1. 程序结构简介

缓存程序(下文中简称cache)，在整个缓存系统中承担着数据的最终存储的任务。在cache内部的缓存通过最近最少使用（LRU）算法进行管理。它能够接受来自客户端(client)的请求，读出键值或者改变键值，也能够向控制程序(master)通过传送心跳包的方法向其报告自身状况，并根据master的指令传递向其他cache传递扩/缩容信息。

另外，为了保证整个缓存系统的灵活性和可靠性，在cache程序内分别实现了系统的扩容和缩容功能，和缓存的备份和容灾功能。为了实现这些功能，cache程序由数个相互独立的模块构成，分别是LRU缓存模块、一致性哈希模块、cache-cache通信模块、cache-client通信模块和master-client通信模块。

2. 底层算法和数据结构

2.1 LRU缓存算法

程序底层的缓存算法的更新采用最近最少使用（Least Recently Used, LRU）算法进行维护。该算法的原理是，将一段时间内最近最少被使用的数据剔除出缓存中。

2.2 一致性哈希算法

2.3 地址表

3. 程序功能

3.1 程序功能简介

为了实现缓存系统的键值管理、键值查询功能，并且为了保证缓存系统的可靠性和灵活性，在缓存程序中实现了键值查询功能、扩容和缩容功能、容灾功能、心跳包发送功能。其中，键值查询功能用来处理来自客户端的请求，并根据客户端的请求更改缓存或查询缓存中的值。扩容和缩容功能用来根据控制端的请求，进行缓存的扩容和缩容中涉及到的数据传输工作。容灾功能实现了主缓存模块和备份模块之间的同步和通信功能。一旦主缓存意外失效，备份缓存将第一时间顶替主缓存的位置。

3.2 键值查询功能

程序的键值查询功能是由LRU模块提供的。对于LRU模块而言，它提供了两个方法，分别是put(key, val),get(key),分别对应着更改值和查询值的功能。当客户端发出请求后，会通过报文的形式通知给缓存端，缓存端则通过IO复用的方法将这些不同的客户端所发出的请求打包成一个一个任务，并传到线程池中。

在线程池中，有数个事先启动并等待任务执行的工作线程。任务在被加入到线程池的任务队列后，会被一一派发给这些工作线程。在工作线程的启动和运行方面，采用了future-promise机制实现了线程的调用和同步。这些线程在完成任务后，会被统一将任务结果发送给future队列中，再由主线程统一应答。线程池的工作原理如图所示。

3.3 扩容和缩容功能

缓存中的扩容和缩容功能主要是依靠缓存间的通信模块和一致性哈希模块实现的。当控制端发出扩容命令时，只需要将新增的缓存节点加入自身的地址表中即可，反之，控制端发出缩容命令时，则需要指定一个节点进行数据转移和关停。此时，则需要分配该节点所掌握的所有键值对。

在程序中，采用了一致性哈希算法实现了键值对的分配。在节点收到关停命令时，节点将会利用一致性哈希算法，根据自身的地址表，为每个键值对分配一个地址，并将该键值对发送到对应的缓存中去。此时，节点完成了数据的转移，并顺利关停。

对于该节点的备份节点而言，当控制端向主节点发送关停命令时，也会向所有缓存节点广播将亡节点的地址。此时，备份节点将会将该地址与自己的主节点的地址进行比较，如果二者一致，则意味着该备份节点也将会被关停。



图 缩容时的通信时序图

3.4 心跳功能

3.5 容灾功能

为了提高缓存系统的可靠性，程序为每个主节点设置了一个备份节点。主节点和备份节点一同产生，且一同受控制端关停。当主节点接收到客户端的请求后，主节点也会将该请求转发给备份节点，使二者的缓存是同步的。当主节点意外关停，控制端无法接收到来自主节点的心跳时，控制端则会通知备份节点转为主节点，并将该节点的地址广播给其他的缓存，使得其他缓存更新自己的地址表。容灾功能的模式图如图所示：



（a） 正常情况下的主节点和备份节点之间的通信



（b） 主节点意外关停时的通信

4. 程序模块简介

4.1 LRU缓存模块

4.2 cache-cache通信模块

4.3 client-cache通信模块

4.4 master-cache通信模块