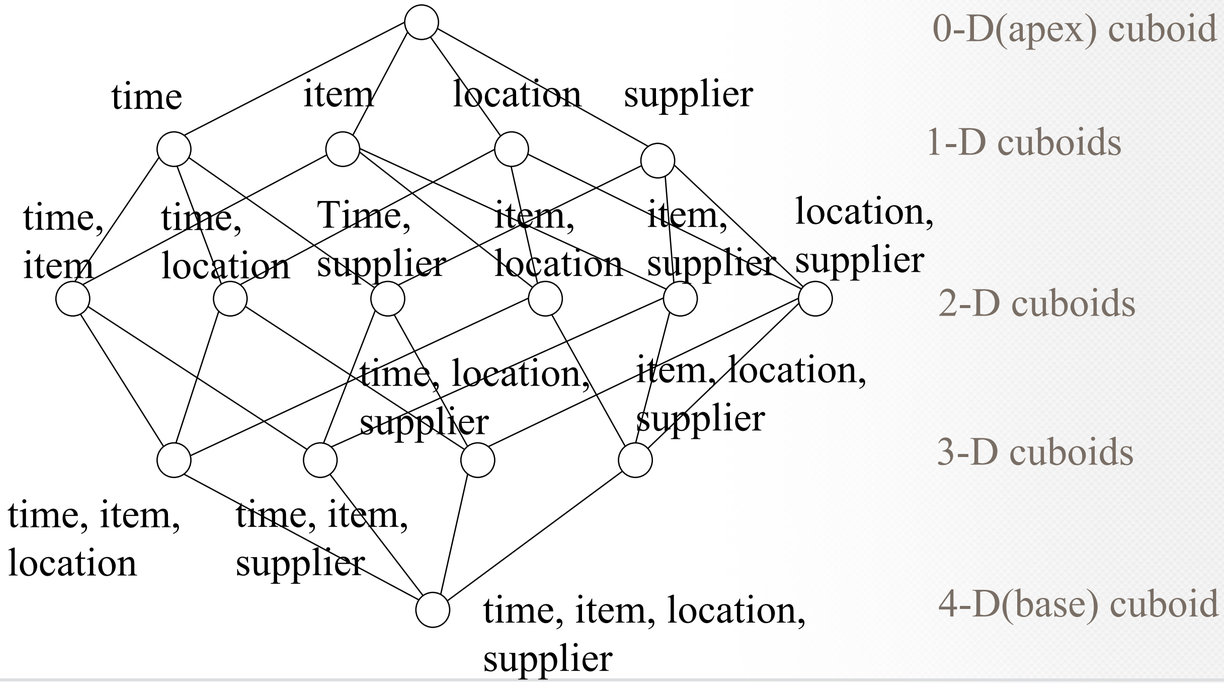


Apache Kylin 关键在于预计算

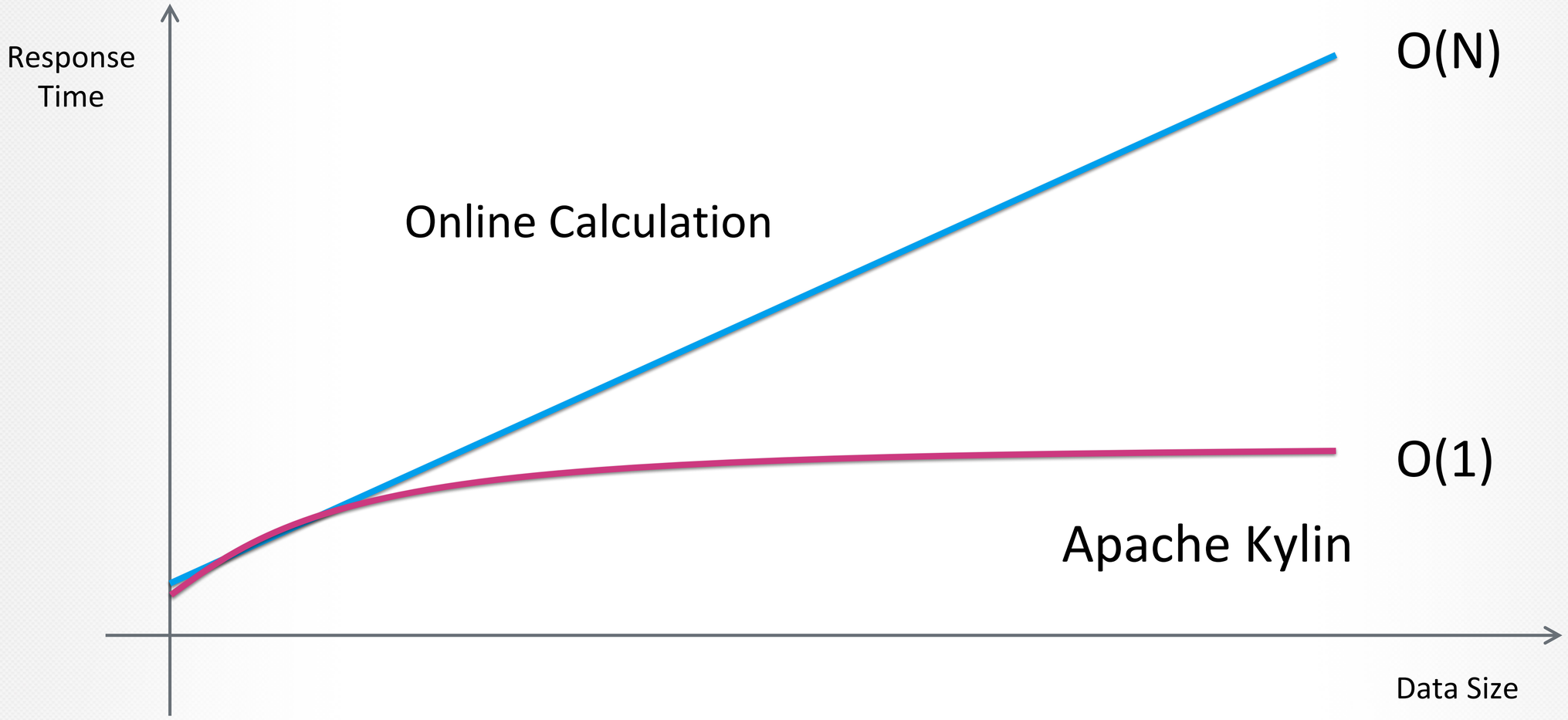
- 基于cube理论
- Model 和 Cube 定义了预计算的范围
- Build Engine 执行构建任务
- Query Engine 在预计算的结果之上完成查询



预计算



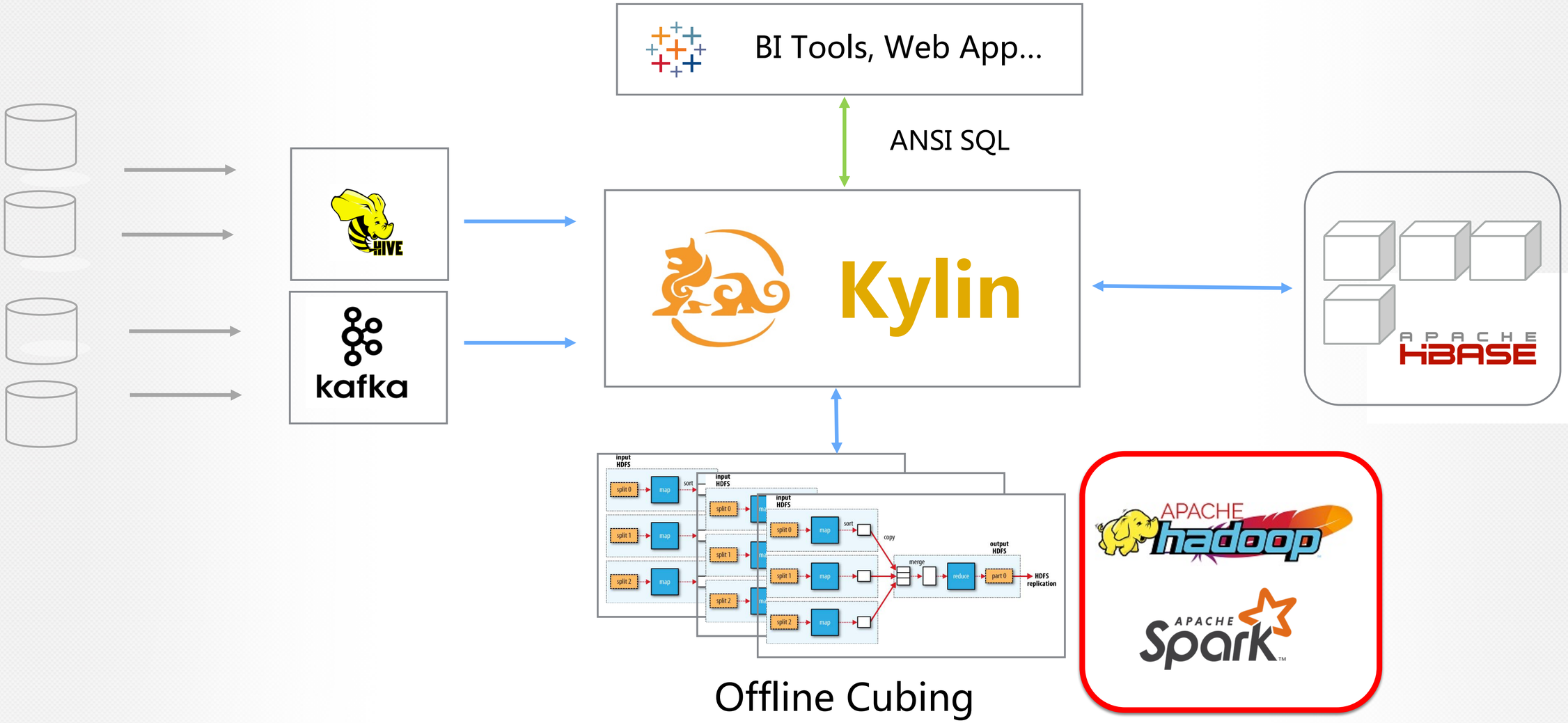
O(1)复杂度



Spark 构建引擎

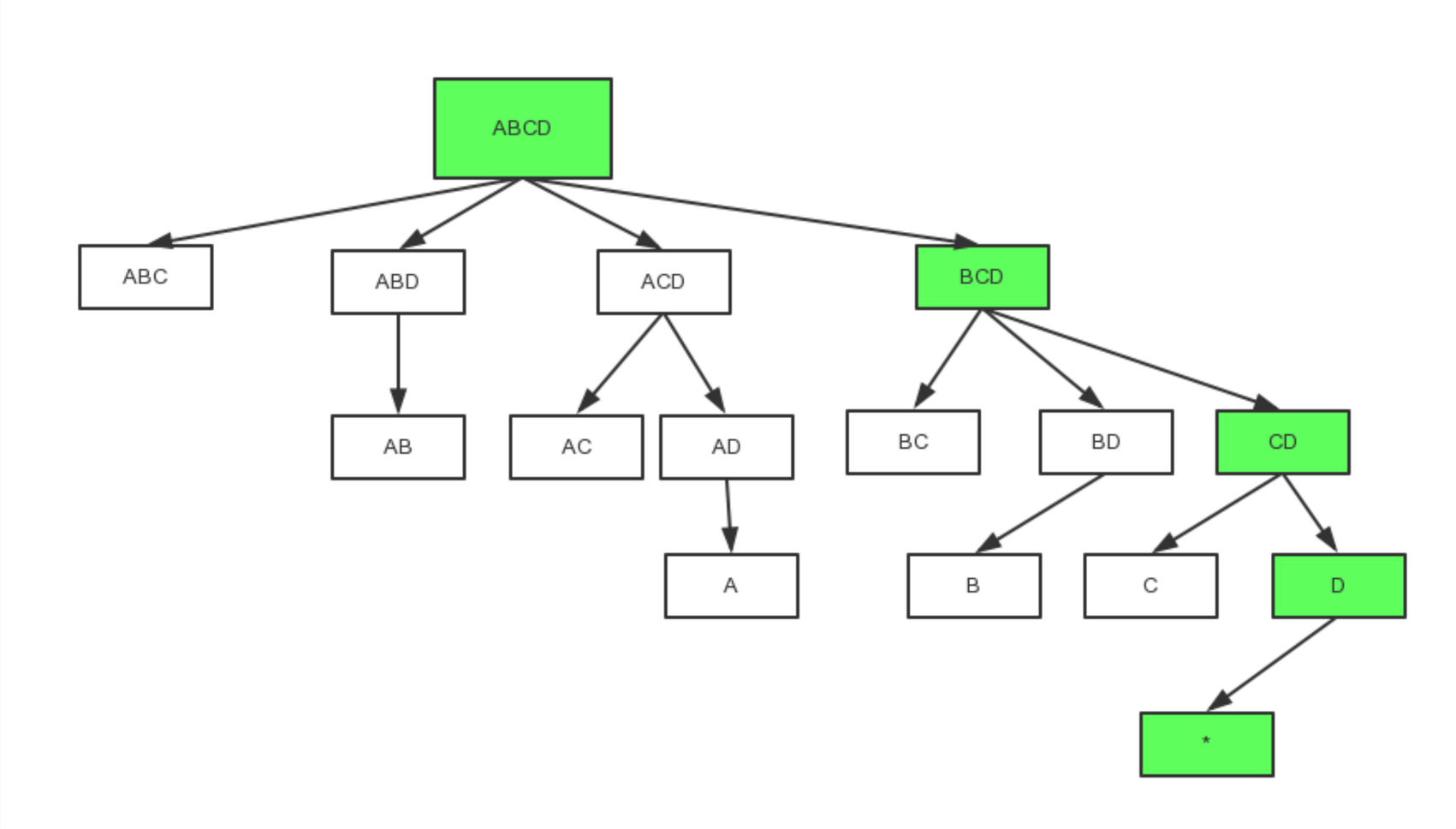
减少一半的构建时间！

Cubing with Spark



Offline Cubing

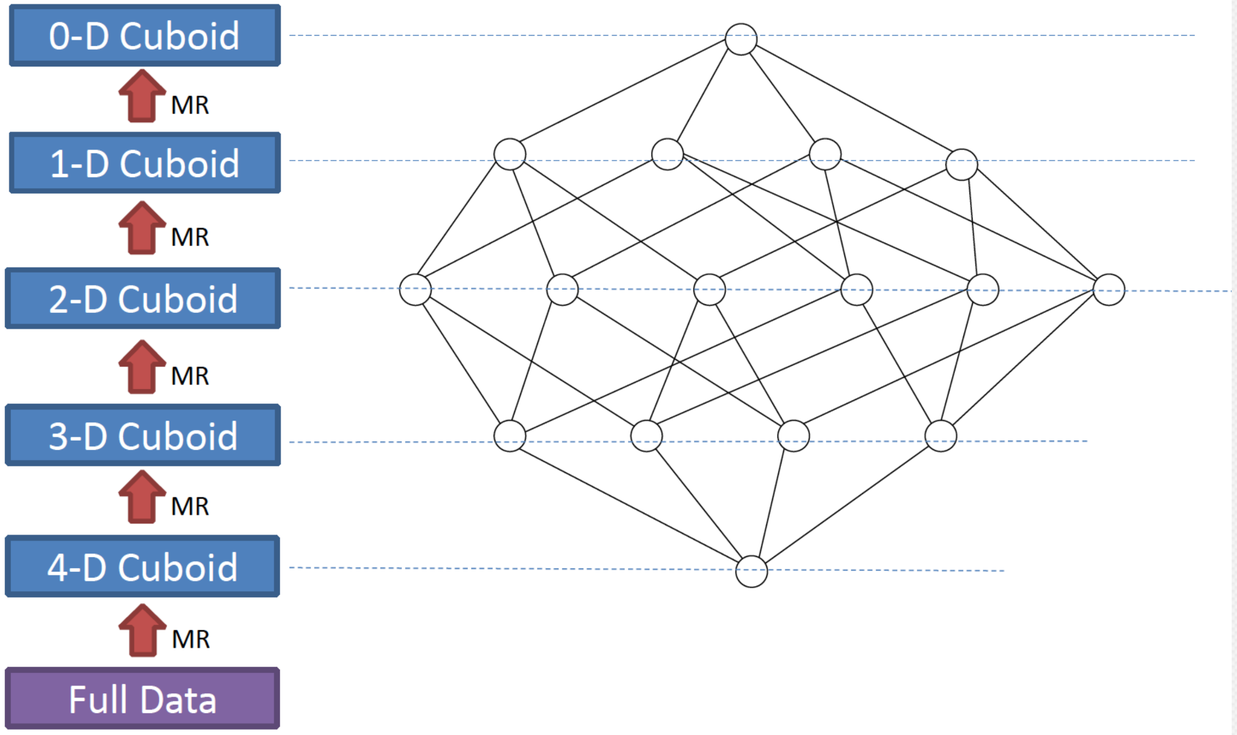
Cuboid 生成树



构建算法 - Layer Cubing

标准的构建算法：Layer Cubing

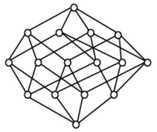
- 启动多轮MR任务
- 将大型shuffle切分到多个stage
- 稳定，但是在构建时间上并不是最优的

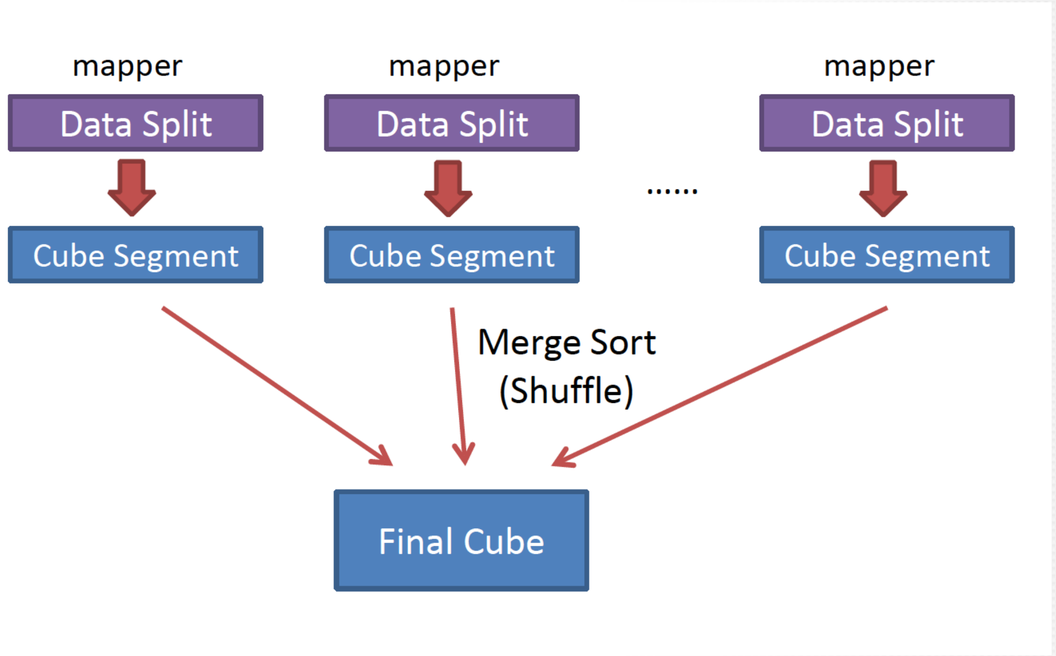


构建算法 - In-memory Cubing

In-memory Cubing是对Layer 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





MR 构建引擎

比较稳定

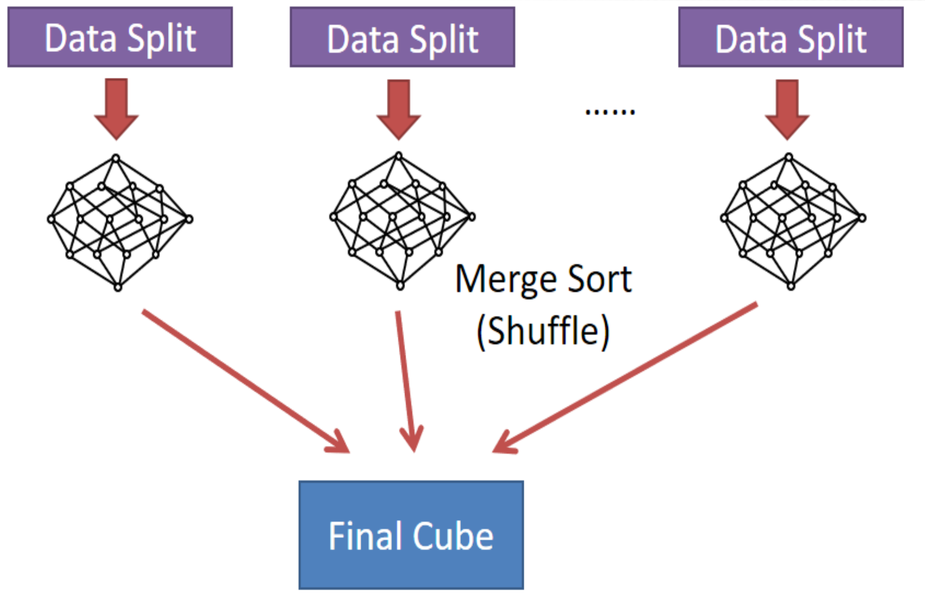
在某些场景下性能都有待提高

社区迫不及待地想要尝试使用其他技术来加速cubing

Spark Cubing in 1.5 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

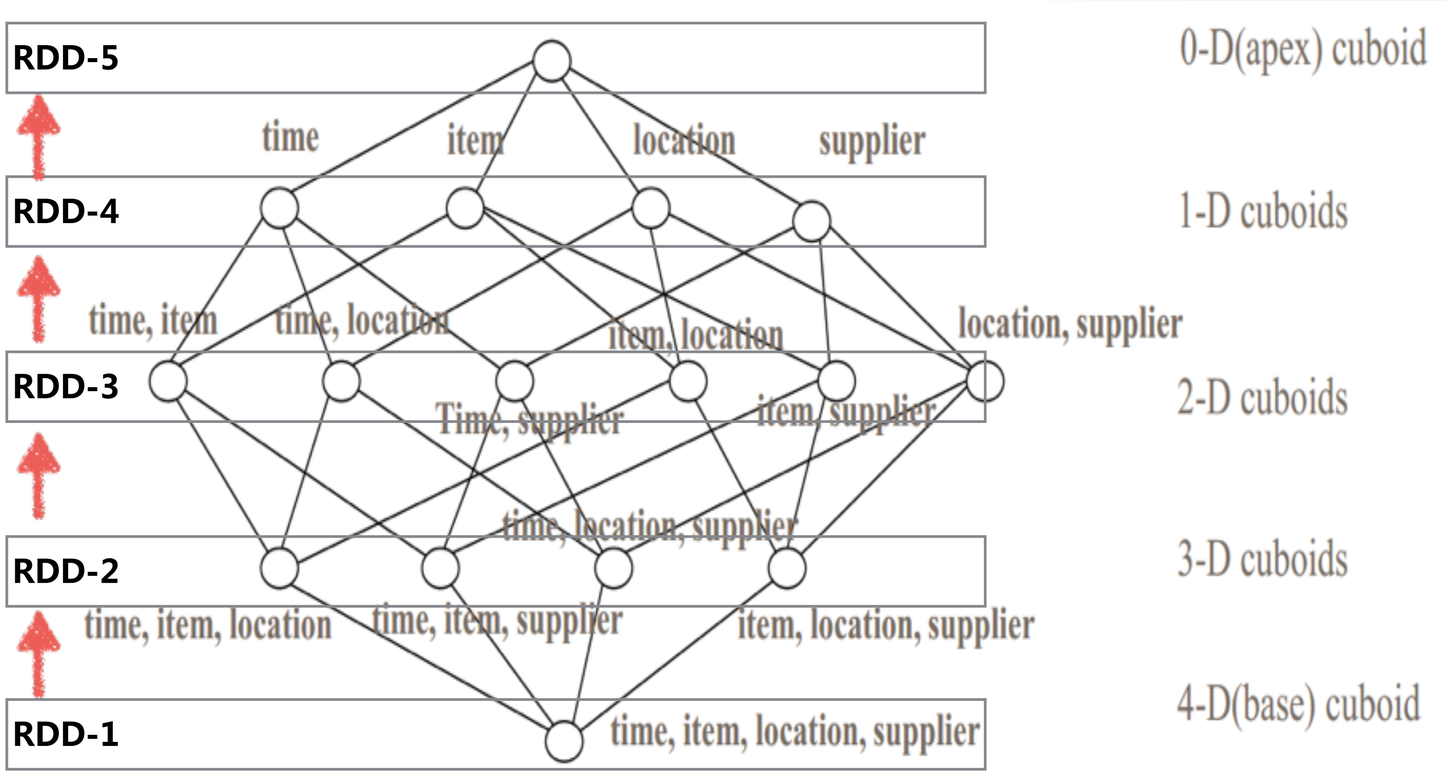
- 它只是简单地将In-memory cubing移植到Spark上
- 使用一轮RDD转换计算整个cube
- 并未观察到明显改进
- Spark 计算方式与MR并无明显区别



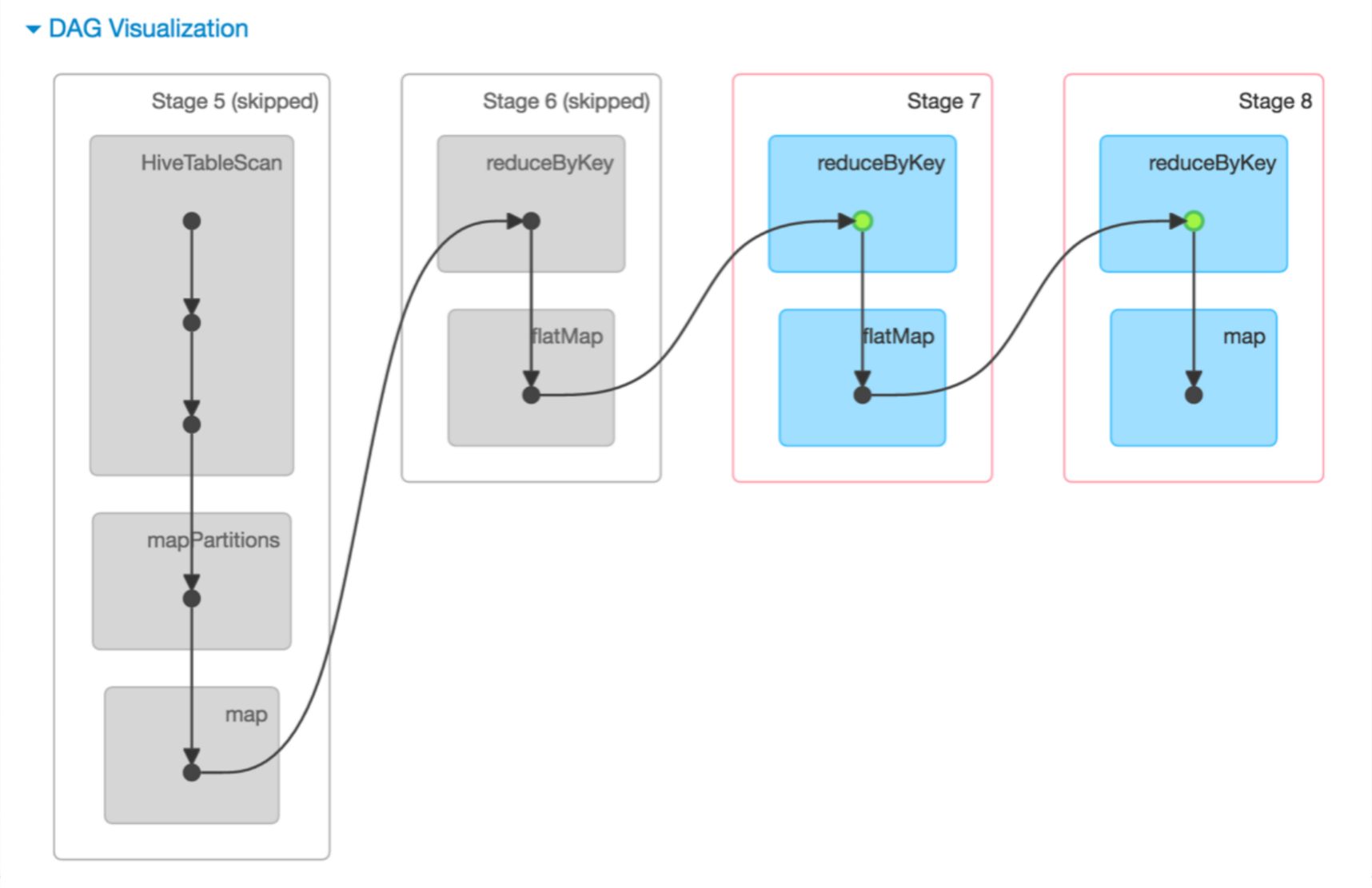
Spark Cubing in 2.0

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, 把reduce替换为reduceByKey, 可以复用大部分代码



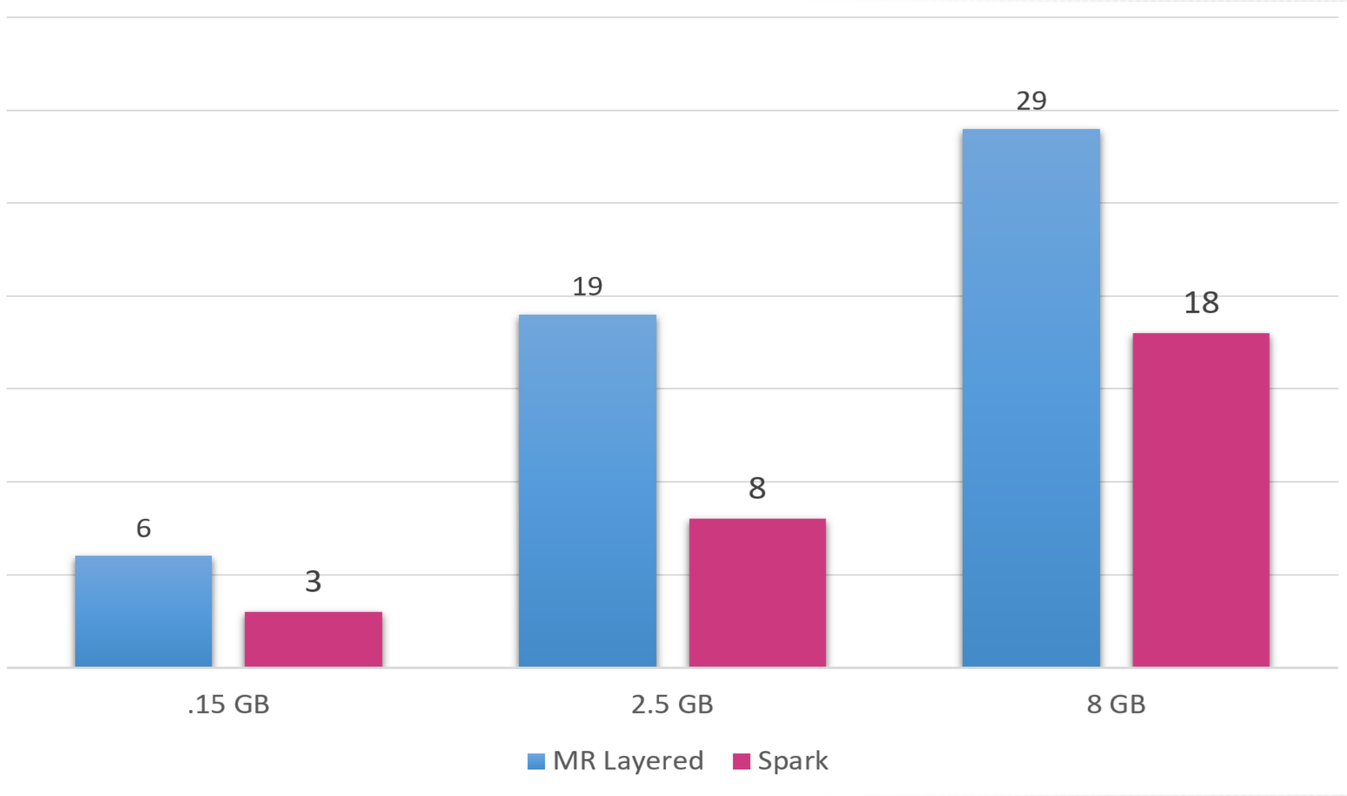
计算3-D Cuboids的DAG



Spark Cubing vs. MR Layered Cubing

减半构建时间， 但是可以观察到优势随着数据量的增加而减少

- 4 节点的集群
- Spark 1.6.3 on YARN
- 24 vcores, 30 GB memory
- 3 data sets of increasing size:
  .15 GB / 2.5 GB / 8 GB

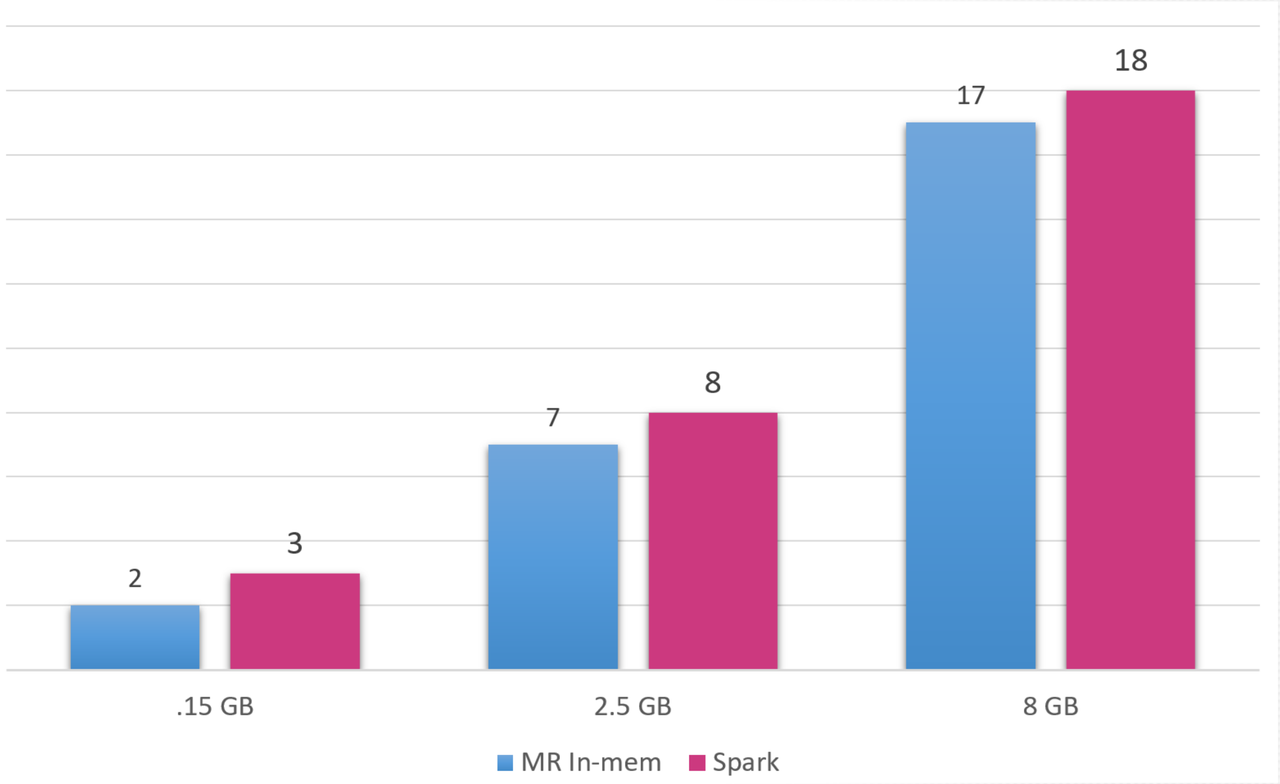


Spark Cubing vs. MR In-mem Cubing

几乎一样快，但是更适合通用的数据集

- In-mem cubing 期望良好分区的数据，
  在随机分布的数据上表现次优

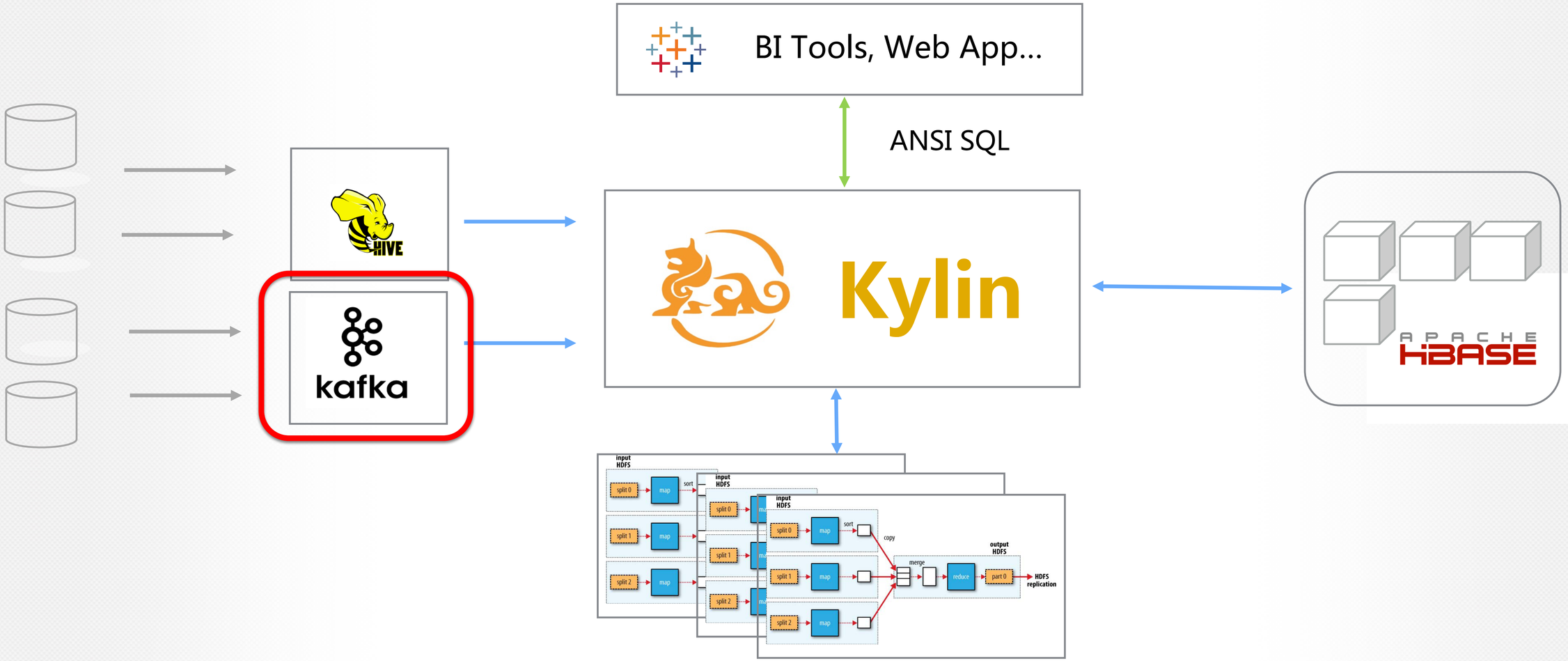
- Spark cubing 更适合随机分布的数据



近实时流数据处理

构建分钟级别延迟的cube

Cubing from Kafka



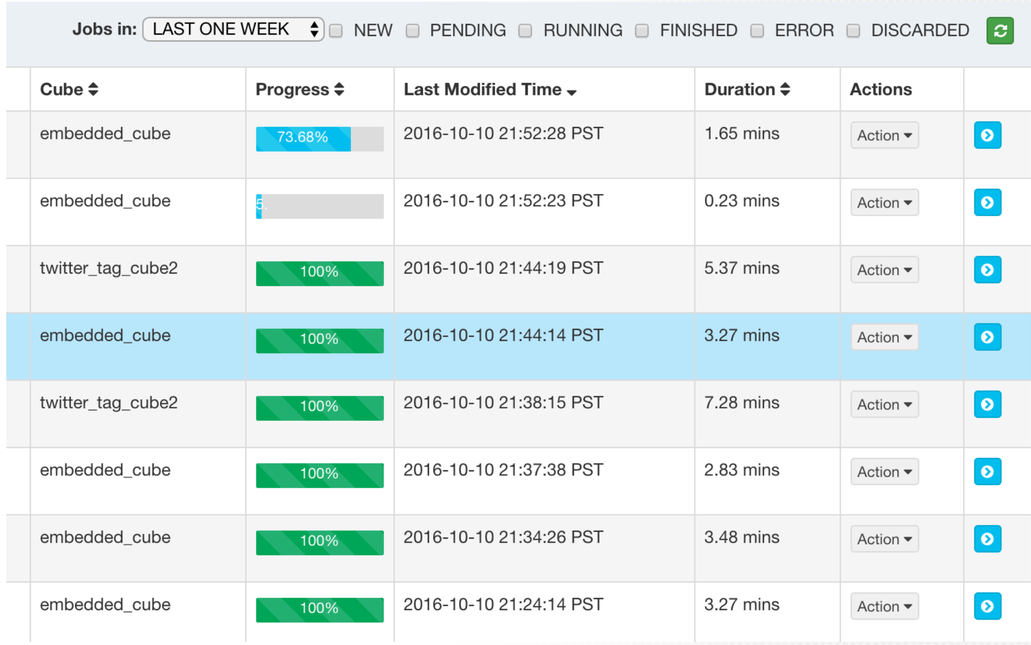
Offline Cubing

Demo of Twitter Analysis

http://hub.kyligence.io

每2分钟触发一次增量构建，每次构建在3分钟完成。

- 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



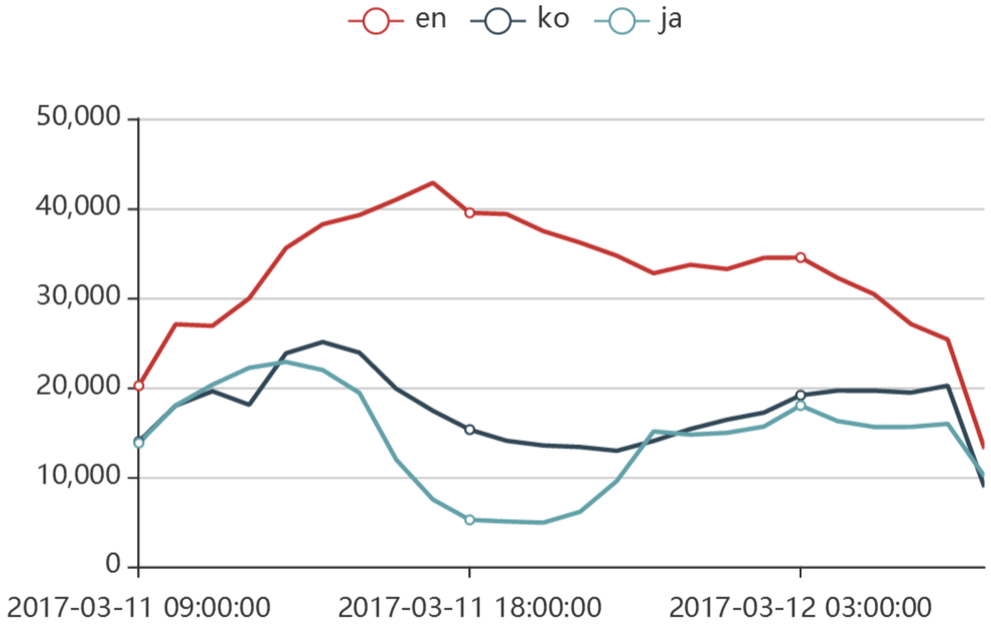
Real-Time Streaming Analytics for Twitter

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

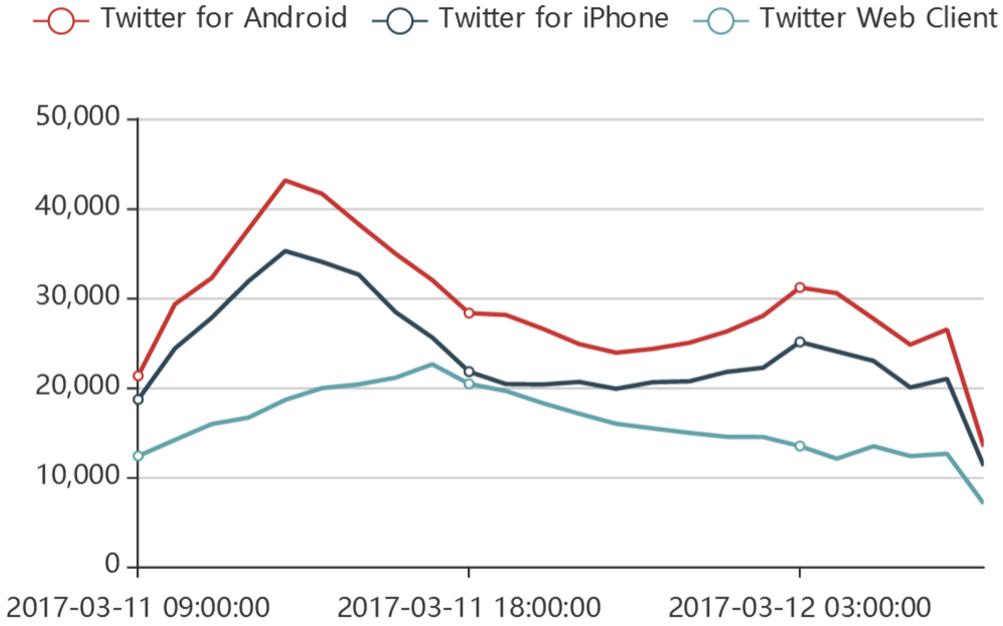
Language

en, ko, ja



Device

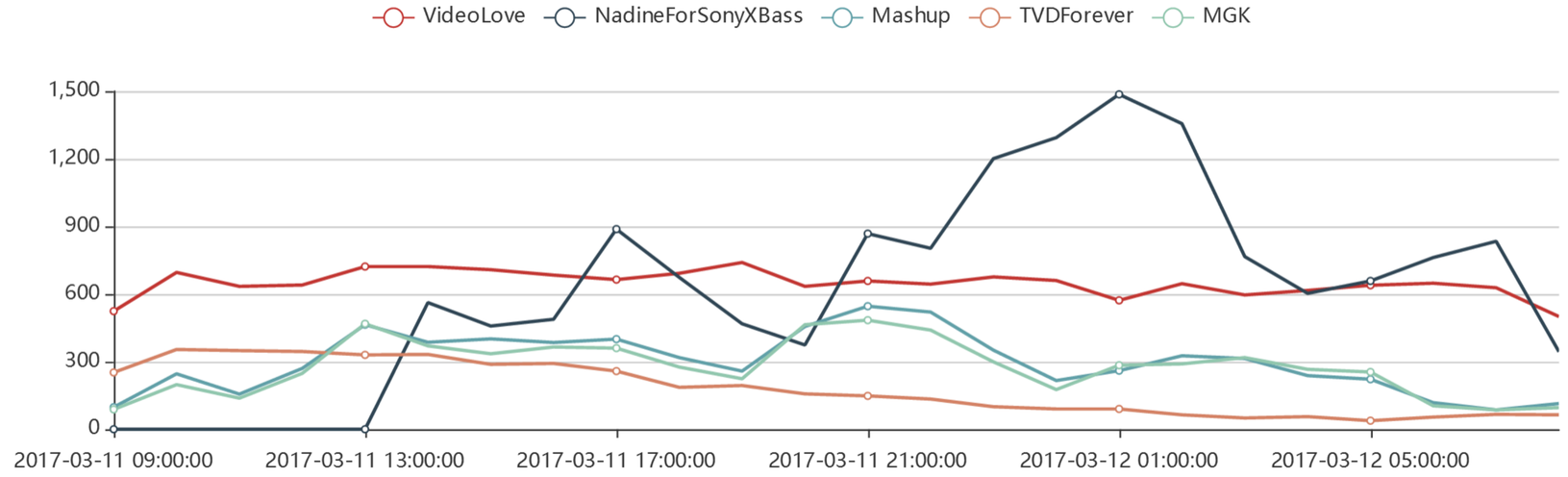
Twitter for Android, Twitter for iPhone, Twitter Web Client



en

Tag

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



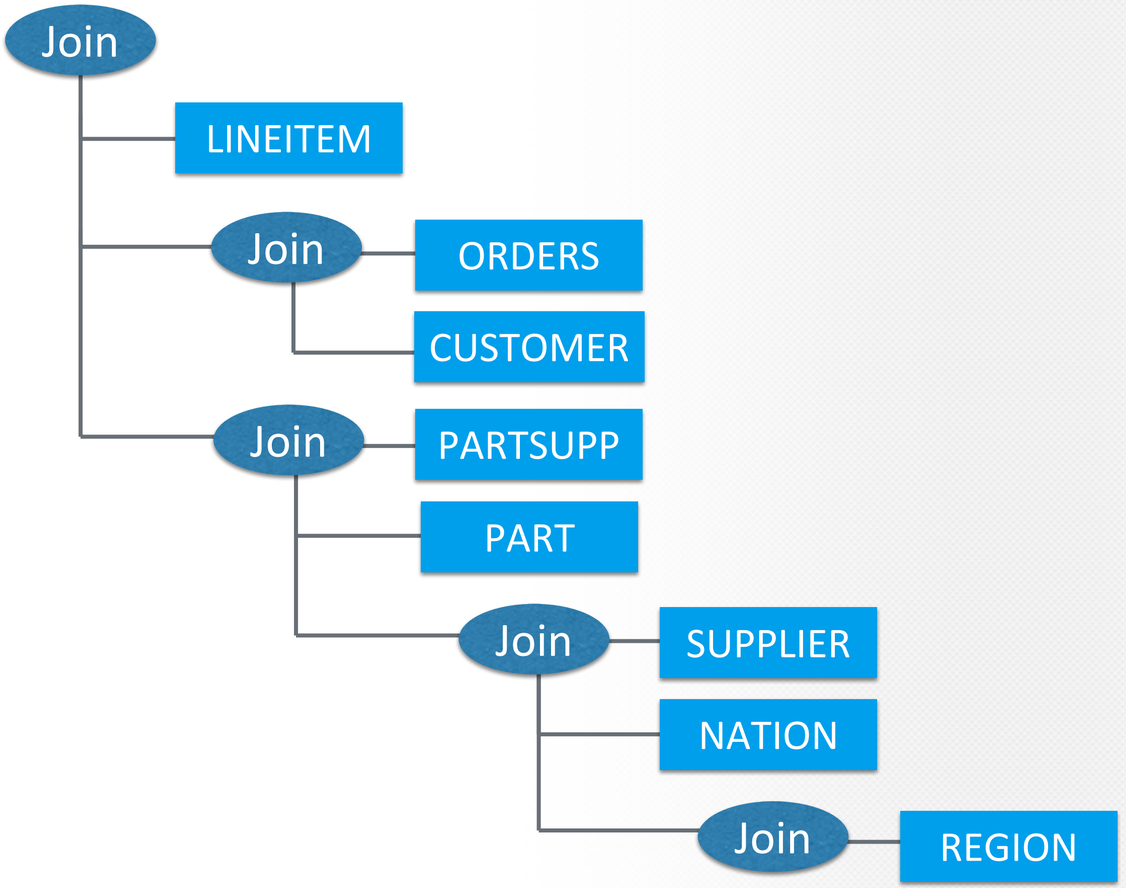
TPC-H Benchmark

全面的雪花模型支持

Kylin 2.0 支持雪花模型

解决了Kylin 1.0的很多功能限制

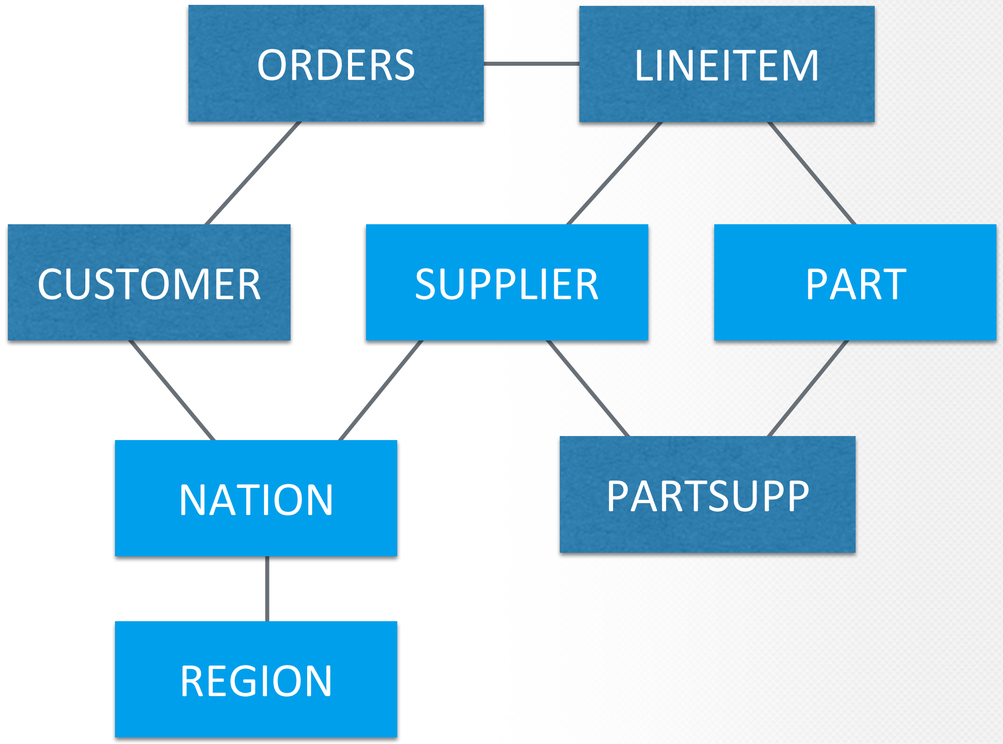
- Snowflake schema support (KYLIN-1875)
- Allow table be joined multiple times
- Many bug fixes regarding joins and sub-queries
- Support complex models of any kind, support flexible queries on the models



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)

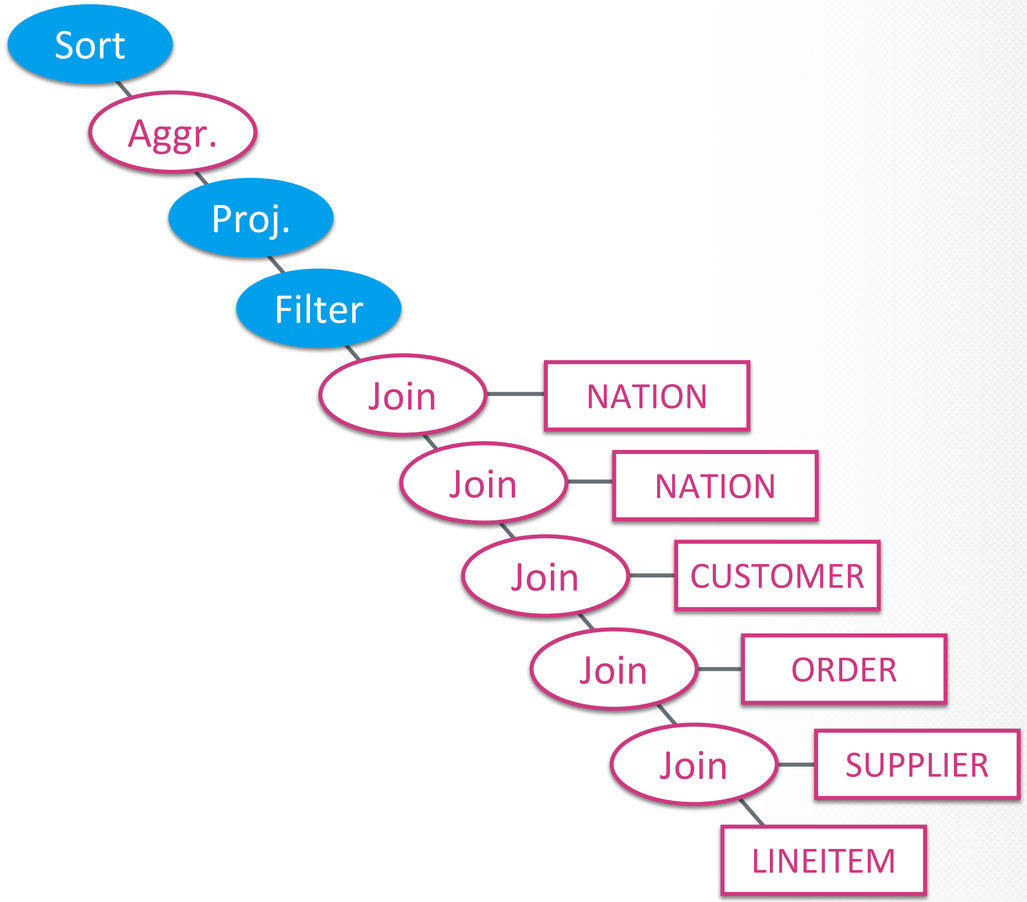
- Pre-calculation can answer very complex queries
- Goal is functionality at this stage
- Try it: https://github.com/Kyligence/kylin-tpch

复杂查询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
```



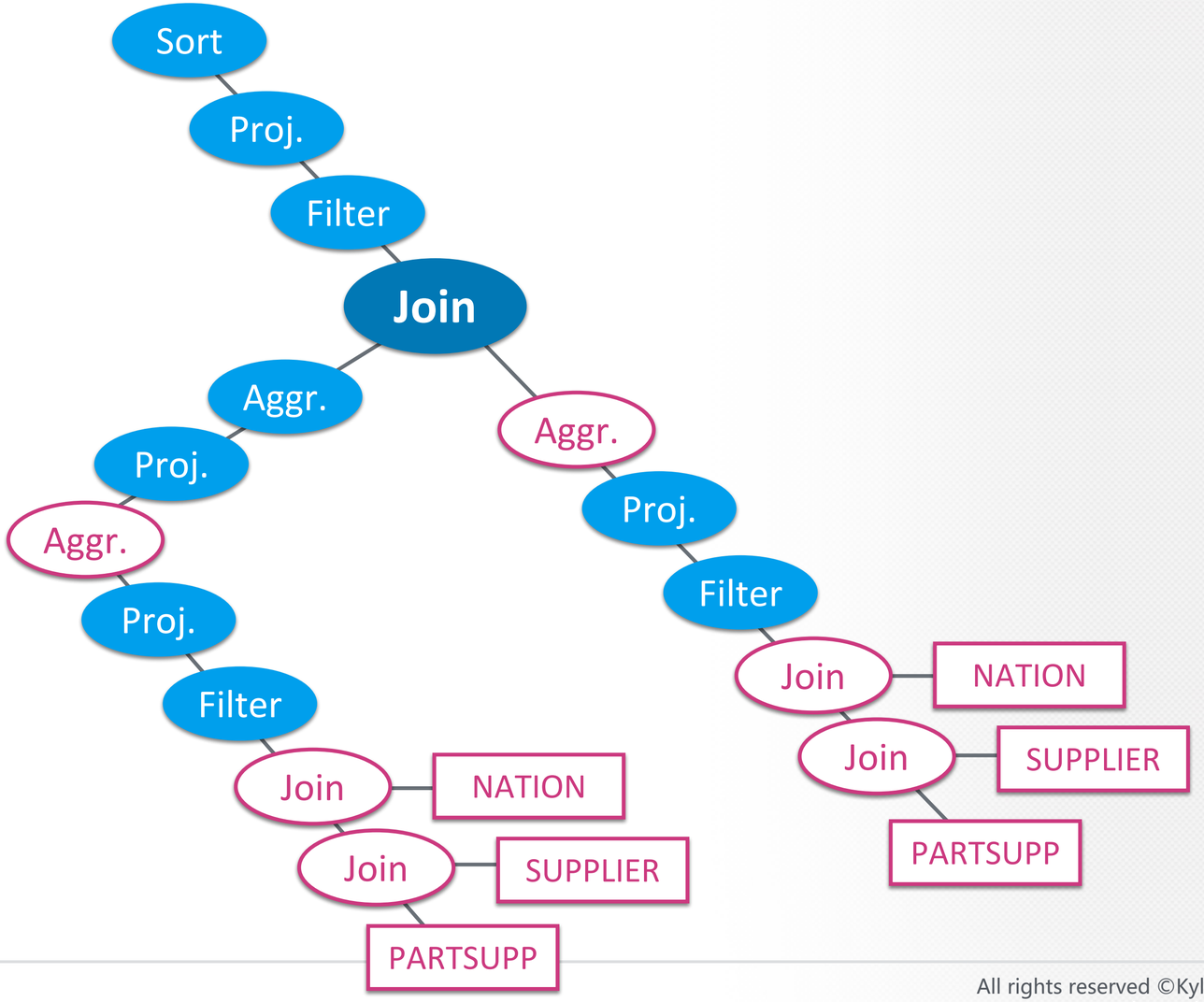
复杂查询2

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;
```

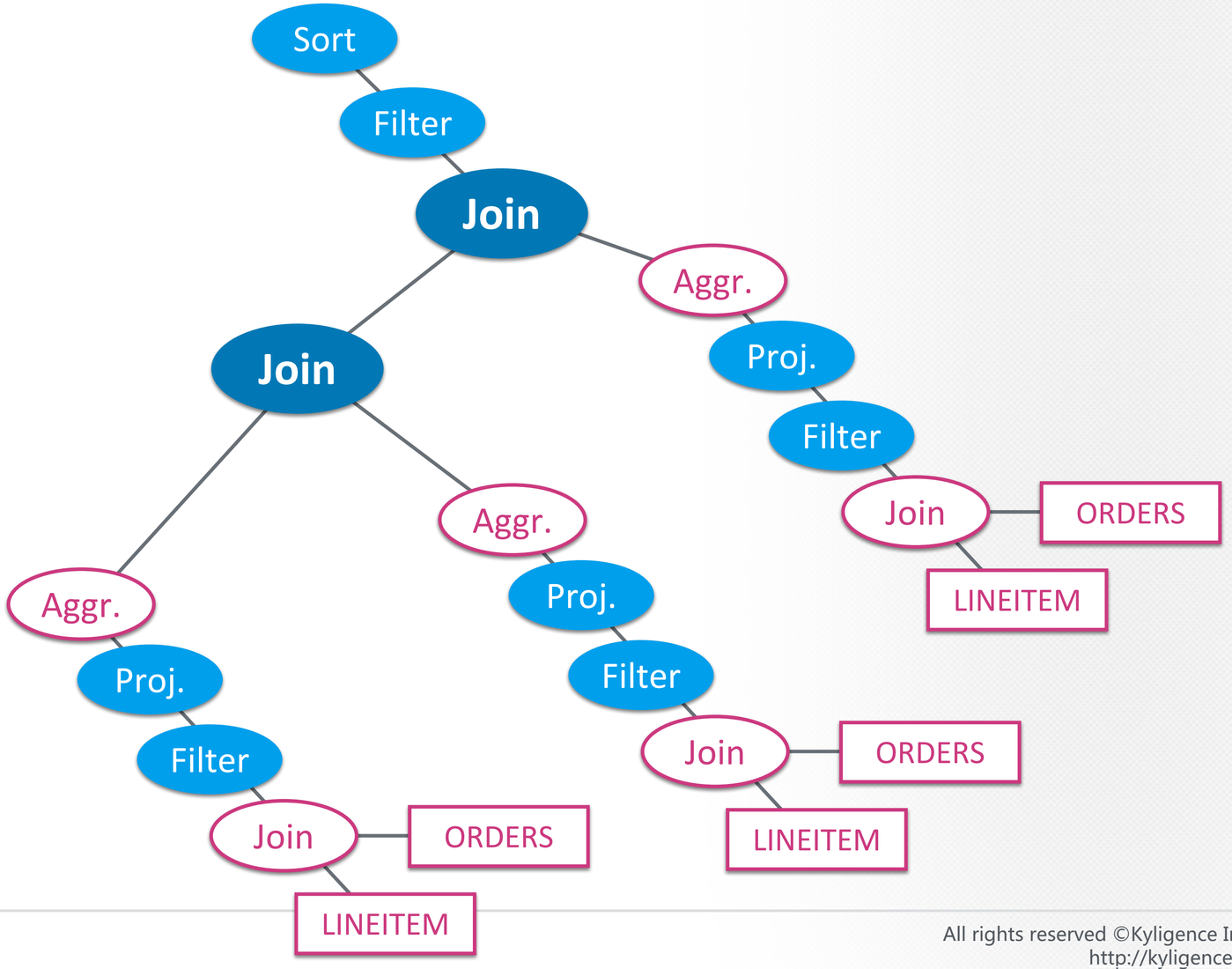


复杂查询3

TPC-H query 12

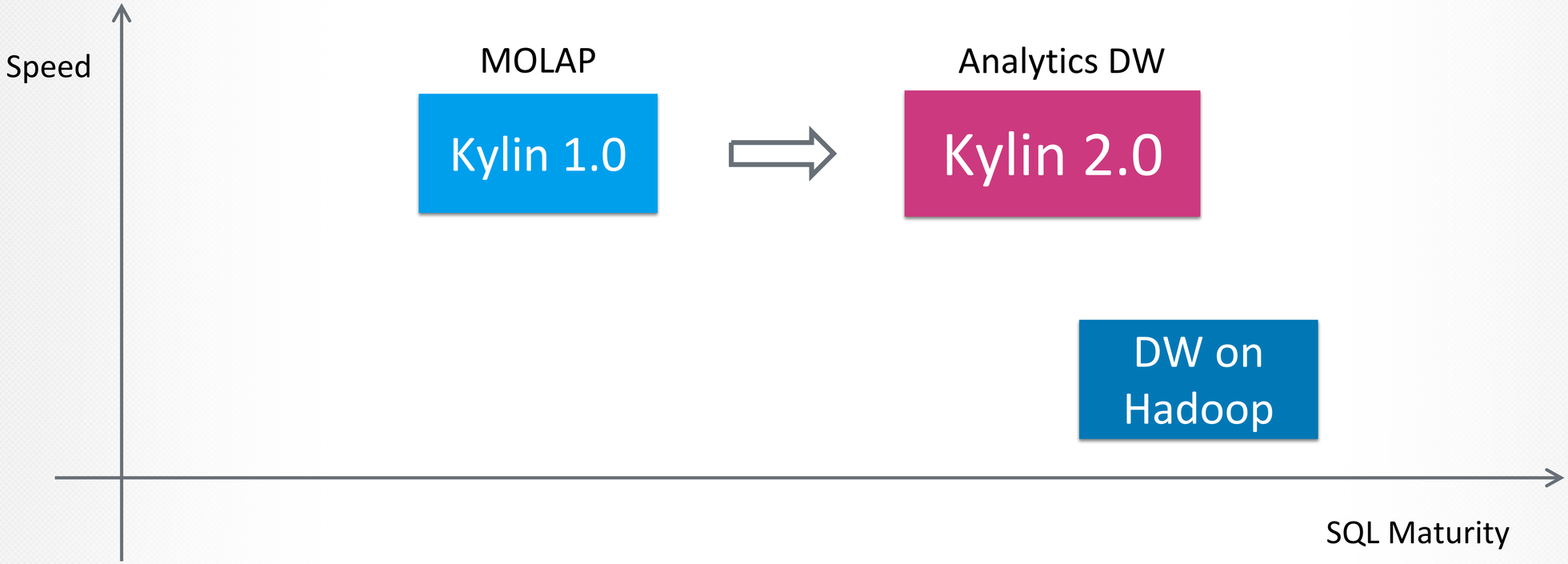
- 7.66 sec (Hive+Tez 12.64 sec)
- 5 sub-queries, 2 online joins

```
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
```



不仅仅是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构建引擎
- 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

感谢聆听！

OSC源创会 Opensource Innovation Meetup | IT大咖说 知识分享平台

Kyligence Inc

info@kyligence.io

Twitter: @Kyligence
http://kyligence.io

WeChat：Kyligence

WeChat：ApacheKylin

Apache Kylin

dev@kylin.apache.org

Twitter: @ApacheKylin
http://kylin.apache.org