



平民软件  
OneXSoft

# Spark over OneProxy/PostgreSQL

楼方鑫



# 简介

- 个人
  - 约20年数据库相关经验，年青心态
  - 程序员 -> DBA -> 部门主管/架构师 -> (?)
  - 小公司 -> eBay -> 阿里巴巴(支付宝) -> 平民软件
  - 个人产品：SQLULDR2、AUL/MyDUL
- 公司
  - 数据相关基础技术产品和方案设计
  - 产品介绍
    - OneSQL：深度定制的MySQL版本
    - OneProxy：基于各种数据库协议的中间件（Oracle/SQL Server/MySQL/PostgreSQL/Redis）

# 现有方案

- 数据库
  - Oracle RAC
  - Oracle Exadata
- Storm
- MPP
  - Greenplum
- 大数据
  - Hadoop/HBase + Hive
  - Hadoop/HBase + Spark

# 数据库

- 计算能力有限
  - 数据量增长过快
  - 计处需求越来越多
- 成本过高
  - 存贮依赖
  - License依赖

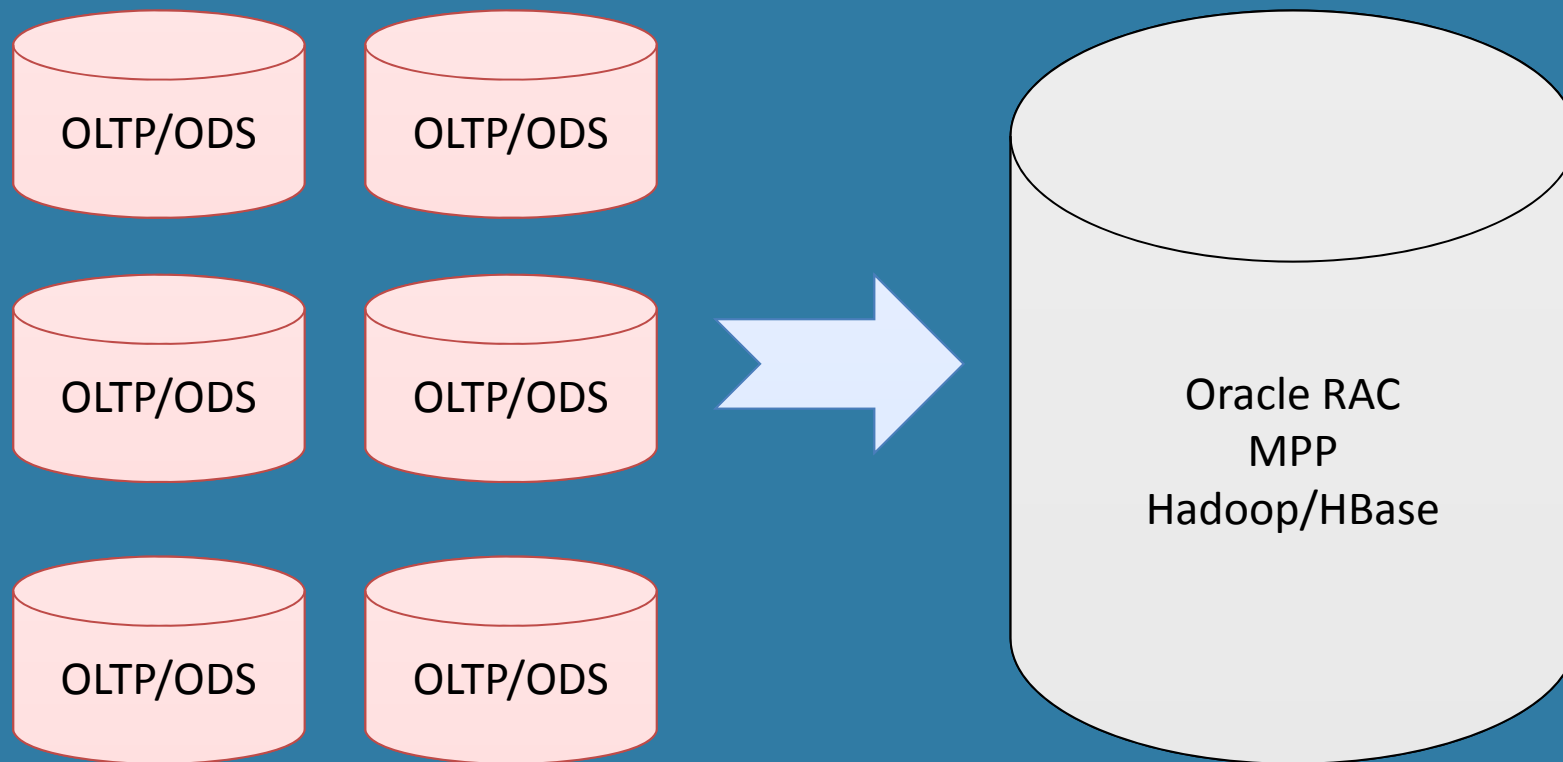
# Storm

- 数据分析的特殊形式
- 常态化成本比较高

# MPP/大数据

- 并发能力
  - 小查询难满足
  - 数据更新不灵活
- 数据同步
  - 须有数据落地层
  - 须有数据装载过程

# 双份数据



# 用户场景

- 大量对象
  - 数十万个监控对象
- 秒级数据
  - 每秒或几秒钟生成大量数据
- 实时计算
  - 对最近的数据做实时计算
  - 对历史数据做汇总分析



# RowKey

- 时间优先
  - 所有的写入集中于少数节点
  - 按时间范围查询无法充分并行
- 对象优先
  - 按时间范围查询需扫描所有数据
- 哈希均衡
  - 无法做分区过滤，影响查询效率

# 三大要素



# 条件过滤

- 分区方式
  - 按时间组织数据
- 第二索引
  - 加速多唯度查询
- 条件下放
  - 底层过滤数据

# 充分并行

- 数据分布
  - 数据均衡地分布到所有机器
- 条件下放
  - 根据条件进行分片过滤

# 计算下放

- 过滤操作
  - 在数据存贮层过滤
- 关联操作
  - 大表同唯度拆分
  - 小表全结点复制
- 计算操作
  - `Select count(*) from big_table`

# OneProxy

- OLTP中间件
  - 数据路由，故障检测等
- 超高性能
  - 单实例40W QPS（C/C++编写）
  - 并行查询
- 灵活数据分片机制
  - Range/List/Hash/Composite
  - 小表广播

# 分片算法

- Global
  - 全局表，每个节点留一份
- Range
  - 按某个字面的范围进行分表
- List
  - 按某个这段的值进行分表
- Hash
  - 按某个字段的值的Hash值进行分表（OneProxy算法）
  - 数字类型，hash值等于数值
  - 分区数取模
- Crc32
  - 同Hash分区，Hash值算法为标准的CRC32
  - 数字类型，hash值等于数值
  - 分区数取模
- 复合分区

# 分片配置

```
{  
  "table" : "my_range",  
  "pkey"  : "id",  
  "type"  : "int",  
  "method" : "range",  
  "partitions":  
  [  
    { "suffix" : "_0", "group": "server1", "value" : 100000 },  
    { "suffix" : "_1", "group": "server2", "value" : 200000 },  
    { "suffix" : "_2", "group": "server3", "value" : 300000 },  
    { "suffix" : "_3", "group": "server4", "value" : null   }  
  ]  
}
```





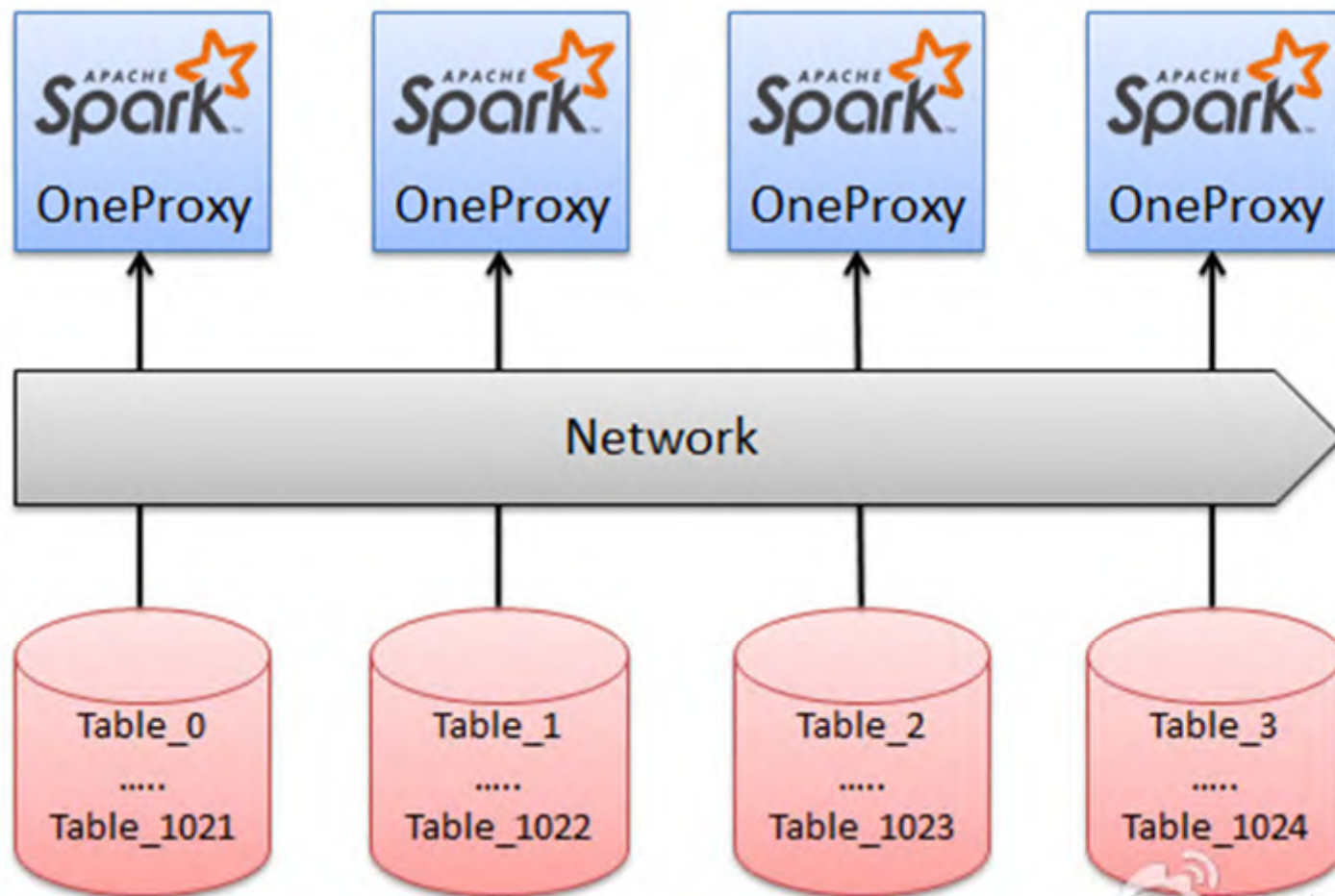
# 协议支持

功能	MySQL	PostgreSQL	SQL Server	Oracle
连接池	支持	支持	支持	支持
读写分离	支持	支持	支持	支持
分库分表	支持	支持		
防火墙	支持	支持	支持	支持
SQL Audit	支持	支持	支持	支持
SQL Spy	支持	支持	支持	支持
.....				

# Spark

- 分布式计算框架
- Spark-SQL接口
- 数据分析算法
  - 集成MLIB/GraphX/R
- 支持JDBC数据源
  - 分区管理较弱，可依赖OneProxy
- Hot Development

# Spark over OneProxy

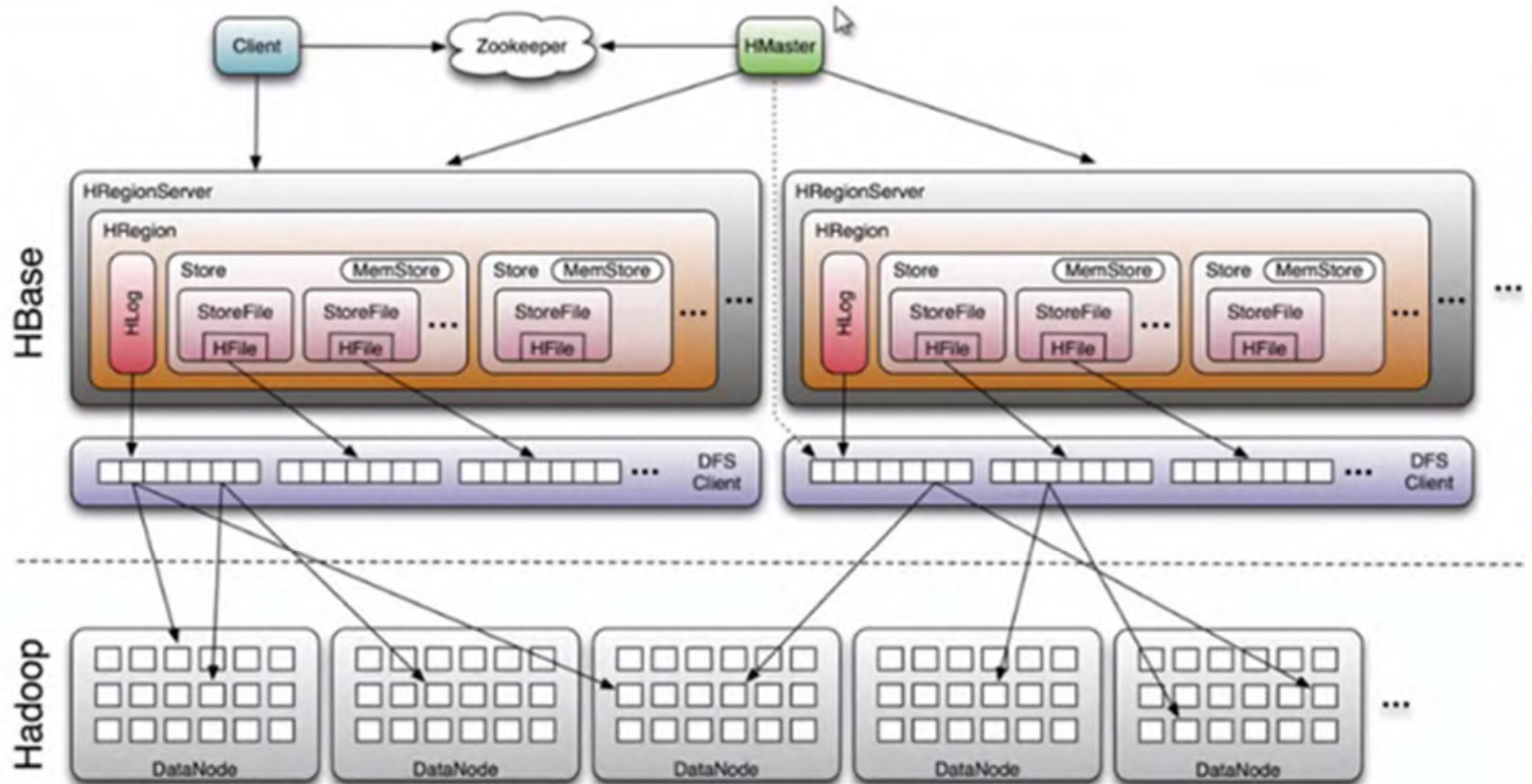


Postgre

@平民架构  
weibo.com/dbatools

QL

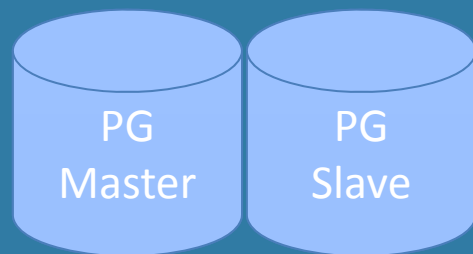
# Hadoop/HBase



# 单份数据



OneProxy



数据分析

分片、更新  
实时查询

数据存贮

# 条件过滤

- 分区方式（OneProxy）
  - Hash + Range复合分区
- 第二索引（PostgreSQL）
  - 灵活的数据库索引
- 条件下放（Spark）
  - Spark下放Where条件

# 充分并行

- 数据分布（OneProxy）
  - Hash + Range数据打散
  - 主备读写分离
- 条件下放
  - 条件下放（Spark）
  - 分区过滤（OneProxy）

# 计算下放

- 过滤
  - 数据库层过滤记录
- 关联（OneProxy）
  - 大表同唯度拆分
  - 小表全结点复制
- 计算（人工）
  - 用SQL语句做DataFrame
  - SQL语句（Join、分组汇总）



# 单份数据



# Spark Connector

- 分区管理
- 接口介绍
- 简单测试

# 接口介绍

- 获得连接
- 获得结构
- 根据SQL返回RDD
- 根据SQL返回DataFrame
- 根据表名返回DataFrame

# 测试表

- 表名：my\_hash4
- 分区：根据ID哈希做256个分区
- 自增：tid自增作分表主键

# 获得连接

- 类名：com.onexsoft.oneproxy.SparkConn
- 方法：
  - 连接OneProxy for MySQL : mysql
  - 连接OneProxy for PostgreSQL : pgsql

```
scala> import com.onexsoft.oneproxy.SparkConn
import com.onexsoft.oneproxy.SparkConn

scala> val oneproxy = SparkConn.mysql("127.0.0.1:3307", "test", "test")
oneproxy: org.apache.spark.sql.OneProxyContext = org.apache.spark.sql.OneProxyContext@c0c8f96
```

# 获得结构

- 根据SQL语句，取得字段结构，以方便从RDD构建DataFrame对象

```
scala> val schema = oneproxy.desc("select col8 from my_hash4_1 where 1=0")  
schema: org.apache.spark.sql.types.StructType = StructType(StructField(col8, StringType, true))
```

# 返回RDD

- 根据SQL语句获得RDD对象
- 自动从OneProxy获得分区信息

```
scala> val myrdd = oneproxy.query(spark, "select * from my_hash4 where tid < 100")
myrdd: org.apache.spark.api.java.JavaRDD[org.apache.spark.sql.Row] = ONEPROXYQUERYRDD[0] at query at <console>:26

scala> myrdd.count
[Stage 0:>                                     (0 + 0) / 25

res0: Long = 25344
```

# 返回DataFrame(1)

- 根据SQL语句和表结构返回DataFrame

```
scala> val mydf = oneproxy.openQuery(spark, "select * from my_hash4 where tid  
< 100", schema)  
mydf: org.apache.spark.sql.DataFrame = [tid: int, id: string ... 22 more field  
s]  
  
scala> mydf.count  
[Stage 0:> (0 + 0) / 25  
[Stage 0:> (0 + 24) / 25  
[Stage 0:=====> (91 + 24) / 25  
[Stage 0:=====> (203 + 24) / 25  
  
res0: Long = 25344
```



# 返回DataFrame(2)

- 根据表名返回DataFrame

```
scala> val mydf = oneproxy.openTable(spark, "my_hash4")
mydf: org.apache.spark.sql.DataFrame = [tid: int, id: string ... 22 more fields]

scala> mydf.createOrReplaceTempView("my_hash4")

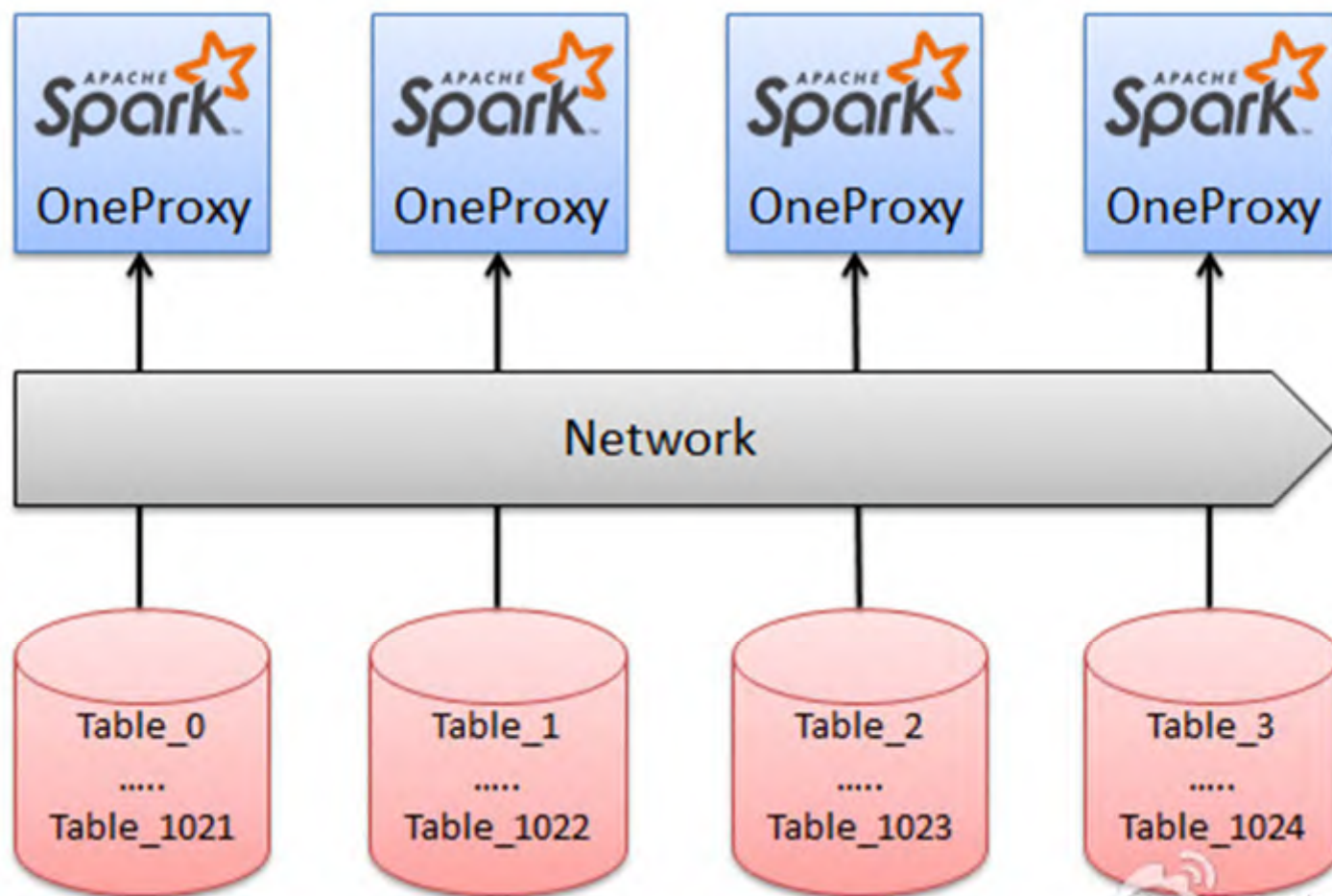
scala> spark.sql("select count(*) from my_hash4 where tid < 100").collect
res1: Array[org.apache.spark.sql.Row] = Array([25344])
```

# 计算下放

- Spark计算能力下放不够好
- 可以使用自定义SQL来构造DataFrame

```
/* group=shixun2 direct */ SELECT 1 FROM my_hash4_253 my_hash4 WHERE (tid < 100)
SET character_set_results = NULL
SET character_set_results = NULL
/* group=shixun1 direct */ SELECT 1 FROM my_hash4_252 my_hash4 WHERE (tid < 100)
/* group=shixun3 direct */ SELECT 1 FROM my_hash4_254 my_hash4 WHERE (tid < 100)
/* group=shixun1 direct */ SELECT 1 FROM my_hash4_255 my_hash4 WHERE (tid < 100)
```

# 集群直连



Postgre

@平民架构  
weibo.com/dbatools

QL

# 总结

- 优势

- 数据共享
- 均衡分片
- 灵活查询
- 多唯索引

- 劣势

- 自动扩容（面向OLTP的缺陷）
- 执行计划（不够优化，Spark在努力）

- Thanks!

Q&A