分布式存储系统中的数据分区

如何选择数据在集群中的写入位置是分布式存储的一个非常关键的问题：

- 数据均衡：集群在长时间运行后，是否会出现单点由于容量不足而出现不可写的情况
- 影响写入性能：将并发写入的数据分散到多个目标的话，能够提高写入的并发度，减少网络和数据节点成为性能瓶颈的可能
- 影响读性能：数据的存放位置，会决定数据能否从多个目标并发读取
- 更多：主机资源的均衡（CPU、内存、硬盘），资源的损耗（硬盘），数据热点问题

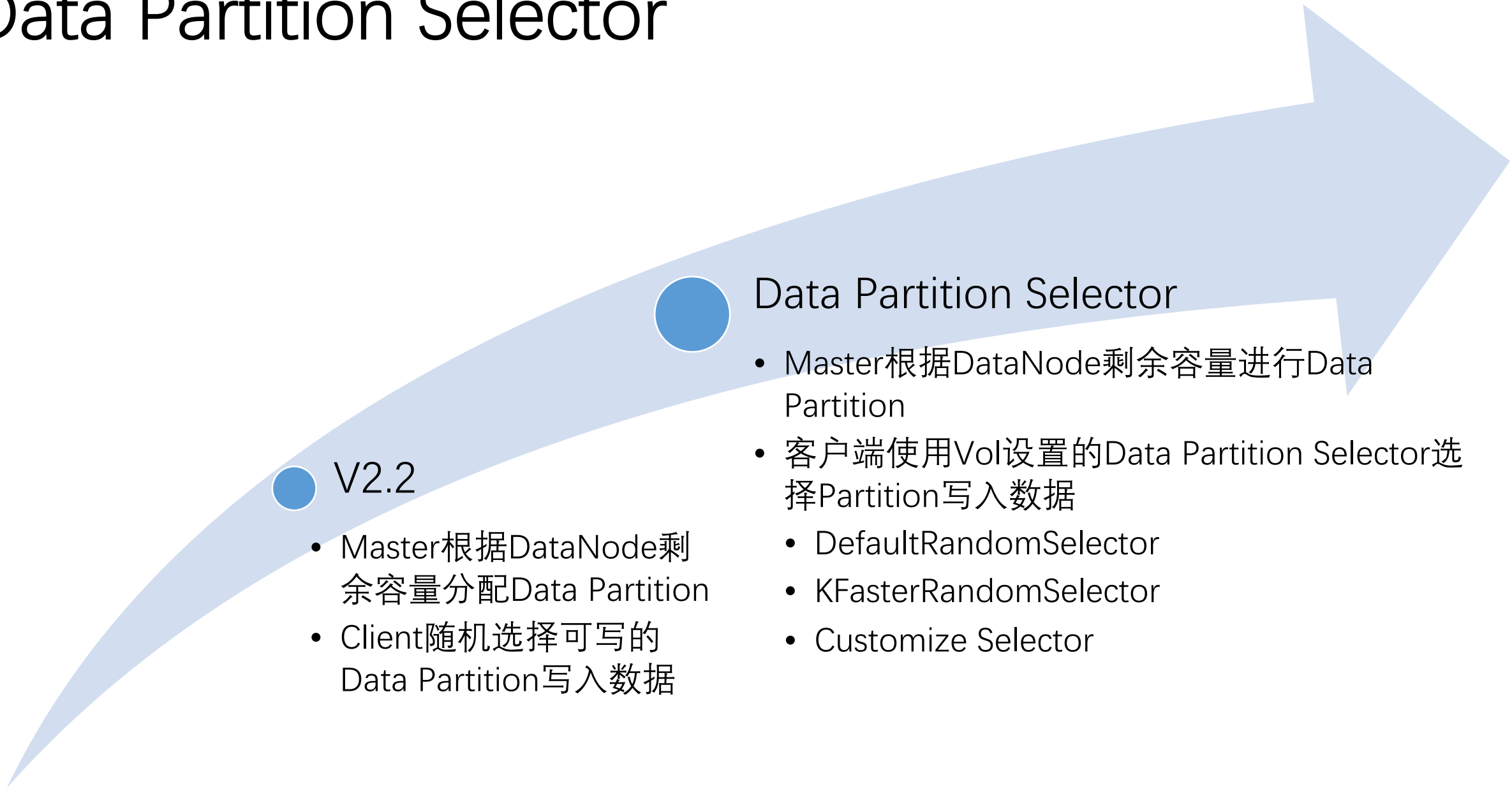
常见的数据分区策略：

- 随机分区（实现简单，抗结构变化能力强，均衡性较好，但需要保存大量的分区数据）
- 基于关键字的区间分区（数据连续性好，但容易出现数据倾斜和热点，少量的分区数据）
- 基于关键字的哈希分区（均衡性较好，区间连续性不好，抗结构变化能力较差，少量的分区数据）
  - *CEPH的CURSH通过straw算法提供了较好的抗结构变化能力，但也导致与复制协议结合使用时，对CRUSH MAP的变化极为敏感，且需要处理大量数据迁移的问题）

除了常见的分区策略，还有很多特定客户的，特殊场景的，定制的分区策略

- 剩余容量
- IP地址距离
- 基于分区权重（优先写某些机房、机柜、机器、磁盘等）

Data Partition Selector



V2.2

- Master根据DataNode剩余容量分配Data Partition
- Client随机选择可写的Data Partition写入数据

Data Partition Selector

- Master根据DataNode剩余容量进行Data Partition
- 客户端使用Vol设置的Data Partition Selector选择Partition写入数据
  - DefaultRandomSelector
  - KFasterRandomSelector
  - Customize Selector

如何定制自己的Selector

注册

- 通过初始化函数init()调用Selector的注册函数，两个参数分别为Selector名称和构造函数

构造函数

- 参数为Selector的参数，为string类型

外部调用接口

```
type DataPartitionSelector interface {
    // Name return name of current selector instance.
    Name() string

    // Refresh refreshes current selector instance by specified data partitions.
    Refresh(partitions []*DataPartition) error

    // Select returns an data partition picked by selector.
    Select(excludes map[string]struct{}) (*DataPartition, error)

    // RemoveDP removes specified data partition.
    RemoveDP(partitionID uint64)
}
```

Tips

- 1、对Selector的访问，Client已经进行了锁保护，但是对于Selector内部的数据访问，需要Selector的实现中进行并发控制
- 2、构造函数中如果需要多个参数，需要拼接成单个字符串
- 3、Master设置接口如下：curl http://masterPort/vol/update?name=volName&authKey=VolKey&dpSelectorName=a&dpSelectorParm=b，dpSelectorName和dpSelectorParm必须同时指定，单独指定一个无效
- 4、支持动态修改Selector，不需要重启Client，直接通过Master接口设置即可
- 5、当指定的Selector不存在或者初始化失败时，会启用默认的DefaultSelector