## 分布式的服务器框架

整体框架：

一个负载集群+后台N个服务器集群。负载集群可以根据消息内容进行分发。比如图片下载，电影下载都专门发给后台A服务器集群。如果是充值的消息，我们也可以进行单独处理，用专门的服务器集群进行管理。还有很多很多，我做的只是一个框架，还有很多后台服务器需要自己扩充，扩充不同类别的服务器之后，需要修改相应的代码，进行专门的转发。

**每个DtbtNginx都监听三个端口：**

1.用于DtbtNginx内部消息转发

2.用于监听后台server的消息（当前系统没有使用到这个功能，因为后台就只有web服务器而已。但是接口我都是设计好了的，将来如果有特殊的服务器，那么利用protobuf进行通信）

3.监听客服端的请求

客服端的请求会先经过父进程，父进程通知子进程去accept，然后子进程根据Consistent Hash可以先出一个服务器进行处理。

每个子进程都有可能去连接后台的每一个服务器，因为每个子进程都可以看做是单独的服务器进行负载。

服务器负载会均衡么？答：父进程通知子进程使用轮循算法，Consistent Hash也是一个负载均衡的算法，所以最终所有子进程对服务器应该负载都是比较均衡。

那既然每个子进程功能都一样，为什么不干脆使用单进程？答：其实单进程没有任何问题。redis就是单进程，性能，吞吐量照样很高，但是有个缺点就是：无法发挥多核的优势。所以针对于这个问题，我使用了多进程，有几个核就开几个子进程。

做成分布式的目的：

1.防止负载均衡挂掉，导致整个系统都不能使用

2.防止后台服务器挂掉，可以缓存一部分后台服务器的数据在负载集群中

比如你在文件下载的时候，你可以把下载的位置除了原本的服务器有备份，负载集群中也备份一个

可以为每个后台服务器创建一个缓冲区，备份用户的数据，以免后台主机挂掉

Note:客户端怎么知道连接哪个？通过dns可以自动选择leader进行连接，dns服务器相当于第一层负载均衡

用多进程的目的：

先说说为什么一定要用进程池，最开始一直用的都是进程池，而且nginx也是用的进程池。但是在写分布式的时候出现一个问题，就是有些数据需要共享，那么每个子进程之间就需要用共享内存来进行数据共享，而且还必须的加锁。最开始我也这样尝试过，但越写越感觉很low，所以又想着用线程池，但是线程池有一些地方还是比不上进程池。比如：

*1.程序的健壮性很强，一个进程挂了，我再创建一个新的就好，不会把整个程序给拖垮*

*2.通过增加CPU，就可以容易扩充性能*

*3.可以尽量减少线程加锁/解锁的影响*

*4.从资源的占有量上来说比线程池大很多*

但是用进程池得共享数据，还要加锁。怎么办呢？于是我想到了管道通信，父进程通知子进程进行数据的增删改查就好了，也不需要共享数据，更不需要加锁。于是这个问题就这样解决了。Note：当前系统并不需要数据共享，所以只是提供了接口和思路，如果将来要共享什么数据，需要自己修改代码。

会话保持：

遇到一个问题，client可以通过map找到对应的server，因为client就只有一个，但是server发来的消息负载怎么知道发给哪个client呢？由于代码是非阻塞的，所以不能一直等到数据发过来。

所以我想到利用一个list链表，里面每个节点都是(client, server)，这样我每次把发过来的数据都发给链表的第一个节点就好。但是如果server有很多呢？那么每次就查找链表中第一个和severfd匹配的节点，那么对应的client就一定是需要转发的client。

**思考**：1.不会产生第2个client发的数据是第一次返回来的消息么？答：由于使用的是tcp，不可能乱序。

2.那这样做如果第一个节点的数据请求阻塞了，后面请求的数据都会被耽搁？答：不会阻塞，如果出错会直接返回错误码和错误信息，不存在前面的节点消息把后面的阻塞掉。

3.性能上影响大么？答：基本没有影响，就算是高并发也只是多占用点内存罢了，因为每次基本都是发给链表前面几个节点中的某一个。加上使用的是链表，删除只需要O(1)时间，可以说对性能基本没啥影响。