# 一致性Hash实现说明

# 介绍

一致性哈希算法在1997年由麻省理工学院提出，是一种特殊的哈希算法，目的是解决分布式缓存的问题。 在移除或者添加一个服务器时，能够尽可能小地改变已存在的服务请求与处理请求服务器之间的映射关系。一致性哈希解决了简单哈希算法在分布式哈希表( Distributed Hash Table，DHT) 中存在的动态伸缩等问题。

详细介绍参考：https://blog.csdn.net/qq\_40551994/article/details/100991581

# 2.名词解释

**Builder File**：Hash设备配置文件，存储用于构建Hash Ring的设备信息，包含设备ID、设备名称、设备权重、设备中的分区数量等；

**Ring File**：Hash Ring文件，用于存储虚拟设备到物理设备的映射关系；

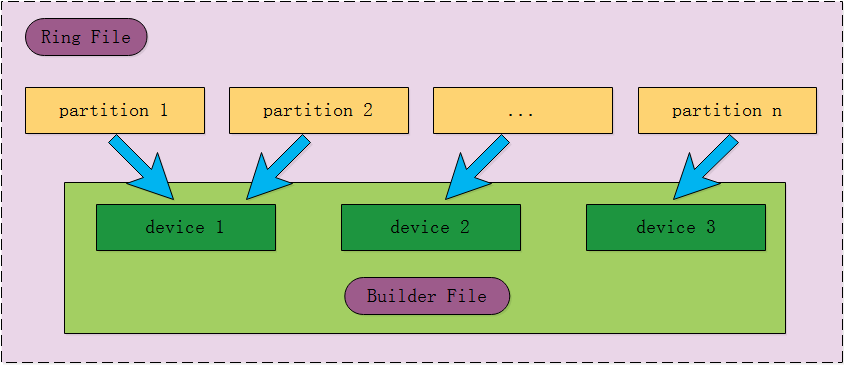
**rebalance**：物理设备信息变更时，需要执行rebalance调整虚拟设备和物理设备的映射关系。

**partition**：虚拟设备的引入，主要是为了缓解数据分布不均匀的问题，一般会根据物理设备的实际数量，乘以一定比例得到虚拟设备的数量。

# 概要设计

## 2.1.模块原理

Ring File和Builder File存储信息关系如下图所示：



说明：

1、Builder File仅用于保存设备信息，增删和编辑设备权重等信息时，均通过修改此配置文件完成；

2、Ring File记录了虚拟设备（partition）到物理设备的映射关系，添加和删除设备时，需要执行rebalance操作，实际就是根据新的Builder File配置信息调整Ring File。

## 2.2.静态结构设计

### 2.2.1.Builder File结构设计

Builder File所包含的字段信息（供参考）如下表所示，其存储结构自行定义，可以采用json、xml、二进制文件格式等。

|  |  |  |
| --- | --- | --- |
| 字段名 | 字段类型 | 含义 |
| dev\_id | int | 设备ID，必须唯一 |
| dev\_name | str | 设备名 |
| dev\_weight | int | 设备权重，权重越大的可容纳的分区数量越多 |
| part\_num | int | 设备中的分区数量 |

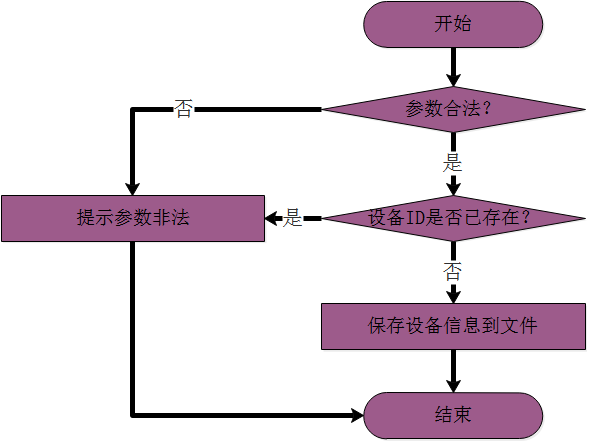
### 2.1.2.Ring File结构设计

Ring File包含虚拟设备（partition）到物理设备的映射关系，其结构自行定义。

这里值得一提的是，一般而言，虚拟设备到物理设备的映射关系通常是固定的，那是不是只需要Builder File记录相关信息就可以了。如果只是本次课题，那当然可以，但实际生产环境中，虚拟设备和物理设备的量级往往非常庞大，映射策略也比较复杂，仅记录设备信息，等到实际Hash时再动态生成映射关系，将会比较耗时，且可能存在大量重复计算带来的无谓的性能开销，所以还是提前处理为妙。

## 2.3.流程设计

### 2.3.1.添加device



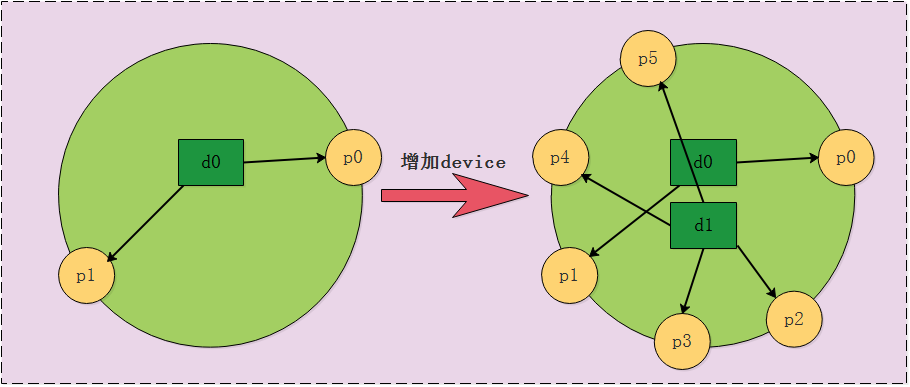
### 2.3.2.移除device

移除device的流程和添加device类似，需要修改并更新Builder File。

### 2.3.3.rebalance

rebalance主要是在删除和增加物理设备后，对虚拟设备和物理设备的映射关系进行rebalance。

下图展示了在已有device0（weight 100）的情况下，添加device1（weight 200）后rebalance的过程：



### 2.3.4.Builder File读写管理

增加、删除和编辑设备权重都是基于文件来操作的，需提供对应读写接口。

### 2.3.5.Ring File读写管理

rebalance后，需更新虚拟设备到物理设备的映射关系到Ring File，需提供对应读写接口。