SACC 2014中国系统架构师大会
SYSTEM ARCHITECT CONFERENCE CHINA 2014
发现架构之美

Palo: a MPP-based Interactive Data Analysis SQL DB

maruyue@baidu.com
Tel: 15810293784
2014.9.17

目录

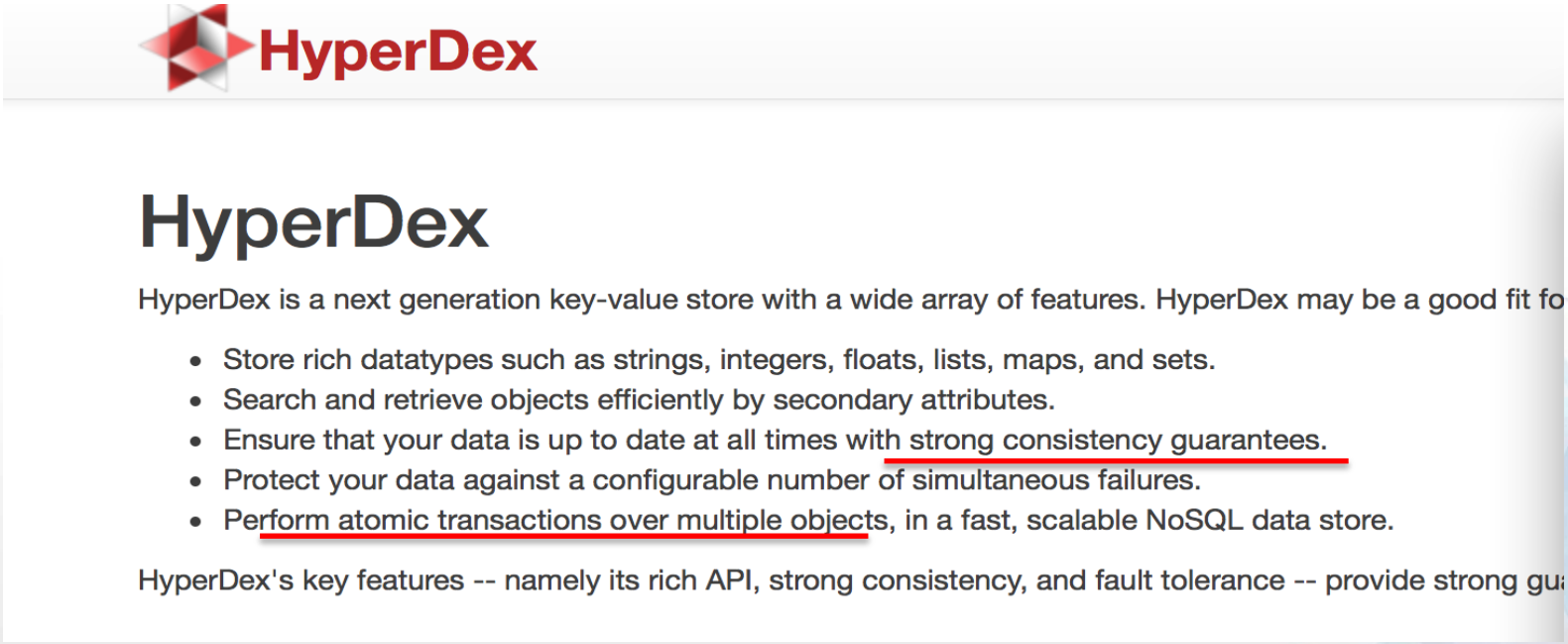
- 做Palo的背景
- Palo整体架构
- Palo关键技术
- 与竞品的比较
- 我想使用Palo

数据系统分类

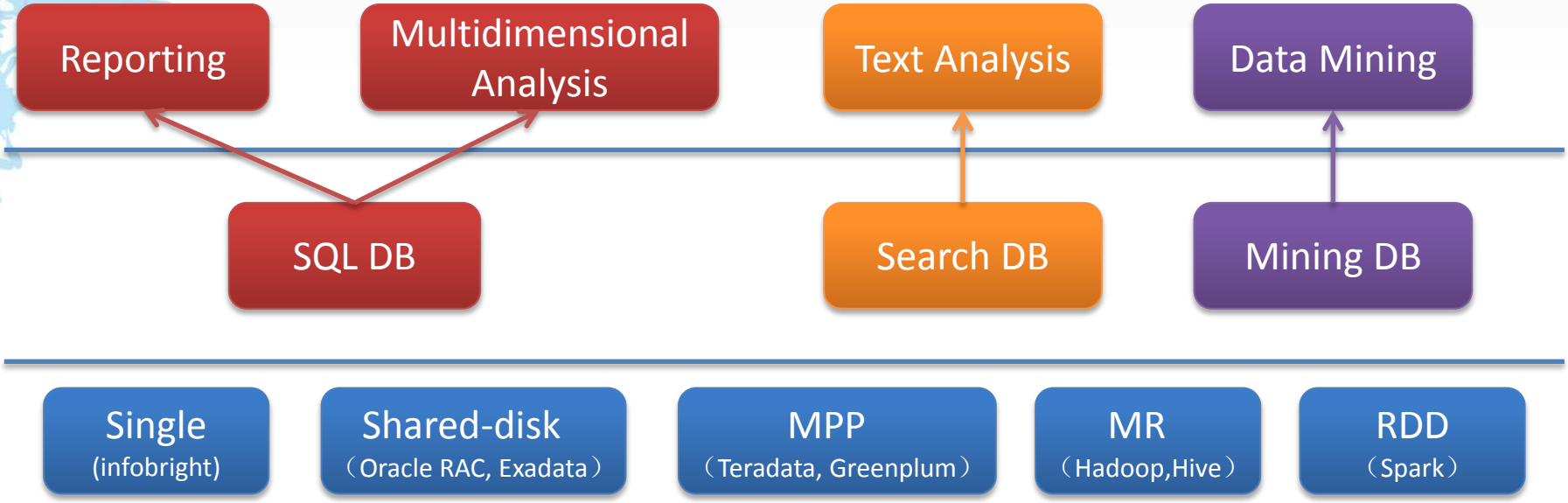
- OLTP vs. OLAP
- Operational system vs. Data warehousing
- Transactional database vs. Analytic database
- Implementation Architecture
  - ***Data Engine = Function Engine(Storage Engine)***
  - MySQL: Querying Engine over Storage Engine
  - Hadoop: MapRed over HDFS

OLTP

- SQL DB(MySQL),NoSQL DB(MongoDB), NewSQL DB(Spanner)
- High Concurrency, Strong Consistent, Transactional
- Function: CURD(Create，Update，Read，Delete)



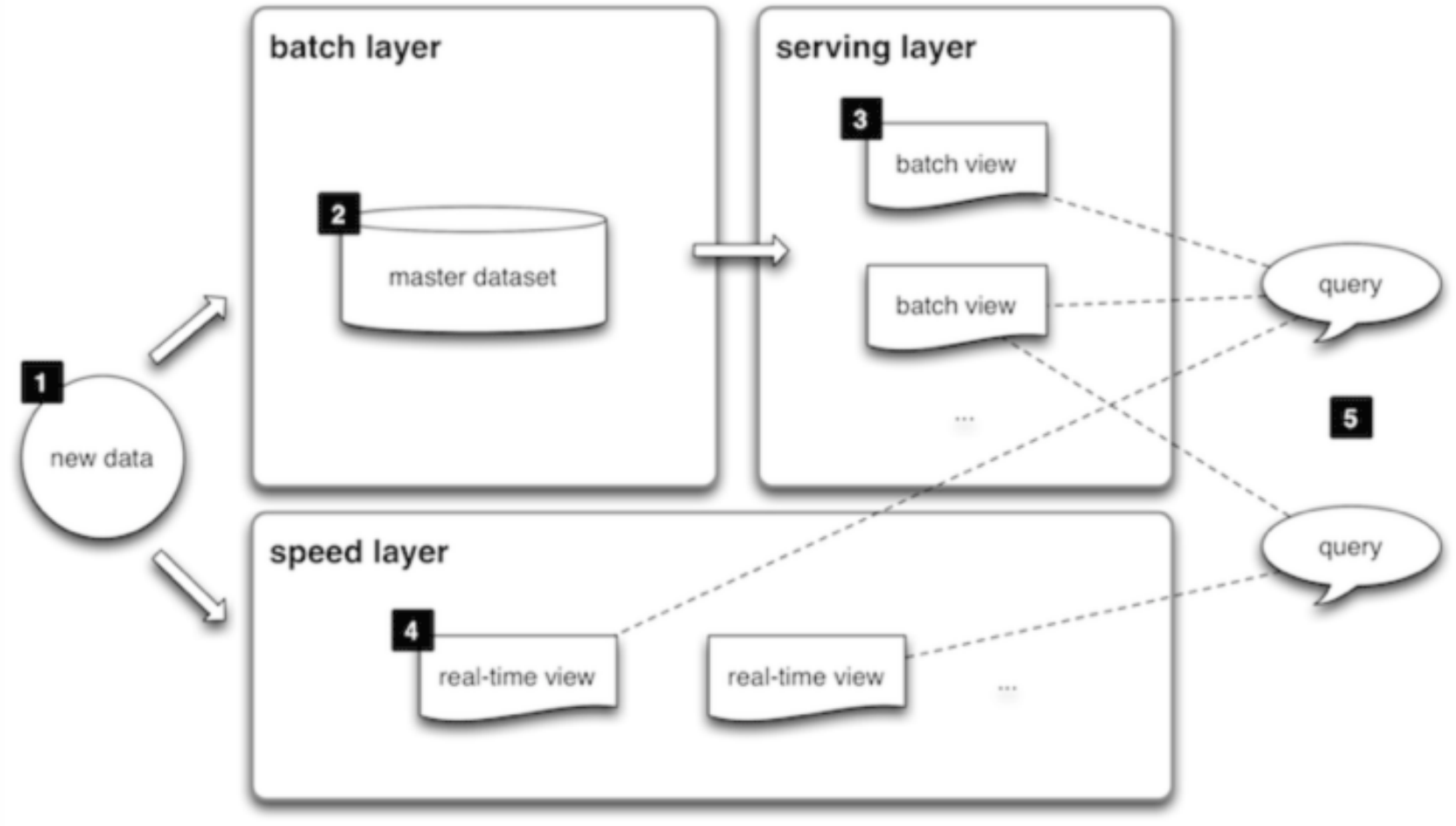
OLAP



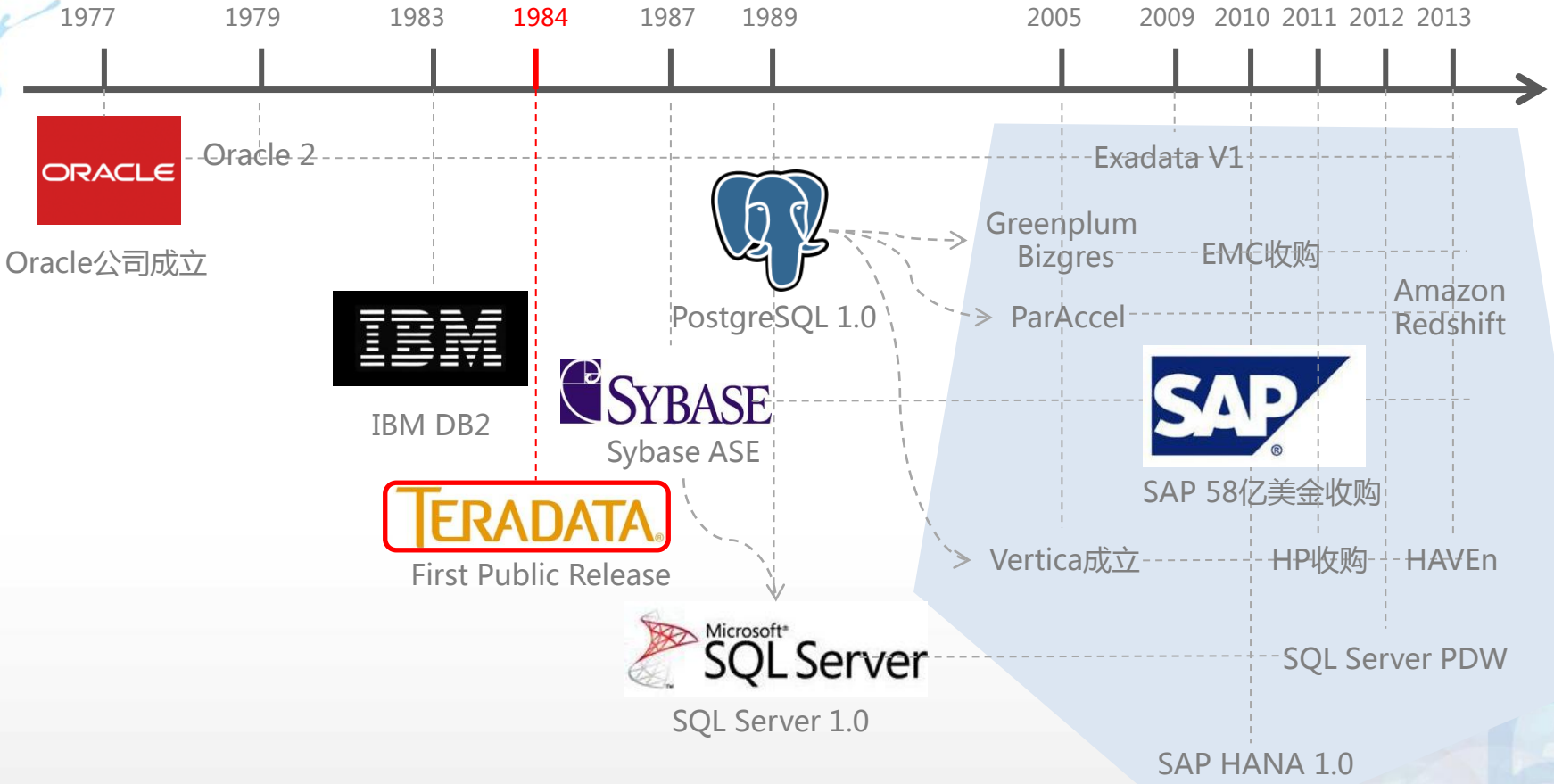
- High Throughput(大查询高吞吐 兼顾 小查询高并发)
- Function: Read(query、analysis、mining)
- ***Batch Data Processing vs. Interactive Data Analysis***
- SQLDB(Impala+hdfs,Mesa), SearchDB(ElasticSearch), MiningDB(R+?, Julia+?)
- ***Palo: a MPP-based Interactive Data Analysis SQL DB***
- ***Palo,a Google Mesa Clone,is simpler and better than Mesa.***

Batch Data Processing vs. Interactive Data Analysis

Big Data Lambda Architecture – Storm author



传统数据库研发逐渐从OLTP转向OLAP



大数据促使传统数据库领域格局发生变化，并购渐多

千禧年以前，数据库领域并购次数不多，金额也不大；千禧年以后，在OLAP领域美国与欧洲成立了许多小型创业公司，像Vertica、Vectorwise、ParAccell、Exasol都是其中的佼佼者，数据库领域呈现出前所未有的活跃气氛；2010年，SAP率先以58亿美金巨资收购Sybase，拉开领域并购狂潮；许多IT公司开始进行领域布局。

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，EMC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

传统数据库面临的问题

问题	描述
成本	✓ 传统分析型数据库往往采用一体机形式交付，维护、升级、扩容成本非常高
扩展性	✓ 数据增长速度比以往更快，数据量超出配额限制成为常态，要求数据库有很好的扩展性
可用性	✓ 支持automatic fail over，对用户透明，不影响用户查询性能
查询性能	✓ 传统数据库平均综合查询性能（复杂查询、即席查询、高并发小查询）达不到海量数据分析型应用，大数据查询性能往往要求提升10倍以上
数据加载性能	✓ 大数据处理必然会对数据加载速度有很高的要求，传统数据库的索引结构将不再适用
开源	✓ 开源会促进技术的透明化，让系统不断迭代，吸收大众之智慧。传统数据库大都闭源。

开源及互联网企业也抓紧布局

Hortonworks

Enterprise Hadoop | Products | Hadoop Training | Commu

The Stinger Initiative: Making Apache Hive 100 Times Faster

February 20th, 2013 Alan Gates

cloudera

Ask Bigger Questions

WHY CLOUDERA | PRODUCTS | SOLUTIONS | PARTNERS | RESOURCES | SUPPORT | ABOUT

Hadoop & Big Data

Cloudera Impala: Real-Time Queries in Apache Hadoop, For Real

by Marcel Kornacker & Justin Erickson | October 24, 2012 | 14 comments | Tweet

Search Images Maps Play YouTube News Gmail Documents More Andrew Brust

Google bigquery

COMPOSE QUERY

Query History

Job History

BigQuery Sandbox

MyDataSet

NYBabyNames

NameData

WordCounts

publicdata:samples

github_timeline

gsod

natality

shakespeare

trigrams

wikipedia

Compose Query

```
SELECT corpus, word_count FROM
publicdata:samples.shakespeare ORDER BY word_count DESC
LIMIT 200;
```

RUN QUERY | Query running (1.9s)...

Recent Queries

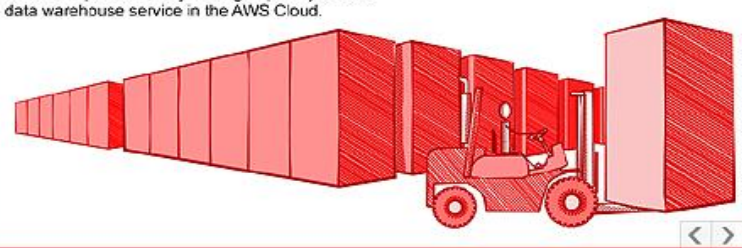
SELECT corpus, word_count FROM publicdata:samples.shakespeare ORDER BY word_count DESC LIMIT 200;	3:17pm
SELECT word, COUNT(word) AS wordcount FROM publicdata:samples.shakespeare WHERE word >= "A" AND word < "B" GROUP BY word HAVING COUNT(word) > 10 ORDER BY word LIMIT 10;	12:53pm
SELECT word, COUNT(word) AS wordcount FROM publicdata:samples.shakespeare WHERE word >= "A" AND word < "B" GROUP BY word HAVING COUNT(word) > 100 ORDER BY word LIMIT 10;	12:53pm

Apache Drill Distributed system for interactive analysis.

Apache Drill (incubating) is a distributed system for interactive analysis of large-scale datasets, based on Google's Dremel. Its goal is to efficiently process nested data. It is a design goal to scale to 10,000 servers or more and to be able to process petabyes of data and trillions o records in seconds.

Introducing Amazon Redshift

A fast and powerful, fully managed petabyte-scale data warehouse service in the AWS Cloud.



MemSQL, The Real-Time Analytics Platform.

MemSQL's real-time analytics platform is built on the world's fastest, most scalable in-memory database, capable of simultaneously handling real-time transactions and analytic workloads. MemSQL unleashes the full potential of Big Data by consuming and returning data instantly.

Shark: Real-time queries and analytics for big data

Print

Listen

Shark is 100X faster than Hive for SQL, and 100X faster than Hadoop for machine-learning

by Ben Lorica | @bigdata | Comment | November 27, 2012

Mesa: Geo-Replicated, Near Real-Time, Scalable Data Warehousing

Ashish Gupta, Fan Yang, Jason Govig, Adam Kirsch, Kelvin Chan
Kevin Lai, Shuo Wu, Sandeep Govind Dhoot, Abhilash Rajesh Kumar, Ankur Agiwal
Sanjay Bhansali, Mingsheng Hong, Jamie Cameron, Masood Siddiqi, David Jones
Jeff Shute, Andrey Gubarev, Shivakumar Venkataraman, Divyakant Agrawal
Google, Inc.

ABSTRACT

Mesa is a highly scalable analytic data warehousing system that stores critical measurement data related to Google's ness critical nature of this data result in unique technical and operational challenges for processing, storing, and querying. The requirements for such a data store are:

目录

- 做Palo的背景
- Palo整体架构
- Palo关键技术
- 与竞品的比较
- 我想使用Palo

设计原则与定位

- **至繁归于至简** - 苹果公司产品理念
- 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

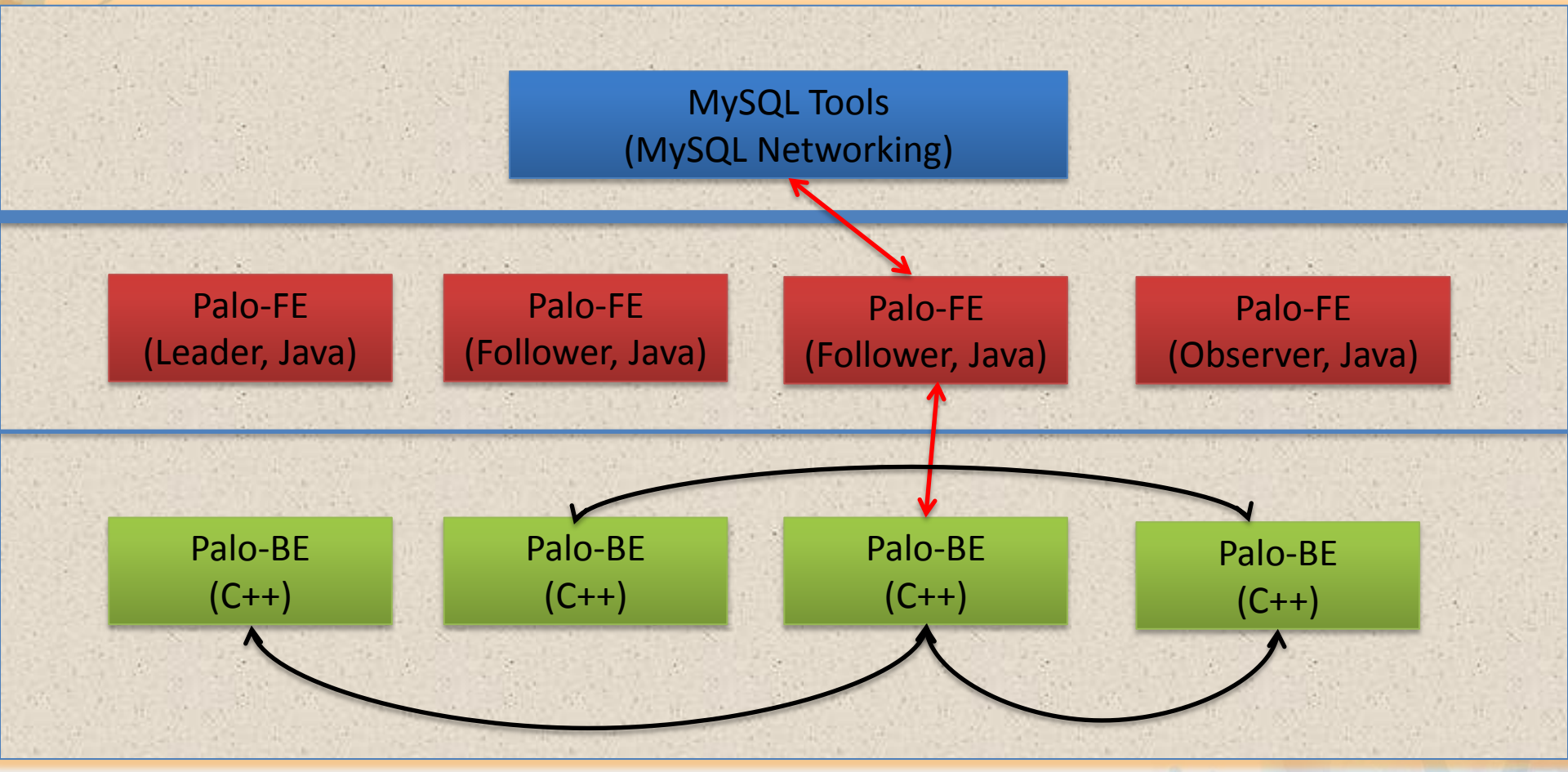
高查询性能

10W QPS/ 100GB/s

高加载性能

10 TB / Hour

实现架构

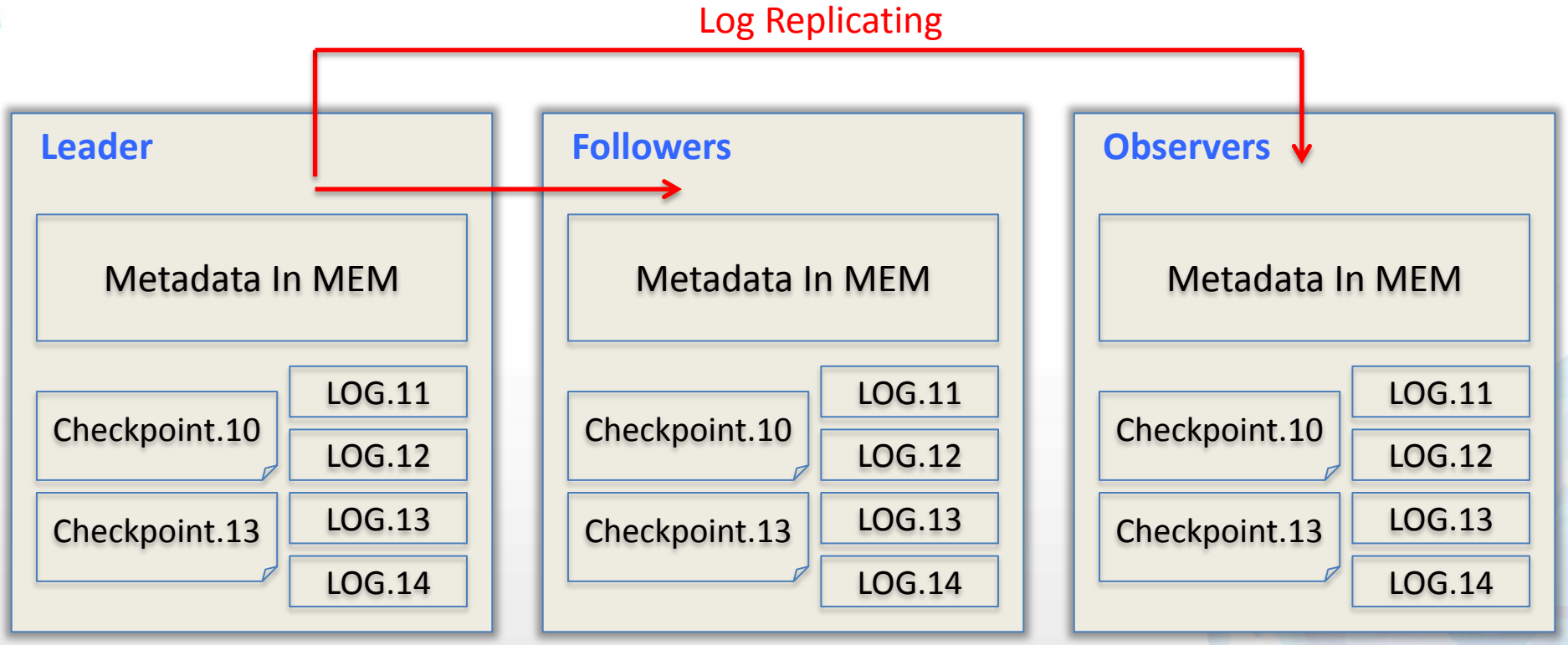

MySQL Tools
(MySQL Networking)
Palo-FE
(Leader, Java)
Palo-FE
(Follower, Java)
Palo-FE
(Follower, Java)
Palo-FE
(Observer, Java)
Palo-BE
(C++)
Palo-BE
(C++)
Palo-BE
(C++)
Palo-BE
(C++)

目录

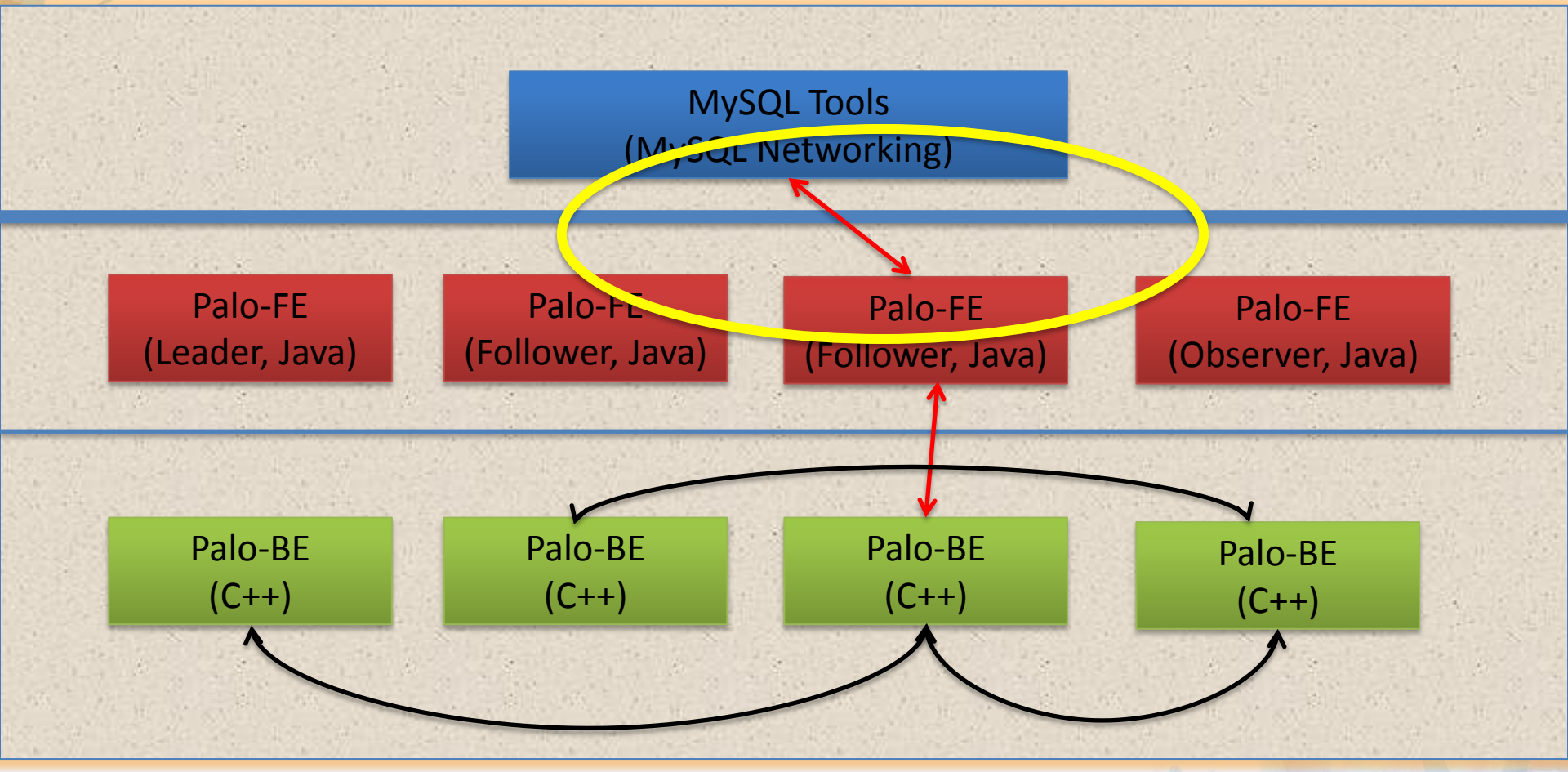
- 做Palo的背景
- Palo整体架构
- Palo关键技术
- 与竞品的比较
- 我想使用Palo

Frontend Metadata Management

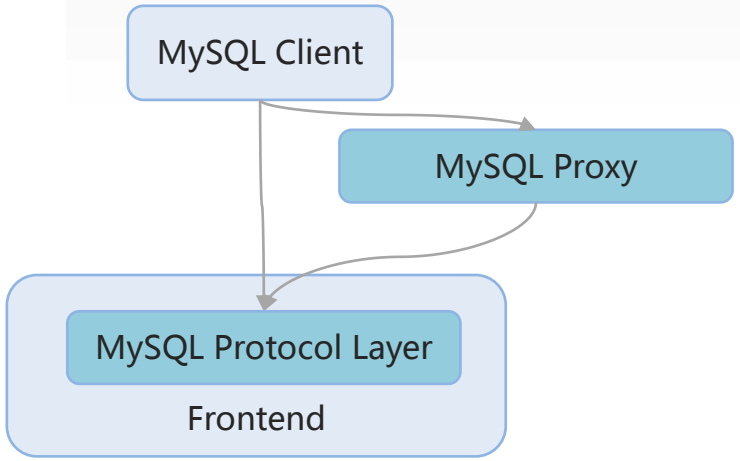
- State Machine + Replicated Log
- 类似Raft协议思想



MySQL Networking Protocol

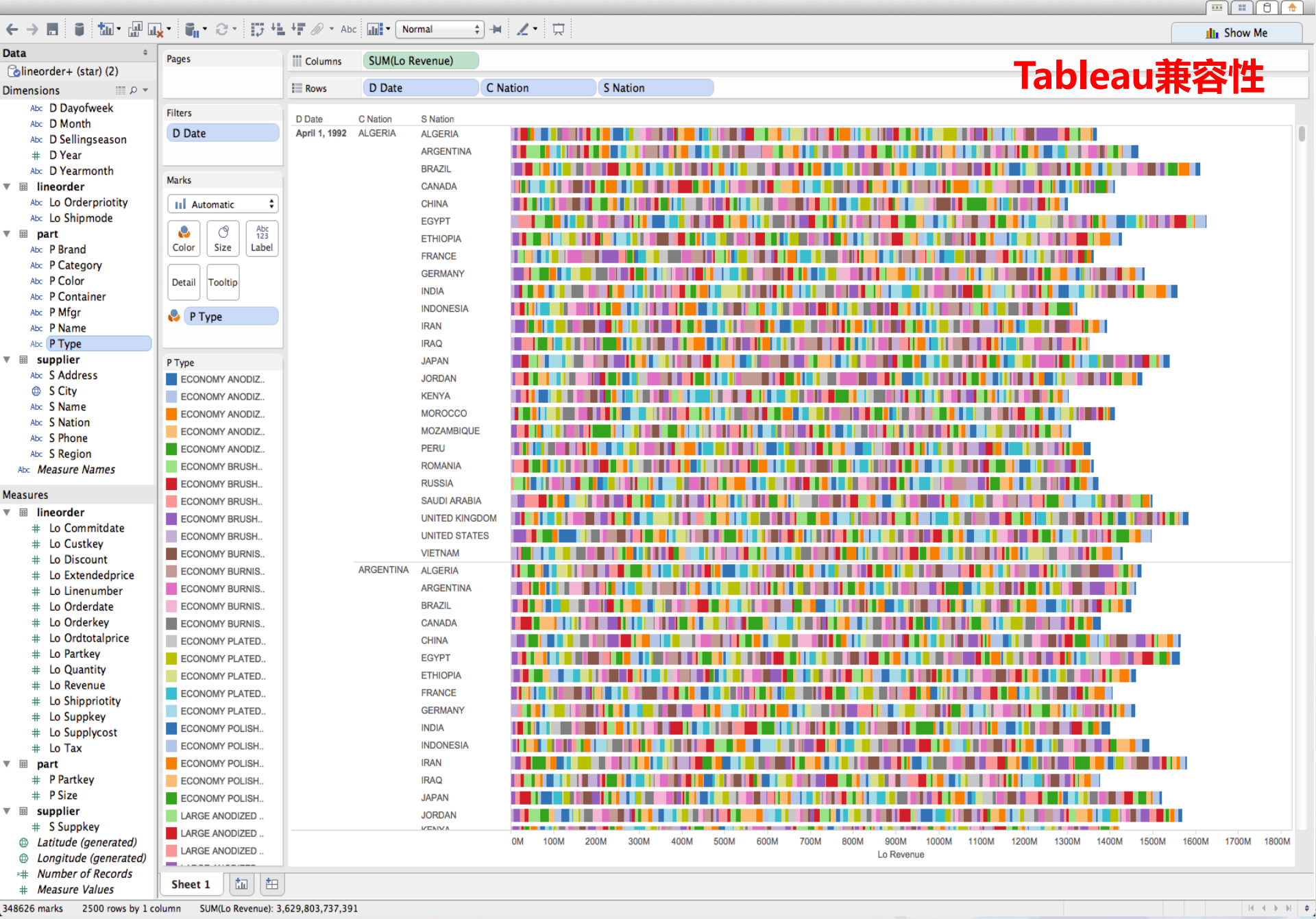


```
test@mry-laptop:~$
test@mry-laptop:~$ mysql -h tc-inf-devop01.tc.baidu.com -P 8276 -u maruyue
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 0
Server version: 4.1.2 (Powered by Palo 2.0 Beta)

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)

mysql>
```



- ✓ 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

Tableau兼容性

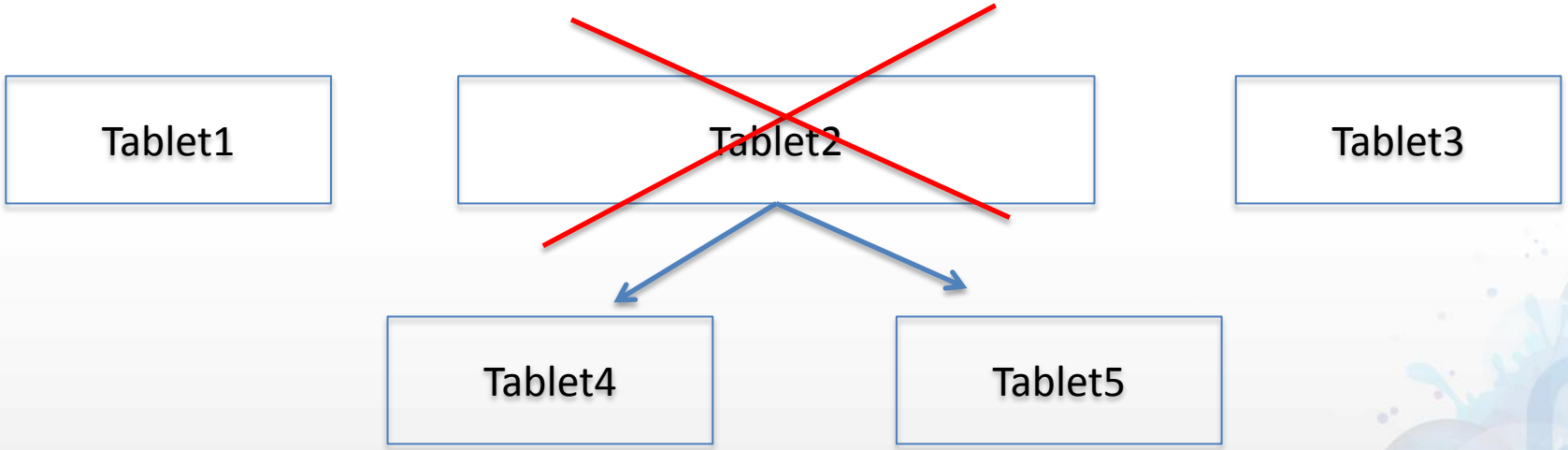


R语言兼容性

```
➜ 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
>
```

Elastic Range Partition

- 支持Hash Partition
- 也支持一种Elastic Range Partition



Palo Storage Design

Keys

Values

Date	PublisherId	Country	Clicks	Cost
2013/12/31	100	US	10	32
2014/01/01	100	US	205	103
2014/01/01	200	UK	100	50

(a) Mesa table A

Date	AdvertiserId	Country	Clicks	Cost
2013/12/31	1	US	10	32
2014/01/01	1	US	5	3
2014/01/01	2	UK	100	50
2014/01/01	2	US	200	100

(b) Mesa table B

Base表

AdvertiserId	Country	Clicks	Cost
1	US	15	35
2	UK	100	50
2	US	200	100

(c) Mesa table C

Rollup表

Figure 1: Three related Mesa tables

引自Google Mesa Paper

Values聚合方式Sum, Replace

Date	PublisherId	Country	Clicks	Cost
2013/12/31	100	US	+10	+32
2014/01/01	100	US	+150	+80
2014/01/01	200	UK	+40	+20

(a) Update version 0 for Mesa table A

Date	AdvertiserId	Country	Clicks	Cost
2013/12/31	1	US	+10	+32
2014/01/01	2	UK	+40	+20
2014/01/01	2	US	+150	+80

(b) Update version 0 for Mesa table B

Delta更新

Date	PublisherId	Country	Clicks	Cost
2014/01/01	100	US	+55	+23
2014/01/01	200	UK	+60	+30

(c) Update version 1 for Mesa table A

Date	AdvertiserId	Country	Clicks	Cost
2013/01/01	1	US	+5	+3
2014/01/01	2	UK	+60	+30
2014/01/01	2	US	+50	+20

(d) Update version 1 for Mesa table B

Figure 2: Two Mesa updates

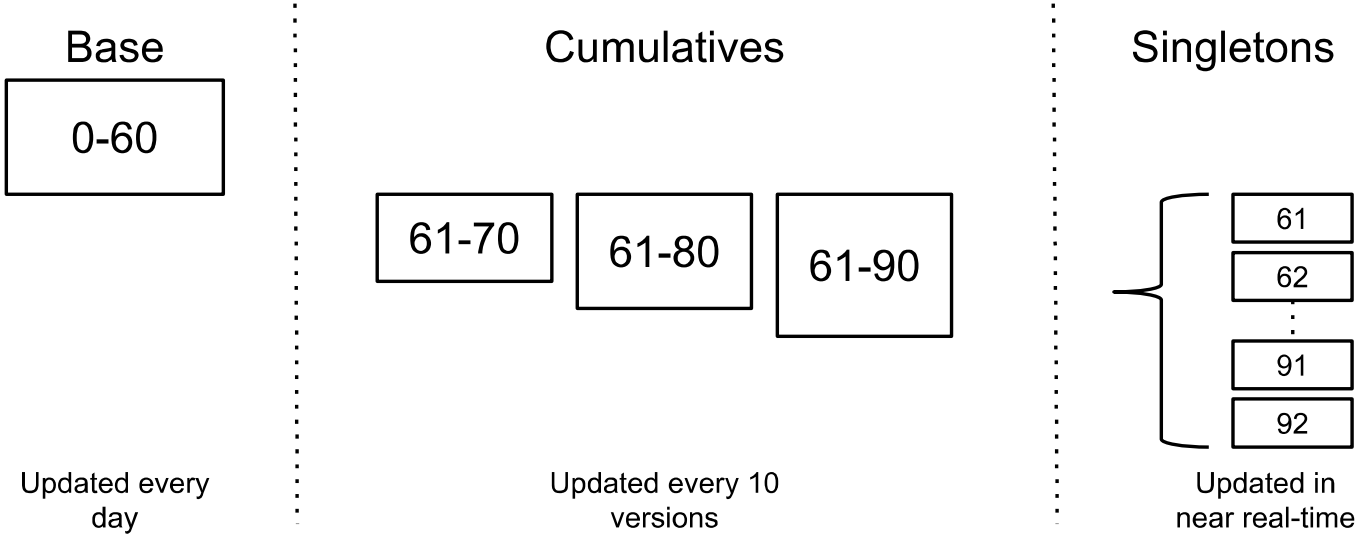
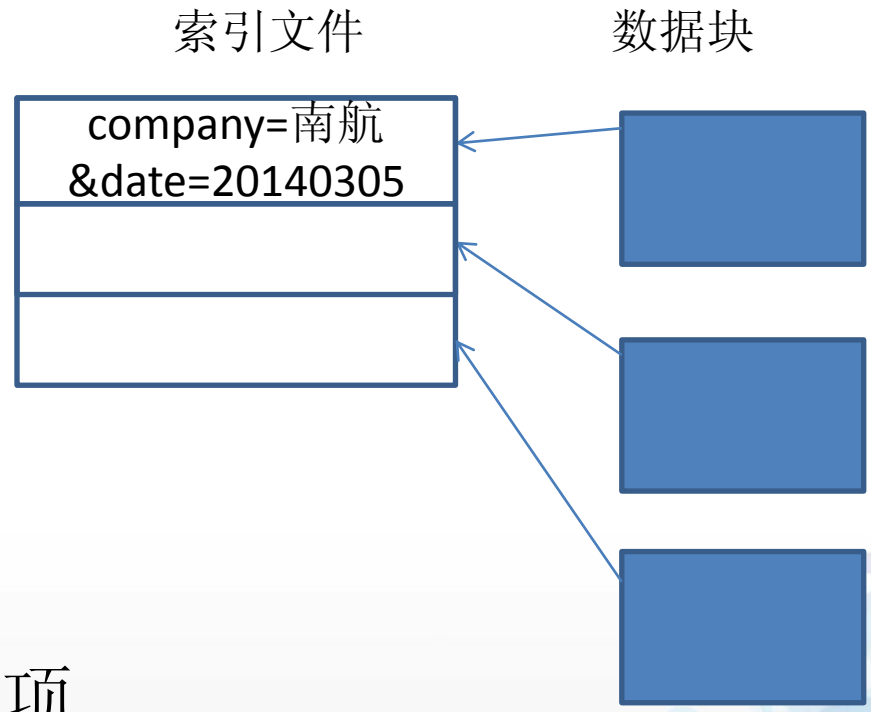


Figure 3: A two level delta compaction policy

引自Google Mesa Paper

Palo Storage Format – 行列存储

- 数据块存储
  - 每个块含256行
  - 块内部列存储
  - 块整体压缩
- 稀疏索引
  - 索引常驻内存
  - 每个块对应一个索引项
  - 索引项只保存key的前几列 – short key



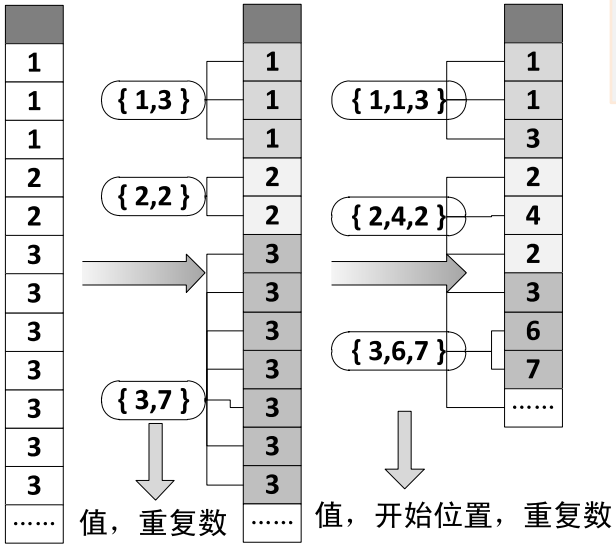
Palo Storage Format – 列式存储

SSN	Name	Age	Addr	City	St
101259797	SMITH	88	899 FIRST ST	JUNO	AL
892375862	CHIN	37	16137 MAIN ST	POMONA	CA
318370701	HANDU	12	42 JUNE ST	CHICAGO	IL

101259797 | 892375862 | 318370701 | 468248180 | 378568310 | 231346875 | 317346551 | 770336528 | 277332171 | 455124598 | 735885647 | 387586301

Block 1

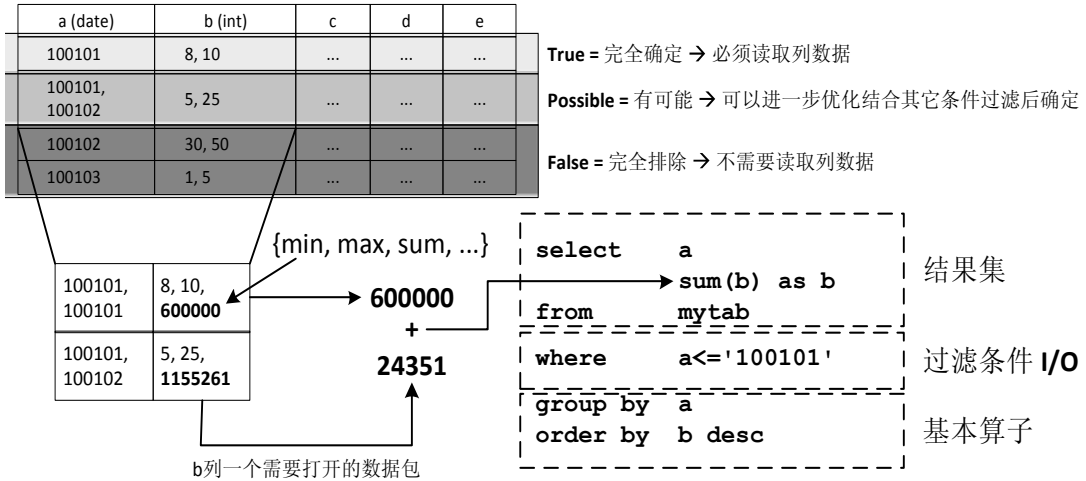
- ✓ 数据按列存储，每一列单独存放
- ✓ 只访问查询涉及的列，大量降低I/O
- ✓ 数据类型一致，方便压缩
- ✓ 数据包建索引，数据即索引



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



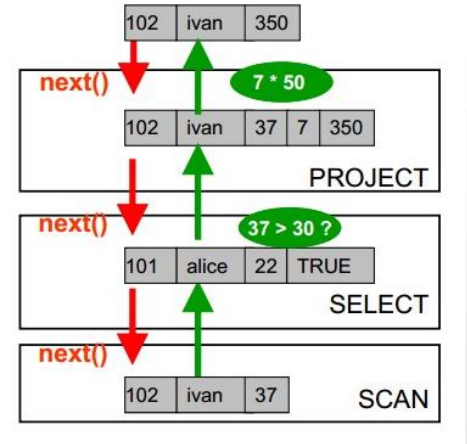
高并发小查询

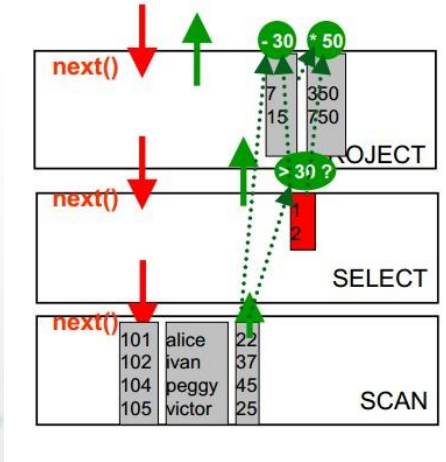
- 多FE节点解析（非Leader单调一致性，SYNC原语）
- 执行规划时的Partition Pruning
- 运行时的Partition Pruning(还未开展)
- Sorted，带有智能索引的存储格式
- Rollup表
- Delta更新策略

高性能

- 启发式预聚合
- 谓词下推，复杂谓词下推
- 向量化执行

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

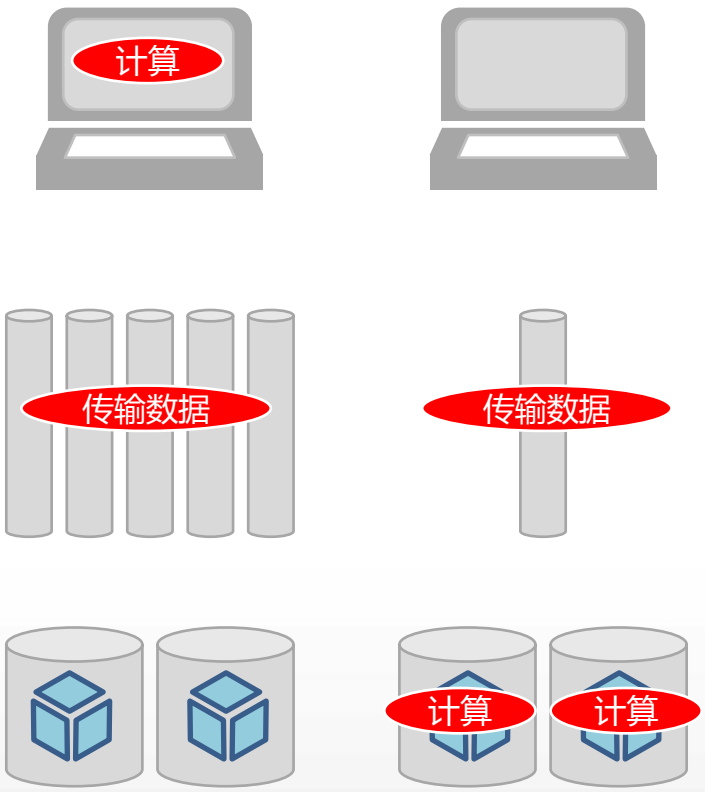




容错、稳定性

- 多副本存储，副本自动修复
- 多硬盘支持，硬盘自动容错
- 查询时副本切换机制
- 数据包序列号验证
- 黑名单机制，快速屏蔽宕机机器
- 服务器结果延迟发送

In-Database Analysis



time-stamp	userid
10:00:00	238909
00:58:24	7656
10:00:24	238909
02:30:33	7656
10:01:23	238909
10:02:40	238909

(a) Raw click data

time-stamp	userid	session
10:00:00	238909	0
10:00:24	238909	0
10:01:23	238909	0
10:02:40	238909	1
00:58:24	7656	0
02:30:33	7656	1

(b) Click data with session information

- UDF
- UDAF
- UDTF

```
SELECT ts, userid, session
FROM sessionize (
        ON clicks
        PARTITION BY userid
        ORDER BY ts
        TIMECOLUMN ('ts')
        TIMEOUT (60)
        );
```

其它关键技术

- 批量数据导入的原子更新，MVCC
- Schema Change/Create Rollup/Data Recovery
- Shared-nothing,MPP
- 自动扩展和收缩
- 基于Hadoop的分布式导入系统

TODO 技术点

名称	示例

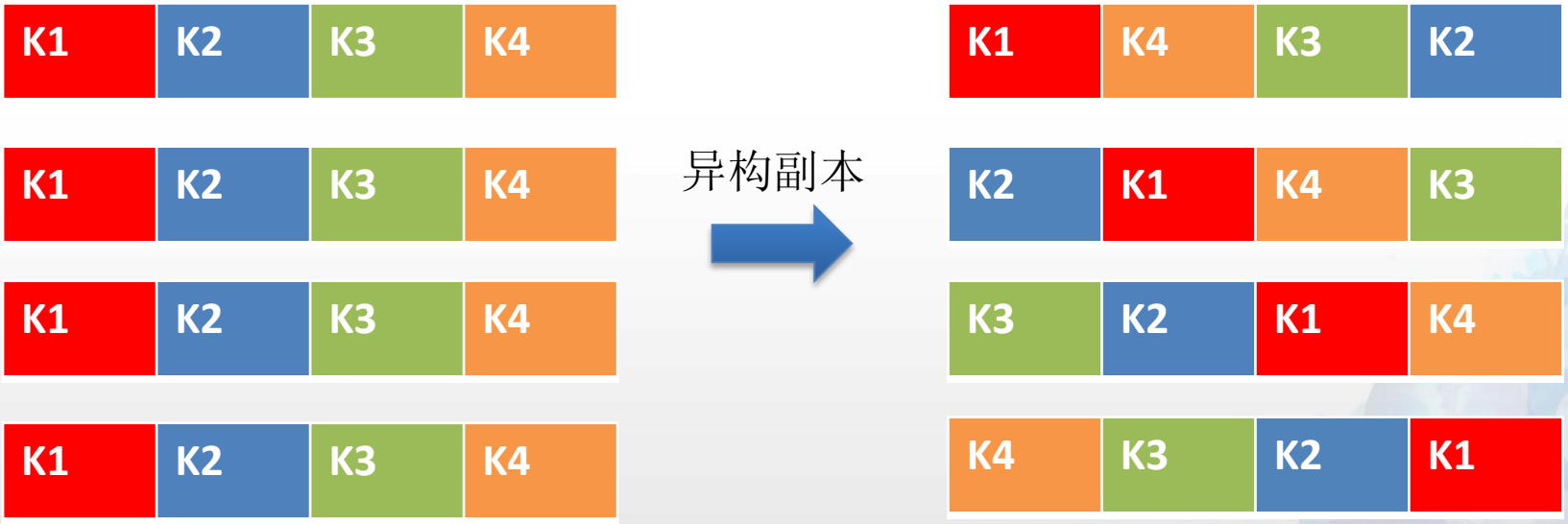
SQL-DAG & Multi-SQL执行	// 创建内存临时表 mt
	create memtable mt as select * from table1 where url = “http://test.com”;
	// 返回按照省份聚合的pv数据
	select province, sum(pv) from mt group by province;
	// 返回按照浏览器类型聚合的pv数据
	select browser, sum(pv) from mt group by browser;

代码执行速度优化	• 使用llvm进行运行时的代码生成 • 计算的更多向量化执行
复杂分析计算层	• 复杂分析层 – 类R、Matlab和Python的分布式科学计算语言、常用分析工具包（矩阵计算、统计分析、机器学习、信号处理）
嵌套数据类型	• 引入Array、Map、Struct select userid from user where user_tags contains（‘单身’，‘IT’）;

user_id	user_tags
101	[“单身”，“IT”，“上地”]

TODO 技术点

名称	示例
存储格式统一	Column Group Storage Format 列组间为纯列式存储，列组内为行列式存储 一个列可以在多个列组出现，支持动态修改
节点、Disk分组	对节点和Disk可以分组，对表可以指定各个副本需要放置的组 通过此类功能，比如可以使得最近数据的一个副本放置到SSD上，加快查询



目录

- 做Palo的背景
- Palo整体架构
- Palo关键技术
- 与竞品的比较
- 我想使用Palo

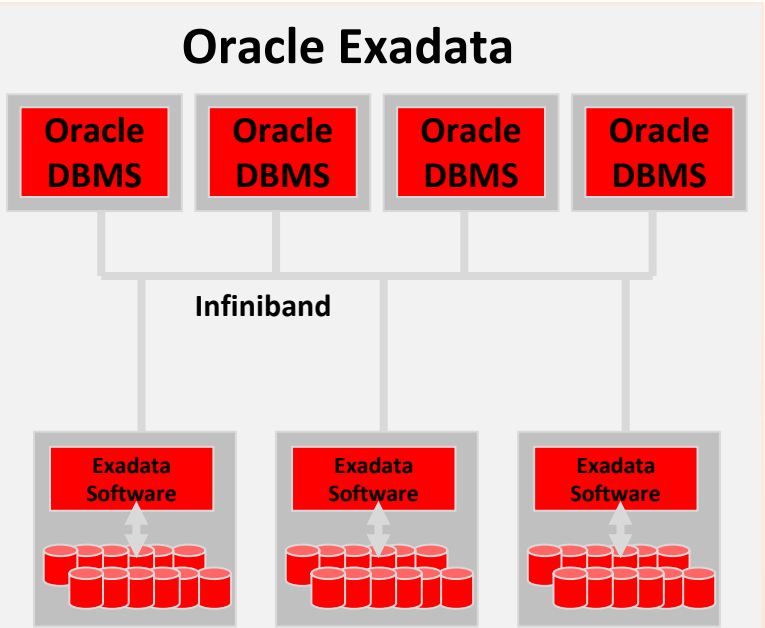
Palo vs. Oracle Exadata

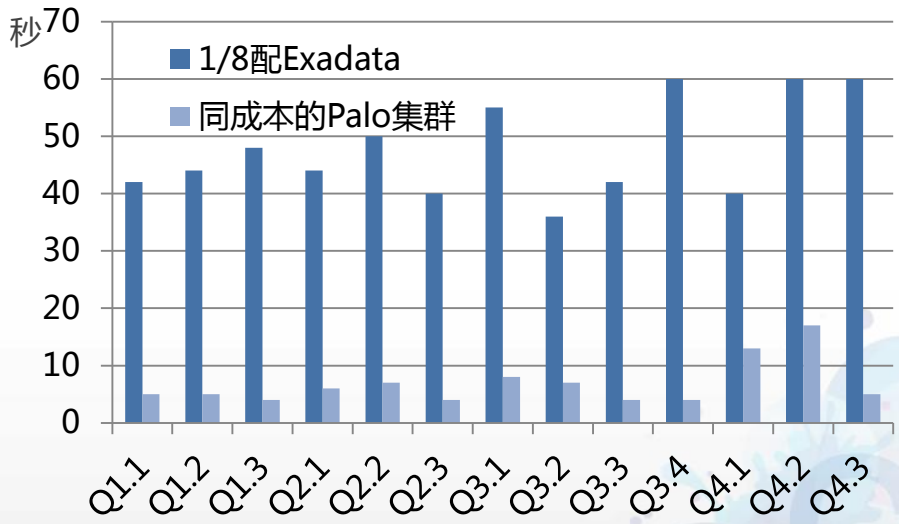
Exadata

- ✓ Shared-Disk架构
- ✓ 一体机，无法扩容，无法利用最新硬件技术
- ✓ 导入速度非常慢

Palo

- ✓ Shared-Nothing、MPP架构
- ✓ X86-64服务器，可以使用当前最新硬件技术
- ✓ 导入速度非常快





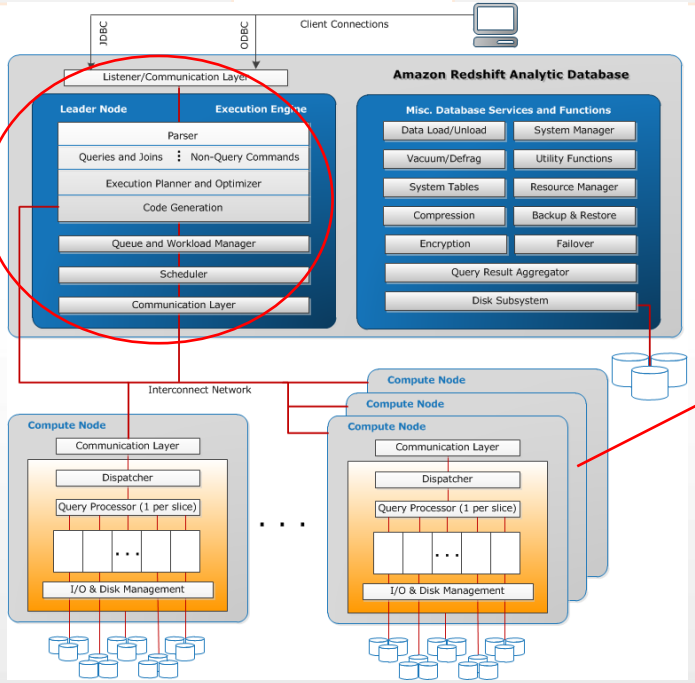
Star Schema Benchmark，

性能是Exadata的7倍。

Palo vs. Amazon Redshift

Amazon Redshift	Palo
✓ 只有一个Leader Node负责接受SQL查询和协调Compute Node执行	✓ 所有前端节点都可以接受连接，并执行SQL解析和规划。前端节点可以在线自由扩展
✓ 扩容采用copy一个新集群的方式，并且在扩容过程中，要停止写入数据	✓ 扩容可以在旧集群上透明进行，不影响任何读写操作

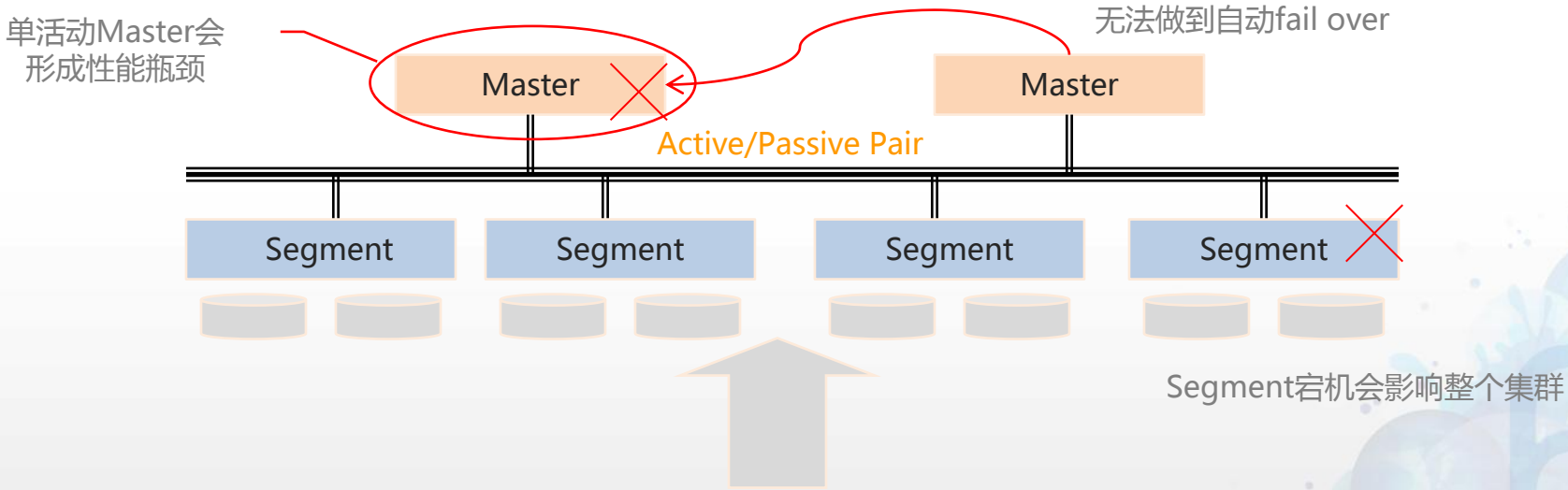
只有一个
Leader Node



扩容要停止数据写入

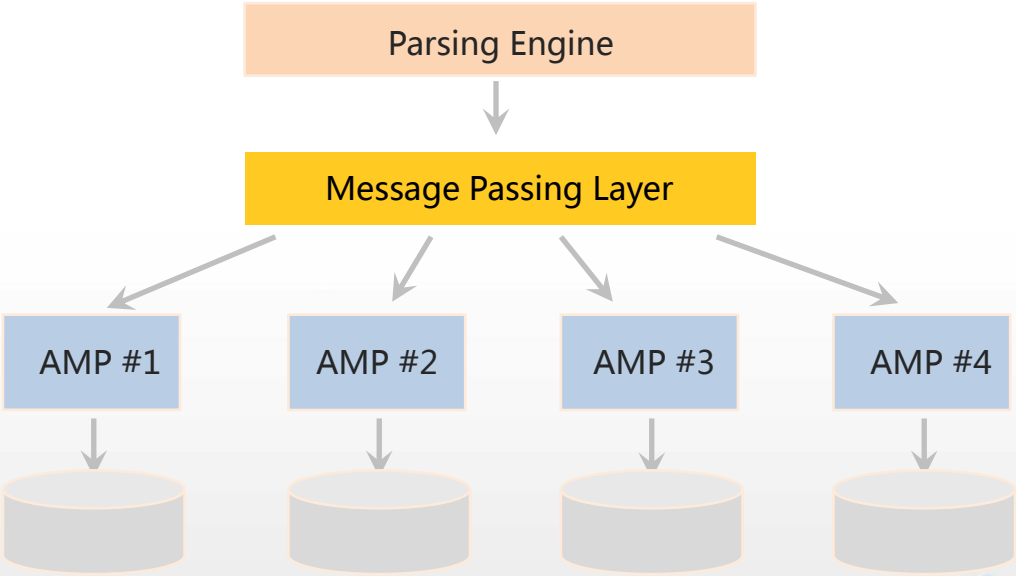
Palo vs. EMC Greenplum

EMC Greenplum	Palo
✓ 单Master设计，可用性切换方案复杂，造成读可用性较低	✓ 单Master设计，多前端节点设计，读可用性很高
✓ 只有Master可以接受连接，并执行SQL解析和规划，是全系统的性能瓶颈面	✓ 所有前端节点都可以接受连接，并执行SQL解析和规划，前端节点可以在线自由扩展
✓ 数据修复采用Mirror方式，原始并低效	✓ 数据修复使用全部机器修复，修复效率很高



Palo vs. Teradata

Teradata	Palo
✓ Shared-Nothing架构，但扩展有上限，扩容升级需要专业的CS人员参与进行	✓ Shared-Nothing架构，扩展无理论上限，而且非常容易
✓ 性能很好，价格太贵，单节点（100w美金？）	✓ 性能很好，性价比高，单节点(5w人民币?)
✓ 需要专业的PS团队辅助建库、开发和运维	✓ 开放系统架构，不需要专业PS团队辅助开发
✓ 一体机方式提供，云数据库方案不成熟	✓ 成熟的云数据库解决方案



目录

- 做Palo的背景
- Palo整体架构
- Palo关键技术
- 与竞品的比较
- 我想使用Palo

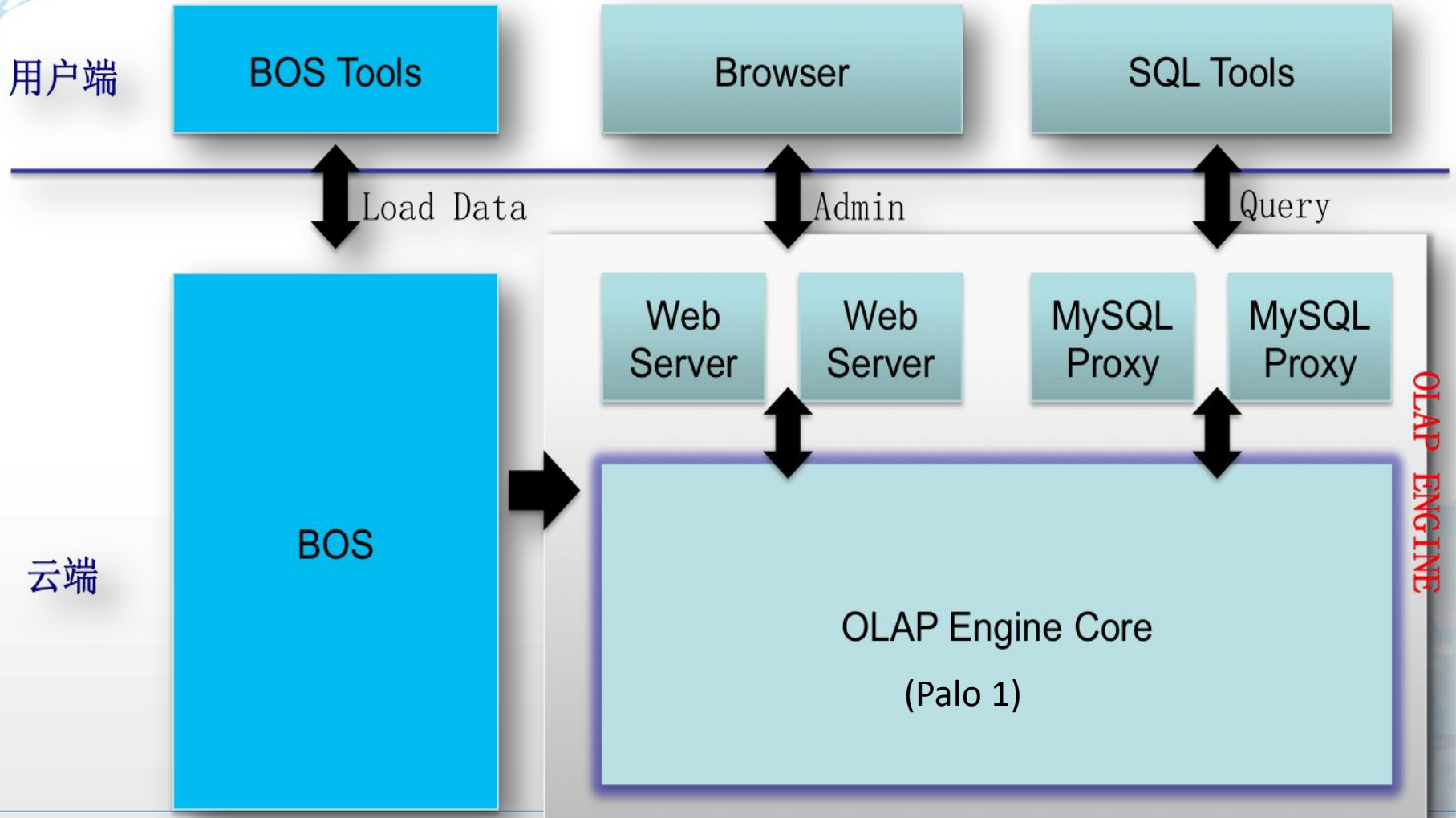
多种部署方式

- Standalone
- Cloud
  - Software as a Service
  - Public Cloud & Private Cloud

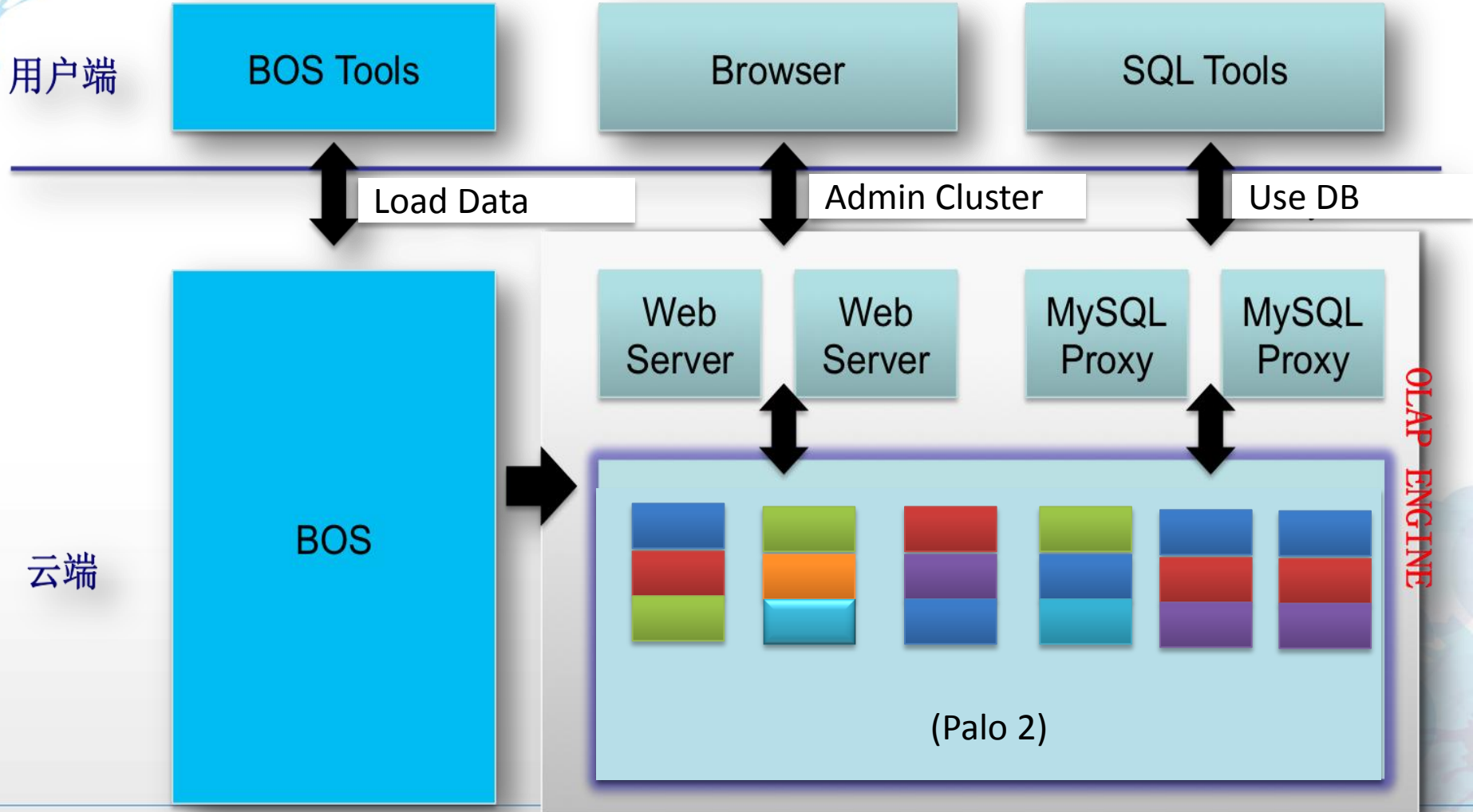
Palo-based Cloud Product – OLAP Engine

- http://factory.baidu.com/olapengine

- OLAP Engine Alpha – 2014.9
- BigQuery模式



- OLAP Engine Beta – 2015.5
- Redshift模式



1976
Apple I

1977
Apple][

- OLAP Engine GA – 2015.12
- Open Sourcing Palo and OLAP Engine – 2015~2016
- 商业合作
  - 百度已经提供功能部分受限的云端Palo供大家申请试用。
  - 如果您急需此类技术，云端产品又不能解决您的问题，欢迎与我们联系。

Q&A

THANKS

SequeMedia 盛拓传媒 | IT168 www.it168.com | ChinaUnix | ITPUB