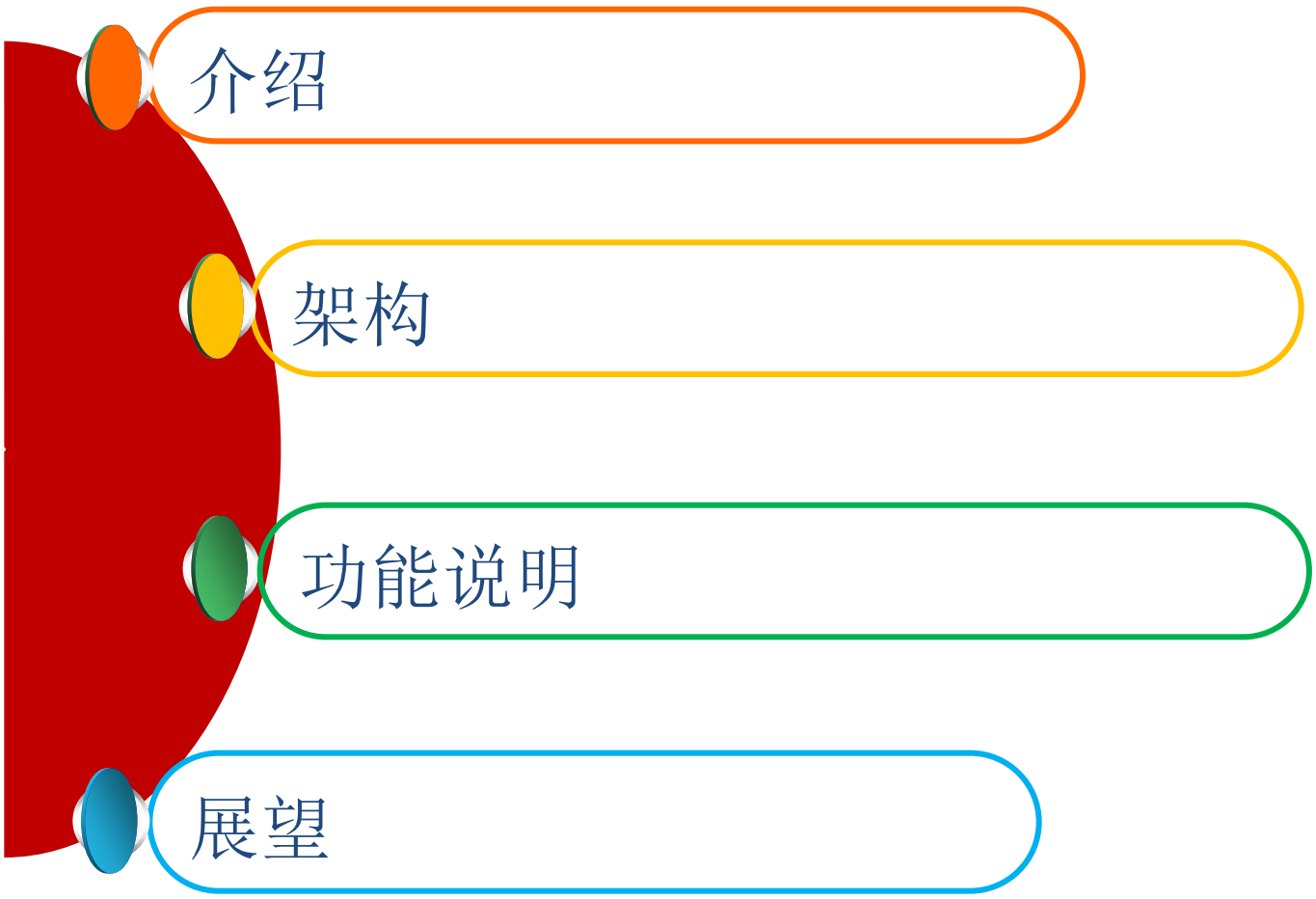
京东MYSQL分布式数据库集群系统技术交流

秦波

JD.京东.COM

内容大纲

JD.COM 京东



介绍

JD.COM 京东

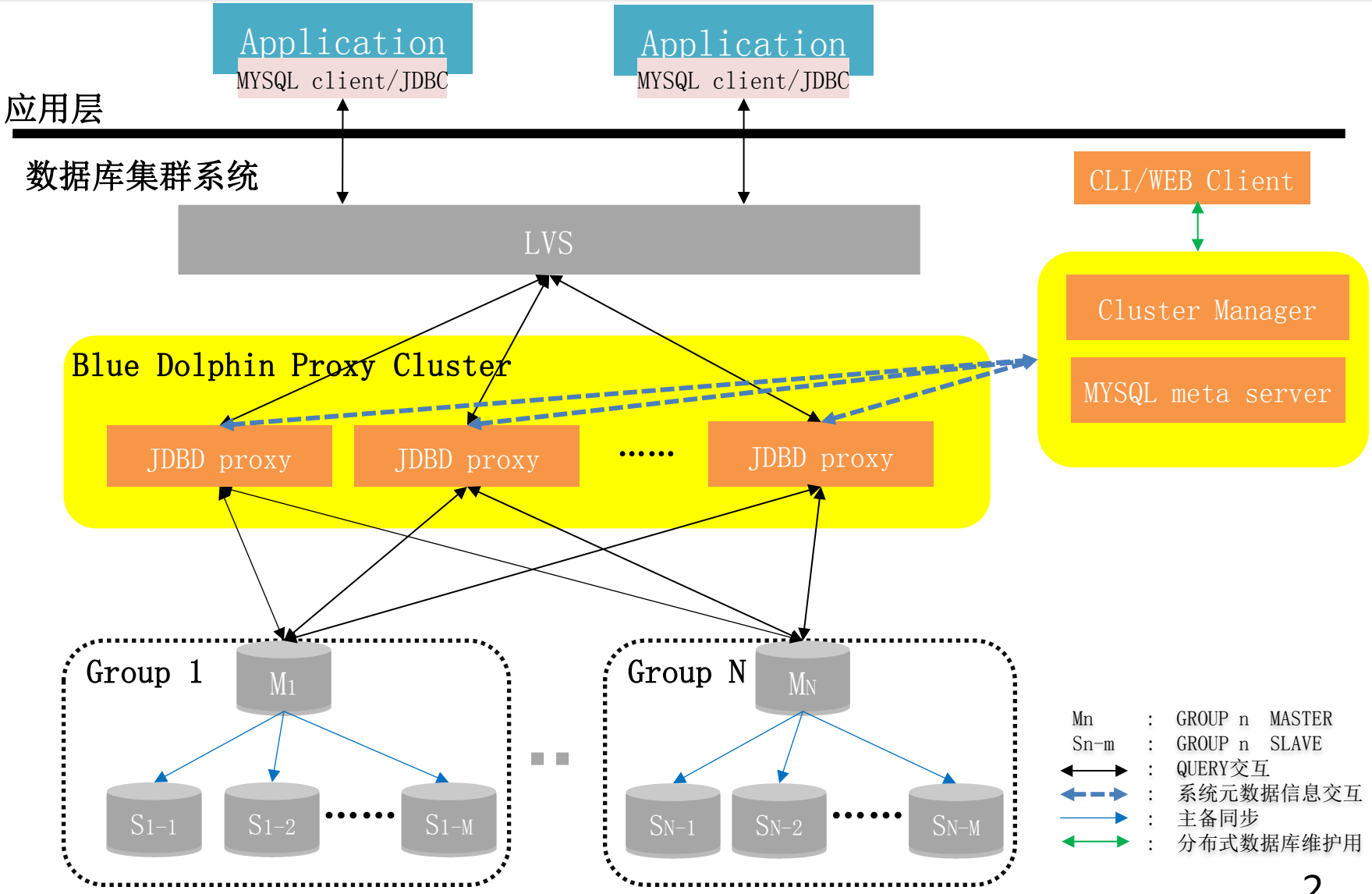
- **目前技术痛点**
  - 业务驱动
  - 运维需要

- **目前分布式中间件的一些开源实现**
  - 功能简单，不满足需求
  - 非自主开发难维护

- **自主开发的蓝海豚分布式MYSQL集群系统**
  - 集群系统，支持高并发、高可用、高扩展性
  - 进度可控，迭代开发

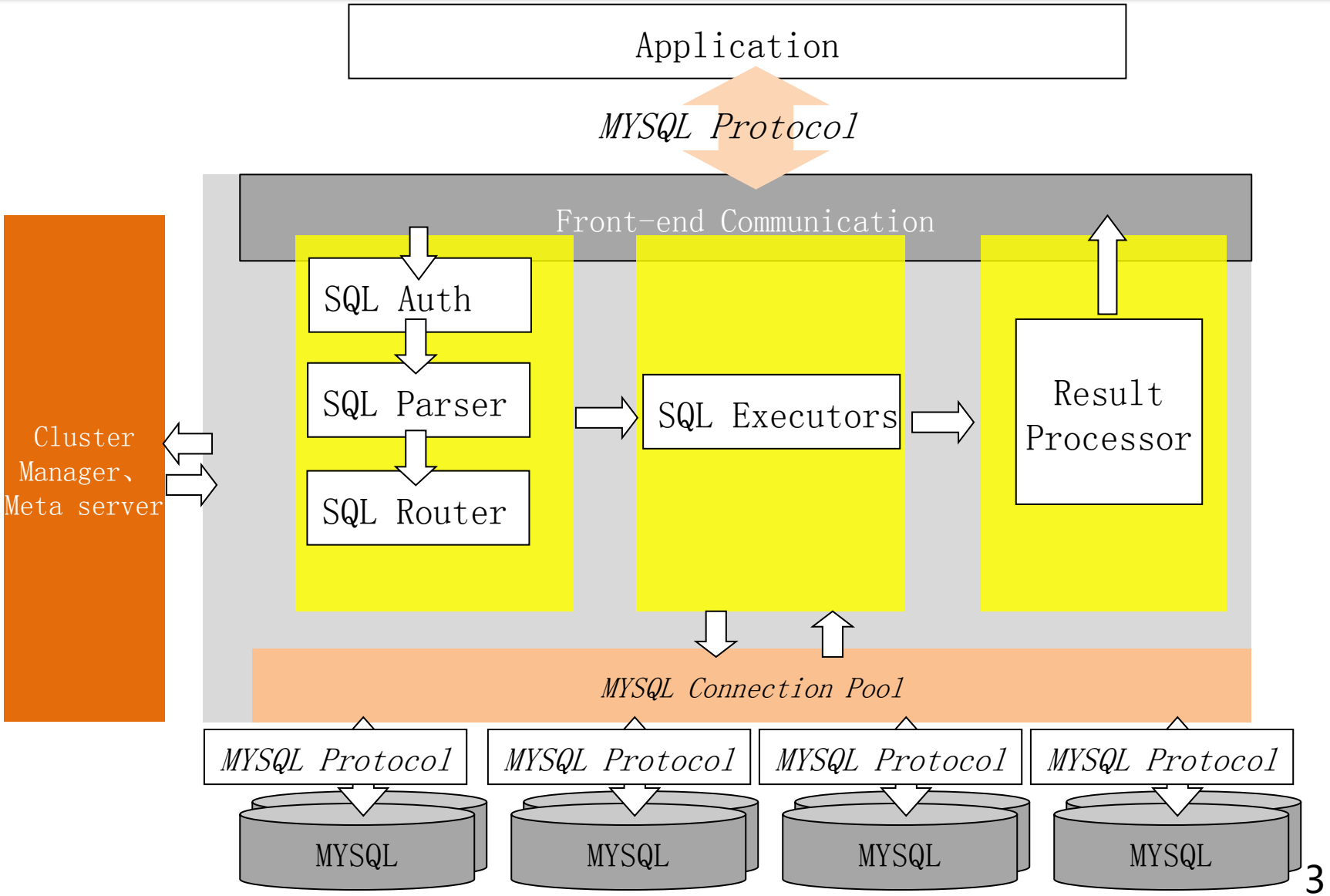
架构 - 拓扑

JD.COM 京东



架构 - 系统实现

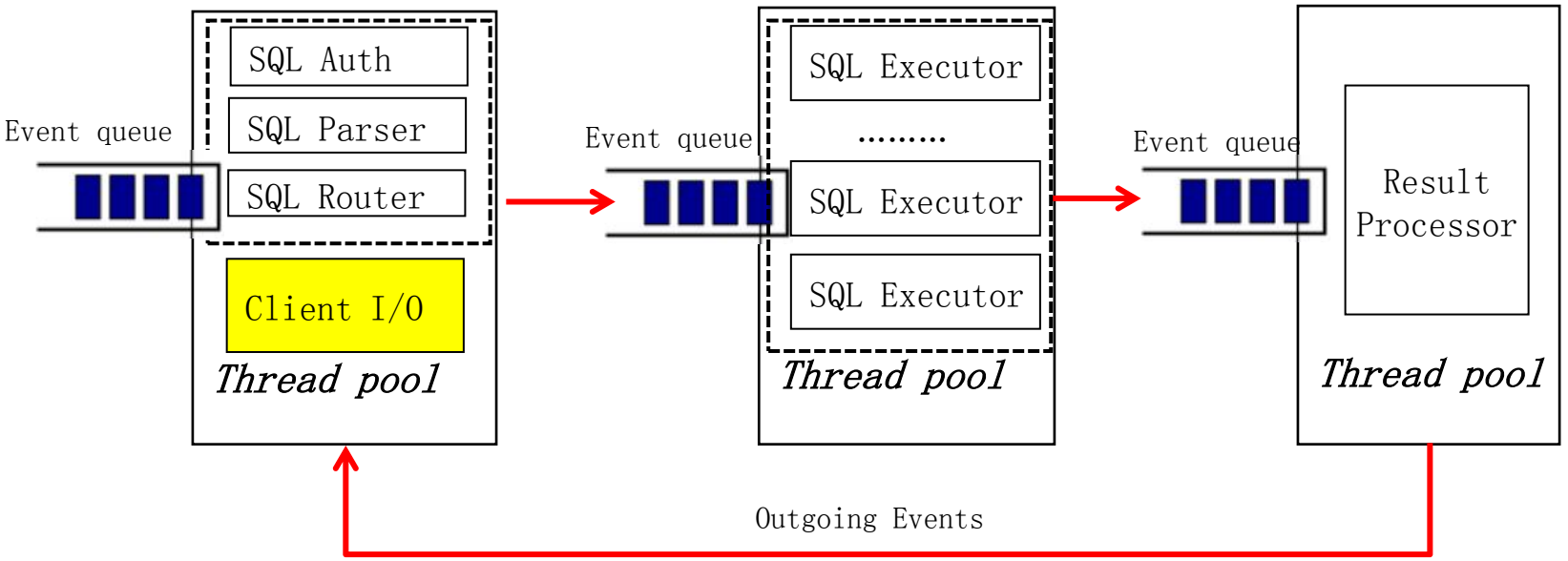
JD.COM 京东



架构 - SEDA并行处理框架

JD.COM 京东

SEDA: Staged Event Driven Architecture



- **线程池&连接池**

  3个线程池；

  可伸缩连接池；

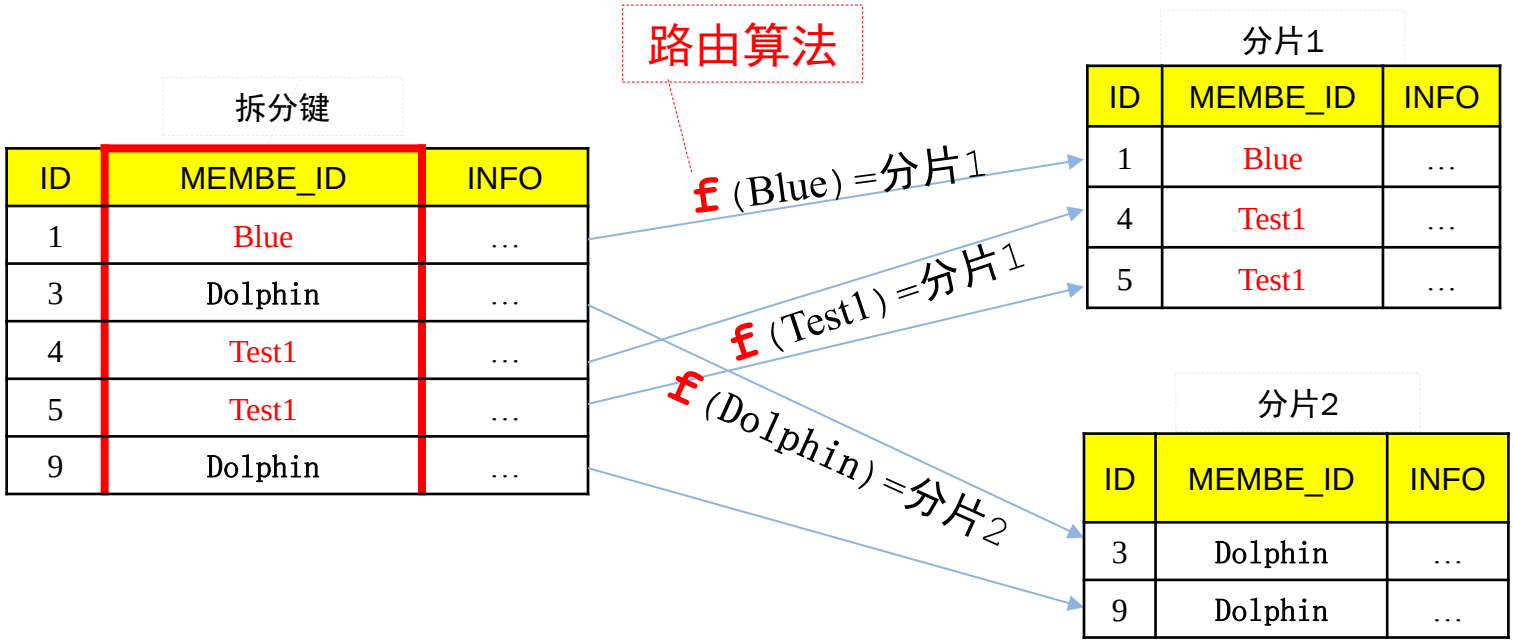
  两者无绑定关系；

功能

JD.COM 京东

跨机分库分表	• consistent-hash • range
SQL支持	• DDL • DML(JOIN) • DCL
表级权限控制	• GRANT SELECT ON T.TBL TO …
数据扩展/自动迁移	• 以分片为单位的数据迁移和自动扩展 • 支持数据重构
HA	• JDBD Proxy层HA • 数据库层HA（Group内自维护）
meta信息在线更新	• JDBD Proxy节点在线更新meta信息

分库分表



- **拆分键&主键**

  拆分键可不等同于主键，目前单表支持指定一个列作为拆分键；
  支持自增键可作为拆分键

- **分片&MYSQL SERVER**

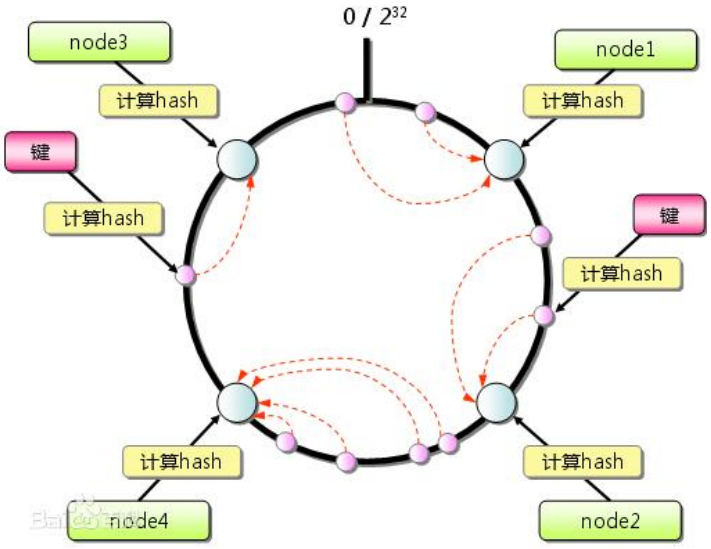
  分片可以理解为子表；是分布式系统管理库表的**最小单位**；在SERVER上分散分布。

分库分表

JD.COM 京东

目前支持两种分库分表路由算法

- consistent hash路由



- RANGE路由

  按范围进行分片

例：

1）表tt,拆分键id,设定的分片数128
分片表明为tt_0~tt_127
每个分片对应一致性HASH余数段

2）执行INSERT INTO tt (id, name)
VALUES (10240, 'Blue Dolphin')

3）locate分片
10240 % CONSISTANT_HASH_VALUE = 10
10位于中；
#当前代码中CONSISTANT_HASH_MAX_VALUE值固定为1024*1024
#tt_0对应范围0~CONSISTANT_HASH_VALUE%128

4）改写insert语句并
发送到tt_0所在的GROUP master server中
执行SQL:
INSERT INTO tt_0 (id, name)
VALUES (10240, 'Blue Dolphin')

SQL处理

JD.COM 京东

➢ SQL解析

词法/语法解析器通过flex/bison实现，可灵活定制，处理结果为语法树，SQL解析是分布式MYSQL系统处理的**基础**。

select语句语法树

固定有14个子节点：

1）记录ALL或者DISTINCT关键词处理
2）select clause
3）from clause(JOIN的处理在此处)
4）where clause
5）group by clause
6）having clause
7）set operation （记录UNION类型 T_SET_UNION）
8）UNION类型（是否重复）
9）former select stmt（UNION关键词前一个sql语句）
10）later select stmt（UNION关键词后一个sql语句）
11）order by clause
12）limit clause
13）for update 是否需要锁
14）hints

例：

语句*SELECT Persons.LastName, Persons.FirstName, Orders.OrderNo FROM Persons INNER JOIN Orders ON Persons.Id_P = Orders.Id_P ORDER BY Persons.LastName* 语法树结构

```
|-T_STMT_LIST
  |-T_SELECT
    |-NULL
    |-T_PROJECT_LIST
      |-T_PROJECT_STRING
        |-T_OP_NAME_FIELD
          |-T_IDENT : persons
          |-T_IDENT : lastname
      |-T_PROJECT_STRING
        |-T_OP_NAME_FIELD
          |-T_IDENT : persons
          |-T_IDENT : firstname
      |-T_PROJECT_STRING
        |-T_OP_NAME_FIELD
          |-T_IDENT : orders
          |-T_IDENT : orderno
    |-T_FROM_LIST
      |-T_JOINED_TABLE
        |-T_JOIN_INNER
        |-T_IDENT : persons
        |-T_IDENT : orders
        |-T_OP_EQ
          |-T_OP_NAME_FIELD
            |-T_IDENT : persons
            |-T_IDENT : id_p
          |-T_OP_NAME_FIELD
            |-T_IDENT : orders
            |-T_IDENT : id_p
    |-NULL
    |-NULL
    |-NULL
    |-NULL
    |-NULL
    |-NULL
    |-NULL
    |-T_SORT_LIST
      |-T_SORT_KEY
        |-T_OP_NAME_FIELD
          |-T_IDENT : persons
          |-T_IDENT : lastname
        |-T_SORT_ASC
    |-NULL
    |-NULL
    |-NULL
```

SQL处理

JD.COM 京东

➢ SQL改写

依据SQL PARSER生成的语法树，结合路由信息进行SQL优化并改写

例：person表city列做拆分键，客户端输入*select avg(age) from person group by city limit 2,4*

STEP1：SQL改写，最终发到数据节点的SQL语句为：

*select avg(age), **sum(age), count(age), city** from person group by city limit **0,6***

STEP2：中间件根据各个数据分片city先排序，并根据返回count值和sum值，汇总求和，再次计算AVG，最后将offset 2开始的4 row的avg(age)结果返回客户端

➢ 执行计划

目的分片信息: sql要发往的分片信息，里面包含了目的server信息

SQL语句 ：sql经过优化、处理并替换表名之后到数据库层面最终执行的sql语句

例：客户端输入*select name from test order by id limit 2,4*，生成如下一组执行计划：

```
exec_plan_unit shard name: test_0
exec_plan_unit SQL name  : SELECT name ,id  FROM test_0  ORDER BY id ASC LIMIT 0, 6
exec_plan_unit shard name: test_1
exec_plan_unit SQL name  : SELECT name ,id  FROM test_1  ORDER BY id ASC LIMIT 0, 6
exec_plan_unit shard name: test_2
exec_plan_unit SQL name  : SELECT name ,id  FROM test_2  ORDER BY id ASC LIMIT 0, 6
exec_plan_unit shard name: test_3
exec_plan_unit SQL name  : SELECT name ,id  FROM test_3  ORDER BY id ASC LIMIT 0, 6
exec_plan_unit shard name: test_4
exec_plan_unit SQL name  : SELECT name ,id  FROM test_4  ORDER BY id ASC LIMIT 0, 6
```

JOIN支持

JD.COM 京东

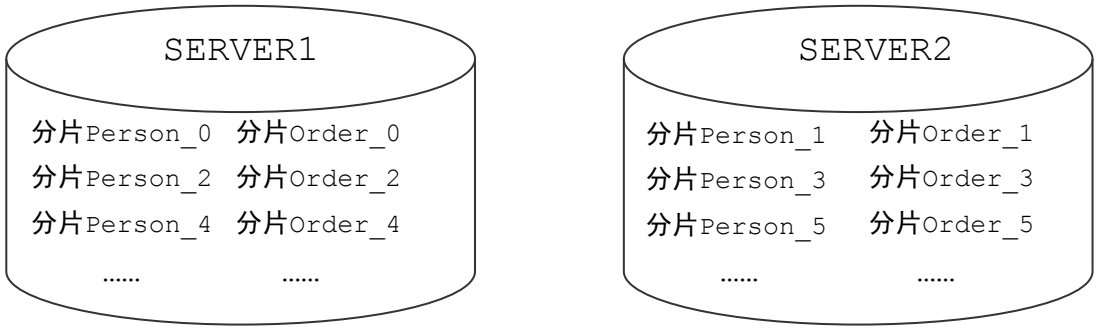
- **Binding table**

  何为binding table?

  按照关联条件将表进行相同拆分。

例：*SELECT Persons.LastName, Persons.FirstName,Orders.OrderNo FROM Persons*
INNER JOIN Orders ON Persons.id = Orders.id
WHERE Orders.id > 100

- ➔ Persons和Orders两表必须均以两表关联列id作为拆分键
- ➔ Persons和Orders两表必须拆分出相同的分片数，比如同为128
- ➔ Persons和Orders两表所有分片在server上的分布方式是一样的

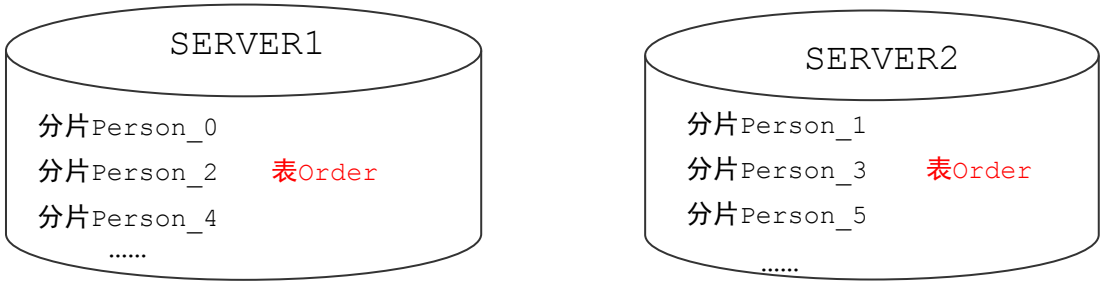


JOIN支持

JD.COM 京东

➢ 小表冗余

将小表分布在所有大表分片所在的server上。
若前面例子中的Order表是小表，分片分布方式：



数据处理原则：
数据的处理尽量做到数据节点本地化处理，尽可能不拿到JDBD Proxy层处理，JDBD Proxy层只做最终的整理和排序

全局唯一ID

JD.COM 京东

- **自增键做拆分键**

  单机环境下的MySQL auto_increment实现的自增键无法使用到分布式系统

- **方案**

  1）数据节点层面自增列当成普通列处理，不再有自增属性
  2）通过MYSQL meta server 分配ID
  　按表维护自增值，指定起始值，步长，当前值

数据迁移/自动扩展

JD.COM 京东

- **数据迁移**

  STEP1：通过自动数据迁移工具开始数据迁移

  STEP2：**数据差异小于某一临界值，停止老分片写操作**（read-only）

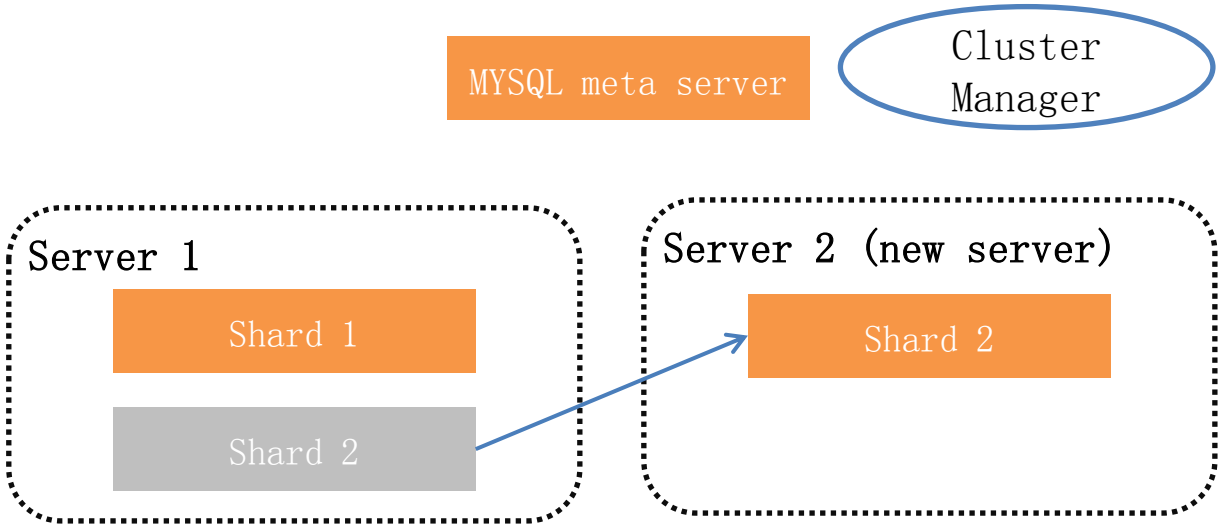
  STEP3：等待新分片数据更新完毕

  STEP4：更改MYSQL meta server中对应分片的路由规则，
  Cluster Manager向所有JDBD proxy**推送**新分片信息

  STEP5：删除老分片

- **自动扩展**

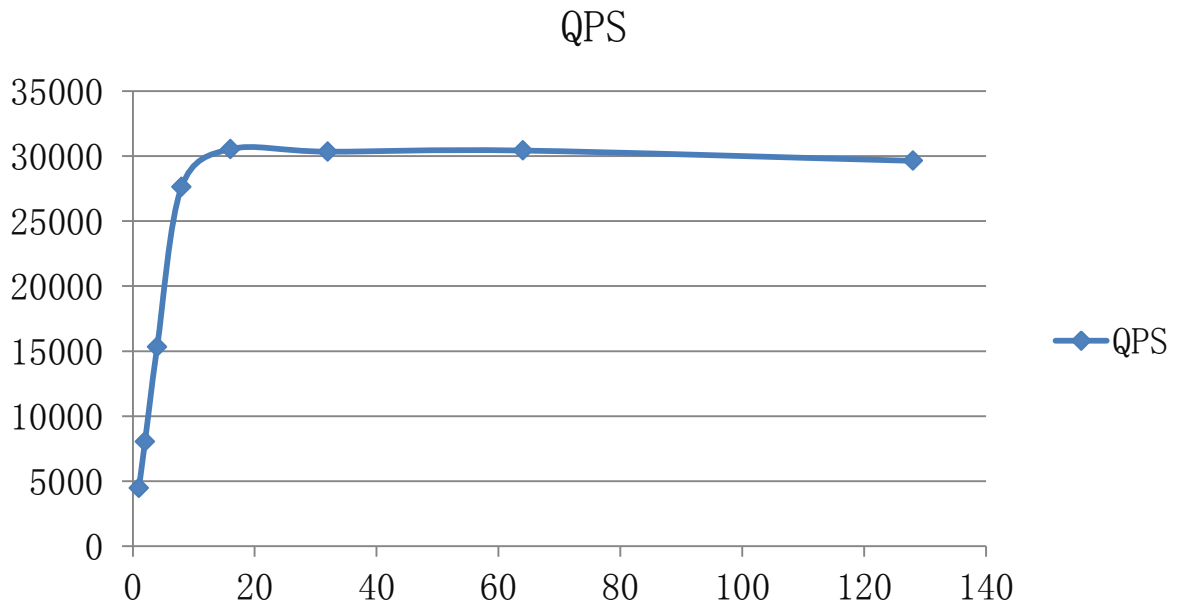
  过程类似于数据迁移



性能

JD.COM 京东

- QPS



注：横坐标是客户端并发数，纵坐标是瓶颈每秒查询处理数，每个query处理2行数据且不带排序操作（最终测试数据也取决于机器性能和基准数据）。

展望 - 事务的支持

JD.COM 京东

- **技术挑战**

1）MYSQL数据库有异步复制

怎么对应所有数据节点Prepare已完成且部分数据节点已commit的情况下，某台未commit的数据库宕机问题？

2）事务协调器JDBD Proxy节点宕机

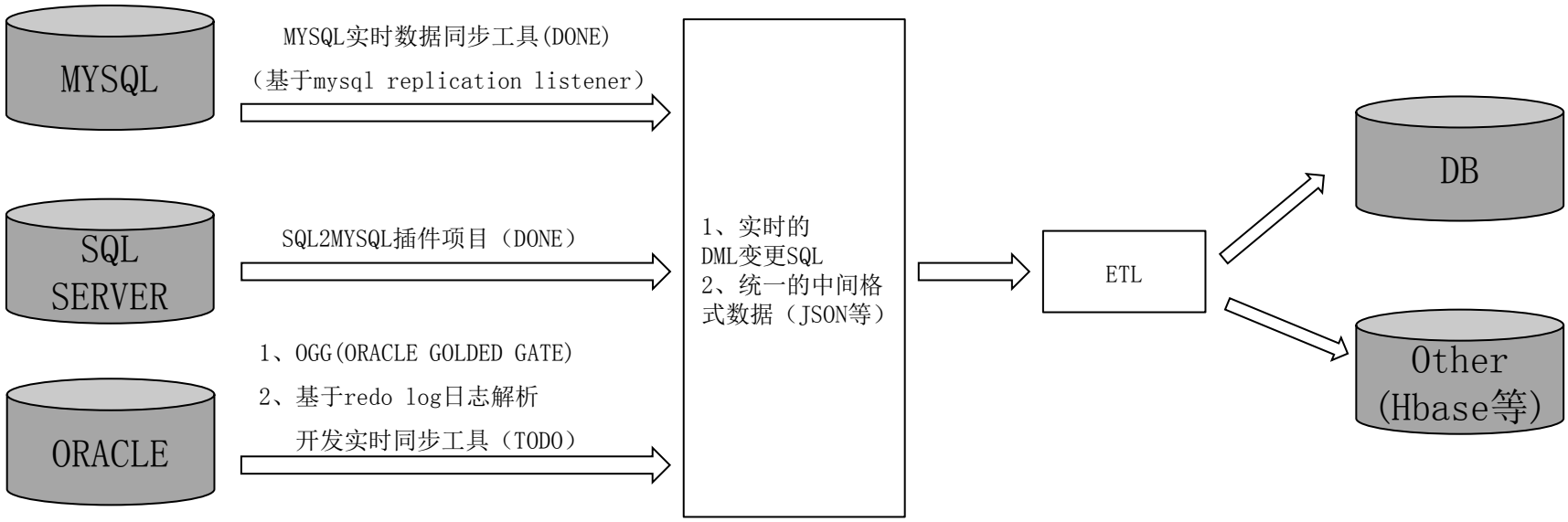
3）XA事务处理效率问题

2PC（2-phase-commit）多机写可以保证事务ACID，但效率较低

展望－分布式实时数据同步系统

JD.COM 京东

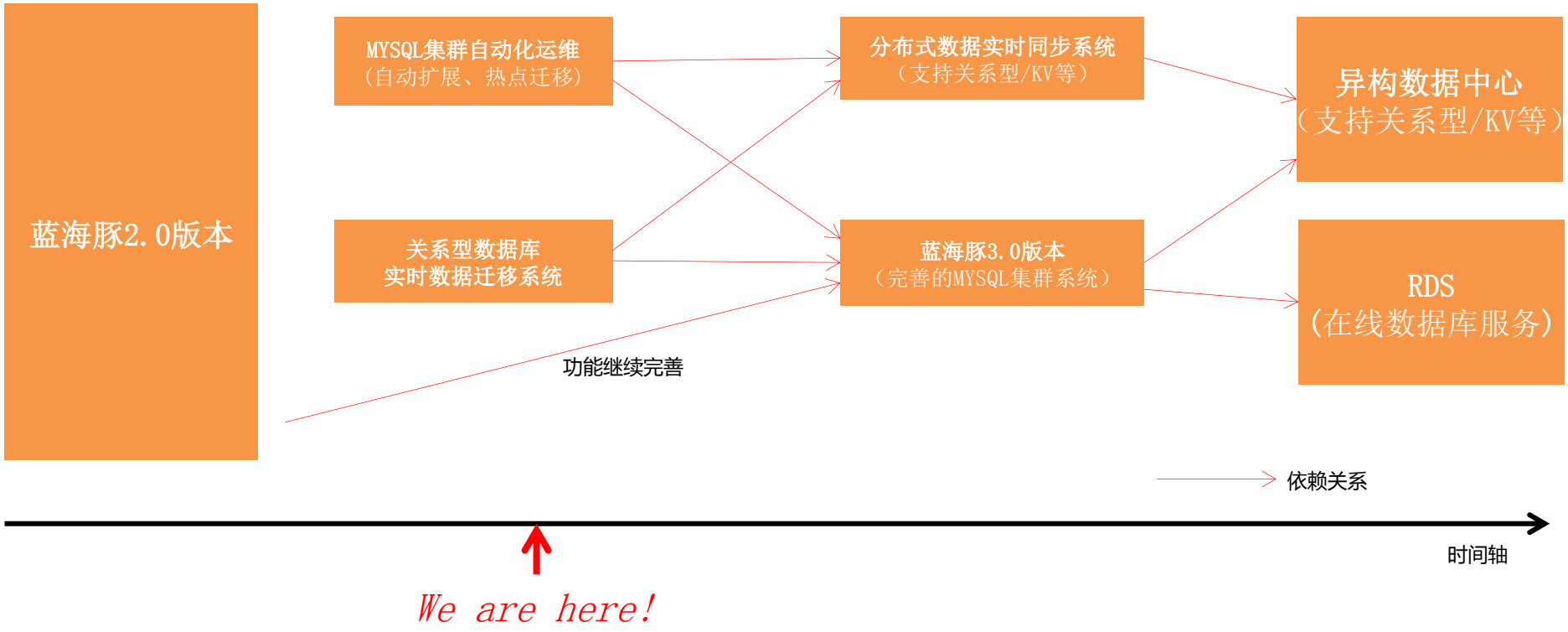
- 分布式实时数据同步系统



1）中间数据在网络中传输（跨/不跨机房）。
2）中间数据可以是sql语句或者统一的第三方格式比如Json或者ProtocalBuffer封装数据
3）一端将数据变更转换为中间数据，另一端负责解释中间数据并还原执行。
4）两端的数据库可以是关系型数据库（MYSQL/SQLSERVER/ORACLE），
也可以是KV系统比如hbase等

展望 - ROADMAP

JD.COM 京东



谢谢大家

E-mail: gqinbo@gmail.com
新浪微博：qinbo0304

JD.京东.COM