QCon
全球软件开发大会

openLooKeng架构解析和性能优化实践

高级开发工程师 / 罗旦

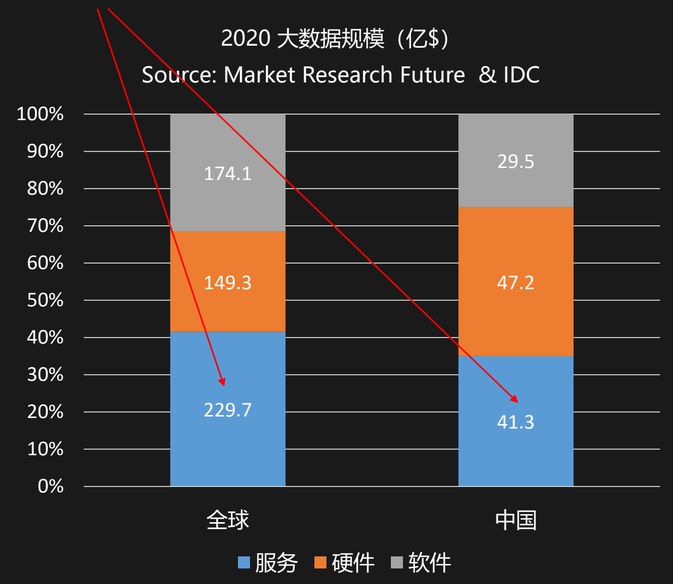
内容

- 大数据分析现状和问题
  - 大数据行业背景
  - 大数据查询面临的挑战
  - openLooKeng聚集的应用场景
- openLooKeng架构介绍
  - 整体系统架构
  - 关键特性解析
- openLooKeng即席查询性能优化实践
  - 即席查询面临的挑战
  - 性能优化关键技术

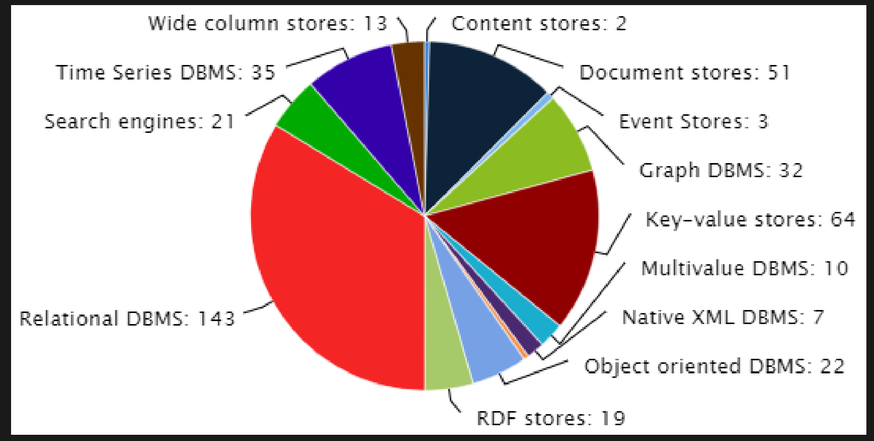
大数据市场现状 – 野蛮生长、使用复杂、格局零散

- 大数据平台市场：53B，格局零散、呼唤整合
  - ✓ 成长迅速占比最大的Splunk只有11%，新进入者空间巨大
  - ✓ 老牌数据库厂商占比都不到10%，Others占50%以上市场

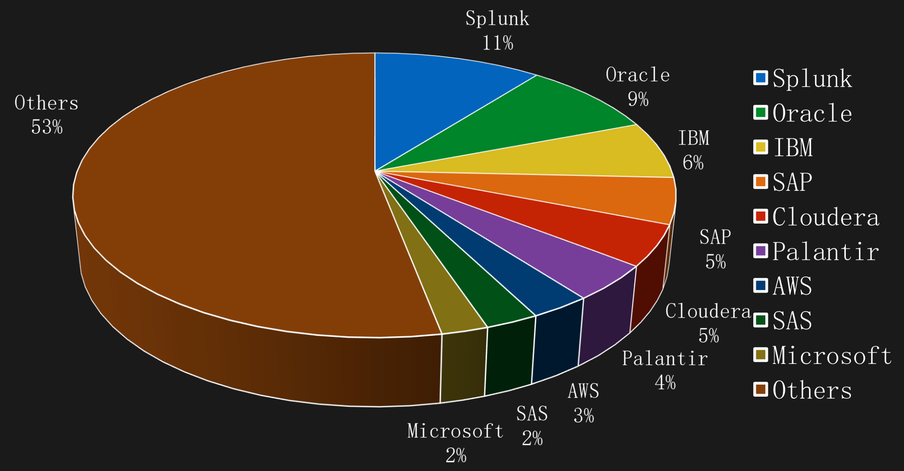
- 极高的服务份额成为大数据项目的一大障碍



Number of systems per category

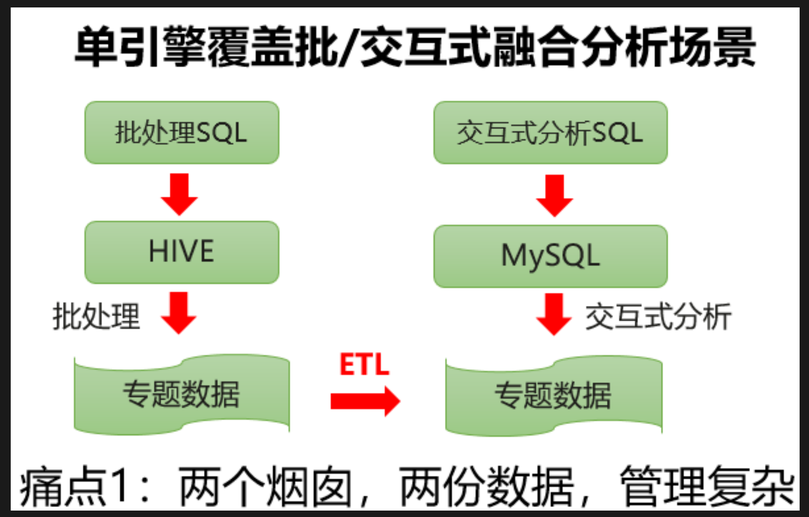


大数据软件提供商市场占比(2017)

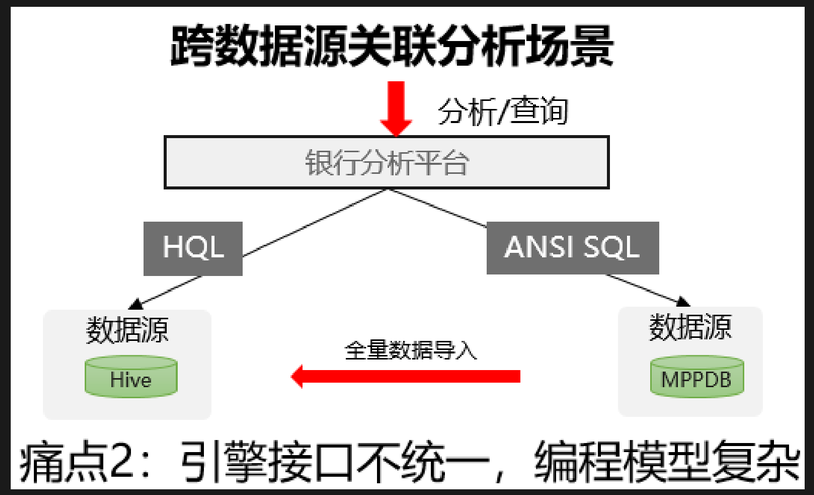


Source:
Market Research Future：Global Big Data Market Research Report: Forecast to 2023
IDC：全球半年度大数据支出指南，2018H2

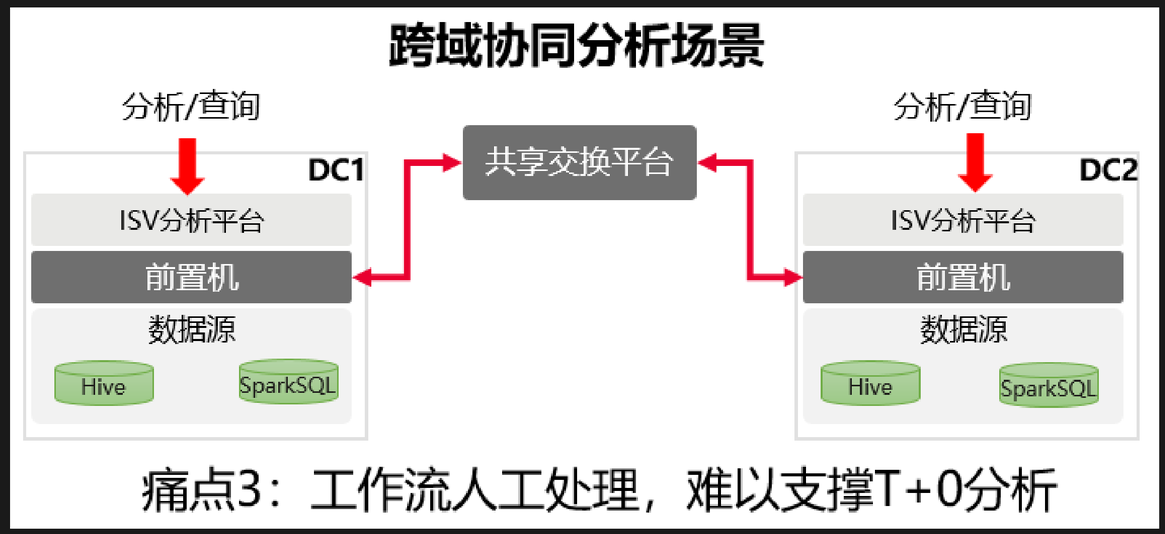
大数据查询面临的挑战



痛点1：两个烟囱，两份数据，管理复杂



痛点2：引擎接口不统一，编程模型复杂



痛点3：工作流人工处理，难以支撑T+0分析

三大需求：

- 批流融合分析
- 跨数据源关联查询
- 跨域跨DC协同分析

openLooKeng统一高效的数据虚拟化引擎，让大数据变简单

openLooKeng

统一数据访问接口

SQL　ODBC　JDBC　REST

引擎内核

融合分析　跨源索引　动态过滤　算子下推　AA 高可用

统一数据源连接框架

Data Source Connector　Data Center Connector

MySQL　Kafka　Hive　Parquet　.....

PostgreSQL　HBase　CarbonData　ORC

数据中心A　数据中心B　数据中心C

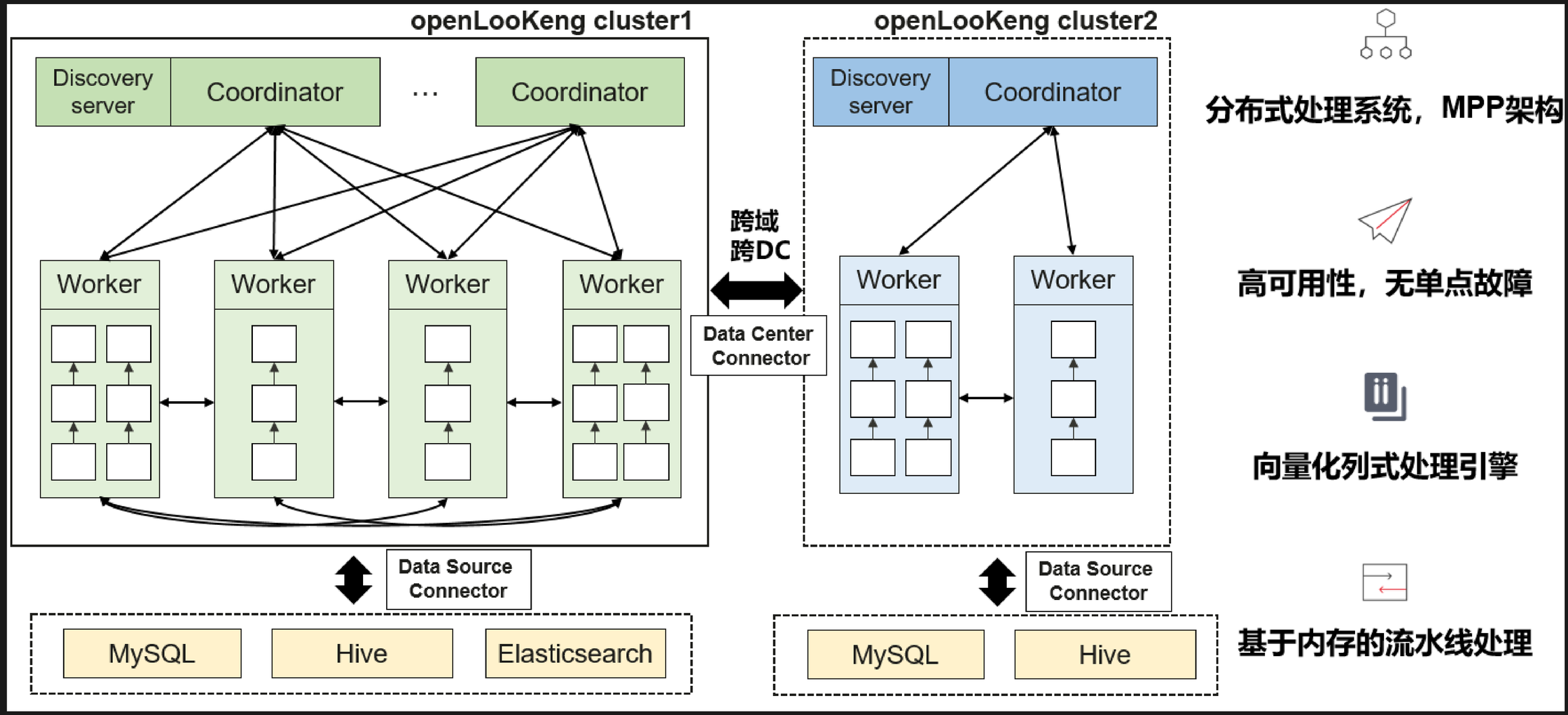
统一入口，化繁为简，单一引擎支持多场景

内核增强，高性能查询

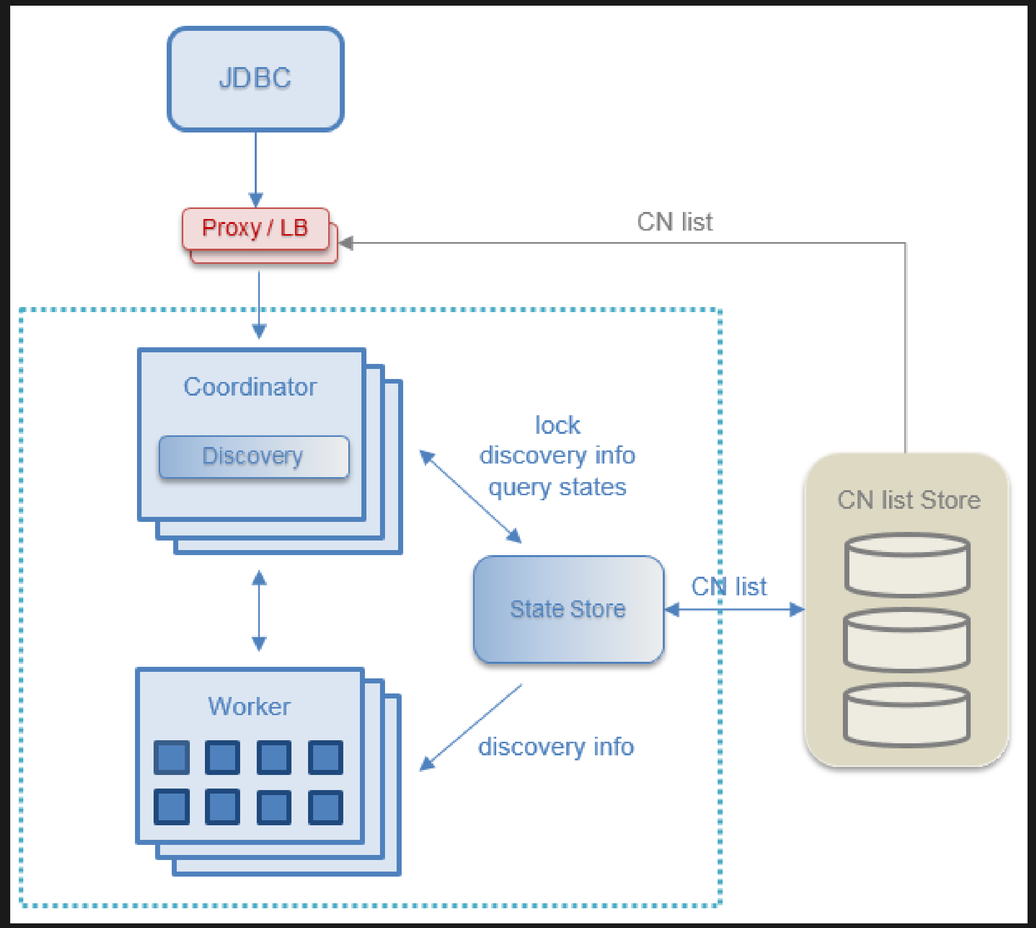
跨源关联分析，数据消费零搬移

跨域协同计算，广域网的部署，局域网的体验

openLooKeng系统架构

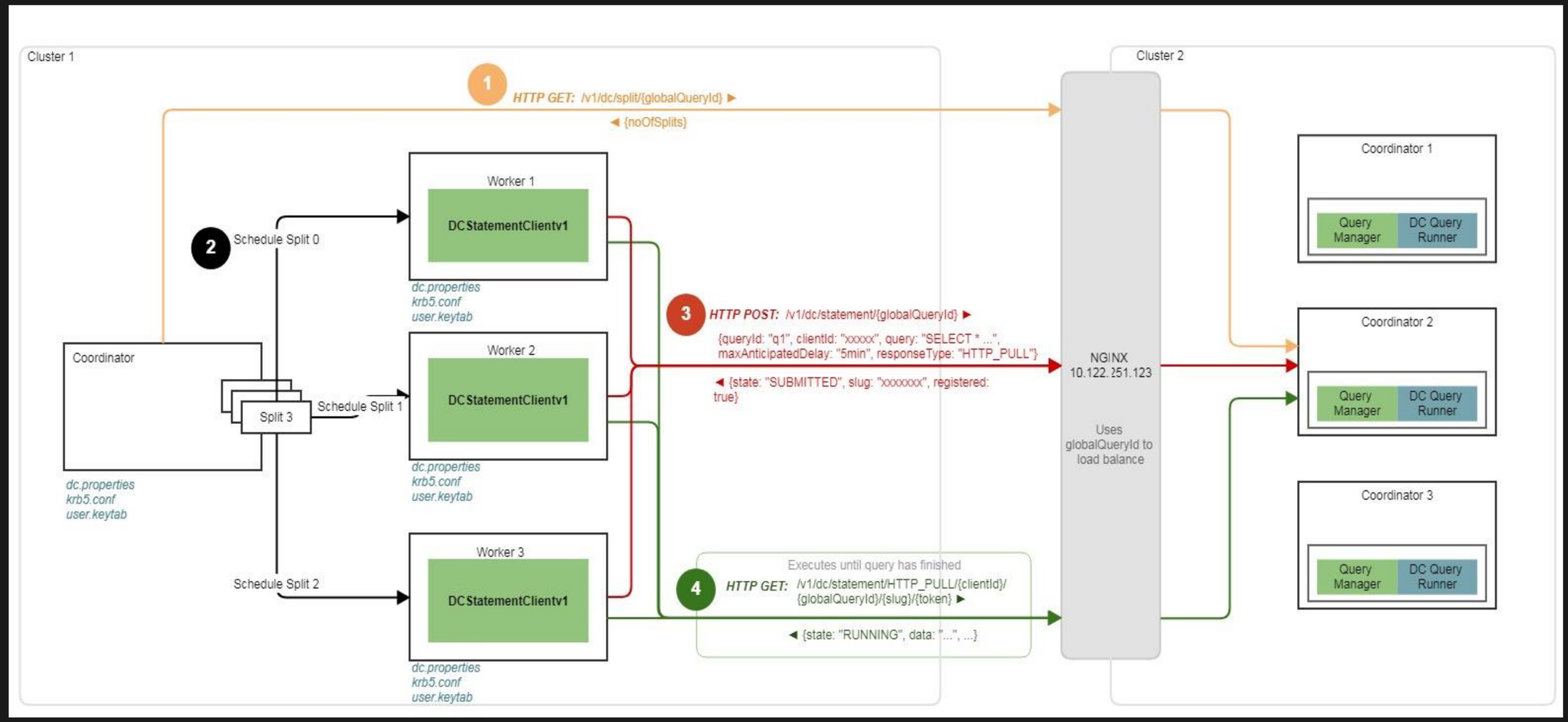


openLooKeng：高可用AA，保证业务连续性



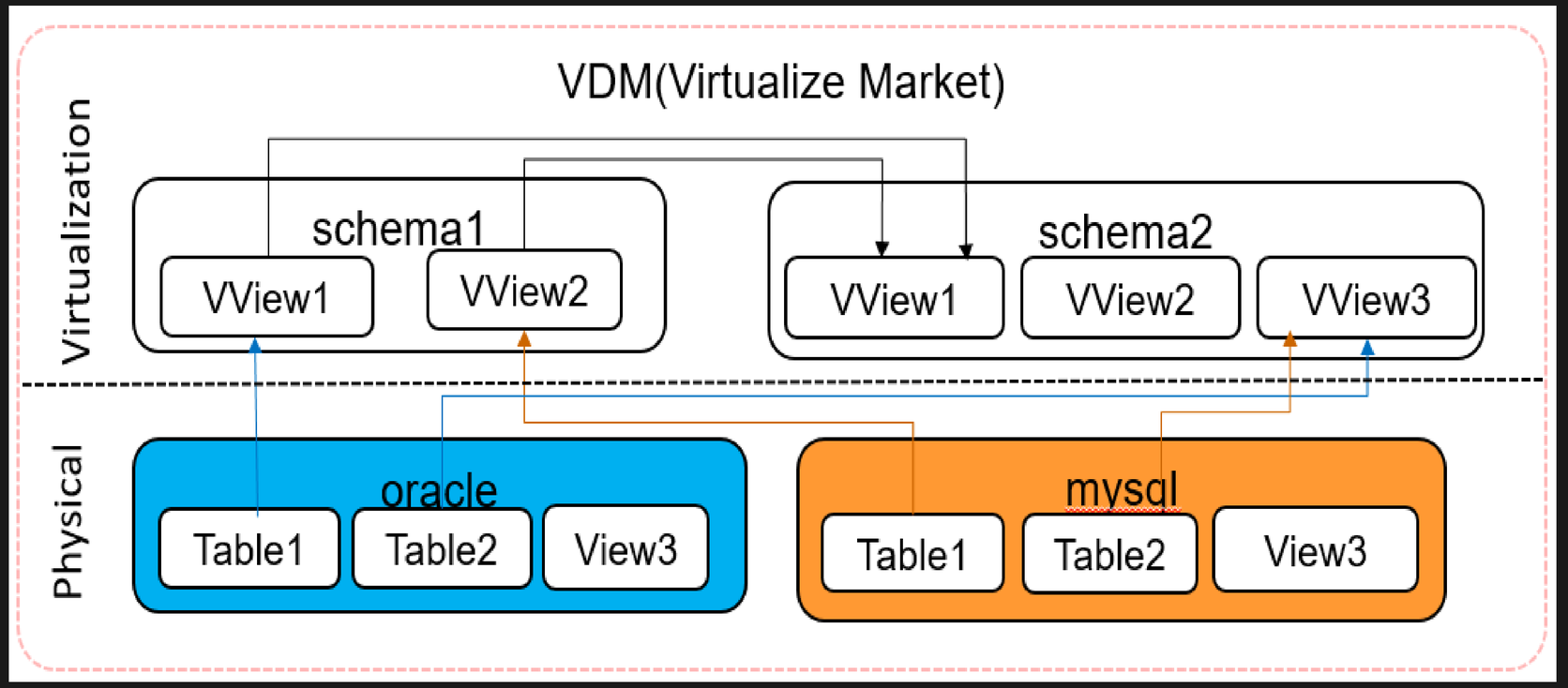
- 多Coordinator**同时运行**并接收客户端的查询提交
- 提供持续的应用**可用性和抗灾能力**，单个Coordinator故障不影响集群的正常运行
- 高并发下，可减轻单Coordinator的压力，提高**吞吐量**
- 结合Nginx等反向代理工具可实现**负载均衡**等高阶特性

openLooKeng: 跨域跨DC协同分析



- 打通异地数据中心数据访问，跨DC数据协同查询**无需依赖数据中转平台**
- 通过**算子下推**与**跨域动态过滤**技术，可获得**广域网部署，局域网的性能体验**

openLooKeng：VDM数据虚拟集市，简化数据开发过程



- 可方便的对底层的数据源、数据表进行管理，通过建立轻量级的**视图**来实现对不同数据源的模式化访问，使得用户不需要每次查询都关心数据的分布以及访问方式

交互式查询特点

随机性
用户SQL不可预测，具有很强的随机性

数据量适中
数据存在data warehouse中

交互式查询

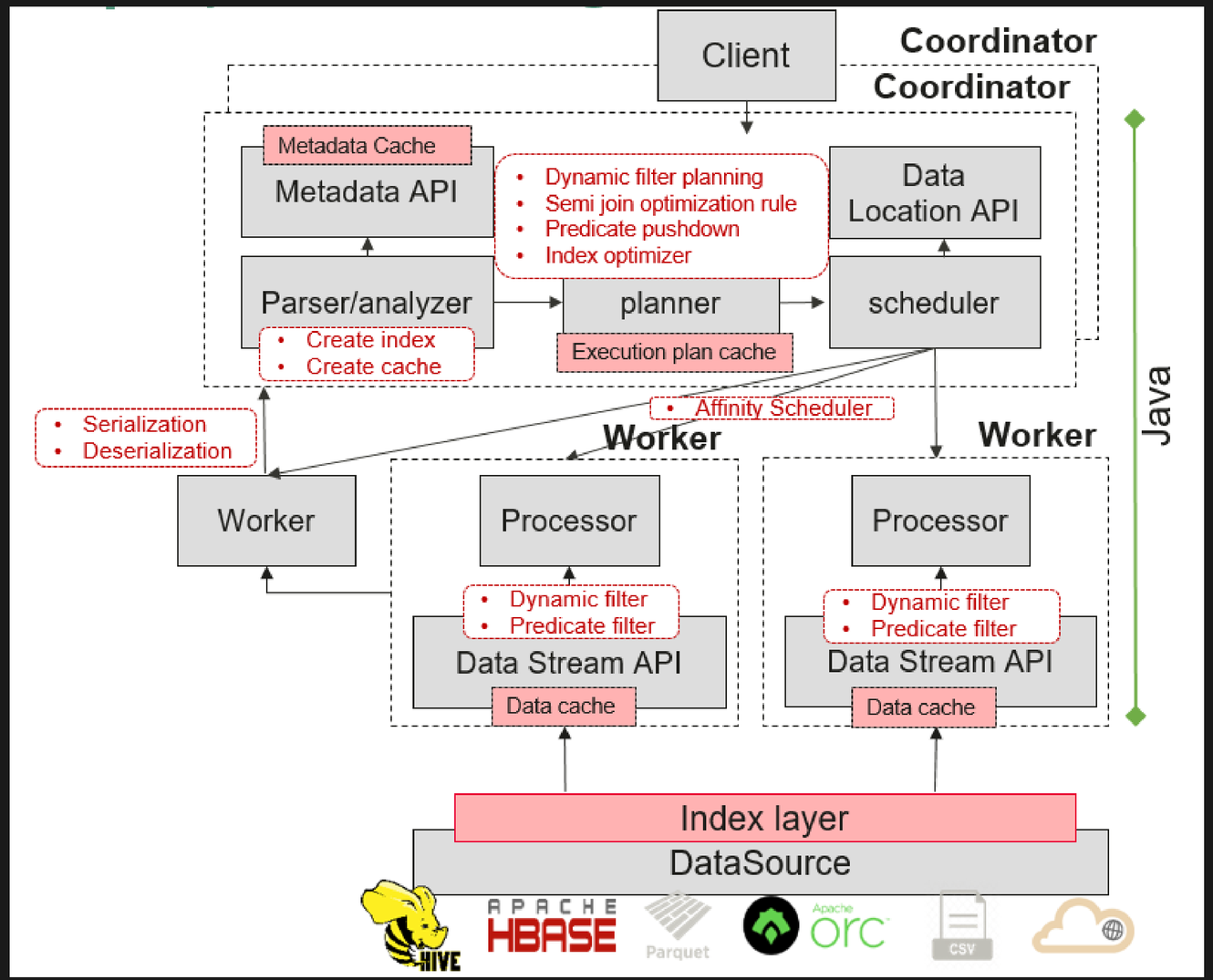
并发支持
一般需要提供50-100并发支持

所需结果集小
通常所需结果集较小，使用limit、取消或者TopN或聚合函数

端到端时间敏感
秒级/分钟级返回，查询涉及资源不敏感

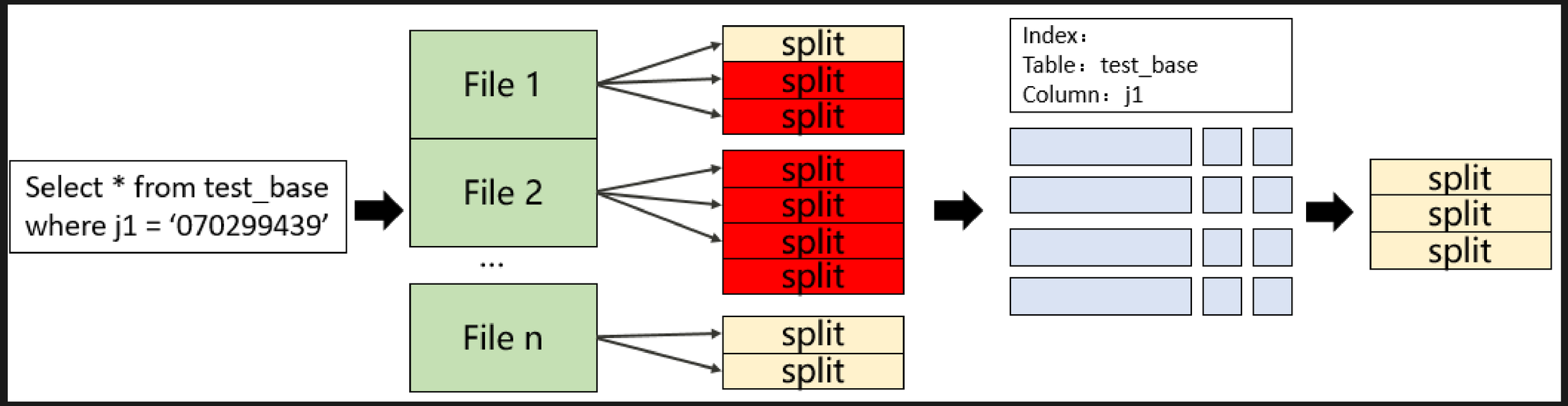
跨源跨域
需要跨DC或者跨源支持

openLooKeng性能优化关键技术



- **数据源侧，更适应openLooKeng**
  - 分桶/分区
  - 小文件合并
  - 查询字段排序
- **引擎层，增强交互式查询能力**
  - 缓存加速：
    - 执行计划缓存
    - 元数据缓存
    - 增量列式缓存
  - 优化器：
    - 谓词下推
    - **动态过滤**
    - RBO&CBO
  - 自适应调度器
- **额外层，加速交互式查询**
  - **Heuristic index layer**
    (bitmap/bloomfilter/min-max)
  - Data cache layer
  - 序列化&反序列化

Heuristic index- 稀疏索引



Bloom filter索引，确定每个split是否包含要搜索的值，并只对可能包含该值的split进行读操作

- 可以快速判断一个集合中有无某个值
- 需要预先通过create index进行索引创建
- 通过在coordinator侧过滤，减少不必要的split生成与处理

客户场景性能测试结果

SQL	数据范围	并发数	最快响应	总时长	查询失败
Sql	1天	10	146ms	206ms	0
		20	158ms	364ms	0
		50	186ms	1.2s	0
		100	145ms	2.4s	0
	10天	10	191ms	309ms	0
		20	338ms	583ms	0
		50	1.1s	2.7s	0
		100	2.4s	5.6s	0
	30天	10	430ms	517ms	0
		20	930ms	1.1s	0
		50	1.9s	2.9s	0
		100	4.4s	9.7s	0

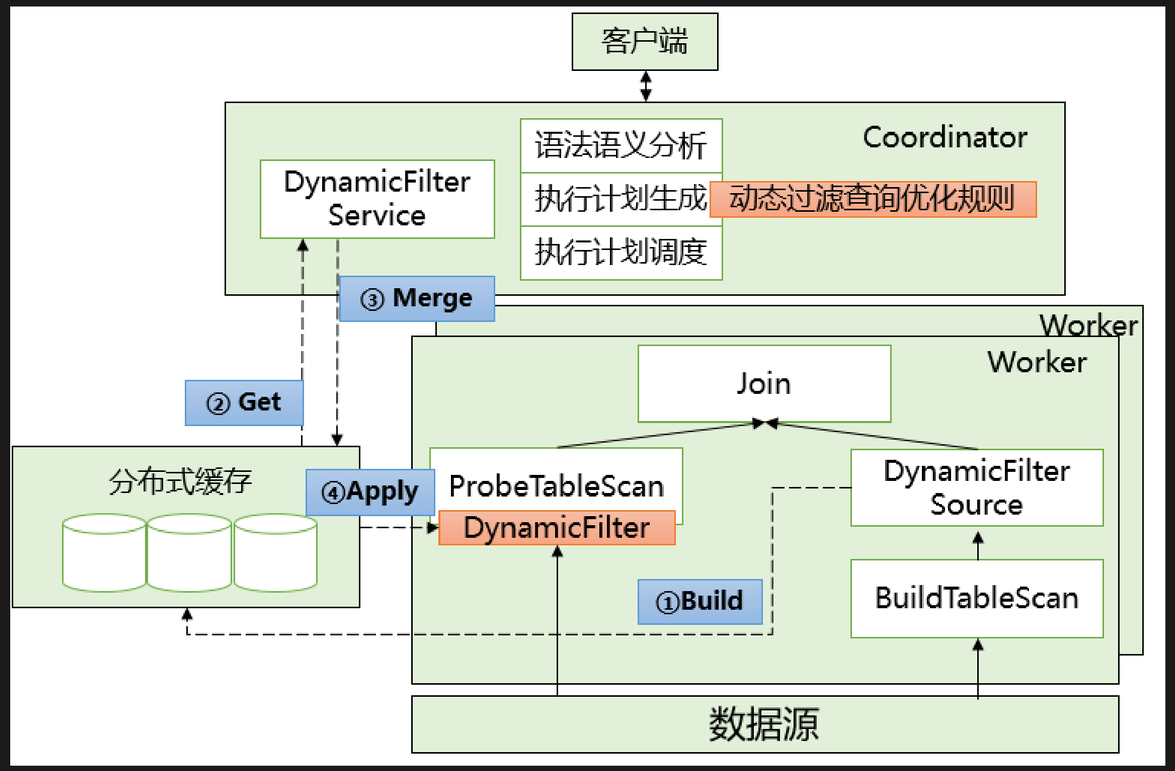
场景分析

- 单表数据量很大，只有天分区，测试数据包含30天
- 谓词包含OR以及AND
- 单表点查询，无join

结论及后续优化

- 结论
  - 100并发20天查询性能满足需求
  - 50并发30天查询性能满足用户需求
  - 100并发30天查询性能存在一定差距
- 后续优化
  - 索引OR支持，并且下推OR操作
  - 聚合场景的Aggregation Stage Cache和StarTree Index

动态过滤

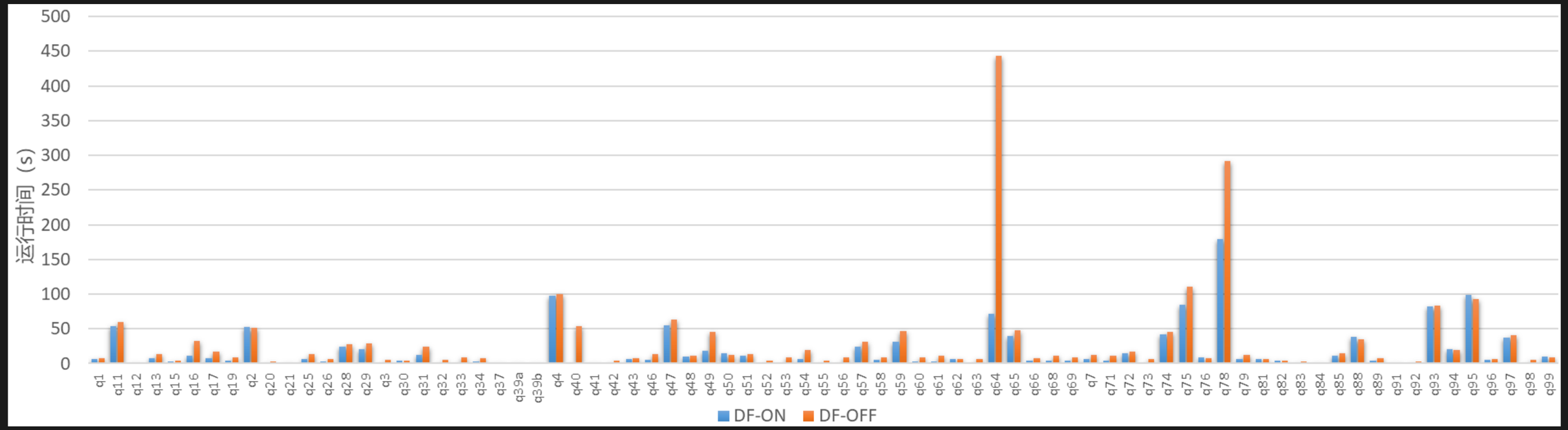


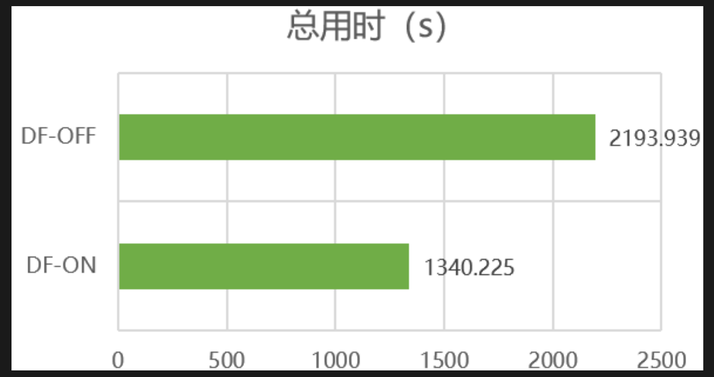
Dynamic Filtering

- 添加DynamicFIlterSource算子，搜集build侧数据
- 依赖分布式缓存进行DF的处理
- 适用于inner join & right join
- **适用于join选择率较高的场景**

依靠join条件以及build侧表读出的数据，运行时生成动态过滤条件（dynamic filters），应用到probe侧表的table scan阶段，从而减少参与join操作的数据量，有效地减少IO读取与网络传输

动态过滤性能测试

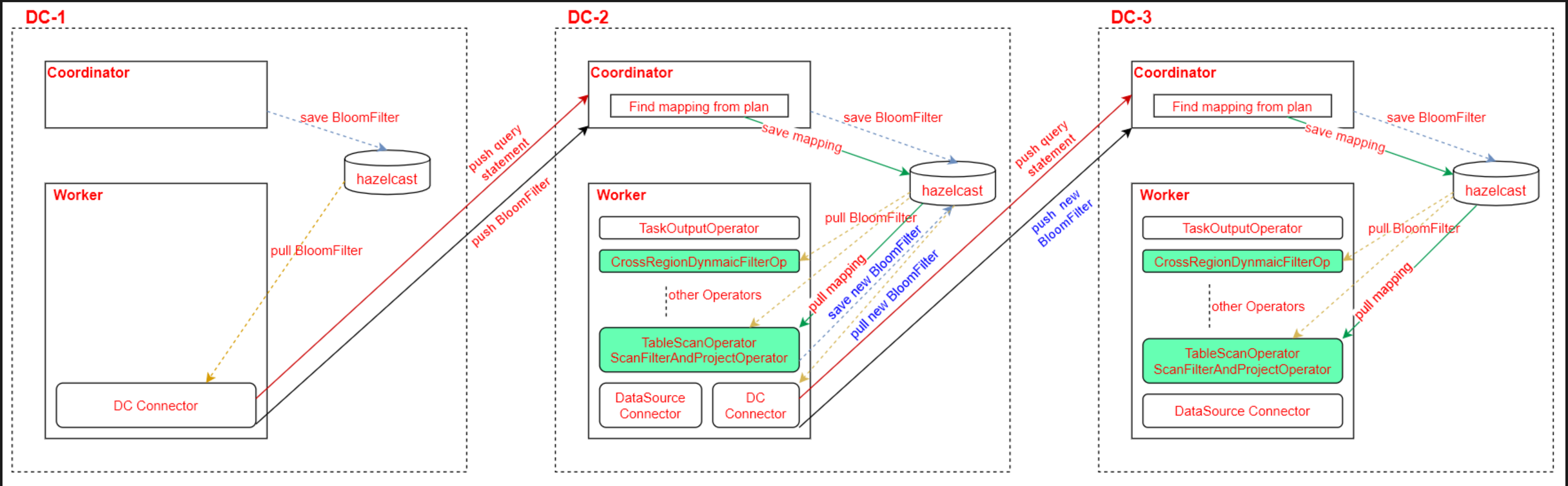




测试背景：
数据集：2TB TPCDS
节点：11计算节点
内存：376GB
CPU：2*Gold 6140 CPU @ 2.30GHz
OS：RedHat 7.3

➢TPC-DS测试用例总用时openLooKeng开启动态过滤，执行时间减少**38.9%**

跨域全局动态过滤



DC-2 Coordinator： 1）将DC-1的BF filter以QueryId为Key存入到hazelcast；2）判断当前query是否存在跨域dynamic filter，存在，设置session中的cross-region-dynamic-filter；3）CN生产执行计划Plan，从Plan中Query的列名到Plan的outputSymbols的映射关系，存入hazelcast；4）判断Plan的TableScanNode是否存在DC table，如存在，则标记，可能存在继续下推BF filter的可能。

DC-2 Worker： 1) CrossRegionDynamicFilterOp从hazelcast中取出BF filter和outputSymbols，判断是否存在过滤列，存在则应用filter对Page进行过滤；2) TableScanOperator应用filter和步骤一类似；3）如果TableScanNode存在DC table，则生成新的BF filter并存入hazelcast，用于发送给下一级DC。

跨域性能测试

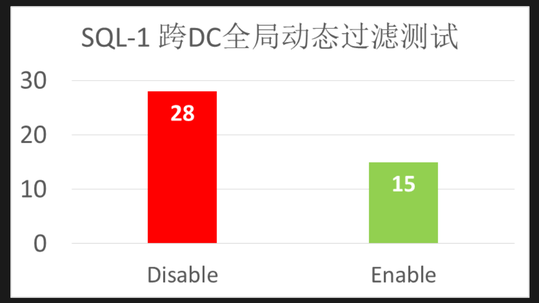
SQL-1：

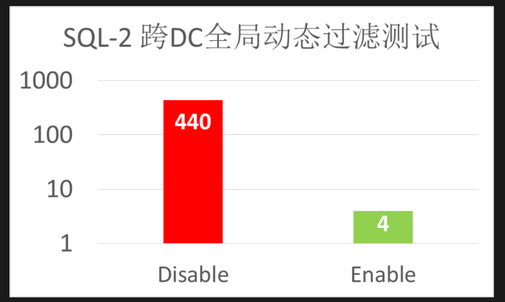
```
Select a.* From
(
        Select collect_place,  first_time, last_time from
        dc.vdm.test.identity_net_stat_10million
        Union
        Select collect_place,  first_time, last_time from
        dc.vdm.test.identity_net_stat_10million
) a
Join hive.test.identity_net_stat_10million b
On a.first_time = b.first_time
Where  b.msidsn =  '13812345678'
```

SQL-2

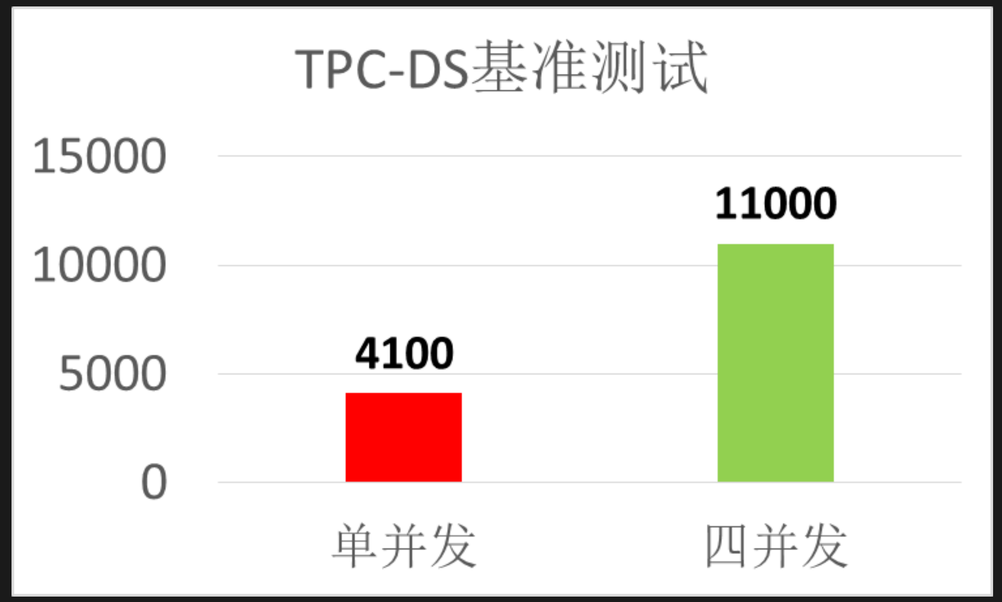
```
Select count(a.first_time) From
(
        Select collect_place,  first_time, last_time From
        dc.vdm.test.identity_net_stat_10million m
        Join
        Select collect_place,  first_time, last_time From
        dc.vdm.test.identity_net_stat_10million n
        On m.collect_place =  n.collect_place
) a
Join hive.test.identity_net_stat_10million b
On a.first_time = b.first_time
Where  b.msidsn =  '13812345678'
```

测试环境：每个DC是一个单节点openLooKeng，内存200GB，CPU：2*Gold 6140 CPU @ 2.30GHz，OS：RedHat 7.3





TPC-DS性能测试



Server	TaiShan 200
CPU型号	2*Kunpeng 920-4826@2.6GHz
核数	96
内存	512GB
磁盘	2*1.6TB NVME
网卡	100GE
OS	openEuler 20.03
openLooKeng集群	1 CN(380GB) + 18 Worker(380GB)
HDFS集群	2 NameNode + 16 DataNode

优化技巧总结:

1：动态过滤：q64，q78，q75，q40，q49

2：Semi join 转 inner join：q95，q14，q93，q58

3：window function + filter 转 Top(N+M)：q67

4：使用group by 消除self join: q95

5：join reorder: q64

总结

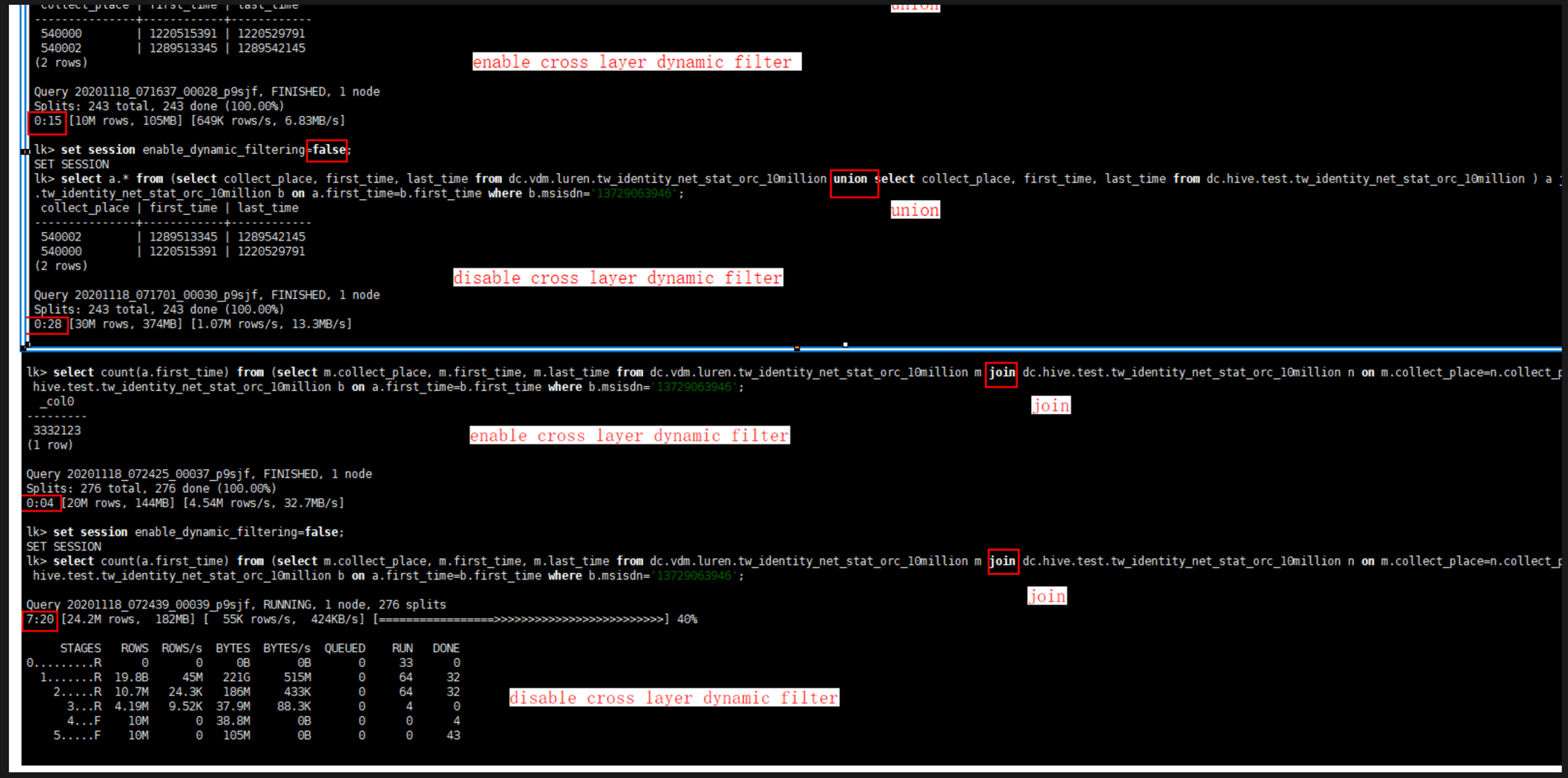
◆统一北向数据访问接口，丰富南向数据源，实现跨数据源数据免搬迁融合分析

◆DC connector提供跨域分析能力，并且通过全局动态过滤，算子下推，压缩断点续传，Coordinator AA等技术来提供高性能和稳定性

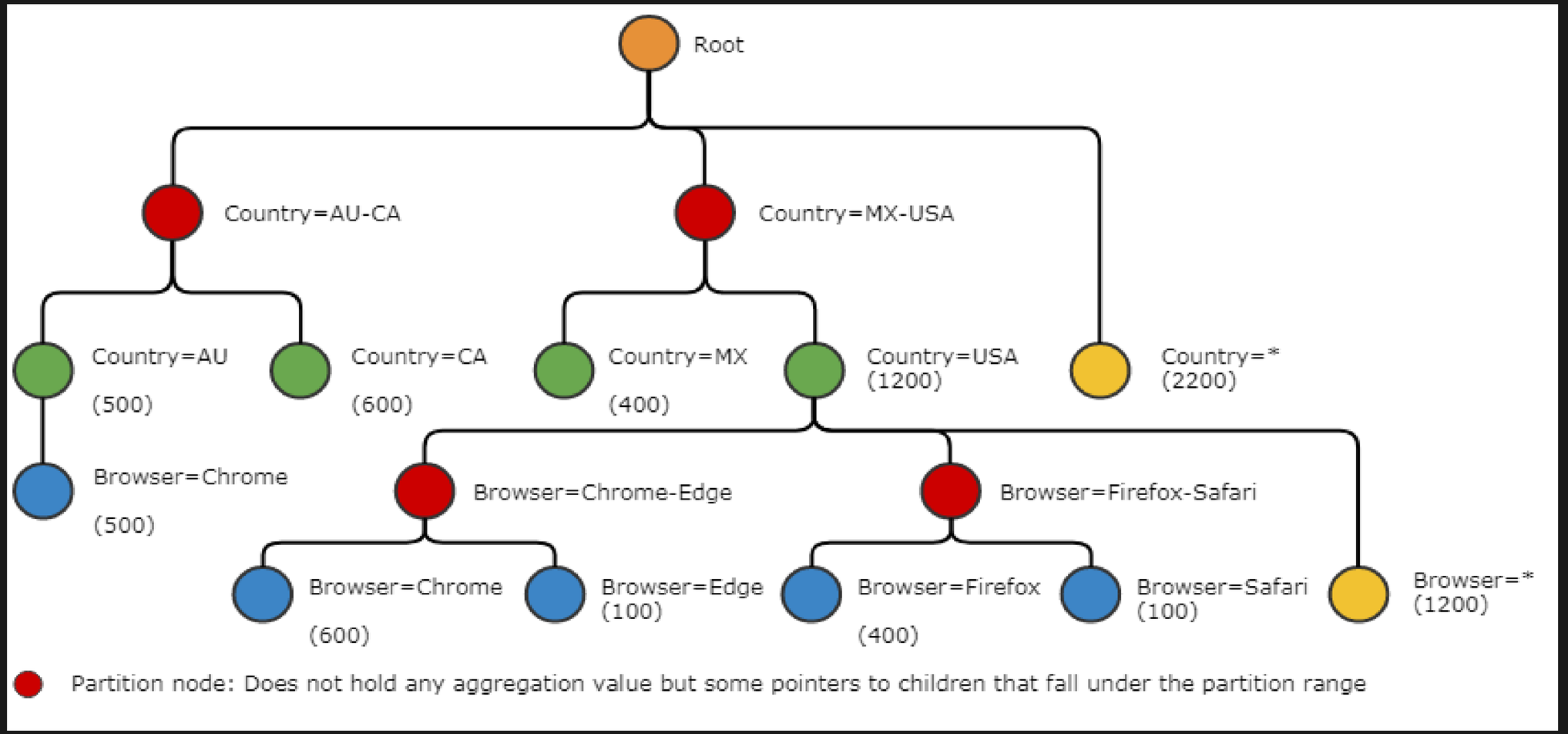
◆通过元数据cache，执行计划优化，索引，动态过滤，算子下推等特性，整体提高openLooKeng的性能


THANKS

备注


union
enable cross layer dynamic filter
union
disable cross layer dynamic filter
join
enable cross layer dynamic filter
join
disable cross layer dynamic filter

备注


Root
Country=AU-CA
Country=MX-USA
Country=AU
(500)
Country=CA
(600)
Country=MX
(400)
Country=USA
(1200)
Country=*
(2200)
Browser=Chrome
(500)
Browser=Chrome-Edge
Browser=Firefox-Safari
Browser=Chrome
(600)
Browser=Edge
(100)
Browser=Firefox
(400)
Browser=Safari
(100)
Browser=*
(1200)
Partition node: Does not hold any aggregation value but some pointers to children that fall under the partition range