# 分库分表技术演进&最佳实践

## 如何存储海量数据？目前比较普遍的方案有3个

分区；

分库分表；

NoSQL/NewSQL；

不使用NoSQL/NewSQL的原因：不成熟

不使用分区的原因：很多的资源都受到单机的限制，例如连接数，网络吞吐等！虽然每个分区可以独立存储，但是分区表的总入口还是一个MySQL示例。从而导致它的并发能力非常一般，远远达不到互联网高并发的要求！

NoSQL比较具有代表性的是MongoDB，es。NewSQL比较具有代表性的是TiDB。

## 分库分表中间件

阿里的TDDL，DRDS和cobar，

开源社区的sharding-jdbc（3.x已经更名为sharding-sphere）；

民间组织的MyCAT；

360的Atlas；

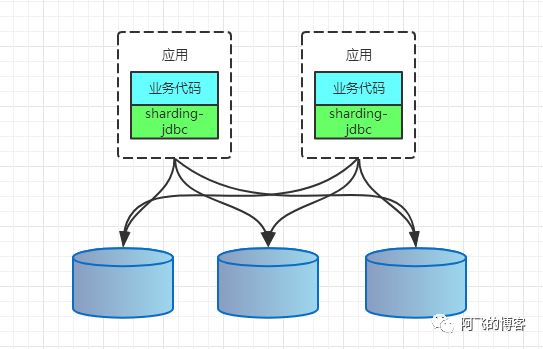
美团的zebra；

## 分库分表中间件全部可以归结为两大类型

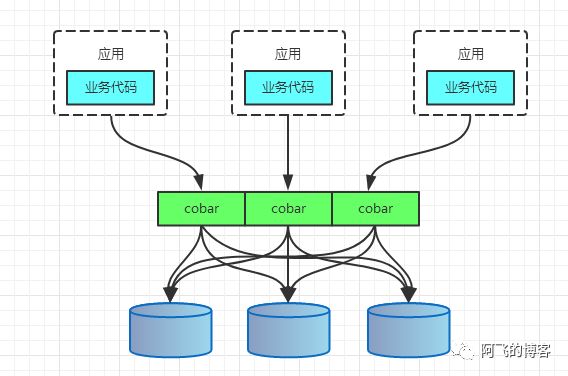
CLIENT模式；

PROXY模式；

CLIENT模式代表有阿里的TDDL，开源社区的sharding-jdbc（sharding-jdbc的3.x版本即sharding-sphere已经支持了proxy模式）。架构如下



PROXY模式代表有阿里的cobar，民间组织的MyCAT。架构如下：



但是，无论是CLIENT模式，还是PROXY模式。几个核心的步骤是一样的：SQL解析，重写，路由，执行，结果归并。

## 实战案例

分库分表第一步也是最重要的一步，即sharding column的选取，sharding column选择的好坏将直接决定整个分库分表方案最终是否成功。选择sharding column的方法最主要分析你的API流量，优先考虑流量大的API，将流量比较大的API对应的SQL提取出来，将这些SQL共同的条件作为sharding column。

这里列举分库分表的几种主要处理思路：

只选取一个sharding column进行分库分表 ；

多个sharding column多个分库分表；

sharding column分库分表 + es(es+HBase)；