透过Session服务学习MCP框架

by derekzhao

本文分享作者透过Session服务学习MCP框架的主要学习线路就。当然，只看干巴巴的文字是没有太多作用的，建议像作者一样的新人，阅读一遍本文，再按照这个学习重点，去阅读代码，理解各个模块和关键类。

另外如果没有socket编程的基础，建议也在学习MCP之前先把socket编程和Linux网络编程的基础学习一遍。

**1.       Session服务简介**

Session服务为登陆鉴权提供一个统一的缓存服务，采用Key-Value方式存储，通过Session API进行统一的访问。

在用户首次登陆时，用户cookies不带SKey（Session Key，Session中存储的数据的Key，以下都使用SKey表示），登陆CGI会调用Session API获得一个可用的SKey（生成方式后面介绍），使用这个SKey来存储用户的数据。其后的用户访问行为，对应的CGI都会通过用户cookies中携带的SKey来获得用户的信息。

除了缓存用户数据以外，Session服务还要做另外一件事情：将用户访问行为，上报到后端审计系统。

简单了解了session服务后，下面说说我透过Session服务理解的MCP框架是如何工作的。

**2.       MCP框架总体结构**

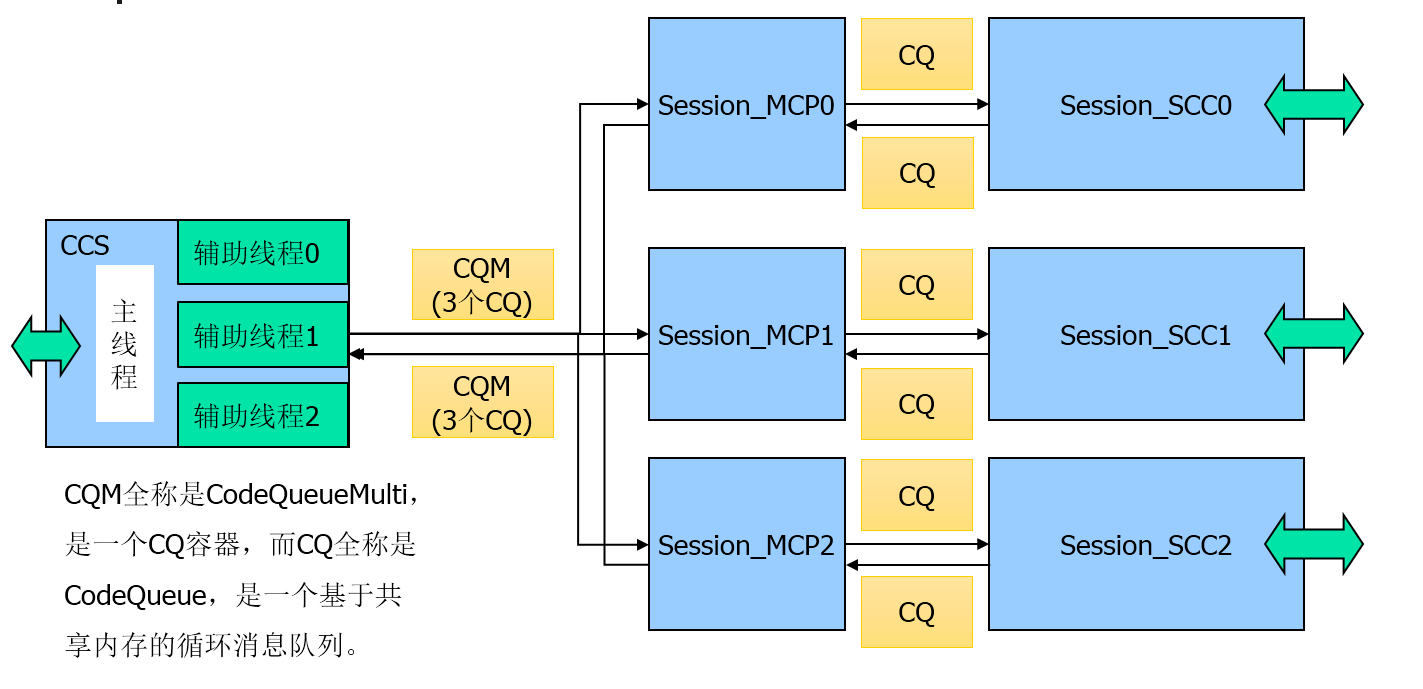


图 1

         图1中每个蓝色方块代表一个进程。

         MCP框架分为三个模块，将具体业务逻辑从前端网络收发和后端服务请求响应中解耦出来，

  CCS模块：负责前方客户端的网络接入和连接管理，以及带缓存池的非阻塞式网络包收发（充当服务器端角色）。

  MCP模块：负责处理具体业务逻辑的模块。

  SCC模块：负责向后端服务进行非阻塞、带缓存池的请求发送和响应接收的模块，也会对TCP连接进行管理（充当客户端角色）

**3.       循环消息队列CCodeQueue**

CCodeQueue对象利用了共享内存，命名管道，实现了高效的循环消息队列，消息队列可以带锁，也可以无锁（当消息存取各只有一个线程或进程时，就无需用锁，因为此时队列的头和尾的修改者各只有一个，图1中的CQ均为无锁）。

此外当CQ对象内有消息时，取方可以直接访问共享内存取出消息；当CQ内消息被消耗完时，取方可以通过命名管道描述符进行同步等待通知（可设置超时时间），当CQ在发方对消息放入时，会检查CQ内消息是否由无到有，如果是，CQ会给命名管道写一个通知（注意，这里有可能出现脏通知的情况，即收到通知，但CQ内没有消息，但MCP框架的服务逻辑不会因为脏通知出现问题，有兴趣的可以思考一下这个细节）。

从图1中可以看到，每两个进程间，是通过两个CQ来进行消息传递的，在这里，一个消息也可以认为是一个包，例如CCS和Session\_MCP0之间有两个CQ，一个CQ是用于CCS传递客户端的请求包到Session\_MCP0，一个CQ是用于Session\_MCP0把回复包传递给CCS向客户端进行回复。

**4.       CCS模块**

CCS负责连接的分发和管理，CCS内部有三个辅助线程，分别对应三个MCP进程，一对一合作，为他们管理分给他们的连接，收发网络包。当有一个新连接fdx过来时，CCS主线程选择一个当前负责连接数最少的辅助线程threadx，并把fdx交给该辅助线程管理，后续从fdx对应的客户端发来的包，都由threadx放到CQ中，合作的MCPx再从CQ中取出进行处理，同样，后续回复给客户端的包，都是由MCPx放到CQ中，threadx辅助线程再从CQ中取出回复包并发给客户端。

CCS内主线程与辅助线程间的通信也是通过CQ，与图中其他CQ的区别只是：图中其他CQ负责的通信是双向的，而CCS主线程与每个辅助线程的通信是单向的，因为只有主线程给辅助线程交代辅助线程需要负责的连接这一个通信需求。

所以CCS内主线程负责的主要工作是：监听TCP端口，接受TCP连接，并分发连接给辅助线程。

而CCS内辅助线程要做的四件主要工作是：

  处理来自CCS主线程的消息，加入新的要负责的连接；

  处理来自对应Session\_MCP的消息（从CQ中取），向客户端发送回复包；

  处理来自TCP客户端的消息，通过CQ交给合作的Session\_MCP。

  放弃不活跃的连接，如果配置文件中有配置的话，会定时检查并删除超过一定时间没有消息交互的连接。

上面一直说的都是TCP连接，那对于UDP包CCS模块是怎么处理的呢？一个好的框架不可能只能处理TCP连接吧？

不用怀疑，CCS对UDP当然也是支持的，对于UDP包，CCS模块是在初始化时，就将固定的UDP协议需要监听的IP:Port加入到每个辅助线程的监听集中，当辅助线程工作时，对每个UDP包，谁先抢到，就归谁处理了。当然，处理也只是将UDP包放到对应合作的Session\_MCP的CQ中，辅助线程本身不会对这个包或者这个请求进行类似TCP连接的管理。

所以，如果我们不对CCS进行修改的话，MCP框架只支持传输层基于TCP和UDP的服务。

**5.       MCP模块**

MCP模块负责的就是对包进行处理，也就是服务的具体业务逻辑。

对于Session\_MCP，他要做的主要事情是：

  处理来自CCS的请求包（取自CQ），Session\_MCP需要处理的有两种请求包：

                     i.            一种是系统内部管理的包，包括心跳请求包、更新CKV配置请求包、配置重载请求包

                   ii.            另一种是缓存服务相关的请求包，包括对缓存进行set、get、del（需要异步请求后端CKV服务），还有QueryNode、QueryMemcached（不需后端服务）。

  处理来自后端CKV的响应包，对异步请求CKV返回的响应包，进行不同的处理后（文章前面提到的上报审计也是在这里进行的），通过CQ发给CCS进行发包。

  清除超时的后端CKV异步调用事务。

当Session\_MCP在某次循环中，CCS和后端CKV都没有包到达（对应两个CQ中没有消息）时，则Session\_MCP会同步等待通知（超时为1秒）。

要理解CCS和MCP的合作，需要理解关键类CodeQueue，而要理解MCP和SCC的合作除了CodeQueue以外，还有两个类需要学习：CTimerInfo和CAnsyTimerQueue，其中更关键的是CTimerInfo。

**6.       异步事务基类CTimerInfo**

异步事务基类是在MCP模块中，用于管理向后端服务的请求事务的。

每种向后端服务的请求事务（例如向CKV的get、set、del等），都需要继承CTimerInfo，并按时实现其中两个关键的成员函数（virtual函数）：

  OnMessage：收到响应后的处理函数

  OnExpire：超时后的处理函数

而CAnsyTimerQueue，可以认为是异步事务的一个容器，MCP模块中维护一个这样的容器，将已发出请求的事务加入容器中，并在MCP模块中定时处理并删除其中超时的事务（CAnsyTimerQueue的实现是一个按过期时间有序的队列）。

所以，继承CTimerInfo的异步事务类，其实还需要在被加入CAnsyTimerQueue前，成功向后端服务发出请求。Session服务中为此在所有异步事务类中，都实现了一个Begin成员函数，创建成功一个异步事务后，先成功调用Begin，再将异步事务加入CAnsyTimerQueue。

         所以Session\_MCP与SCC的合作流程应该如下：

a、   根据需要创建一个异步事务；

b、   调用异步事务的Begin函数，向后端服务发送请求（就是将请求包放到与SCC相连的CQ中），如果成功，则将其加入CAnsyTimerQueue；

c、   MCP在TimeTick函数中，处理并删除CAnsyTimerQueue中超时的事务，对它们调用其OnExpire成员函数（这里超时可以选择重新放入，也可以选择直接删除）。

d、   如果收到后端服务的响应（与SCC相连的CQ中的消息），从CAnsyTimerQueue取出事务，并调用其OnMessage进行下一步处理。

**7.       SCC模块**

SCC负责后后端服务发送请求包和接收响应包，并管理与后端服务的连接，在Session服务中，每个SCC进程里只有一个主线程，其服务对象只有一个MCP模块，如图SCC0只为MCP0发送请求、接收响应，他们之间的数据通信，如前面所知，也是通过两个CQ来进行的。

所以每个SCC负责的主要工作是：

  处理来自后端服务的响应，通过CQ交给合作的Session\_MCP；

  处理来自对应Session\_MCP的消息（从CQ中取），向后端服务发送请求包；

  处理来自内部管理的请求，将SCC统计数据发送过去响应。

  放弃不活跃的连接，如果配置文件中有配置的话，会定时检查并删除超过一定时间没有消息交互的连接。

跟CCS一样，在不修改SCC代码的前提下，SCC在传输层只支持TCP和UDP协议，并且同样的，对于UDP的请求和响应，SCC不会使用缓存池。但是不一样的是，SCC会记下后端基于UDP服务的套接字，方便下次请求（超过一定时间没有数据交互的套接字，会被关闭）。

**8.       其余需要学习的类**

  CEPollFlow：对epoll进行封装，实现多路选择，对文件描述符集进行监听的组件。这个组件建议先看一遍，再开始MCP学习，特别是对于像作者一样本来对epoll没有了解的新人。

  IdxObjMng.cpp中的类：其中核心是对象管理器（TIdxObjMng），基于对象管理器，可以把对象组织成哈希表（CHashTab）、缓冲器（CBuffMng）、树（CTree）等。CCS和SCC用到了缓冲器管理收发包缓冲区，另外SCC还用到哈希表管理文件描述符（CCS管理文件描述符是用一个数组，对文件描述符简单取模哈希出在数组中的位置，没有用对象管理器中的哈希表）。这个组件作者目前也没有仔细看一遍，对于学习MCP，只要了解IdxObjMng.cpp里面的东西是干什么用就可以了，不过这些组件都是非常有学习价值的，有空的时候应该要仔细阅读一下代码。

作者对MCP的学习路线基本上如上所述，对于新人，刚来首先第一件事真的应该是学习MCP框架，希望本文对大家学习MCP框架有所帮助。