打造自己的分布式NoSQL

simpcl
2014.11

主要内容

- 分布式策略篇
- Bada篇

分布式策略篇

为什么要分布式

- 容量受限
- 请求压力大

数据分布

- 哈希分布

✓ 简单高效

✓ 数据不连续

- 顺序分布

✓ 支持顺序扫描、范围查找

✓ 复杂

数据分布

- 哈希取模

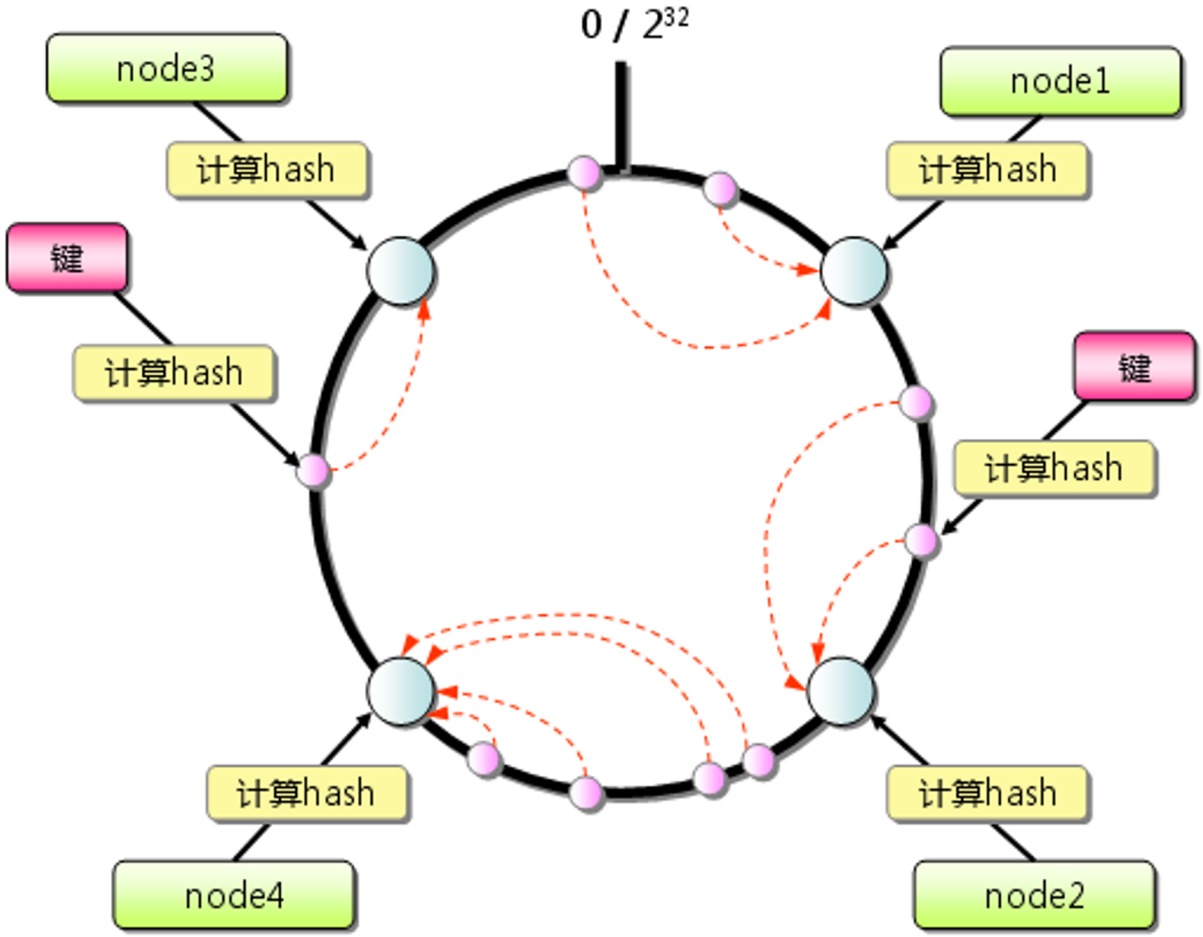
✓ 简单

✓ 数据不连续

✓ 增删节点时大部分数据需要重新定位

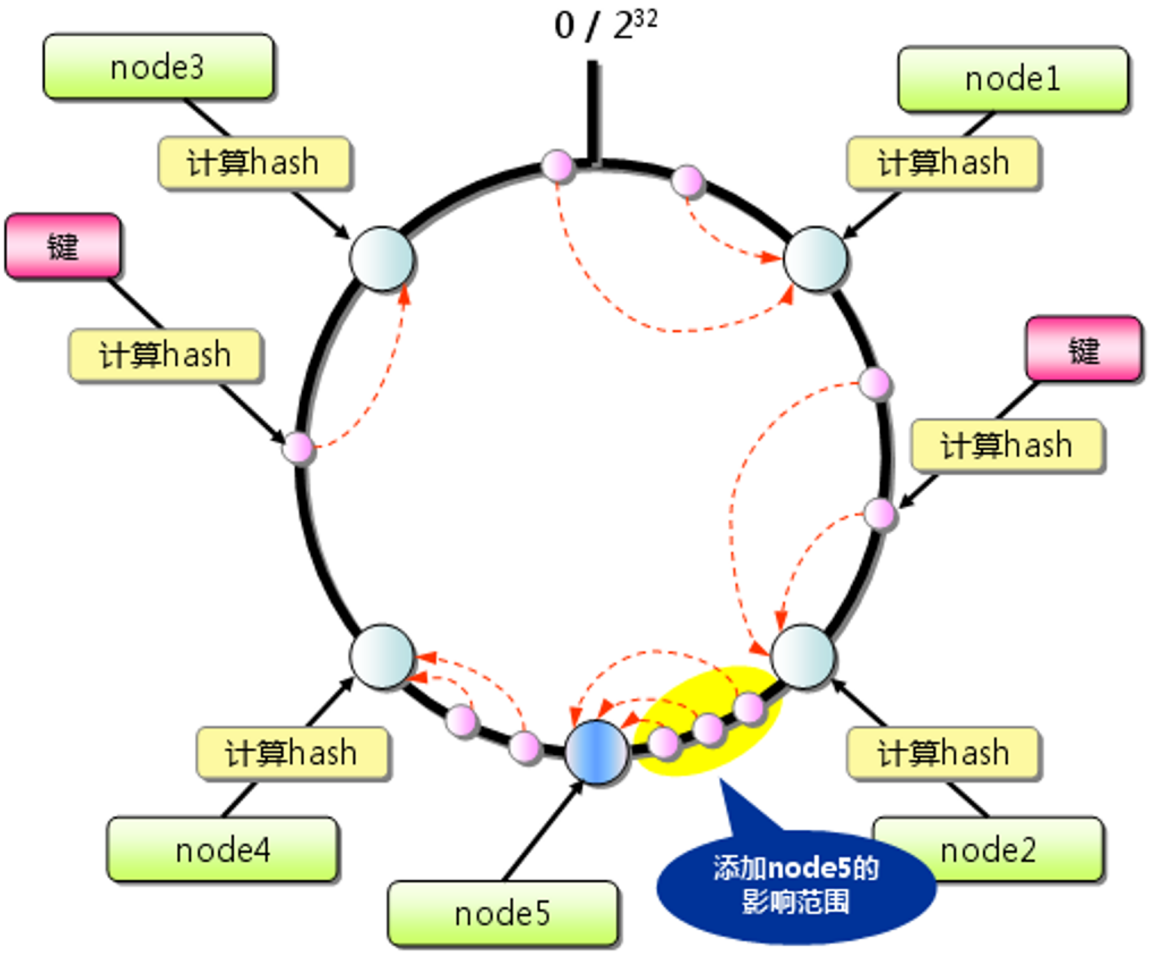
数据分布

- 一致性哈希算法



数据分布

- 一致性哈希算法



数据分布

- 一致性哈希算法的问题

✓ 管理困难

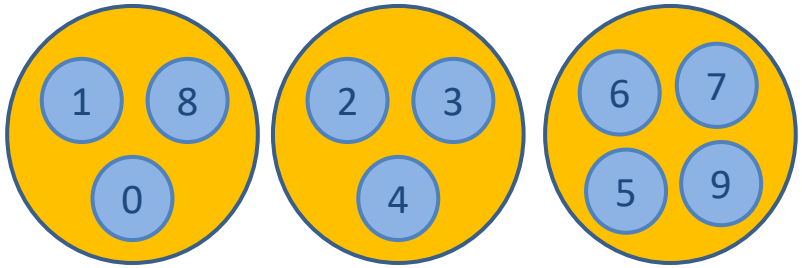
✓ 节点故障局部压力过大

数据分布

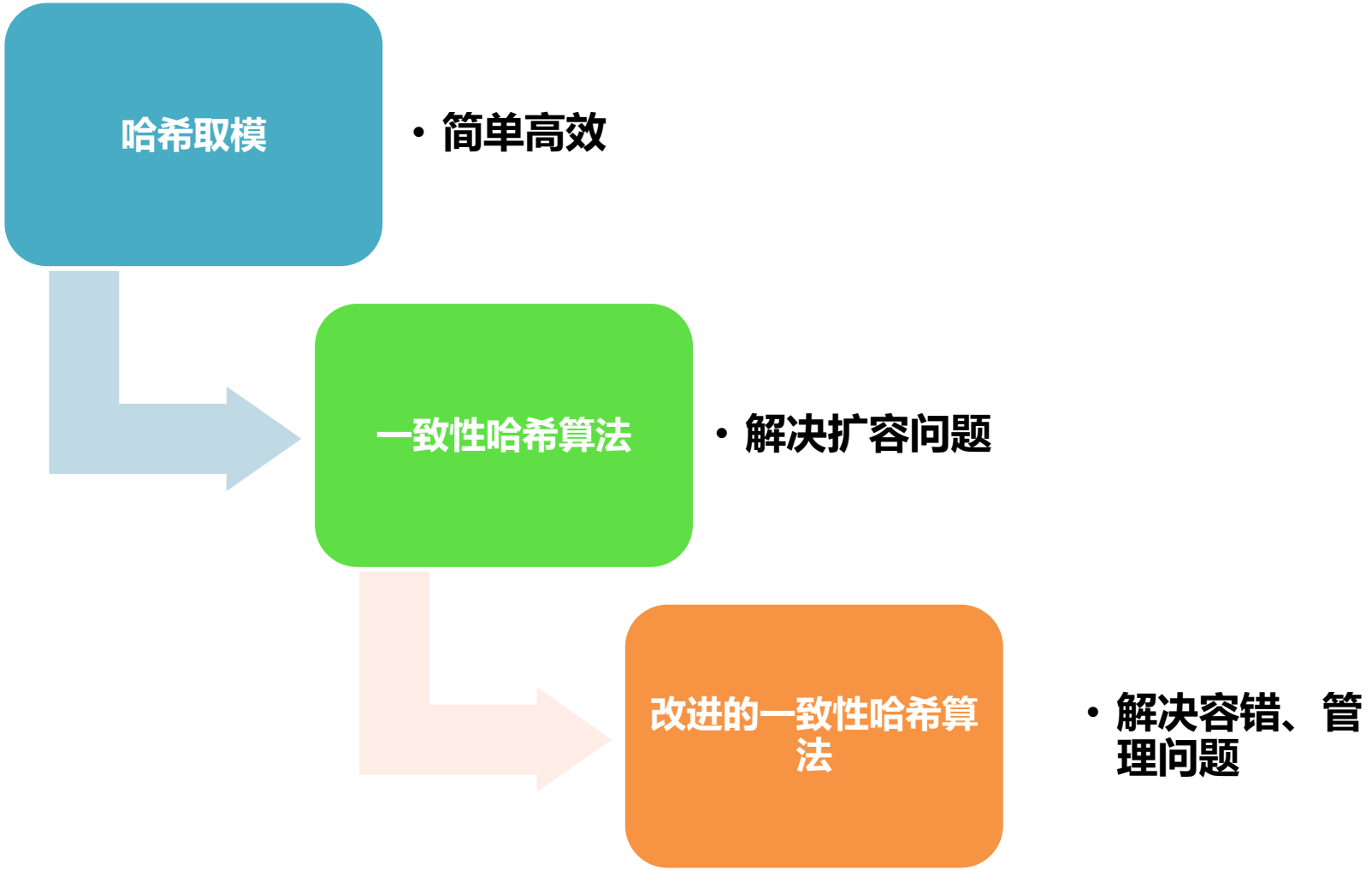
- 改进的一致性哈希算法

✓ 将数据Hash映射到哈希桶

✓ 将哈希桶映射到节点



数据分布



复制

- 主从复制协议（Primary-based protocol）

✓ 前期复杂，后期简单

✓ 一致性稍高

- 复制写协议（Replicated-write protocol）

✓ 前期简单，后期复杂

✓ 一致性略低

复制

- 主从复制的数据同步方式

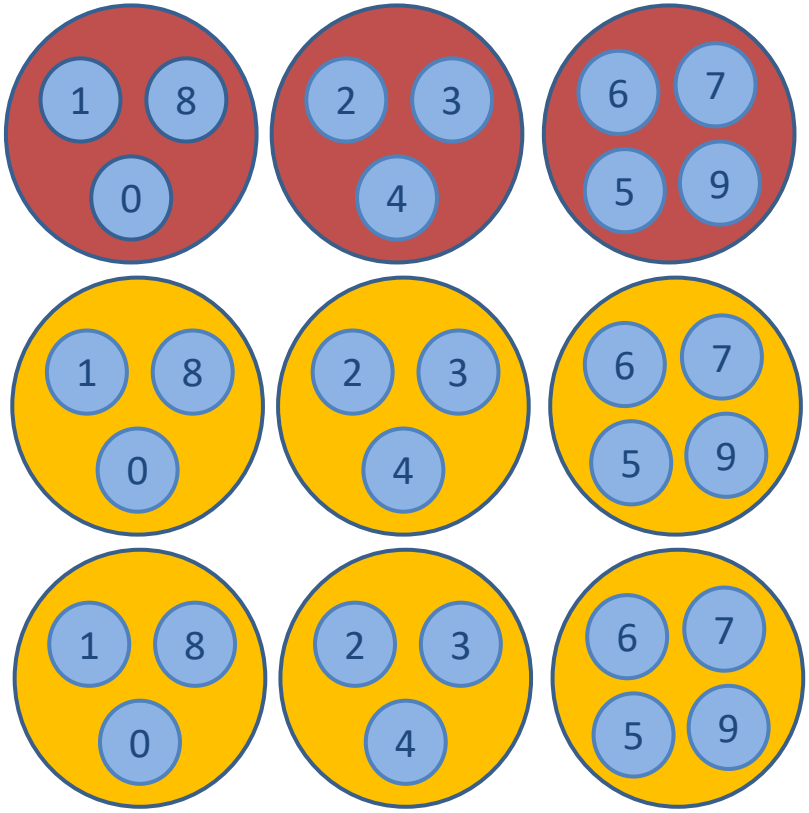
✓ 强一致同步复制

✓ 半同步

✓ 异步同步

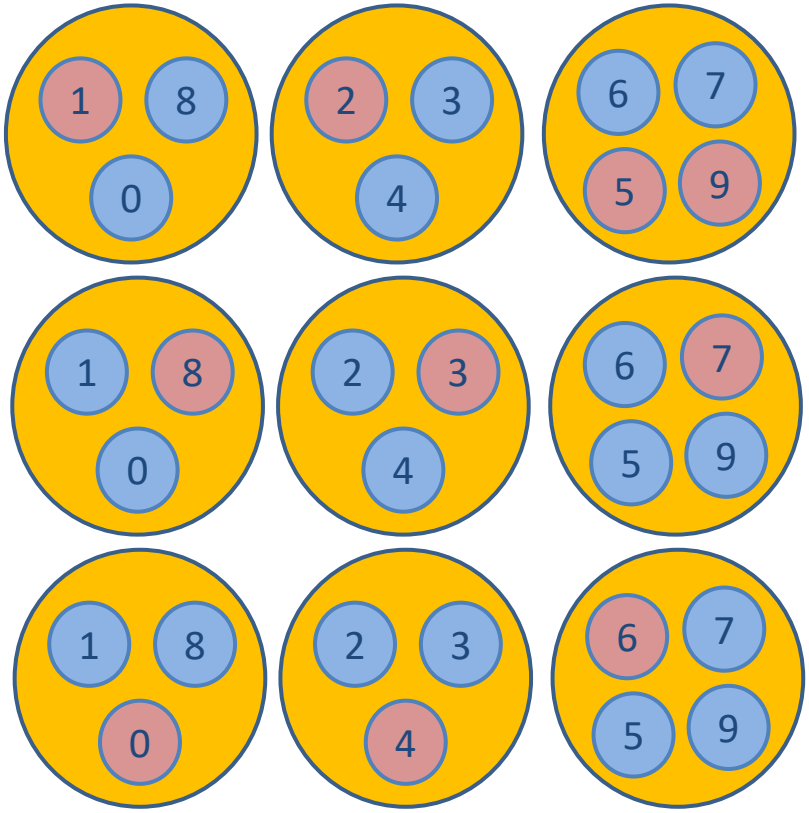
复制

- 副本的放置



复制

- 副本的放置



复制

- 基于分片的主从复制

✓ 每个节点即有主又有从，读写都访问主，一致性高

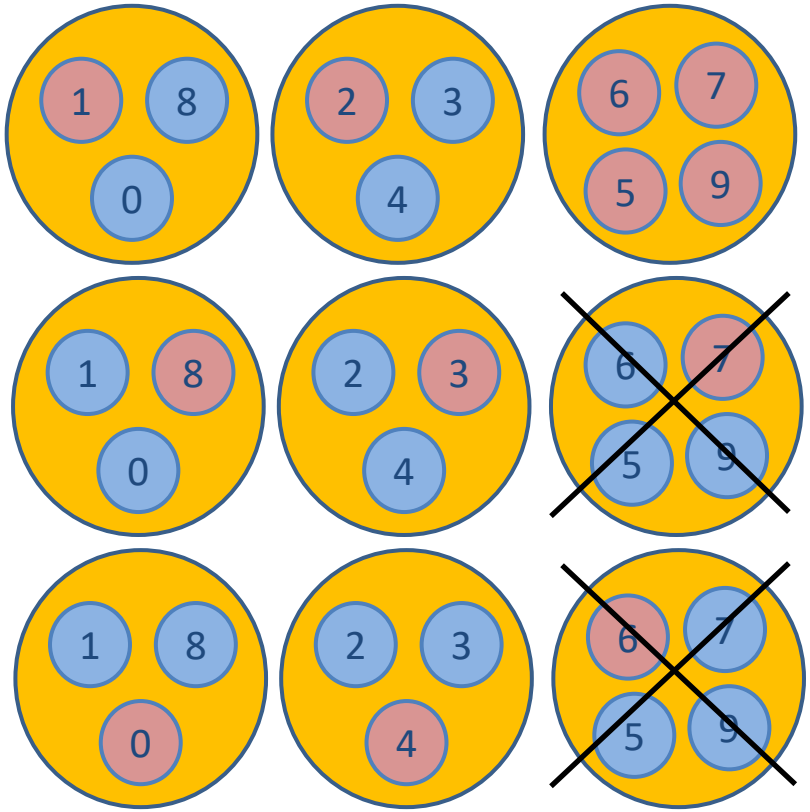
✓ 并行同步，数据同步速度快

容错

- 主从复制协议的容错是重新选主
- 复制写协议的容错是如何修复不一致的数据

容错

- 每个分片的复制集进行选主
- 宕机的局部压力大问题？



负载均衡

- 副本放置均衡

✓ 初始副本均衡

✓ 扩容缩容平滑迁移

✓ 分片数据整体迁移，迁移速度快

- 主副本均衡

✓ 平滑调整主从

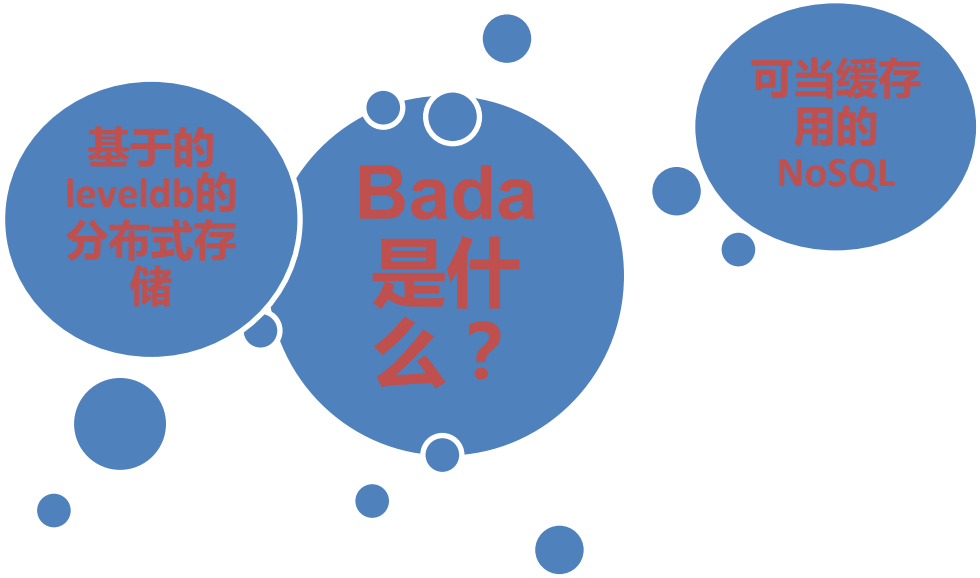
✓ 通过算法保证主平衡

分布式策略总结

- 数据分布： 改进的一致性哈希算法，负载平均，扩容简单
- 复制： 以分片为单位主从异步复制，一致性高，延迟低，同步速度快
- 容错： 分片各自选主，节点与节点包含分片无相关性
- 负载均衡： 分片副本位置均衡和主均衡，以分片为单位迁移速度快

Bada篇

什么是Bada

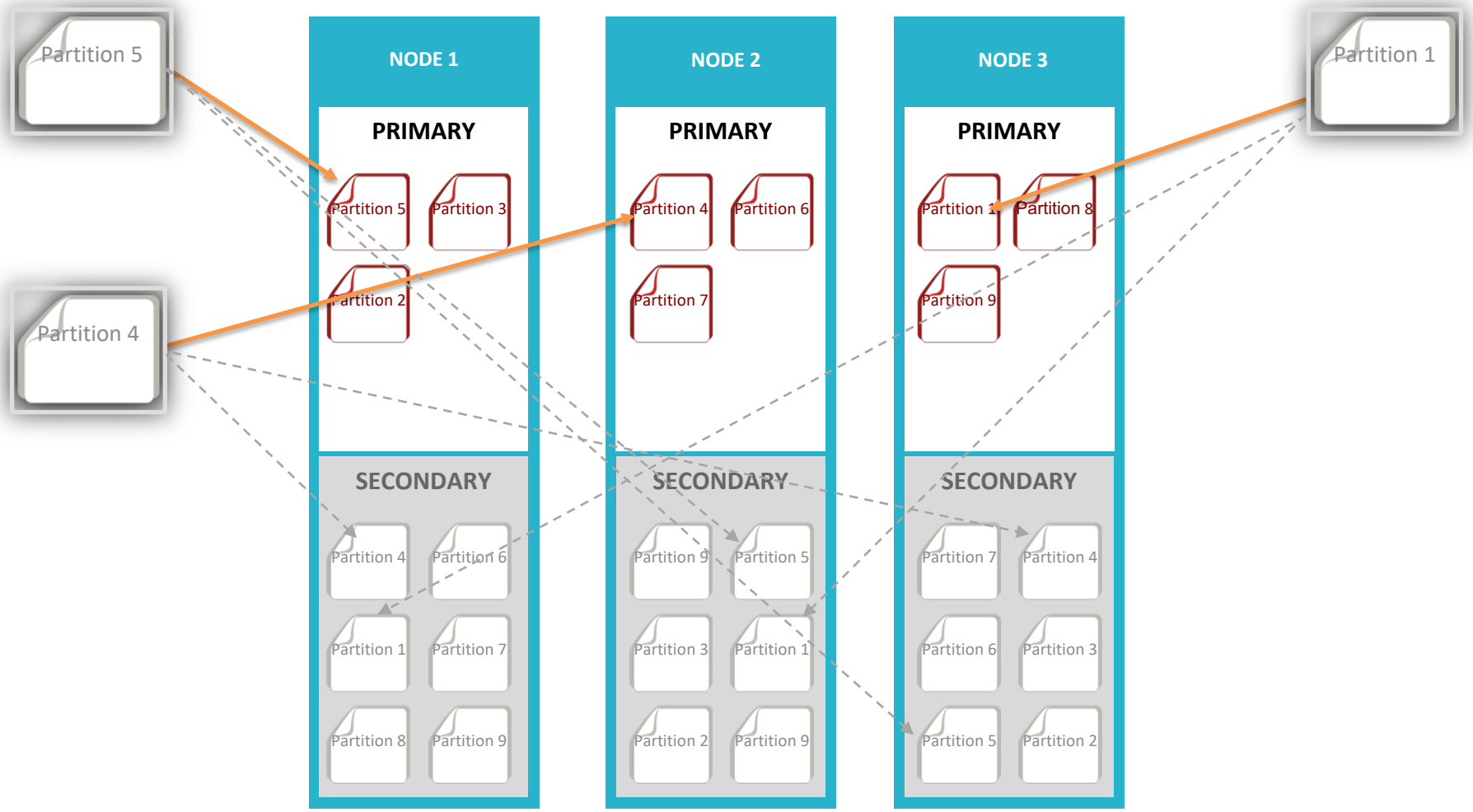


Bada特点	
海量数据	千亿规模
高并发	线性扩展，单台服务器QPS 3w+
低延迟	< 3ms
持久化	支持（leveldb）

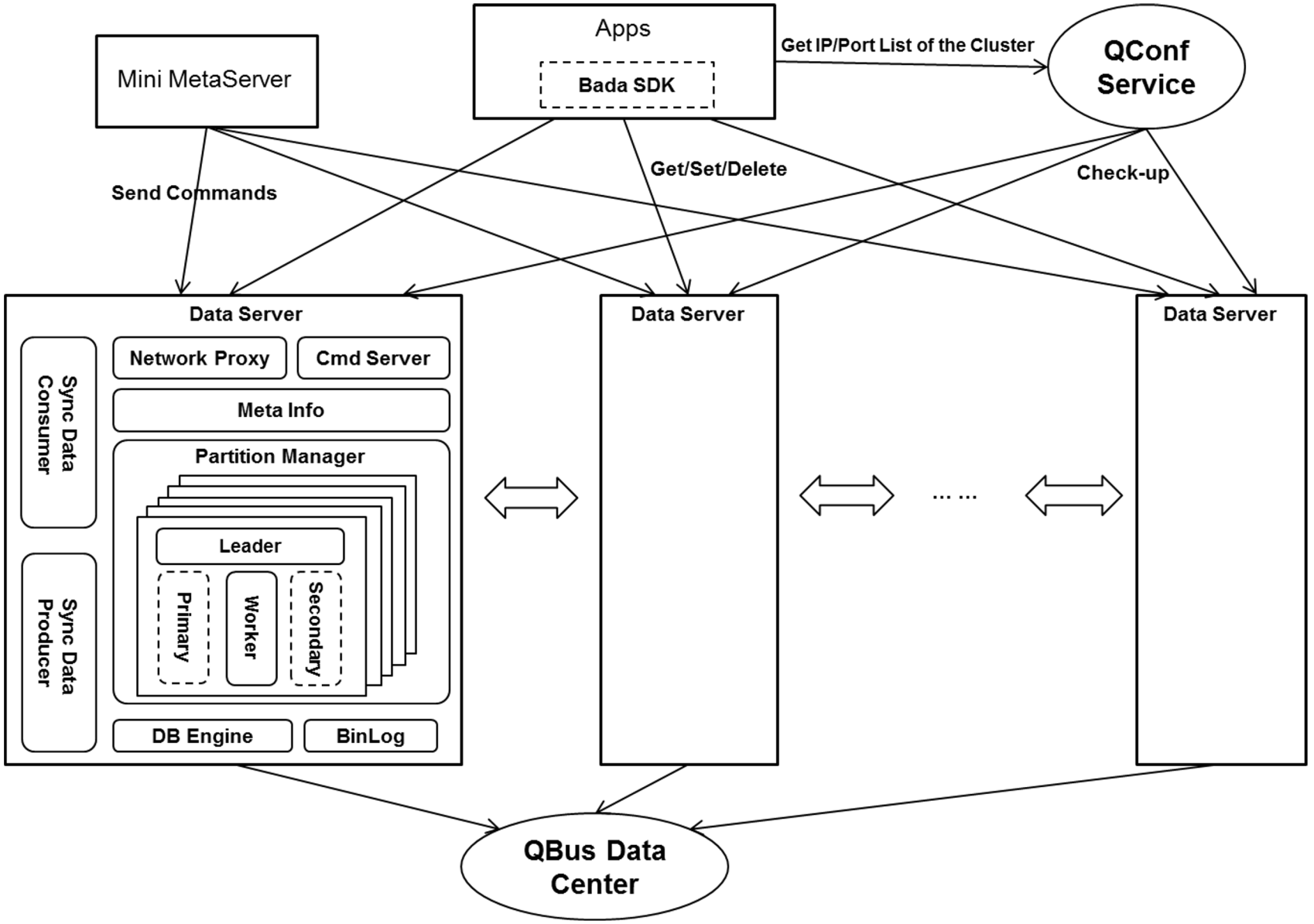
分布式存储 - Bada

- 简单高效的分布式集群方案
- levelDB + SSD的存储，支持Key-String和Key-Structure

Bada分布式原理图


Partition 5
Partition 4
NODE 1
PRIMARY
Partition 5
Partition 3
Partition 2
SECONDARY
Partition 4
Partition 6
Partition 1
Partition 7
Partition 8
Partition 9
NODE 2
PRIMARY
Partition 4
Partition 6
Partition 7
SECONDARY
Partition 9
Partition 5
Partition 3
Partition 1
Partition 2
Partition 9
NODE 3
PRIMARY
Partition 1
Partition 8
Partition 9
SECONDARY
Partition 7
Partition 4
Partition 6
Partition 3
Partition 5
Partition 2
Partition 1

Bada架构图


Mini MetaServer
Apps
Bada SDK
Get IP/Port List of the Cluster
QConf Service
Send Commands
Get/Set/Delete
Check-up
Data Server
Sync Data Consumer
Network Proxy
Cmd Server
Meta Info
Partition Manager
Leader
Primary
Worker
Secondary
Sync Data Producer
DB Engine
BinLog
Data Server
... ...
Data Server
QBus Data Center

Bada的一些特色

1. **多IDC数据同步**
2. **BinLog Merge**
3. **强一致（get_if_all)**
4. **单Key事务(cas)**
5. **过期**
6. **多数据结构(key-structure)**
7. **Partition平均与主平均**

Bada的多IDC同步方案

- **方案1：服务端写单机房--客户端单写多读**

Bada部署到多个机房，只有一个机房的Bada集群被写入，业务只在一个机房的Web服务器有写入（本地机房写），在多个机房读出（本地机房读），机房之间的数据同步由Bada内部自动实现，业务无需关心。

- **方案2：服务端写单机房--客户端多写多读**

Bada部署到多个机房，只有一个机房的Bada集群被写入，业务在每个机房的Web服务器都有写入（只有跟Bada在同一机房的客户端是本地机房写，其它都是跨机房写），在各个机房均有读（本地机房读），数据同步由bada内部实现，业务无需关心。

- **方案3：服务端写多机房-客户端多写多读**

Bada部署到多个机房，多个机房的Bada集群都被写入，业务在每个机房的Web服务器都有写入，每个机房的Web服务器都有读（本地机房读），数据同步由bada内部实现，业务无需关心。按照多个机房同时写入的冲突解决方案可以分为以下三种：

✓ **1. Primary Key（预期2014Q4实现）：每个Key只在一个机房允许写，每个机房都允许读；**

✓ **2. 每个Key在多个机房都可以写，都可以读，通过Vector-Clock方式解决冲突；**

✓ **3. 每个Key在多个机房都可以写，都可以读，冲突只按照时间戳先后来解决；**

Bada服务的SLA

- ✓ **单集群数据规模1000+亿条，2TB+**
- ✓ **单台物理服务器QPS > 3w**
- ✓ **请求平均延时 < 3ms ，99.99%的请求延迟 < 100ms**
- ✓ **集群可用性 99.99%**
- ✓ **数据可靠性 99.9999%**

Q&A

谢谢！