百度在线数据仓库Palo

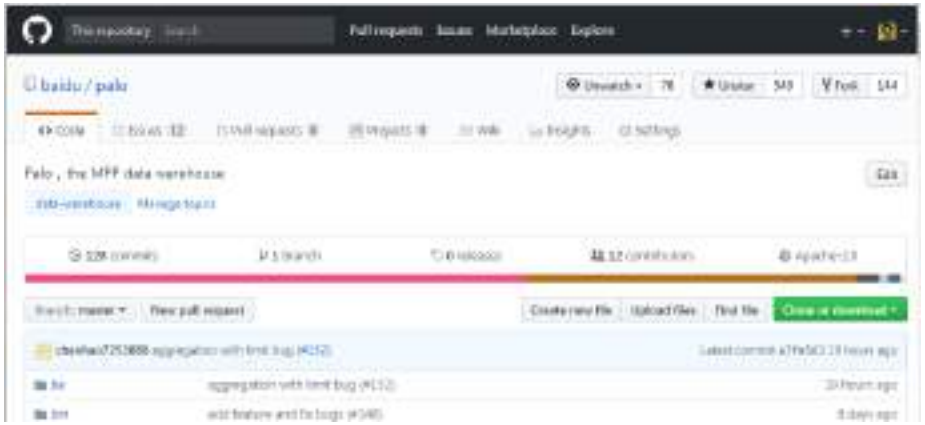
——开源架构解读及应用

百度大数据部 牟宇航

2017.12

Palo

- 名字由来：PALO <->OLAP
- A MPP-based Interactive Data Analysis SQL DB
- 百TB ~ PB级别，结构化数据，毫秒/秒级分析
- 百度大数据部研发，第三代 OLAP 产品
  - Doris -> OlapEngine -> Palo
- 百度内部署1000+台，单一业务最大500TB
- 17年8月开源，10月通过“大数据产品能力评测”

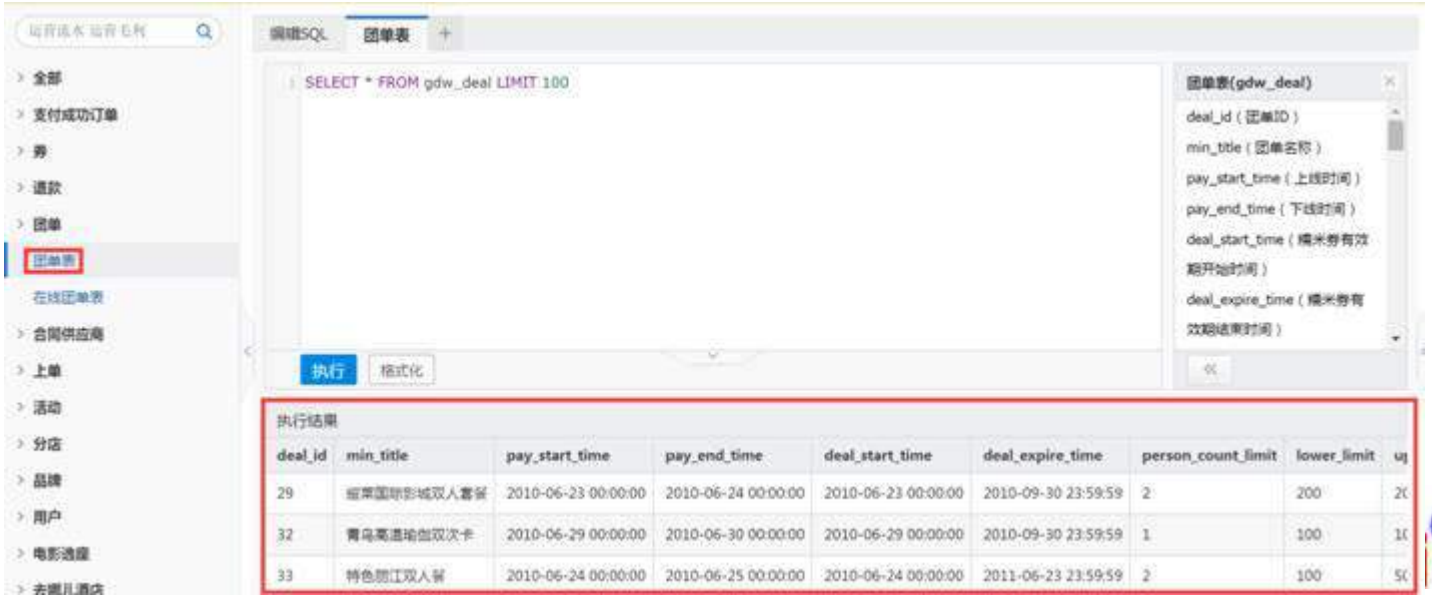


场景一

- **某在线报表业务**
  - 为网站站长提供流量分析，网站分析，受众分析等多种分析服务
  - 300+表，数据清洗结构化后百TB+，单日增量1TB+
  - 查询峰值QPS 2000+，日查询量千万级
  - 一致性（会话内单调一致性、更新一致性）
  - 导入5分钟一次
  - 查询平均延时30+ms

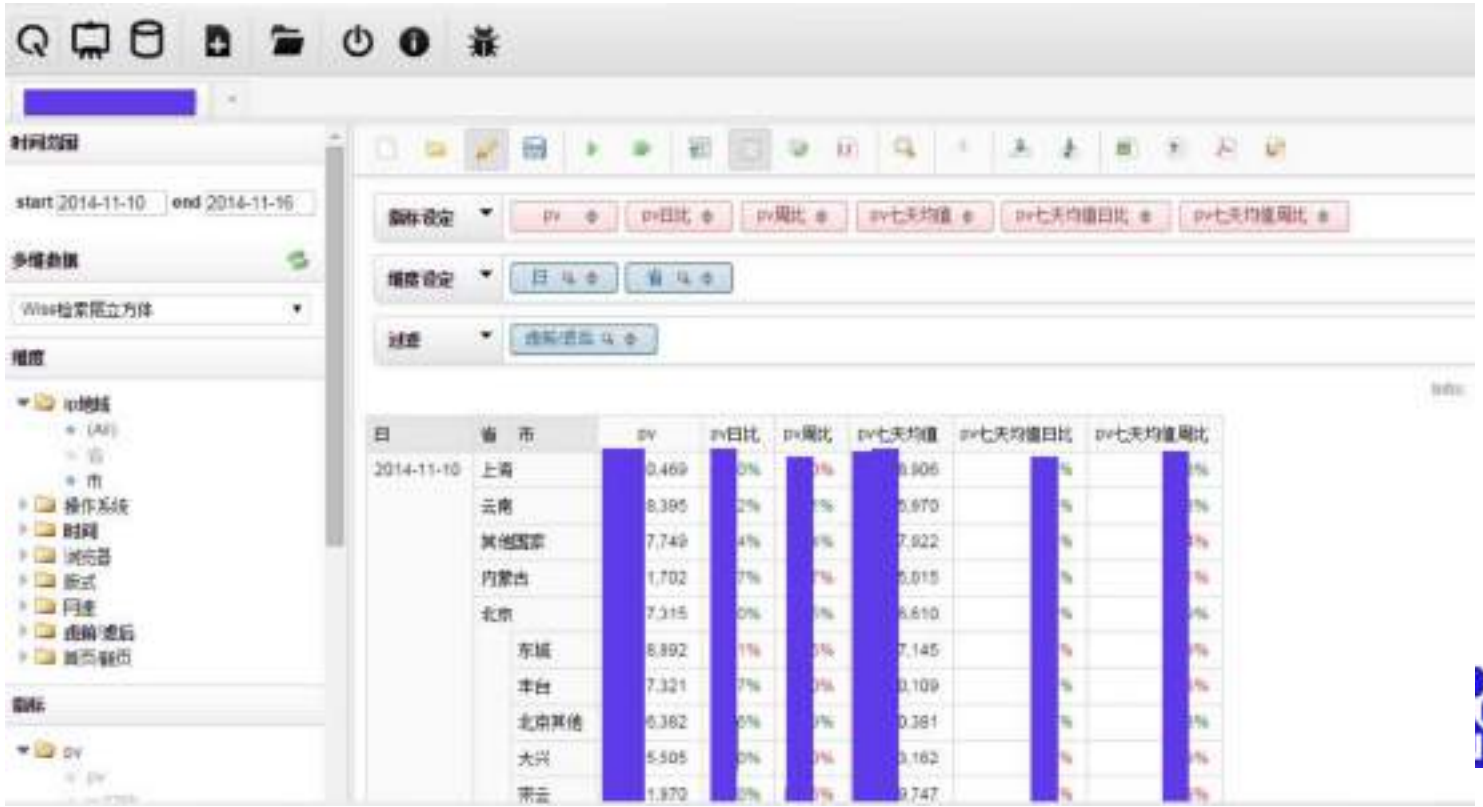
场景二

- **某业务数据集市**
  - 集运营、业务分析、订单管理、会员管理、客户关系管理等数十个管理分析平台一体的综合数据平台
  - 100+主题视图、10-100TB
  - 标准SQL，Ad-Hoc（即席查询），秒级分析



场景三

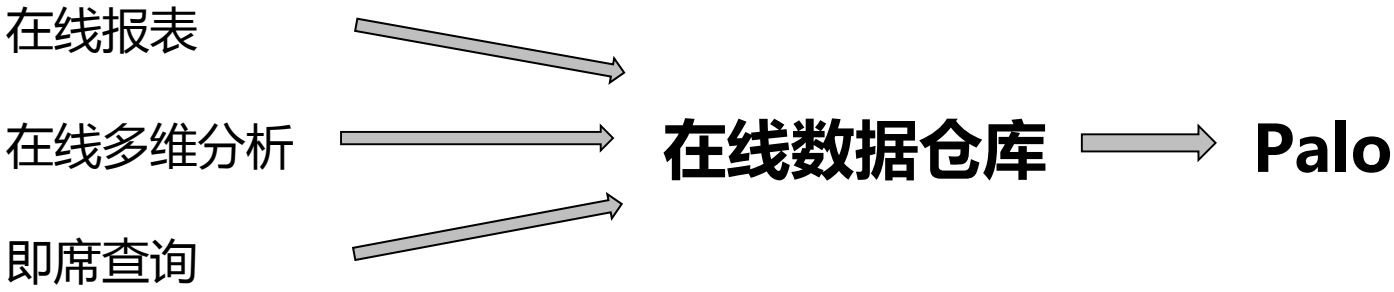
- **某在线多维分析平台**
  - 100+表，最大单表50+维度列、10+指标列，任意组合，秒级分析
  - 10-100TB



场景

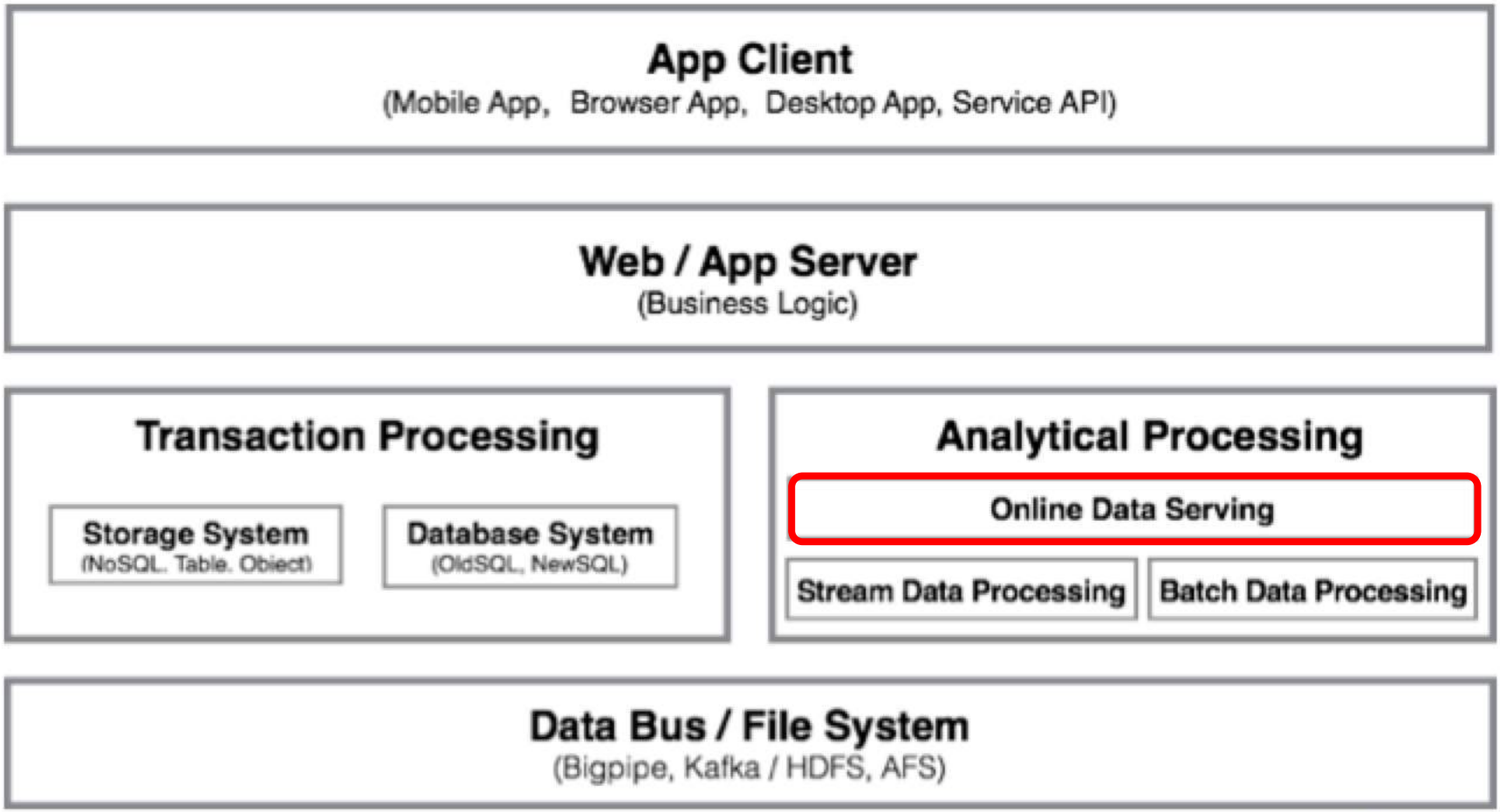
- **以前**
  - 报表：Hadoop + MySQL
  - 分析：Hadoop + Hive

- **现在**



Baidu 百度

Palo在百度大数据技术栈的位置



Palo在友商技术栈的应用

在线数据仓库——OLAP

- **Online Analytical Processing**
  - Online vs. Offline (Interactive vs. Batch)
  - Analytical Processing vs. Transactional Processing

	OLTP	OLAP
面向应用	日常交易处理	明细查询，分析决策
访问模式	简单小事务，操作少量数据	复杂聚合查询，查询大量数据
数据	当前最新数据	历史数据
数据规模	GB	TB ~ PB
数据更新	实时更新	批量更新
数据组织	满足3NF	反范式，星型模型

OLAP-商业产品

产品	简介	技术特点	收购情况
Netezza	2000年在美国成立 Netezza TwinFin	✓ 软硬一体机 ✓ 采用FPGA数据过滤代替索引	2010年9月20日，IBM出资17.8亿美元收购
Greenplum	2003年在美国成立 Greenplum Database	✓ 行存 + 列存 ✓ Shared-Nothing集群	2010年7月6日，EMC出资3亿美元收购
Vertica	2005年在美国成立 Vertica Analytic Database	✓ 列存 ✓ Shared-Nothing集群	2011年2月，HP出资3.5亿美元收购
Aster Data	2005年在美国成立 nCluster	✓ SQL-MapReduce ✓ Shared-Nothing集群	2011年7月6日，Teradata出资2.63亿美元收购
ParAccel	2005年在美国成立 PADB	✓ 列存 + 自适应压缩 ✓ Shared-Nothing集群	2013年Actian出资1.5亿美元收购，Redshift宣称使用ParAccel

Vendor and Appliance	Memory (GB)	Total Cores	Compression	User Storage (TB, Compressed)	List Price
EMC Greenplum Data Computing Appliance	768	48	4 to 1	144	$2,000,000
IBM PureData System for Analytics N1001-010	n/a	112	4 to 1	128	$1,599,000
Microsoft SQL Server 2012 Parallel Data Warehouse[1]	2,304	144	5 to 1	340	$1,569,970
Oracle Exadata Database Machine X3-2	2,048	128	10 to 1	450	$13,580,000
Teradata Data Warehouse Appliance 2690	768	96	4 to 1	146	$1,168,000

OLAP-开源社区

Apache Impala (incubating) is the open source, native analytic database for Apache Hadoop. Impala is shipped by Cloudera, MapR, Oracle, and Amazon.

Follow us on Twitter at @ApacheImpala!

Druid is a high-performance, column-oriented, distributed data store.

百度OLAP功能需求

- 基本需求
  - high availability
  - Scalability
  - High Performance
- 特化需求
  - Consistent View
  - Monotonic Consistency
  - Heterogeneous Storage
  - ……

Palo定位

低成本

1/10 ~1/100 Cost

线性扩展

100~200节点 / 1000 TB

支持云化部署

高可用

99.9999 % Uptime

高查询性能

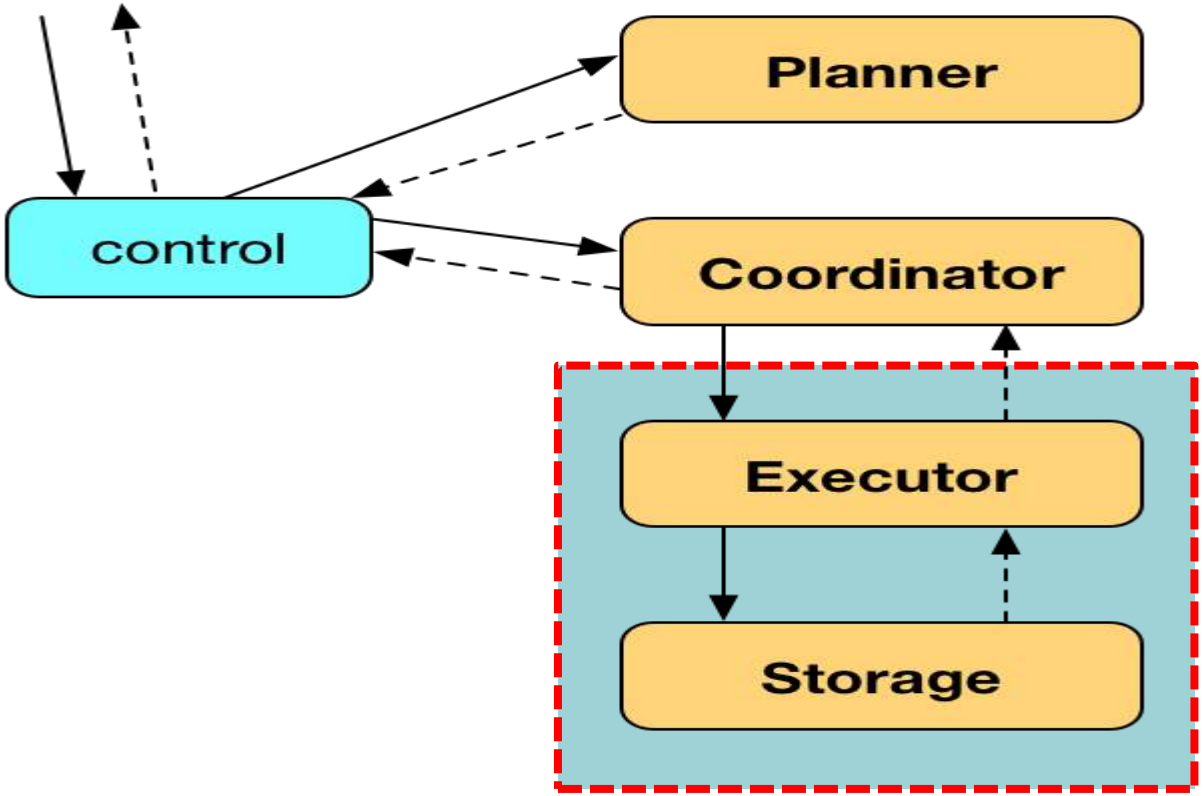
10W QPS/ 100GB/s

高加载性能

10 TB / Hour

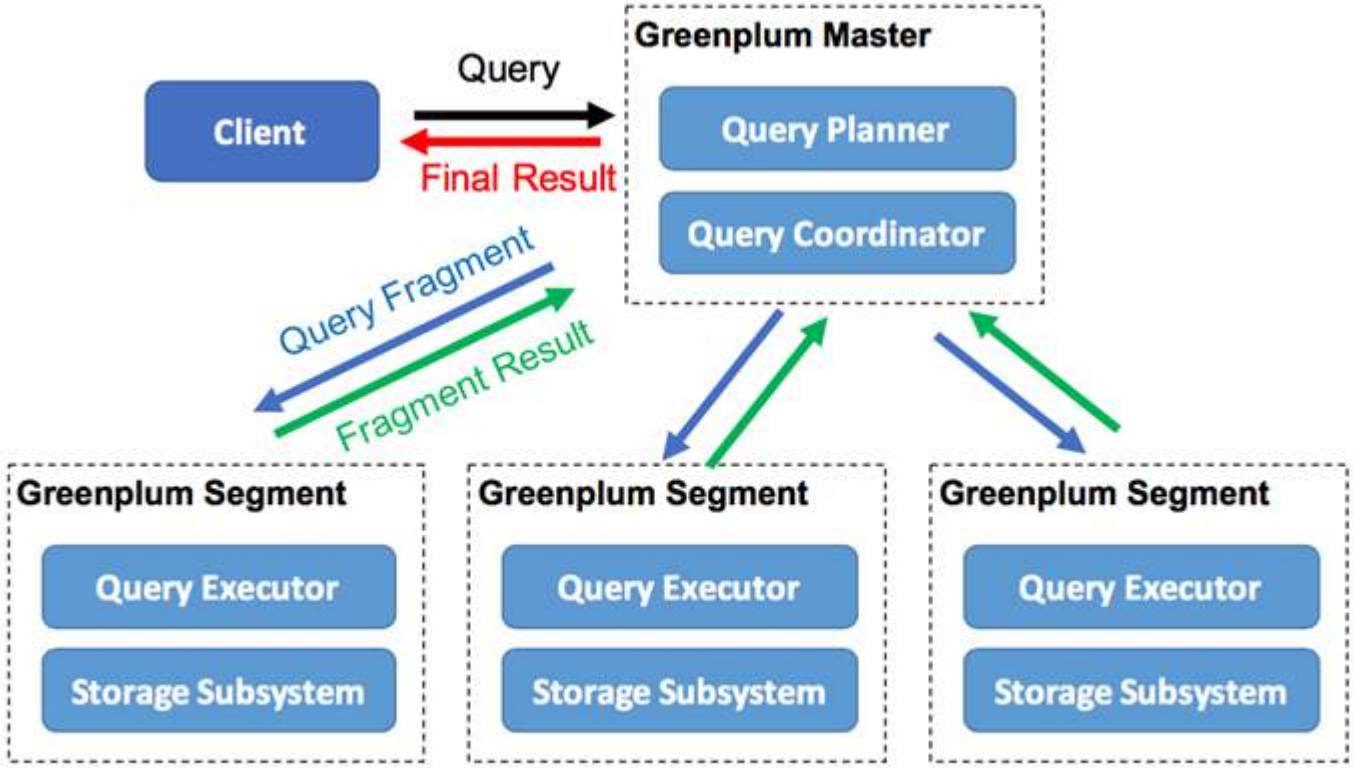
Palo整体架构

- MPP通用构件



中心架构（Centric Architecture）

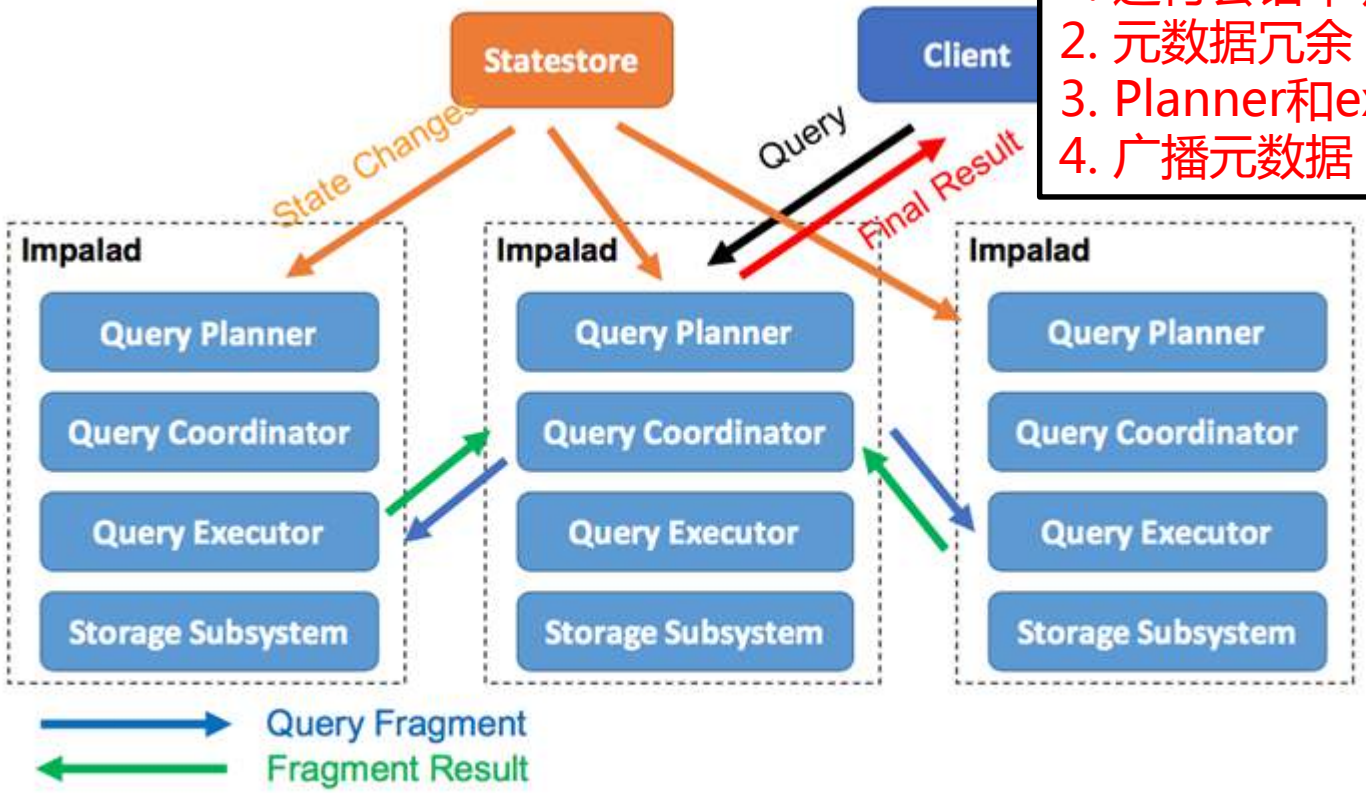
Master may be bottleneck



Centric architecture — Greenplum.

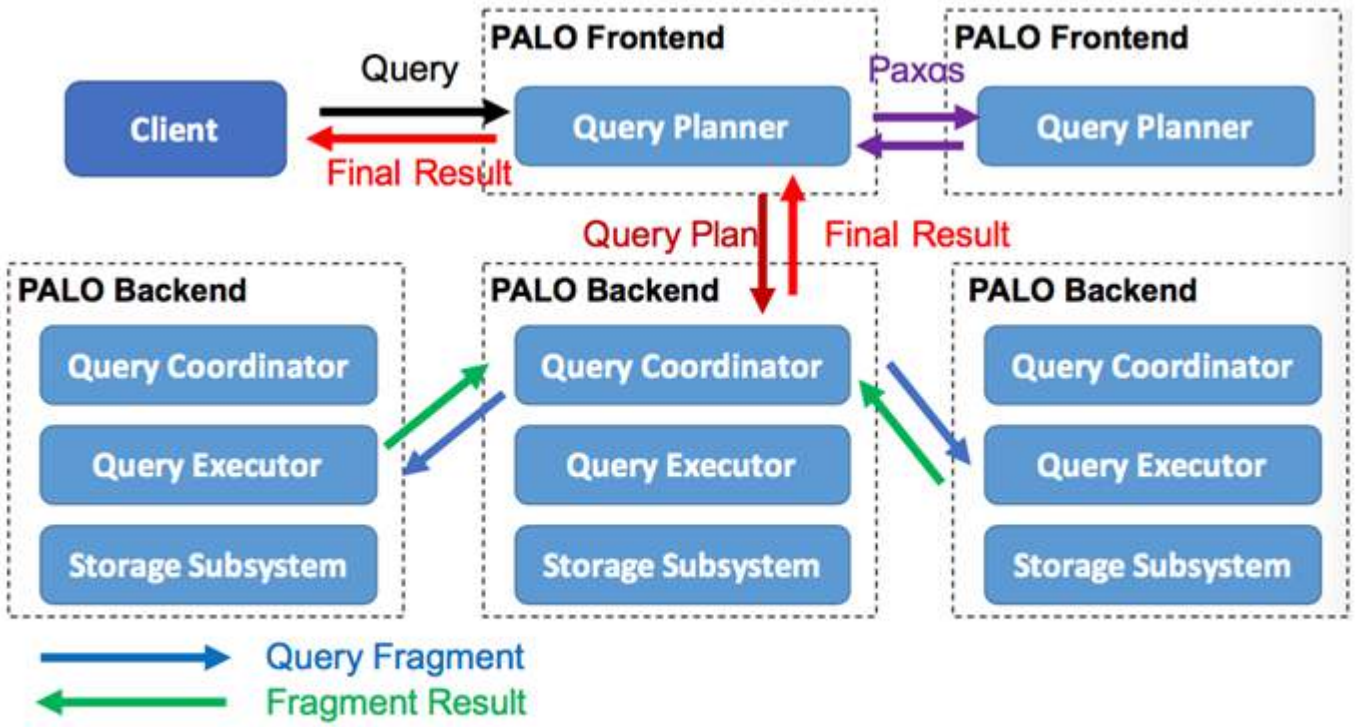
对称架构（Symmetric Architecture）

1. 违背会话单调一致性
2. 元数据冗余
3. Planner和executor争内存
4. 广播元数据，可能有网络瓶颈



Symmetric architecture — Impala.

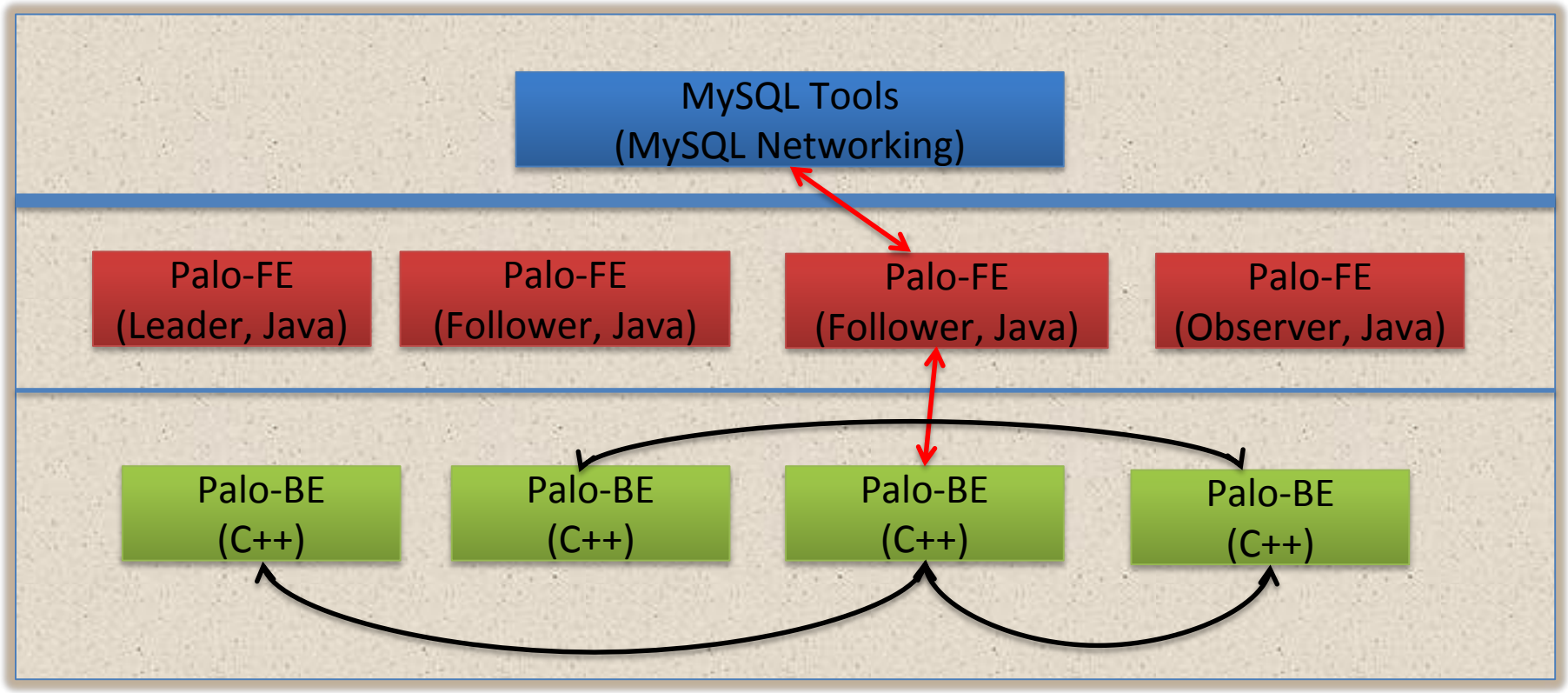
混合架构（Hybrid Architecture）



Hybrid architecture — PALO.

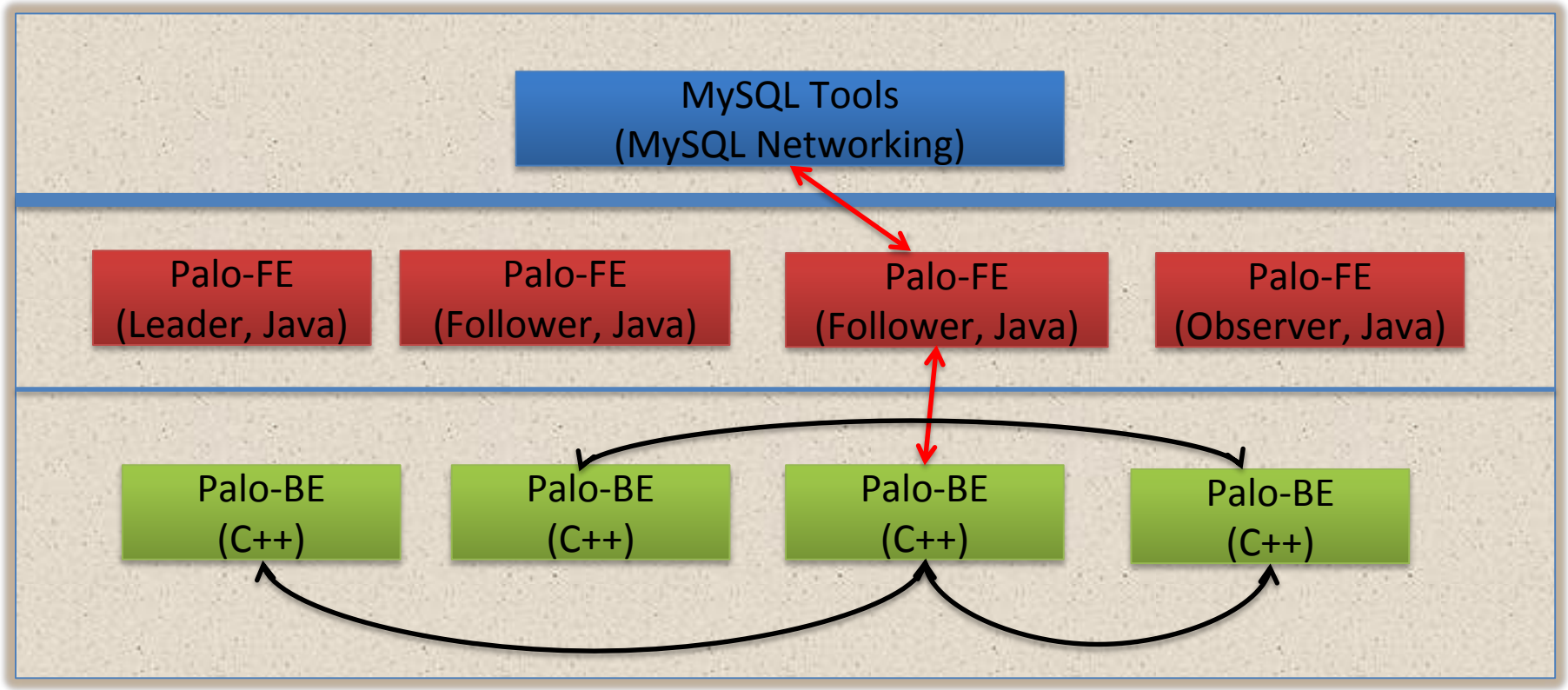
Palo整体架构

- **Hybrid Architecture**
  - Leader * 1 + Follower * 2 + Observer * n
  - 高可靠；高QPS；可动态扩缩容；支持频繁的元数据更改



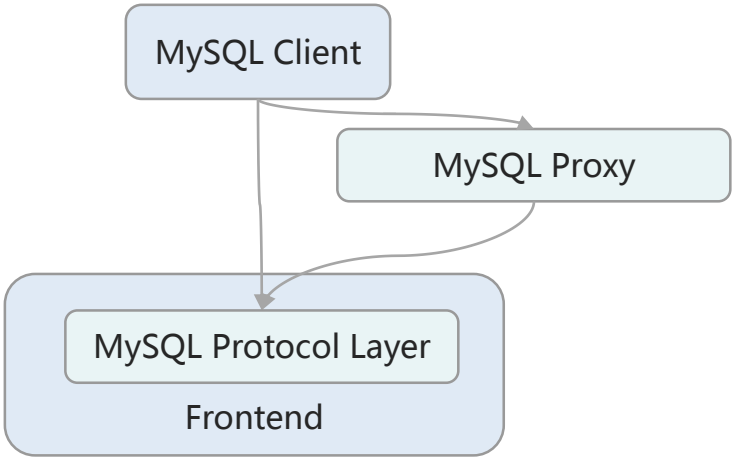
集成式系统

- **Integrated system VS layered method**
  - 多表原子导入
  - 进程内通信 VS 进程间通信
  - 易于开发和调试



Palo使用

```
test@mry-laptop:~$ mysql -h tc-inf-devop01.tc.baidu.com -P 8276 -u maruyue
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 0
Server version: 4.1.2 (Powered by Palo 2.0 Beta)

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| demo               |
| fc                 |
| information_schema |
| lbs                |
| searchbox          |
| test               |
+--------------------+
6 rows in set (0.01 sec)

mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_test  |
+-----------------+
| fc_cmatch_fact  |
| tbldim_pn       |
| tbldim_querytrade |
| tbldim_region   |
| tbldim_wbws     |
| tbldim_wos      |
| tbldim_wpt      |
+-----------------+
7 rows in set (0.01 sec)
```



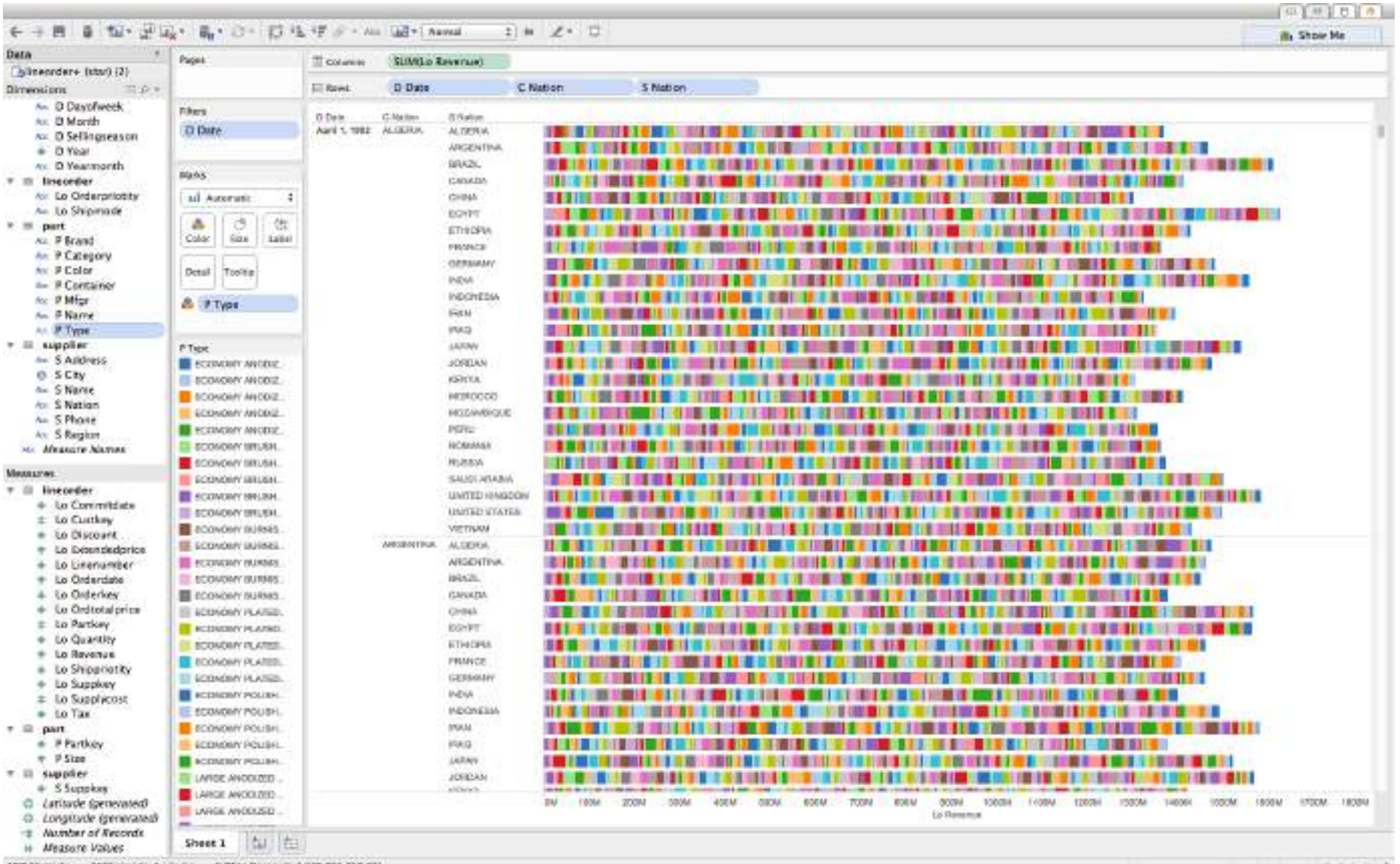
- 轻量级客户端
- 与上层应用兼容容易
- 学习曲线平缓，方便用户上手使用
- 利用MySQL相关工具，比如MySQL Proxy

Baidu 百度

Palo使用

```
./mysql -h PALO_FE_HOST -P PALO_FE_PORT -uYOUR_USERNAME -pYOUR_PASSWORD

CREATE DATABASE example_db;

USE example_db;

CREATE TABLE ps_stats_tbl (
    siteid   INT,         DEFAULT '10',
    day      DATETIME,
    citycode SMALLINT,
    username VARCHAR(32) DEFAULT '',
    pv       BIGINT  SUM DEFAULT '100'
) DISTRIBUTED BY HASH(siteid) BUCKETS 32;

LOAD LABEL ps_stats_20150717 (
    DATA INFILE("hdfs://host:port/ps_stats_data")
    INTO TABLE ps_stats_tbl
);

SHOW LOAD WHERE LABEL = "ps_stats_20150717";

SELECT siteid, sum(pv) FROM ps_stats_tbl WHERE day = "2015-07-17" GROUP BY siteid;
+----------+----------+
| siteid   |  sum(pv) |
+----------+----------+
| 23143    |   114996 |
| 12345    |   318925 |
+----------+----------+
2 rows in set (0.02 sec)
```

Palo使用


Data
Dimensions
O Dayofweek
O Month
O Sellingseason
O Year
O Yearmonth
lineorder
Lo Orderpriority
Lo Shipmode
part
P Brand
P Category
P Color
P Container
P Mfgr
P Name
P Type
supplier
S Address
S City
S Name
S Nation
S Phone
S Region
Measure Names
Measures
lineorder
Lo Commitdate
Lo Custkey
Lo Discount
Lo Extendedprice
Lo Linenumber
Lo Orderdate
Lo Orderkey
Lo Ordtotalprice
Lo Partkey
Lo Quantity
Lo Revenue
Lo Shippriority
Lo Suppkey
Lo Supplycost
Lo Tax
part
P Partkey
P Size
supplier
S Suppkey
Latitude (generated)
Longitude (generated)
Number of Records
Measure Values
Pages
Filters
O Date
Marks
Automatic
Color
Size
Label
Detail
Tooltip
P Type
Columns
SUM(Lo Revenue)
Rows
O Date
C Nation
S Nation
Show Me
Sheet 1

Palo使用


ORACLE Business Intelligence
1.4 计分析
总览
平台
(所有列值)
版式
(所有列值)
日期
应用
重置
说明：用户量为各渠道用户量加和，并未做渠道间的去重处理。
说明标题
分析 - 编辑 - 导出
总览
查询条件
版式：(所有列值)
平台：(所有列值)
日期
9,154
155
23,358
7,305
658
476
485
401
9,812
038
6,831
,509
3,429
566
2,533
,981
6,659
752
8,394
,124
9,167
37
7,940
,264
9,149
70
9,474
869,
2,117
99
4,806
559
011
248
7,238
372

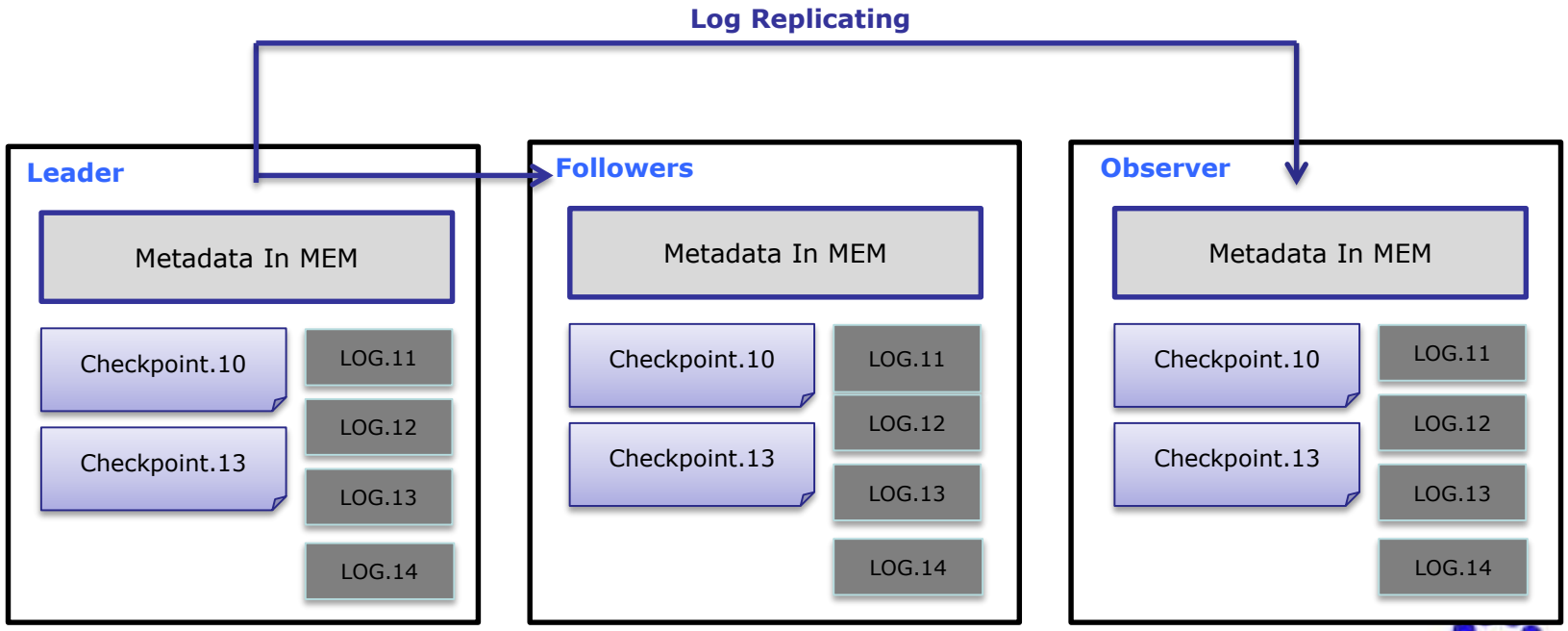
Palo使用

```
→ test_r  R

R version 3.0.1 (2013-05-16) -- "Good Sport"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Previously saved workspace restored]

> library(RMySQL)
Loading required package: DBI
> con <- dbConnect(MySQL(), user="root", password="[redacted]", dbname="demo", host="tc-inf-devop01.tc.baidu.com", port=8276)
> dbListTables(con)
[1] "cumulative_detail_test" "fc_cmatch_fact"         "tbldim_pn"
[4] "tbldim_querytrade"      "tbldim_region"          "tbldim_wbws"
[7] "tbldim_wos"             "tbldim_wpt"             "ud_test"
> rs <- dbSendQuery(con, "select * from tbldim_region")
> d1 <- fetch(rs, n = 10)
> d1
   pid cid province     city
1    1   0     北京 北京其他
2    2   0     上海 上海其他
3    3   0     天津 天津其他
4    4   0     广东 广东其他
5    5   0     福建 福建其他
6    8   0     海南 海南其他
7    9   0     安徽 安徽其他
8   10   0     贵州 贵州其他
9   11   0     甘肃 甘肃其他
10  12   0     广西 广西其他
>
```

Baidu 百度

元数据高可用

- Memory + Checkpoint + Journal
- 采用 Berkeley DB Java Edition，类Paxos协议实现，

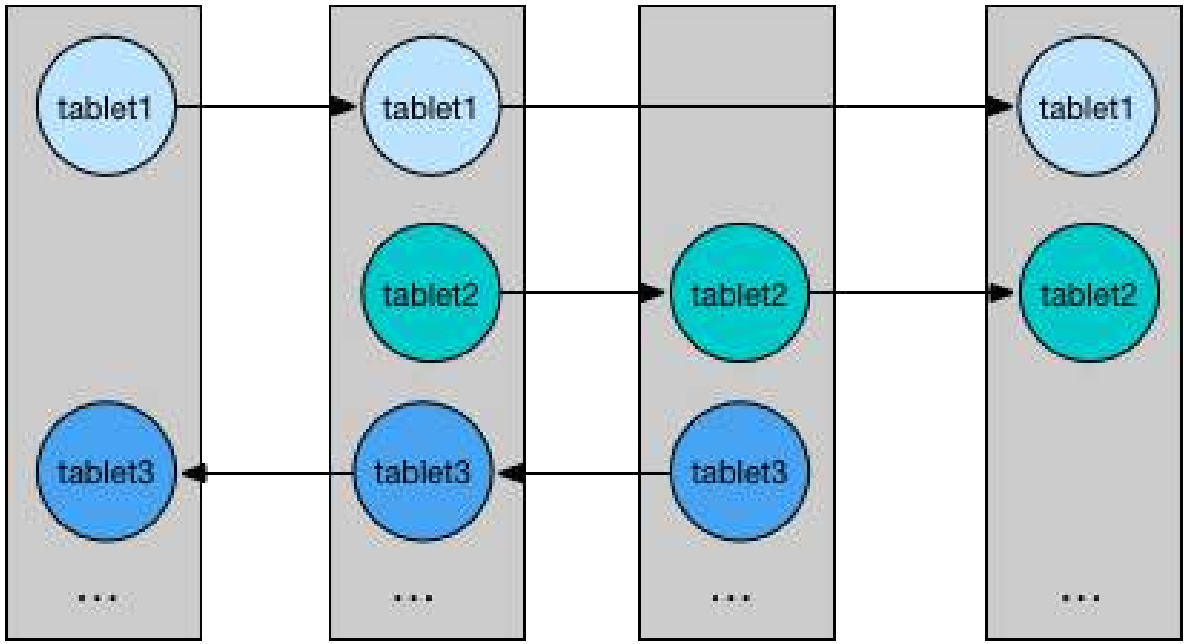


元数据一致性保证

- 会话内单调一致性
  - Master
  - Follower和Observer
- 多个会话间（尤其在失败重试时）
  - 提供sync命令来手动同步

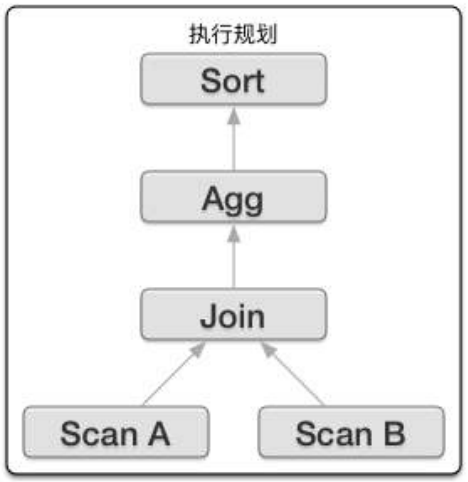
数据高可靠

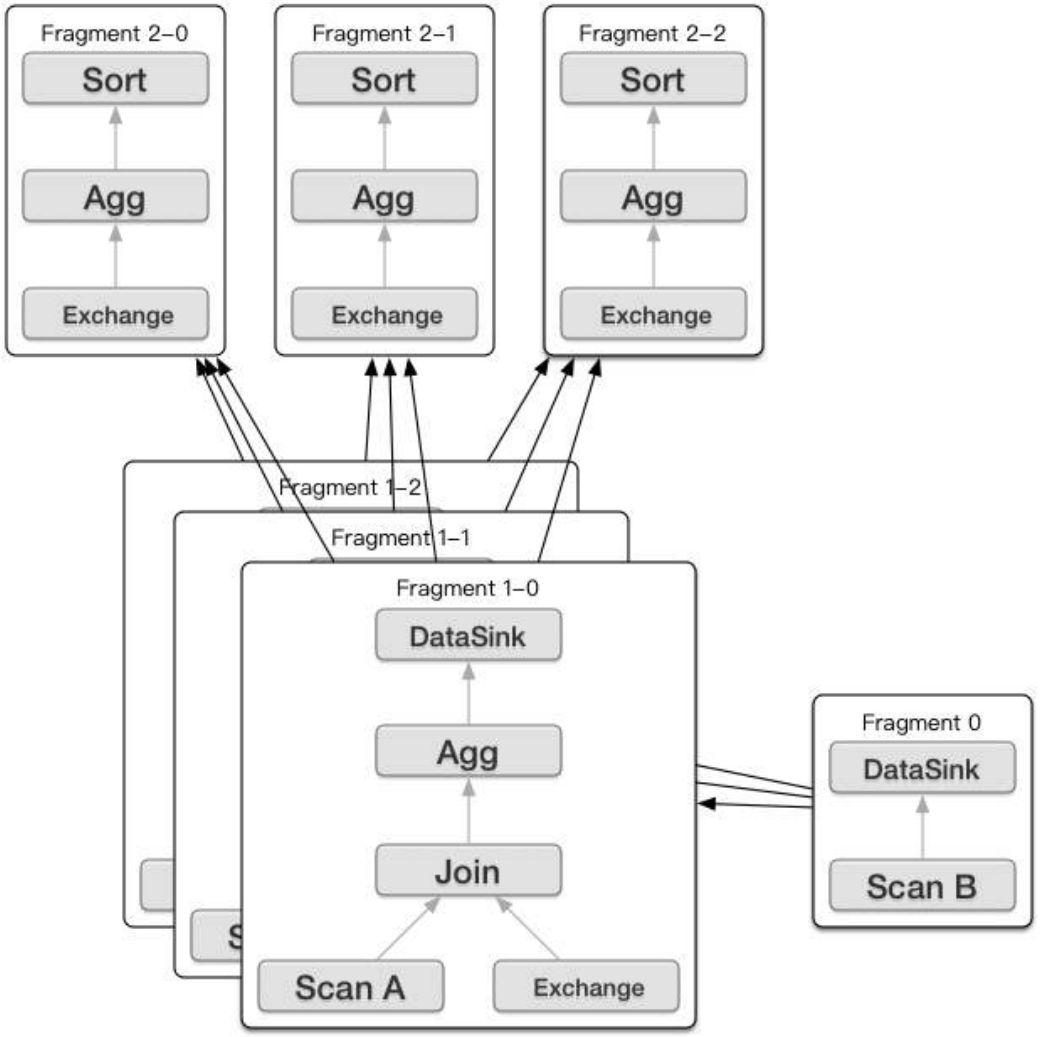
- 默认三副本
- 自动均衡
- 自动补充



MPP执行

```
SELECT k1, SUM(v1)
FROM A, B
WHERE A.k2 = B.k2
GROUP BY k1
ORDER BY SUM(v1)
```

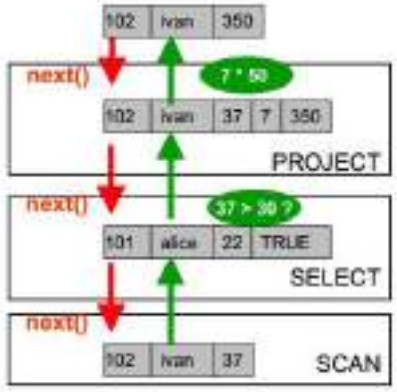


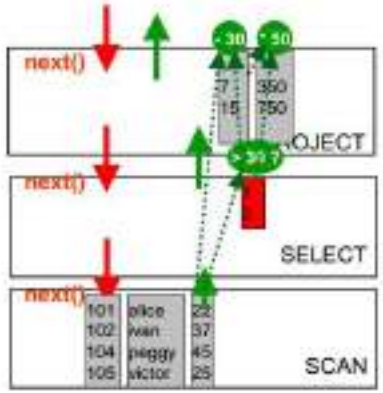


向量化执行&LLVM

- **向量化**
  - 行式执行引擎问题
    - 每行一次函数调用，打断CPU流水，不利于分支预测
    - 指令和数据cache miss
    - 编译器不友好，不利于循环展开，SIMD
  - 设计思想
    - 单条处理到批量处理
    - 行式处理转化为列式处理
  - 效果：star-schema测试整体提升3~4倍

```
SELECT  id, name
        (age-30)*50 AS bonus
FROM    employee
WHERE   age > 30
```





- **LLVM**
  - 运行时代码生成
  - 大型ad-hoc查询可提升5倍以上

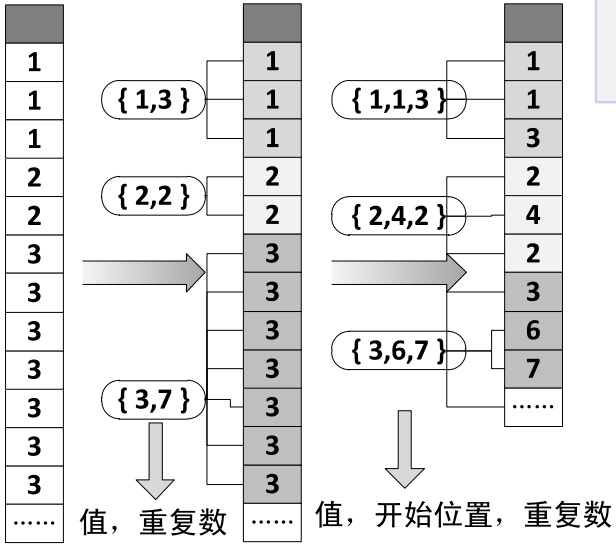
Baidu 百度

存储格式-列存

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	888 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

Block 1

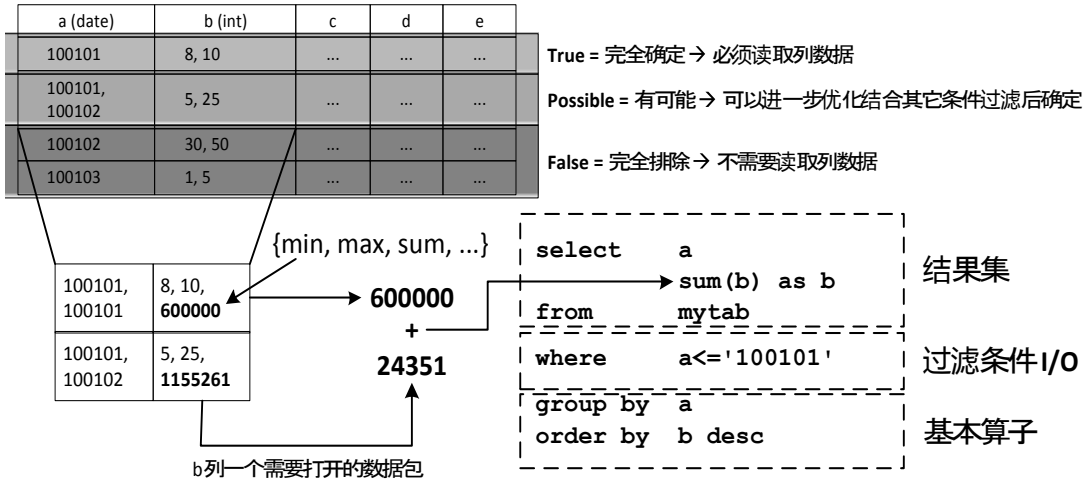
- ✓ 数据按列存储，每一列单独存放
- ✓ 只访问查询涉及的列，大量降低I/O
- ✓ 数据类型一致，方便压缩
- ✓ 数据包建索引，数据即索引



数值类型的行程编码（RLE）压缩

- ✓ Palo存储引擎利用原始过滤条件以及min、max和sum智能索引技术

将数据集查询范围尽可能地缩小，可以大大减少I/O，提升查询性能



数据模型-Schema

- **Google Mesa模型**
  - 维度（Key列），指标（Value列）
- **Key列有序存储**
  - 查询快速定位
- **全Key全局唯一**
  - 相同Key的行，其Value列自动合并(SUM,MIN,MAX,REPLACE)

Time	Id	Country	Clicks	Cost
2016/12/31	1	US	10	32
2017/01/01	2	UK	40	20
2017/01/01	2	US	150	80

数据模型-预聚合

Base

Time	Id	Country	Clicks	Cost
2016/12/31	1	US	10	32
2017/01/01	2	UK	40	20
2017/01/01	2	US	150	80

+

Delta

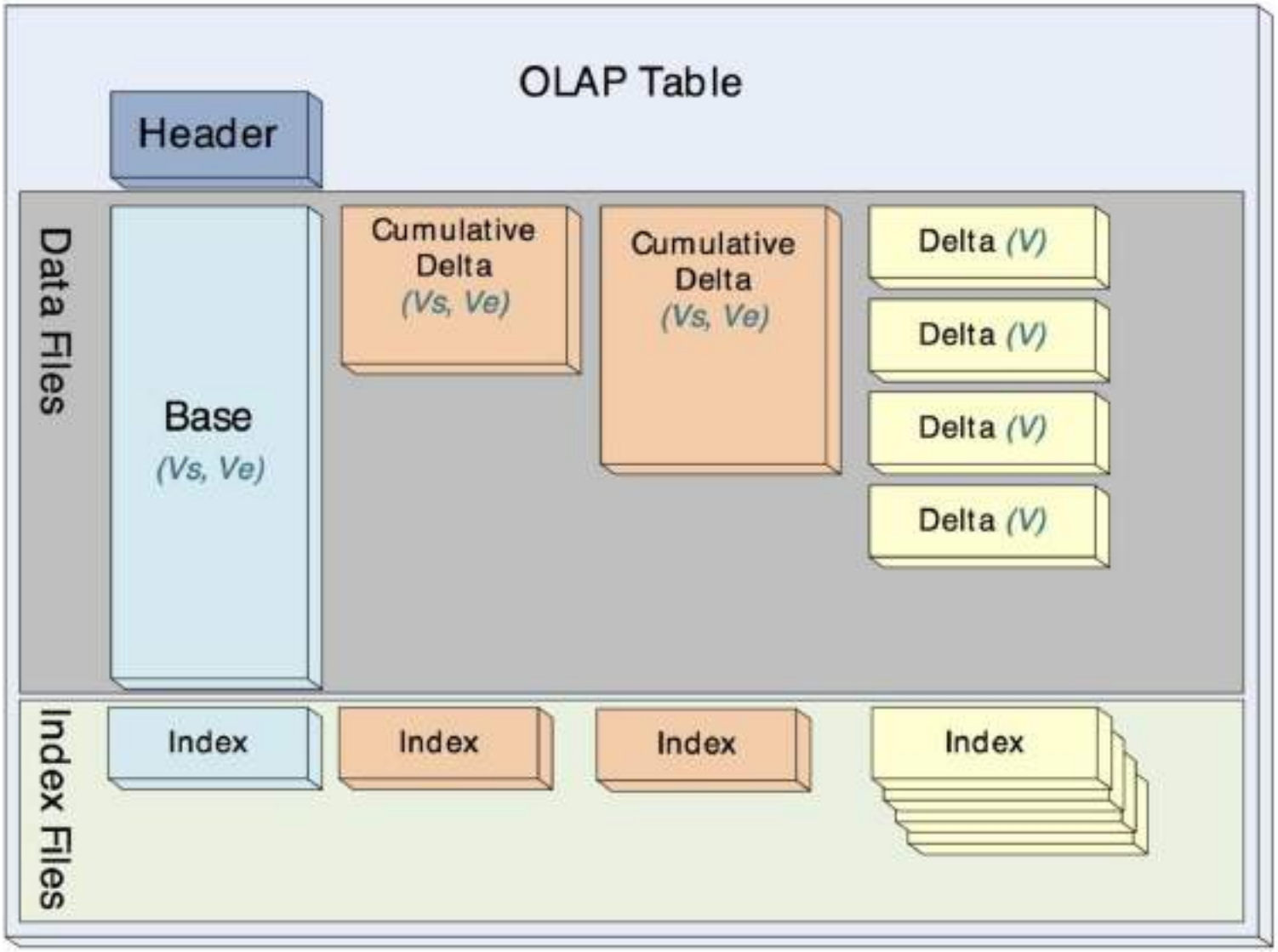
Time	Id	Country	Clicks	Cost
2017/01/01	1	US	5	3
2017/01/01	2	UK	60	30
2017/01/01	2	US	50	20

New Base

Time	Id	Country	Clicks	Cost
2016/12/31	1	US	10	32
2017/01/01	1	US	+5	+3
2017/01/01	2	UK	40+60	20+30
2017/01/01	2	US	150+50	80+20

Baidu 百度

数据模型-多版本


OLAP Table
Header
Data Files
Base
(Vs, Ve)
Cumulative
Delta
(Vs, Ve)
Cumulative
Delta
(Vs, Ve)
Delta (V)
Delta (V)
Delta (V)
Delta (V)
Index Files
Index
Index
Index
Index

数据模型-更多存储模型

- **聚合模型的缺点**
  - 不易理解，某些Olap场景没有聚合需求
  - 读放大

    value列的过滤条件无法下推

    count一个列会造成读取所有列
  - 较多key列时，排序本身可能成为瓶颈
- **更丰富的存储模型选择**
  - DUPLICATED KEY
  - UNIQUE KEY
  - AGGREGATE KEY

物化视图(rollup)

- **以空间换时间**
  - Base表中列的子集
  - key列重新排序
- **查询时自动选择**
- **导入原子生效**

时间	Id	省份	pv
2017.01.01	1	北京	10
2017.01.01	2	天津	30
2017.01.02	1	北京	20
2017.01.02	2	北京	40

Base表

按指定列排序

Id	时间	省份	pv
1	2017.01.01	北京	10
1	2017.01.02	北京	20
2	2017.01.01	天津	30
2	2017.01.02	北京	40

聚合表

Id	pv
1	30
2	70

两层分区 & 分级存储

- **两层分区**
  - 方便新旧数据分离，使用不同的存储介质（新数据SSD，历史数据SATA）
  - 减少了大量历史数据不必要的重复合并，节省了大量的IO和CPU开销
  - 简化了表的扩容，shard调整
- **分级存储**
  - 用户可以指定数据放到SSD上或者SATA盘上，也支持根据TTL将冷数据从SSD迁移到SATA上，高效利用SSD提高查询性能

```
CREATE TABLE example_tbl (
    k1 DATE,
    k2 INT,
    v1 VARCHAR(2048) REPLACE,
) PARTITION BY RANGE (k1) (
    PARTITION p1 VALUES LESS THAN ("2014-01-01")
        properties ("storage_media"="ssd", "storage_cooldown"="2015-06-01 10:00:00"),
    PARTITION p2 VALUES LESS THAN ("2014-06-01")
        properties ("storage_media"="ssd"),
    PARTITION p3 VALUES LESS THAN ("2014-12-01")
        properties ("storage_media"="hdd"),
) DISTRIBUTED BY HASH(k2) BUCKETS 32;
```

Online Schema Change

- 变更期间不停服，用户业务上层不需感知
- 加列、减列，修改列类型等
- 详见 help alter table

数据导入

- 按批导入
- 异步
  - Show load查看状态
- Label机制
  - 防止数据被重复导入

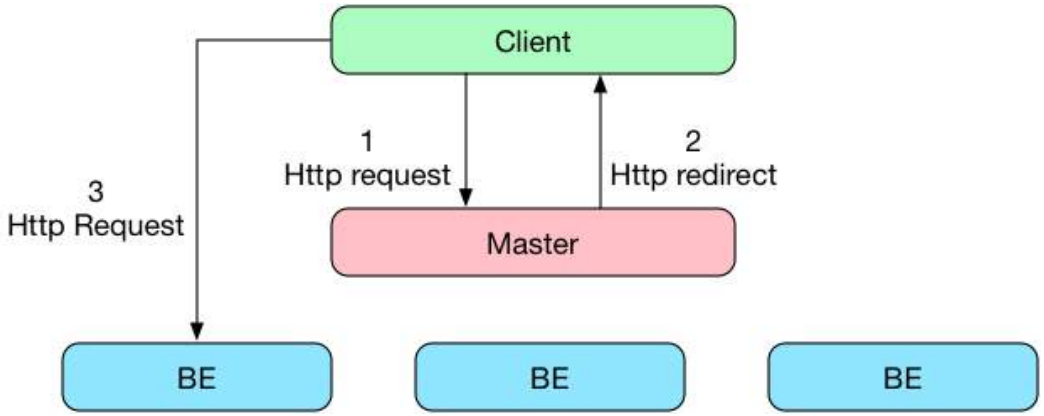
数据导入

- **Broker**
  - 支持直接从HDFS、Baidu BOS上进行数据读取。
  - 每个物理机部署broker进程，提供并发读取能力。
  - 可以访问外部表。

```
LOAD LABEL example_db.label1(
    DATA INFILE("hdfs://dpp.cluster.com:54310/user/palo/data/input/file")
    INTO TABLE `my_table`
)
WITH BROKER hdfs ("username"="hdfs_user", "password"="hdfs_password")
PROPERTIES(
    "timeout"="3600",
    "max_filter_ratio"="0.1"
);
```

数据导入

- **MINI BATCH**
  - 使用http即可导入，减少客户端对其它组件的依赖
  - 实现了多表导入的事务提交




```
-- BATCH DATA LOADIND --
LOAD LABEL ps_stats_20150717 (
    DATA INFILE("hdfs://host:port/input/ps_stats_data")
    INTO TABLE ps_stats_tbl
);

-- Mini-BATCH DATA LOADING --
curl -u username,password -T ./input/ps_stats_data http://fe.host:port/api/db1/ps_stats_tbl/_load?label=ps_stats_20150717
```

资源隔离

- **问题**
  - 多用户影响
  - 单用户多任务影响
- **解决**
  - 线程级cgroup
  - 两级资源组织

```
mysql> show resource;
+-------+---------------+-------+
| User  | Resource type | Value |
+-------+---------------+-------+
| root  | CPU_SHARE     | 1000  |
| zw    | CPU_SHARE     | 1000  |
+-------+---------------+-------+

mysql> show quota;
+------+--------+-------+
| User | Group  | Quota |
+------+--------+-------+
| root | high   | 800   |
| root | low    | 100   |
| root | normal | 400   |
+------+--------+-------+
3 rows in set (0.01 sec)
```

```
|-- cgroup.clone_children
|-- cgroup.event_control
|-- cgroup.procs
|-- cpu.cfs_period_us
|-- cpu.cfs_quota_us
|-- cpu.shares
|-- cpu.stat
|-- cpuacct.stat
|-- cpuacct.usage
|-- cpuacct.usage_percpu
|-- notify_on_release
|-- root
|-- tasks
|-- test_user
|-- yiguolei
```

```
|-- cgroup.clone_children
|-- cgroup.event_control
|-- cgroup.procs
|-- cpu.cfs_period_us
|-- cpu.cfs_quota_us
|-- cpu.shares
|-- cpu.stat
|-- cpuacct.stat
|-- cpuacct.usage
|-- cpuacct.usage_percpu
|-- high
|   |-- cgroup.clone_children
|   |-- cgroup.event_control
|   |-- cgroup.procs
|   |-- cpu.cfs_period_us
|   |-- cpu.cfs_quota_us
|   |-- cpu.shares
|   |-- cpu.stat
|   |-- cpuacct.stat
|   |-- cpuacct.usage
|   |-- cpuacct.usage_percpu
|   |-- notify_on_release
|   |-- tasks
|-- low
|   |-- cgroup.clone_children
|   |-- cgroup.event_control
|   |-- cgroup.procs
|   |-- cpu.cfs_period_us
|   |-- cpu.cfs_quota_us
|   |-- cpu.shares
|   |-- cpu.stat
|   |-- cpuacct.stat
|   |-- cpuacct.usage
```

部分SQL增强

- **支持窗口函数**
  - 计算排名、同环比问题变得简单高效
  - 同一部门内，薪水最高的三位？

    rank() over (partition by dept order by salary)
- **类Proc机制**
  - Show proc ” /XX”
- **HELP**
  - 在线查看帮助文档
  - Web help
  - Help xxx

联系方式

- GitHub：

  https://github.com/baidu/palo

- 百度云：

  https://cloud.baidu.com/product/palo.html

- 邮件：

  palo-rd@baidu.com

- Palo开源讨论群：

  加我的微信，备注“加入Palo技术讨论群”

  myh13161636186

Baidu 百度

谢谢