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

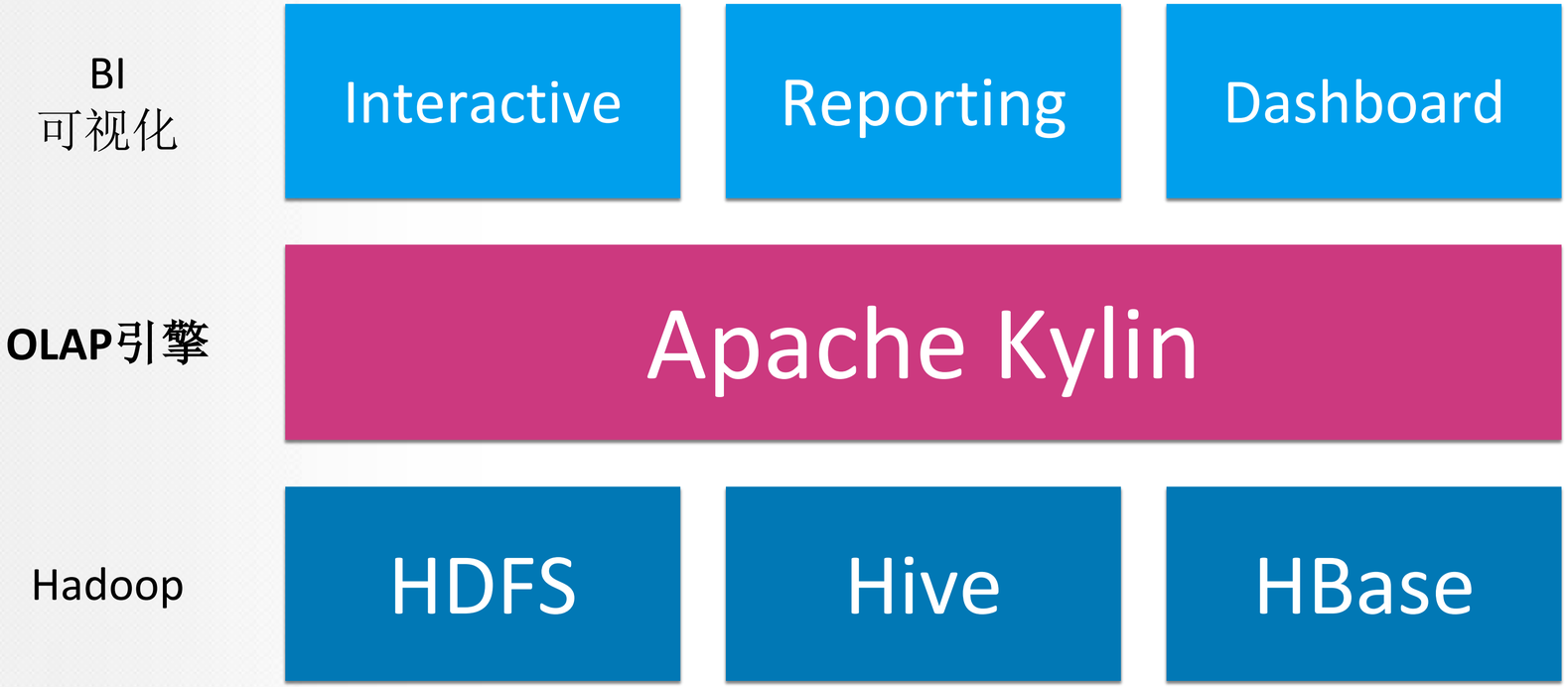
Kyligence

Apache Kylin 2.0 技术解密之Spark Cubing

马洪宾 | ma@kyligence.io

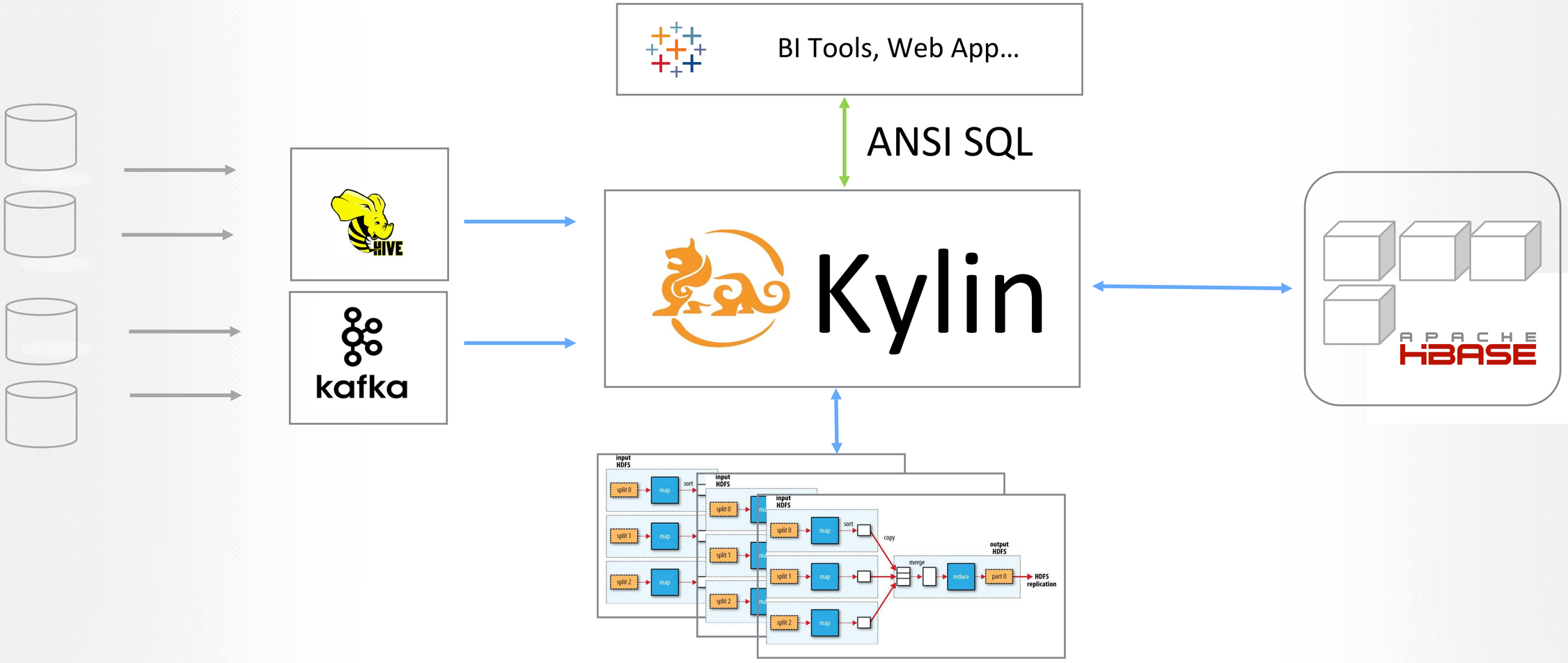
PMC member of Apache Kylin
Kyligence Inc. 技术合伙人 & 高级架构师

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 in the Zoo



Offline Cubing

Apache Kylin全球案例

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.

hadoop → cloudera Hortonworks

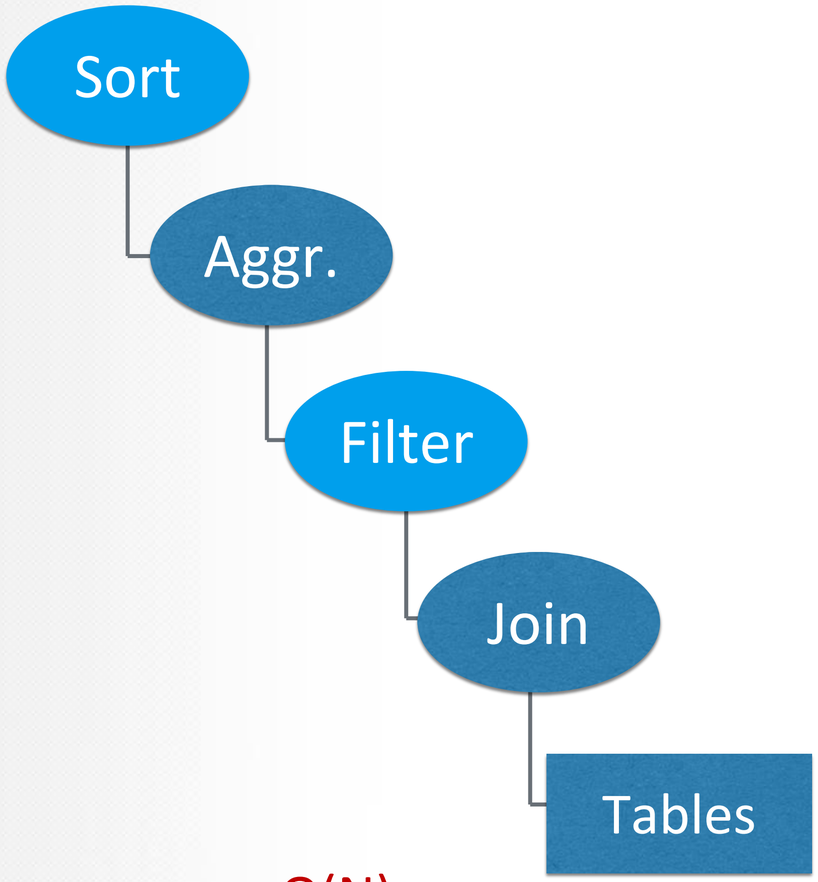
Apache Spark → databricks

Apache Kafka → confluent

Apache Kylin → Kyligence

Kyligence

Kylin为什么快

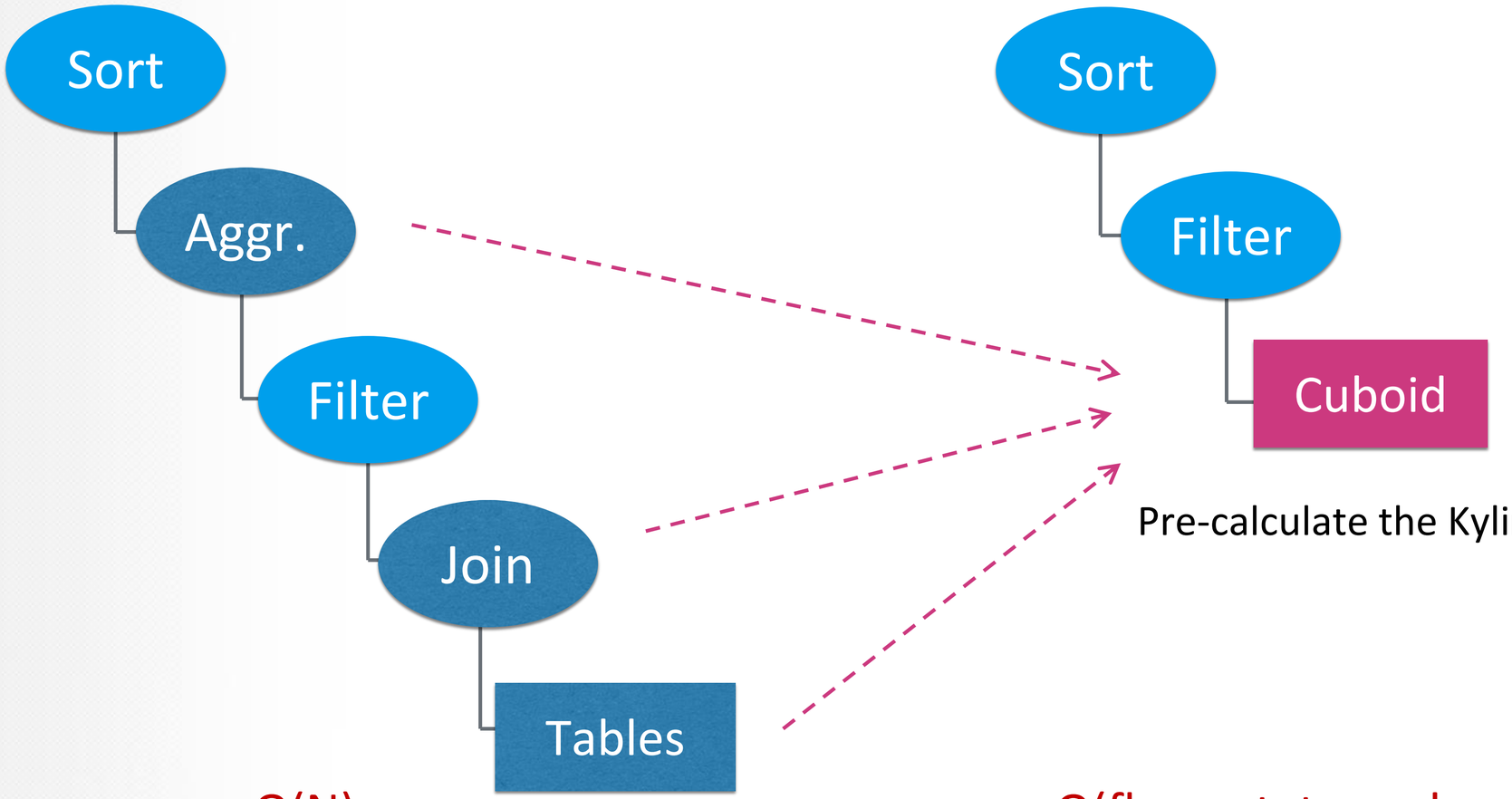


A sample query:

Report revenue by “returnflag” and “orderstatus”

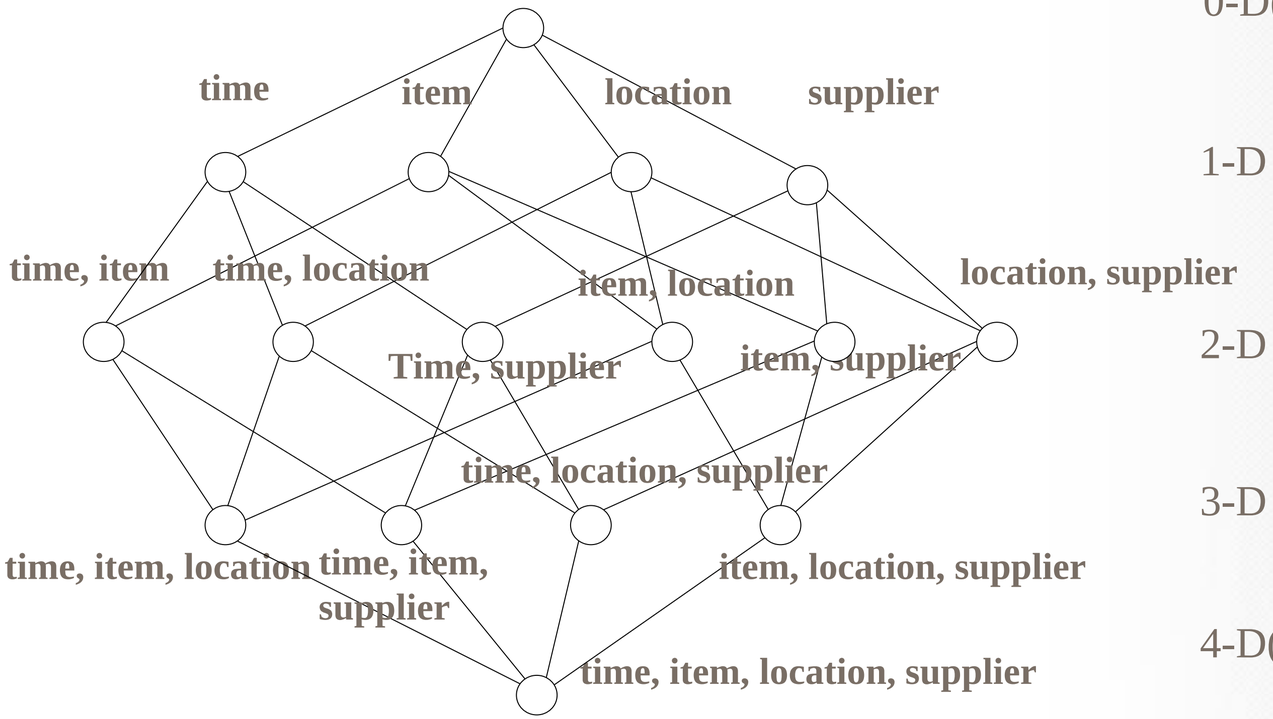
```
select
   l_returnflag,
   o_orderstatus,
   sum(l_quantity) as sum_qty,
   sum(l_extendedprice) as sum_base_price
   …
from
   v_lineitem
   inner join v_orders on l_orderkey = o_orderkey
where
   l_shipdate <= '1998-09-16'
group by
   l_returnflag,
   o_orderstatus
order by
   l_returnflag,
  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

Kylin为什么快



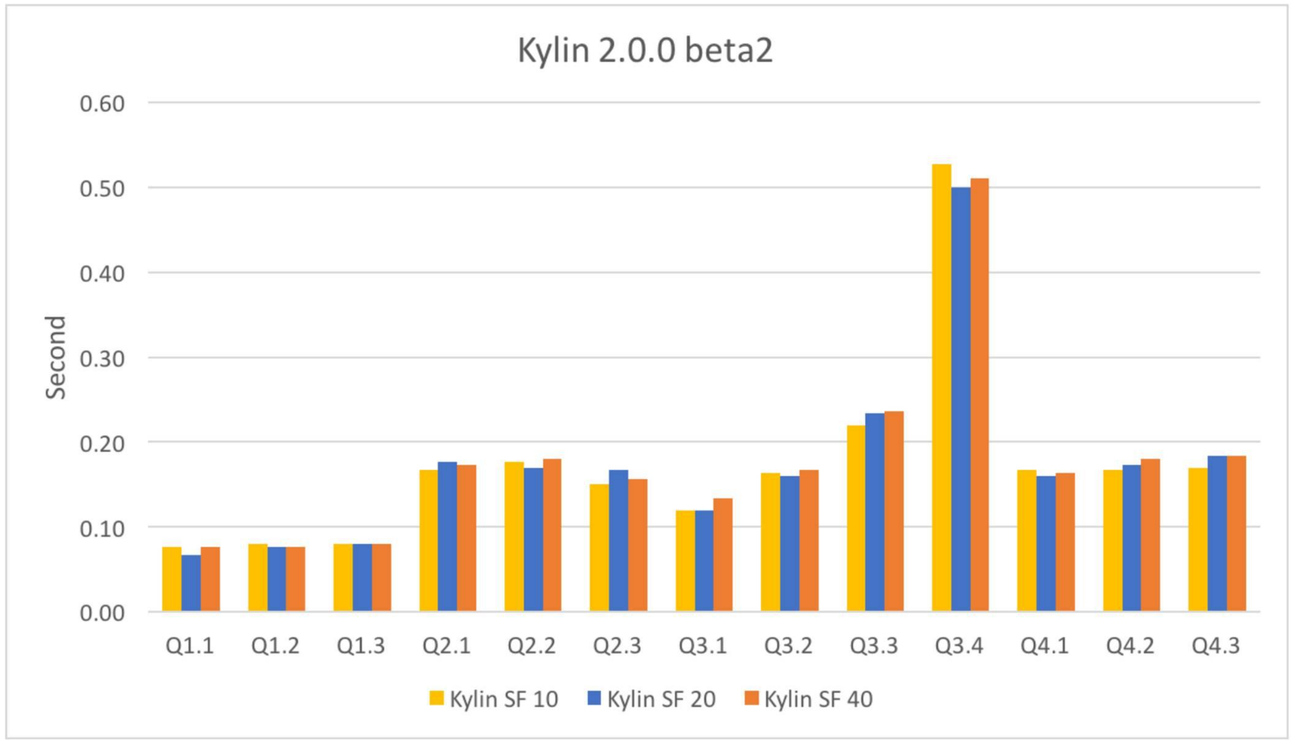
Kylin 关键在于预计算

- 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

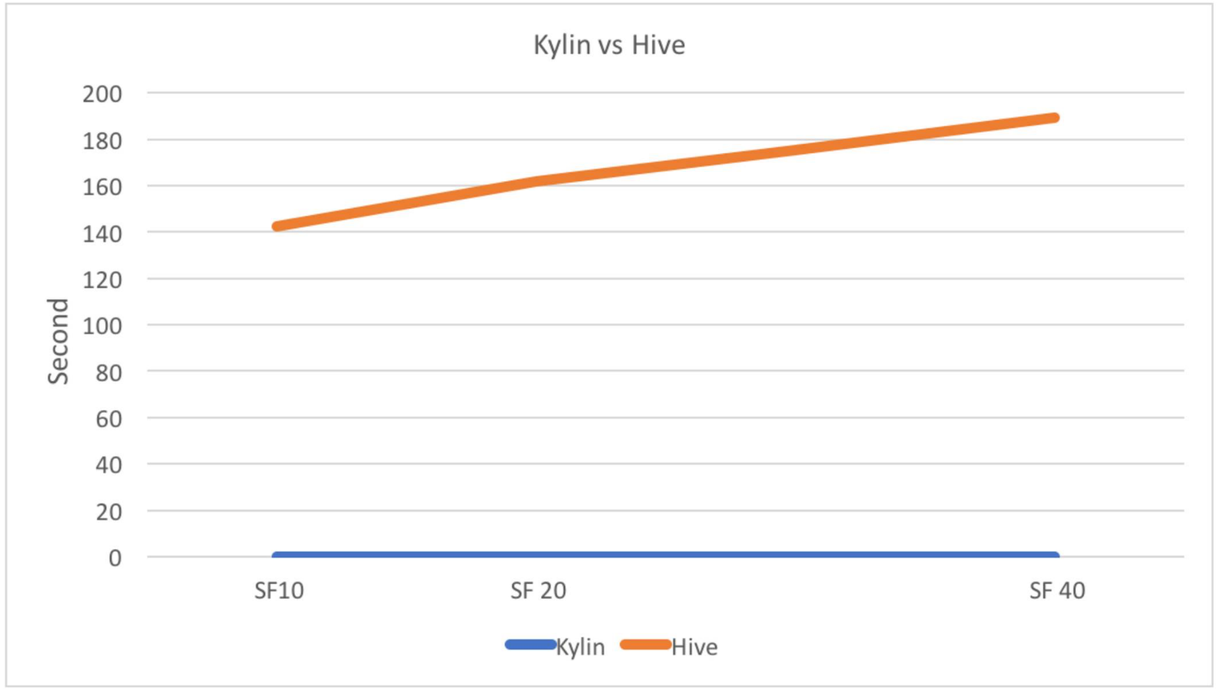


亚线性复杂度 => O(1)

Result of Kylin 2.0.0 beta2



Result of Kylin vs. Hive



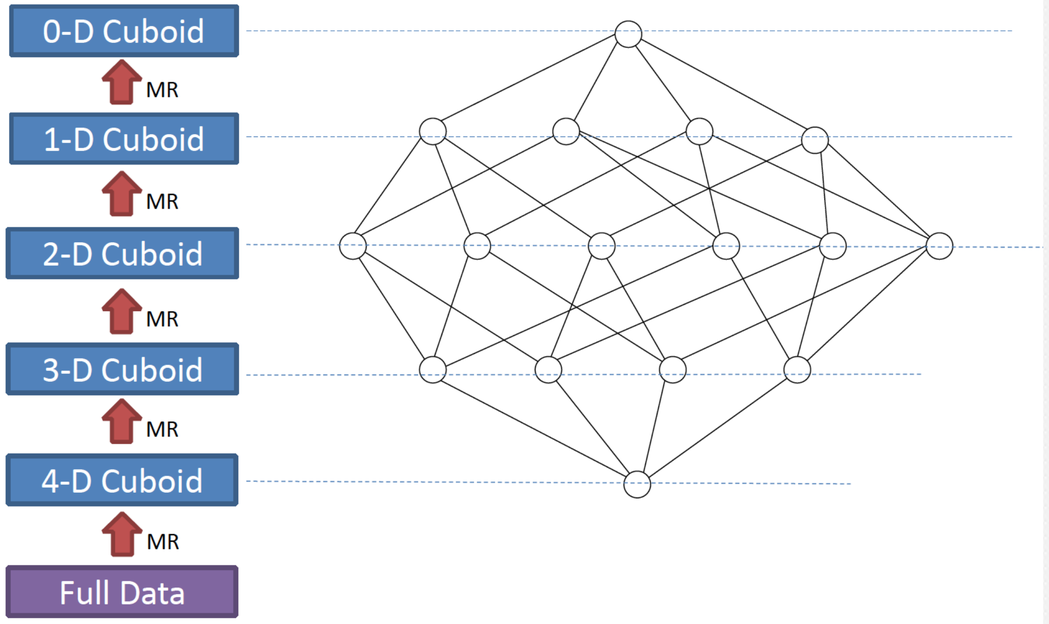
Spark Cubing

目标：减少一半的构建时间

MR Layered Cubing

标准的构建算法：MR Layered Cubing

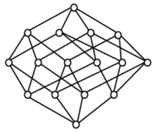
- 启动多轮MR任务
- 将大型shuffle切分到多个stage
- 稳定，但是在构建时间上并不是最优的

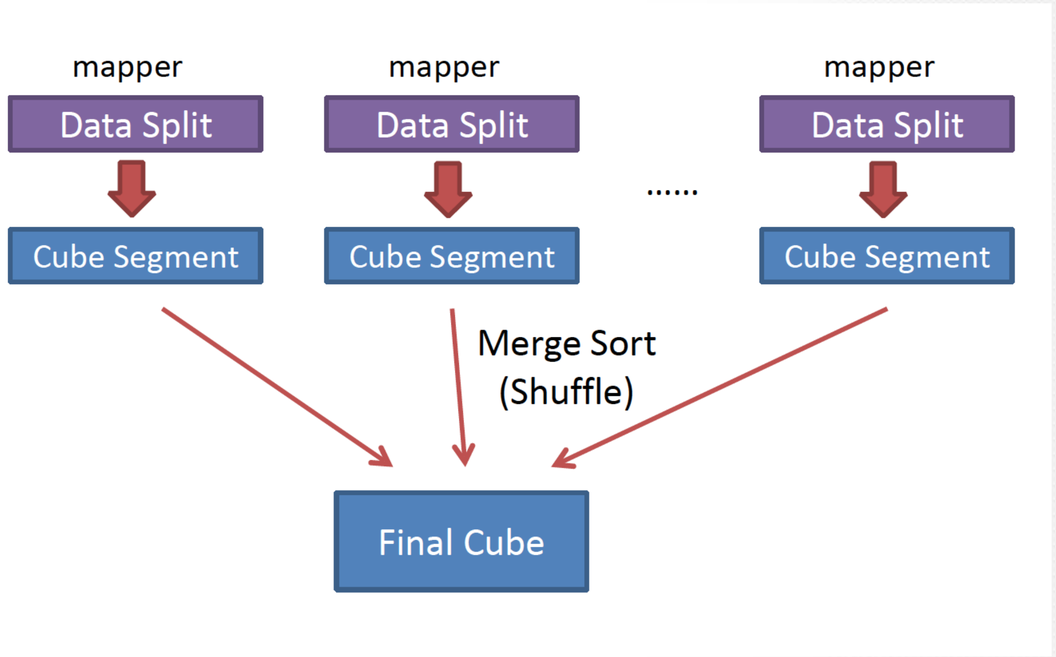


MR In-memory Cubing

MR In-memory Cubing是对MR 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





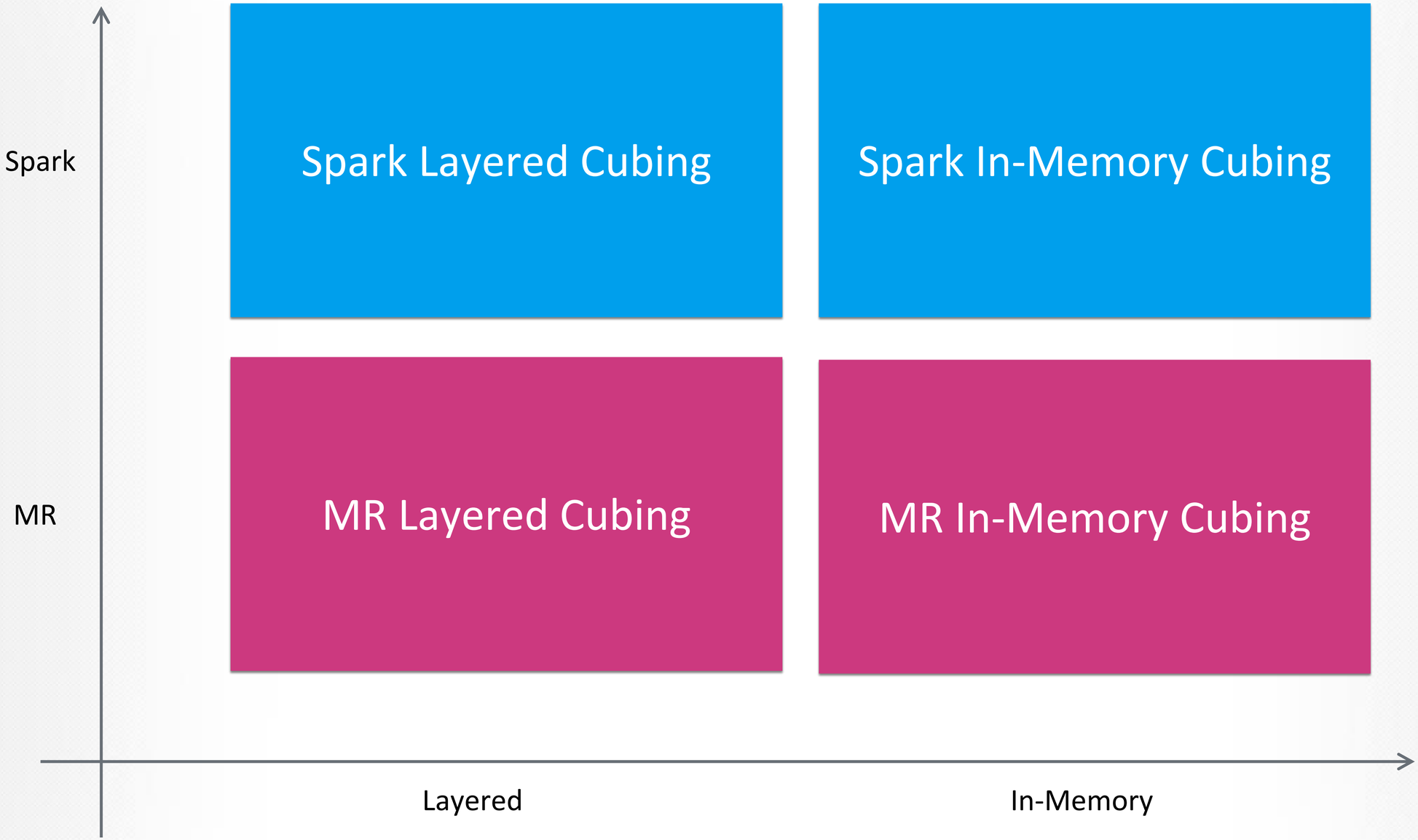
Cubing with MR 总结

比较稳定

MR Layered Cubing在某些场景下性能都有待提高

MR In-mem Cubing适用场景有限

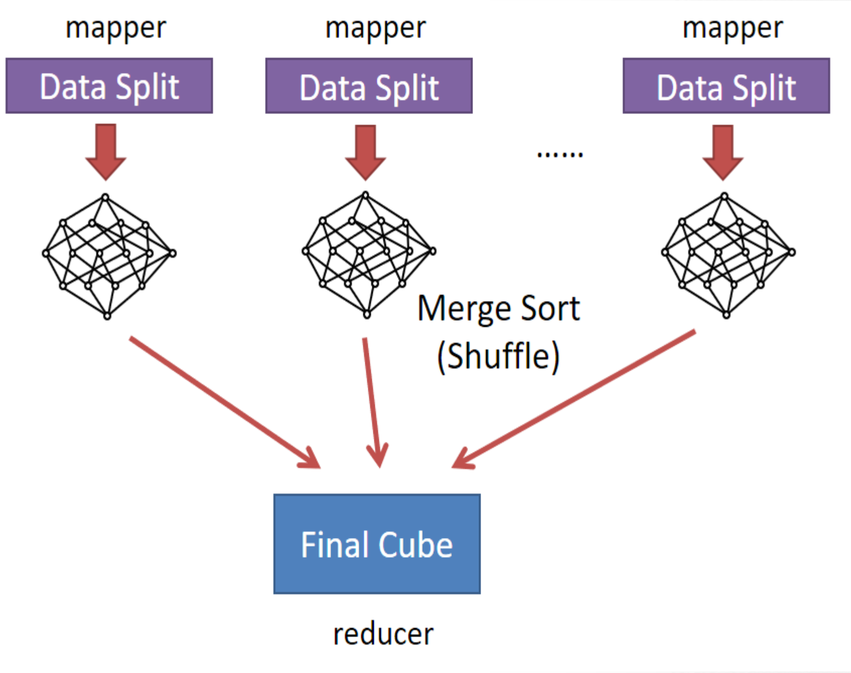
社区迫不及待地想要尝试使用其他技术来加速cubing


Spark
MR
Spark Layered Cubing
Spark In-Memory Cubing
MR Layered Cubing
MR In-Memory Cubing
Layered
In-Memory

Spark In-Memory Cubing:一次失败的尝试

Kylin 1.5曾经尝试使用过Spark In-Memory Cubing，但是从未正式发布

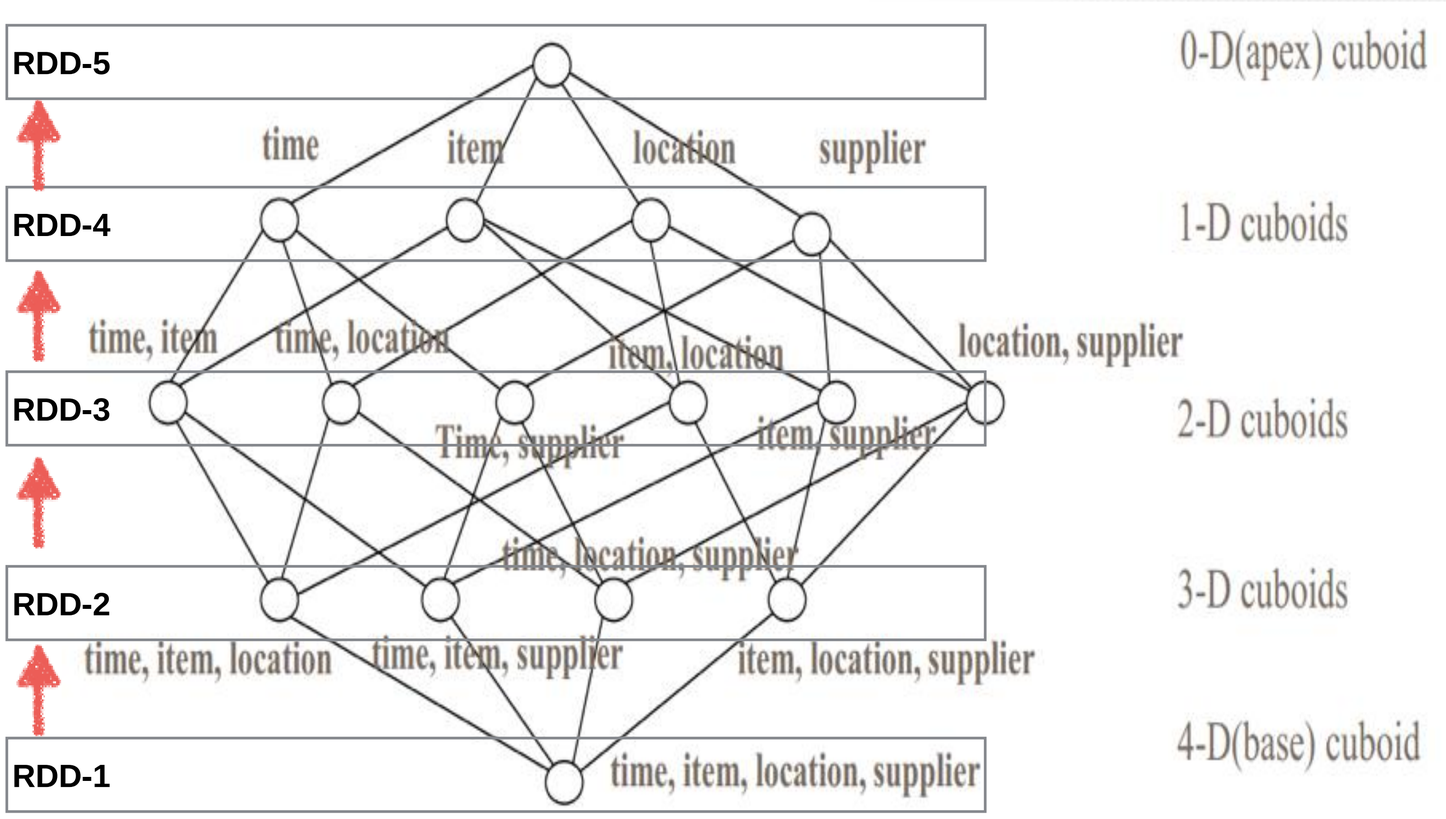
- 它只是简单地将In-memory Cubing移植到Spark上
- 使用一轮RDD转换计算整个cube
- 并未观察到明显改进
- Spark 计算方式与MR并无明显区别



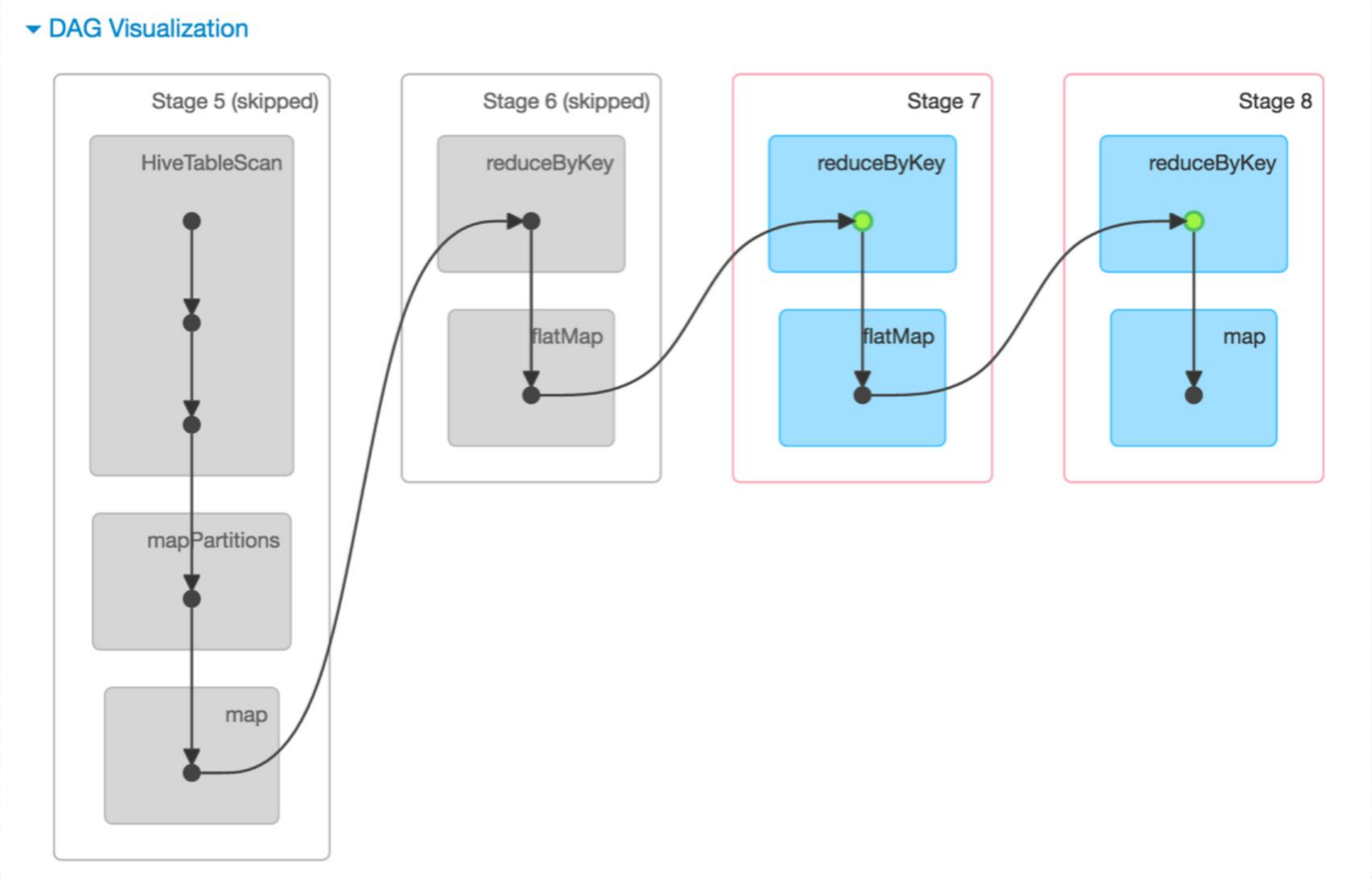
Spark Layered Cubing in 2.0

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



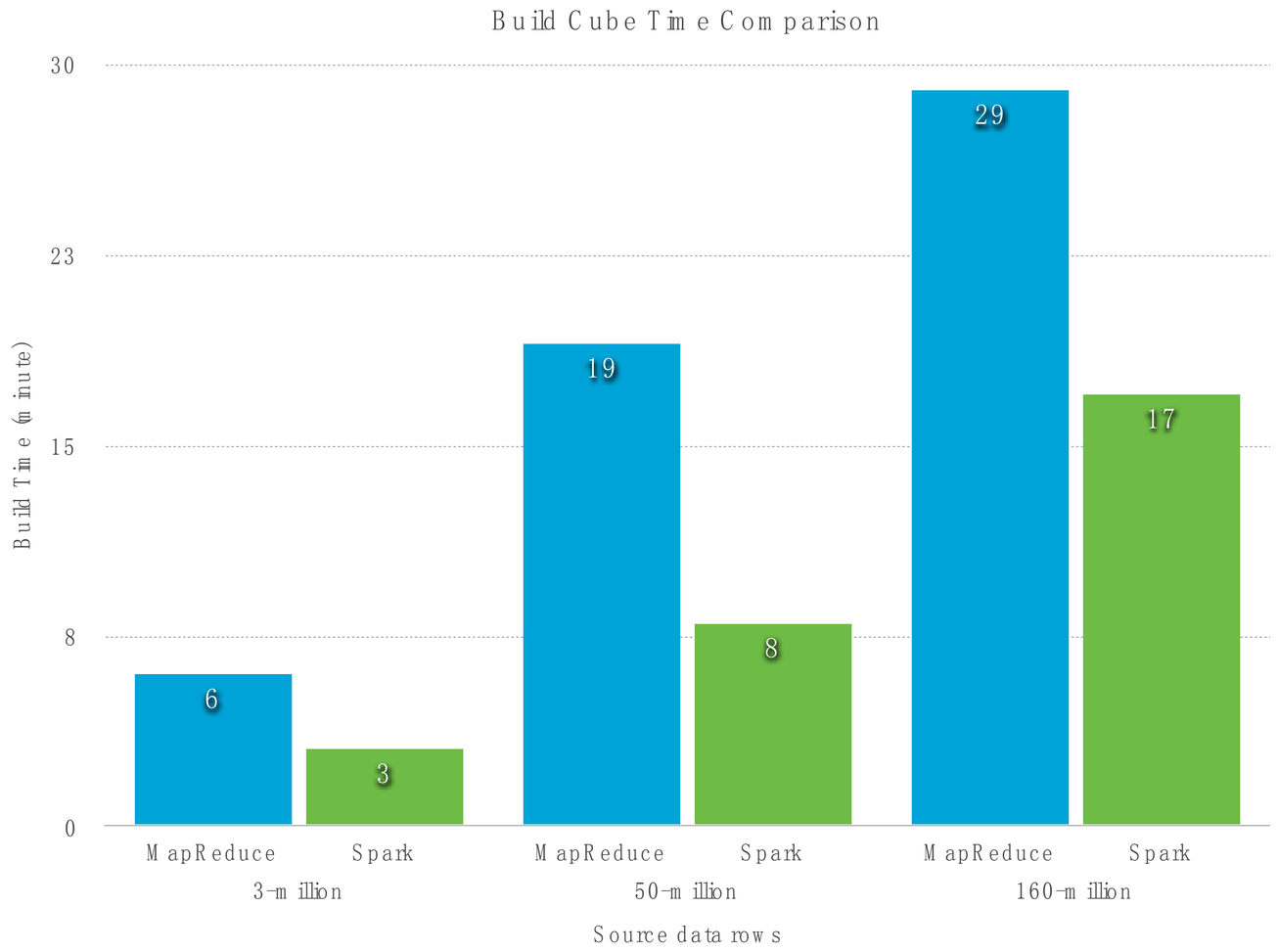
计算第三层cuboid的DAG



Performance Test

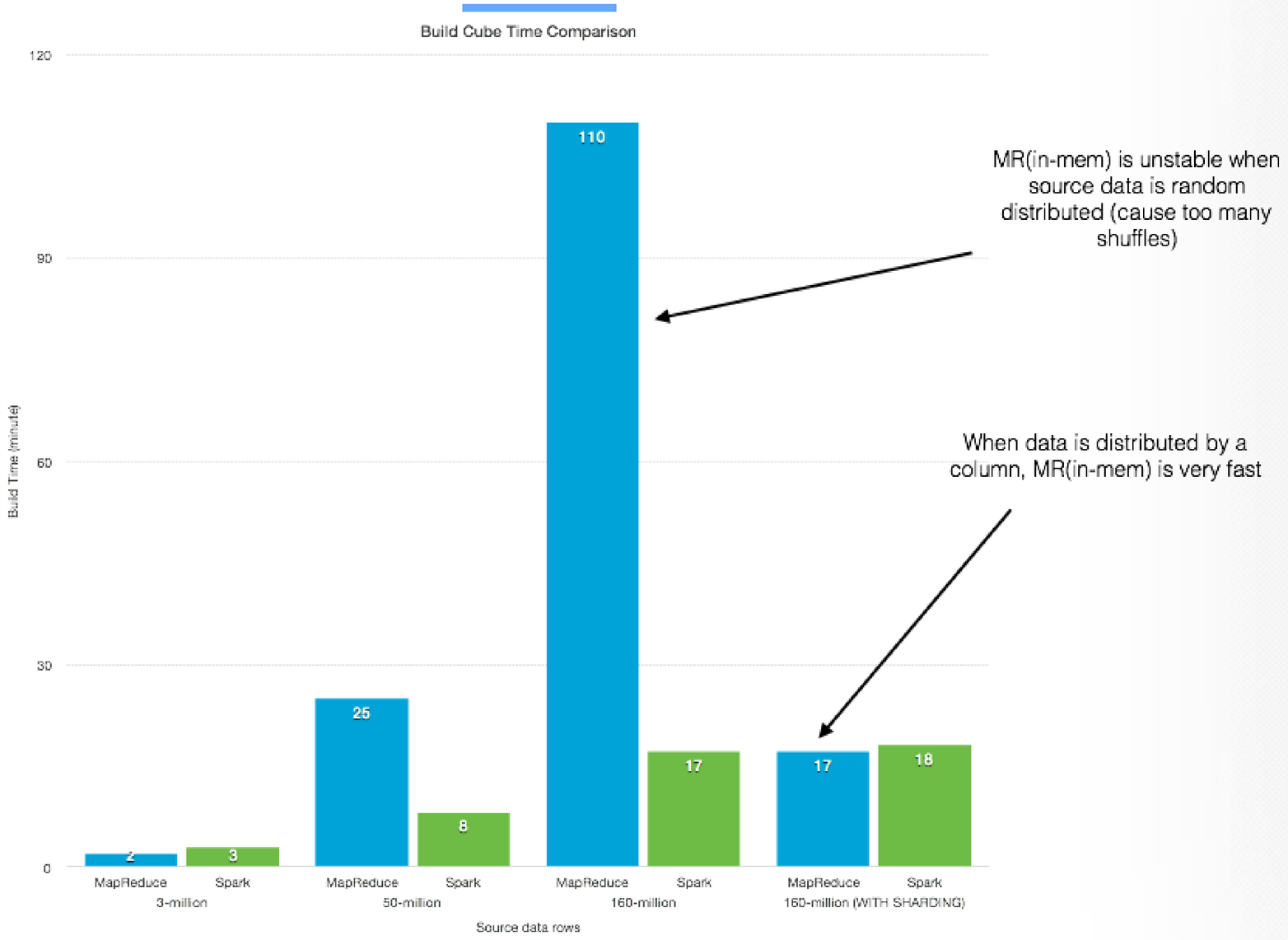
- Environment
  - 4 nodes Hadoop cluster; each node has 28 GB RAM and 12 cores
  - YRAN has 48GB RAM and 30 cores in total
  - CDH 5.8, Apache Kylin 2.0 beta
- Spark
  - Spark 1.6.3 on YARN
  - 6 executors, each has 4 cores, 4GB +1GB (overhead) memory
- Test Data
  - Airline data, total160 million rows
  - 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 Layered Cubing vs. MR Layered Cubing



70% to 130% improvement on Spark

Spark Layered Cubing vs. MR In-memory Cubing



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 Layered Cubing
- When source data is sharded, Spark Layered Cubing still keeps close performance with MR In-Memory Cubing
- There’s still room for improvement

总结

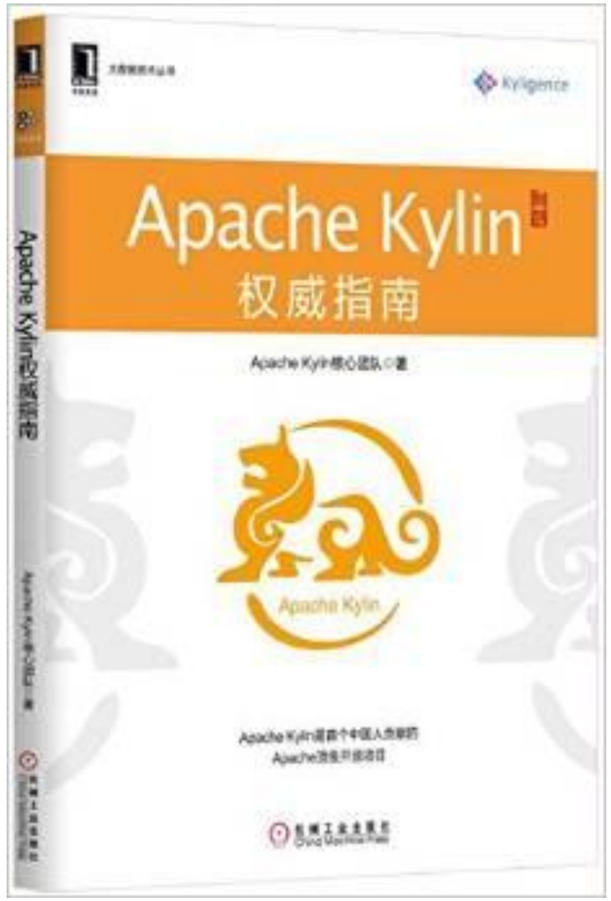
Apache Kylin 2.0

- 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

感谢聆听！



Only Today!