Master

# 程序结构简介

# 底层算法和数据结构

## 一致性哈希算法

在Master收到Client端发送的读写请求之后，为了保证各个Cache的负载均衡以及保证缓存能够有效命中存有缓存数据的Cache，需要利用一致性哈希算法来计算Client端发送的Key和IP的映射关系。一致性哈希算法是通过将每一个IP地址通过哈希算法映射成均匀的哈希值之后，将其均匀的分布在哈希环上面。

当我们收到用户请求的Key之后，通过哈希算法将其映射到环上，然后顺时针找到距离其最近的服务节点作为该Key将要与其通信的节点。所以，利用一致性哈希算法，当某个服务节点挂机之后，仅仅影响到距离他顺时针的最近的节点的负载，而不至于全体的Cache为此付出代价,有效的防止了缓存雪崩的问题。但是当Cache的节点比较少的时候，由于用户的Key无法均匀地铺满整个哈希环，所以会出现数据倾斜地问题，也就是部分节点的负载压力较高。

针对数据倾斜问题，可以采用虚拟节点到真实节点映射的方法将虚拟节点均匀地布满哈希环，这样用户传来的请求通过虚拟节点映射到真实节点之后，可以有效地将负载均衡。

一致性哈希算法的模式图如图2-2所示。

## cache映射

为了保存每个上线的cache的相关信息，将socket连接建立时分配的文件描述符fd与fdmap结构体指针映射起来，数据结构为std::unordered\_map<int, struct fdmap \*> caches\_list;。用fd作为键值是考虑到当有cache发送信息时，其fd是可以直接拿到，通过访问其映射的结构体内容，就能直接得到cache的主备份信息，端口信息等，便于进行数据处理。

fdmap的数据结构：

struct fdmap {

char status; // 'P'/'R'，主备份状态

int fd; // socket的文件描述符

// int vec; //待使用的实际节点映射

int pair\_fd; // pair fd

std::string ip\_port; // IP#PORT\_for\_client

std::string ip\_cache; // IP#PORT\_for\_cache

fdmap(int fd) :status('n'), fd(fd), pair\_fd(-1), ip\_port("0"), ip\_cache("0"){} //无参数的构造函数数组初始化时调用

};

## 存活检测

cache server会周期性向Master发送心跳包，Master接收到cache的心跳包就会更新该cache对应的时间戳，数据结构为：unordered\_map<string, time\_t> cacheAddrMap；。

Master为了确定cache server 的存活，会周期性扫描这个字典，确保接收到最后一次心跳的时间与扫描时刻的差值不超过一个门限。若差值超过门限，说明该cache可能意外关闭，此时Master需要检查该掉线的cache的主备份状态，如果是备份cache，将其对应的主cache中配对cache删除；如果是主cache，检查是否有备份cache，若有，将其转为主cache，若没有，直接删除节点，然后更新cache节点信息。（流程图见图3-1，如果有的话？？？？？？？）。

# 程序功能介绍

## client查询cache地址功能

Master收到client传输来的key后，计算key存储的cache地址。

首先根据key的具体内容通过一致性哈希计算得到虚拟节点对应的值，将虚拟节点映射到实际节点的索引，并根据通过fdmap索引查询到实际节点对应cache的ip和port。最后将ip和port的格式转化为“ip:port”，将该信息返回给client。

## 主备份分配功能

1、当检测到cache上线，添加给它分配的fd到fd\_node列表，并初始化该fd在cache\_list 中的映射值，即结构体struct fdmap，用于保存该cache的相关信息。

2、当收到心跳包后，若cache相关信息不完整，说明是第一次发送信息，根据心跳包内容补充该cache的相关信息，并尝试分配主备份，具体如下。

如果上线的是主cache\_1，判断是否有未配对的备份cache\_2（未配对的备份cache保存在堆栈中，检查堆栈是否非空即可）。如果有备份cache\_2，则设置cache\_1的对应cache为cache\_2, cache\_2的对应cache为cache\_1，弹出备份cache栈（rcache）中的cache\_2。如果没有备份cache，则将主cache\_1放到主cache栈（pcache）中。

在分配完主备份后，master向cache进行回复，通信格式为：配对不成功：“P#None”；配对成功：“P#rcache\_IP#port\_for\_cache”

如果上线的是备份cache\_2，判断是否有未配对的主cache\_1（未配对的主cache与未配对的备份cache处理类似）。如果有主cache\_1，则设置cache\_2的对应cache为cache\_1, cache\_1的对应cache为cache\_2，弹出主cache栈（pcache）中的cache\_1。如果没有备份cache，则将主cache\_2放到备份cache栈（rcache）中。

在分配完主备份后，master向cache进行回复，通信格式为：配对不成功：“R#None”；配对成功：“R#pcache\_IP#port\_for\_cache”

## 心跳功能

进行心跳时间戳更新：master从cache处接收格式为【x#ip#port】的心跳包，对接收信息进行解析，获取该cache对应的ip和port，获取当前的时间戳，更新cacheAddrMap中该cache对应的时间戳。（cacheAddrMap<cache的ip和port， 时间戳>）

检测cache是否存活：对所有给该部分传输cache对应的fd， 根据fd获取该cache的ip和port，通过cacheAddrMap获取该cache现存的时间戳值；获取当前的时间戳；对当前的时间戳和存储的时间戳做差值，当差值不超过设置的心跳间隔时间，则该cache存活，否则不存活。

## cache容灾功能（当cache发生不存活时，完成主备份cache的切换）

1）如果是主cache\_1掉线

将备份cache\_2中的配对cachefd设置为-1，并将状态设置为P（主cache），从而完成备份cache切换为主cache；将cache\_2设置为待配对状态；将fd列表（fd\_node）中主cache\_1的ip和port改为备份cache\_2的ip和port；最后进行信息通知，具体通知为： 如果cache\_1有备份cache\_2，则master通知备份cache\_2现在是主cache，通知所有cache，将原本存ip\_port的数据里，cache\_1的位置更新为cache\_2的位置；如果cache\_1没有备份cache，则master通知所有cache，将原本存ip\_port的数据里,cache\_1的数据删除。

2）如果是备份cache\_2掉线

Master设置本地cache列表（cache\_list），删除备份cache\_2对应的主cache\_1的备份，并将cache\_1设置为待配对状态；删除master中cache列表的备份cache\_2；master通知主cache，它没有备份cache了。

## 扩缩容功能

缩容：首先根据fd列表（fd\_node）获取要删除的cache索引（默认删除最后一个cache），通过一致性哈希中删除节点算法，设置哈希运算。之后根据cache索引获取到缩容cache的ip（killed\_ip）和port(killed\_port)；将killed\_ip和killed\_port以【K#killed\_ip#killed\_port】格式广播给所有的cache。最后减少cachefd列表（fd\_node）中缩容cache的fd；删除cache列表（cache\_list）中缩容cache的信息；关闭缩容cache通信；如果存在备份cache，删除cache列表（cache\_list）中缩容cache的备份cache的信息；关闭备份cache通信。

扩容：当上线的cache为主cache时，进行扩容操作。首先通过一致性哈希中增加节点算法，设置哈希运算。之后根据新上线cache的fd值获取该fd对应的cache的ip(new\_ip)和port(new\_port)，将ip和port以【N#new\_ip#new\_port】格式广播给所有的cache。最后将所有的cache的ip和port以【ip1#port1#ip2#port2...】格式发送给新上线的cache。

# 程序模块简介

## Master-client通信模块

Master对Client的通信在start\_client()中实现，利用IO复用技术，实现socket监听以及与客户端的socket通信。为了方便通信读写，每个cache的socket连接都会分配一个文件描述符fd。

具体处理的工作为：接收Client发送的读/写key 请求，返回该key对应的cache服务器地址。

## Master-cache通信模块

Master对Cache的通信在start\_cache()中实现，基本原理和Master对Client的通信类似。由于需要向完成1）Client分配地址，2）进行一致性哈希运算，3）容灾几项工作，Master需要记录各个Cache的基本信息，包括1）对cache的地址，2）对client对地址，3）主备份状态，4）对应的主备份cache的fd。

具体处理的工作有：

1. 接收Cache的心跳包，回复为其分配的主/备份cache的地址；发送的读/写key 请求，返回该key对应读cache服务器地址。
2. 扩容时，向新上线的主cache发送已有的cache节点信息，并向所有cache节点发送新上线节点的消息
3. 缩容时，向所有cache节点发送删除的cache节点的消息
4. 容灾时，在检测到有主cache失效时，广播发送cache节点的更新消息。

## 周期性心跳检测模块

Master通过心跳检测功能周期性检测所有cache的存活状态，存在不存活的cache时，执行相应的容灾功能。

## 缩容模块

Master循环监听键盘的输入，当输入的字符为’s’ 时，进行缩容操作。