综合业务平台中业务系统与AAA服务器通信的性能优化设计

Optimal Design of Performance for Communication between Service System and AAA Server in Integrated Service Platform

邓中亮 谢 鑫 (北京邮电大学 电子工程学院 北京 100876)

摘要:为了提高业务系统与AAA服务器通信的性能,提出了一种基于对象池技术的优化模型:将socket连接对象置入对象池中,使得多个客户的连接共享。经实验网测试,在满足电信运营商高话务量和高数据通信流量的前提下使得系统平均响应时间缩短,占用系统资源量降低。同时采用面向对象设计模式和类层次结构技术,使得优化模型的模块间耦合度大幅度降低,提高了系统的可扩展性和可维护性。

关键词:综合业务平台 AAA服务器 对象池 设计模式

1 引言

随着通信技术和互联网技术的迅速发展,互联网开始向以IPv6为基础的下一代互联网(NGI)演进,而全球电信网络开始向下一代网络(NGN)演进。两个网络一方面出现了越来越大的融合趋势,同时也提供了更强大的承载通道和丰富的业务支持能力。但是,数据业务在给运营商带来收益的同时,也给运营商的管理带来了巨大的挑战[1]。

综合业务平台(ISP)作为下一代网络中业务开发和运行的开放性业务平台应运而生。它打破了现有的单个业务一套平台的"烟囱"结构,在屏蔽了底层网络的差别的基础上以Parlay X标准提供多种电信网络接口,为业务开发者提供业务开发和运行的环境。业务系统(SS, Service System)是ISP的主要组成部分。它以IP骨干网作为各种异构网络(固定电话网PSTN,移动网2G、2.5G、3G及Wi-Fi、Wi-MAX,分组数据网)融合的基础,集成多类承载网络的各种终端和网络接入能力,并基于Parlay API、Web Service等技术,提供开放的、统一的标准协议接口,综合提供多种类型增值业务。AAA(Authentication, Authorization and Accounting)服务器对接入网络的用户(包括最终用户和SP/CP)进行认证、鉴权,并且支持多种用户类型和业务属性,提供灵活的计费方式。

AAA系统的上述作用,决定了整个通用业务平台是否具有可运营性;而电信网上大话务量和数据流量下AAA服务器与SS通信的效率高低,将直接影响到整个平台的性能。高稳定性、高可用性、可扩展性、容错性等指标都是各大希望建设综合业务平台的运营商十分关心的,也是我们设计和优化的目标。由于SS与AAA通信部分的实现主要取决于各大运营商的内部协议规范,其严格定义了通用业务平台运行时,业务触发的SS与AAA通信消息的时序。这些规范所定义的内容不能进行擅自改动,否则将对系统的互联互通性造成不利影响。

根据需求,必须在SS和AAA服务器之间建立一条TCP通道,然后根据规范将交互信息传送到目的设备的目的进程处理。在电信网中大话务量和大流量的条件下,SS可能会在同一时间收到大量业务请求,这些业务将触发更多认证、鉴权和计费消息。虽然长连接TCP通道可以减少Socket建立的次数,但是经过测算,对于电信级的大型服务器来说,频繁建立Socket连接本身还是占用大量耗费系统和网络资源,如果不采取

相应优化方案，将会导致综合业务平台（ISP）整体性能下降，最终由于资源耗尽而瘫痪[2]。

本文将针对该问题，从共享连接对象的角度出发提出基于 Object Pool 模式的优化模型，并给出了在实验网上的测试结果。

2　基于 Object Pool 的优化模型

2.1　解决方案

为了避免上述情况发生，达到能够统一管理客户请求的目的，在 SS 与 AAA 服务器之间采用了对象池（Object Pool）技术。

在 ISP 启动的时候，对象池根据参数文件首先跟请求处理层建立 n 个 socket 连接并放入对象池中。当 SS 同 AAA 有连结请求的时候，从对象池中申请一个 socket 连接；SS 与 AAA 数据交换完成后，连接立即放回到对象池内。对象池同时还负责处理 socket 连接对象的生命周期，根据系统负载动态地增加和减少池中的连结数。利用 Object Pool 技术不仅可以加快建立连接的速度，而且降低了应用服务器的负荷，使系统更加稳定和可靠。

另外 socket 连接对象是无状态的，后一次从连接池中取出连接对象的用户无需知道前次连接的细节，同时并不为现在获得的连接对象维护任何相关状态信息。因此，socket 对象在每次被客户端调用之后，可以马上为其它的客户端服务。无状态对象的这种特性，使它可以服务众多的客户端，省去建立 socket 连接的时间，提高了系统的实时性，同时也对突发业务流量进行了很好的缓冲。

2.2　模型设计与实现

在大型软件系统中，模块设计必须从软件的复用性、降低模块间耦合度的角度出发。因此 Object Pool 根据具体需求采用类层次结构和设计模式思想。而在面向对象设计模式当中，曾经总结出对象池模式，该模式分为 3 个部分，其类图如图 1 所示。

（1）PoolableObject：被复用的对象或资源。

（2）Client：复用对象的使用者。一个 client 对象可以使用多个复用对象，一个复用对象只能同时被一个 client 使用。

（3）ObjectPool：对象池，也即是复用对象的容器或集合，它用来管理和存储可以复用的对象。一般来说，为了保证对象复用的安全，对象池将按照 singleton（单件）的模式设计为全局唯一实例[3]。

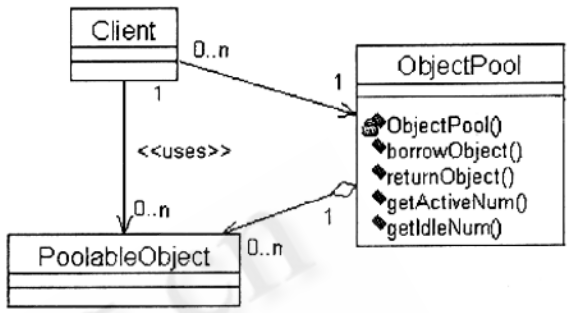

图 1　Object Pool 模式类图

根据上述结构，关键是设计对象池和其中所容纳的对象。首先设计对象池接口 ObjectPool，只要实现了该接口的类，都可以作为对象池处理，而 Socket 连接池则作为实现该接口的具体类。对象池负责对于其中的对象进行管理和统计，并向其他外部提供存取对象方法[4]。ObjectPool 接口 Java 代码如下：

```
interface ObjectPool
{
  borrowObject( ); //取出对象
  returnObject( ); //放入对象
  clear( ); //清除池中对象
  close( ); //清除池中对象并释放池资源
  getActiveNum( ); //得到池中活动对象数
  getIdleNum( ); //得到池中空闲对象数
}
```

而 ObjectPoolFactory 接口利用工厂方法（Factory Method）设计模式，让子类去决定实例化所需要的类，这样就可以将实例化延迟到子类中，使动态创建符合 ObjectPool 接口的对象成为可能，同时也降低了代码类层次结构中的父类和子类的耦合度，提高了灵活性。Java 代码为：

```
Interface ObjectPoolFactory
{
  createPool( ); //创建对象池
}
```

同时能够放入对象池中的对象也要满足一些接口要求，将这些方法设计为工厂类：只要该工厂类包括其子类实例化的对象都可以放入对象池中。这样就可以

让对象池方便的管理其中对象。Java 代码为：

```
interface PoolableObjectFactory
{
  makeObject( ); //生成对象
```

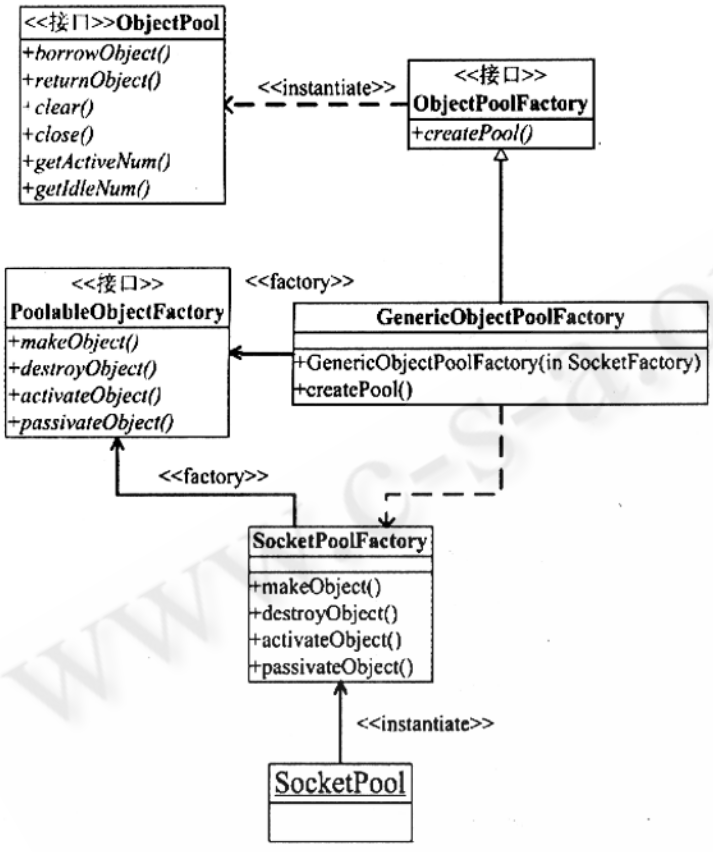

图 2　Socket Pool 模型类图

```
  destroyObject( ); //销毁对象
  activateObject( ); //激活对象
  passivateObject( ); //使对象空闲
}
```

扩展 PoolableObjectFactory，利用 socket 连接的特点实现上述四个方法，得到 SocketFactory 类。该类实例化的对象就是一个个可以放入队相持中的 socket 连接。再将 ObjectPoolFactory 进行方法实现，使其可以接收一个 PoolableObjectFactory 作为参数，Java 代码片段如下：

```
class GenericObjectPoolFactory
{
  GenericObjectPoolFactory(SocketFactory) { … }
  createPool( ) { … }
}
```

这样，通过将 SocketFactory 对象作为参数传入 GenericObjectPoolFactory 的构造方法就可以实例化所需的 SocketPool 对象。这样只要是从 SocketFactory 生成的对象，都可以放入 SocketPool 对象中，完成了 Socket Pool 模型的设计和实现。

上述接口和类层次结构可以用图 2 进行直观的理解。

3　测试结果

某移动运营商实验网平台部分配置如下：AAA 和业务管理服务器（SUN Fire 280R）和业务接入网关服务器（Netrol20）各一台，以 Gigabit Ethernet Network 互联。利用 SP/CP 模拟器，通过业务平台 SendSMS 下发 50000 条短信，经 Mercury LoadRunner Analysis 分析得出优化前后结果对比如表 1 和表 2。

表 1　优化前短信处理时间

	Min	Max	Avg.
TranTime (ms)	5.45	7.08	6.23

表 2　优化后短信处理时间

	Min	Max	Avg.
TranTime (ms)	1.82	2.56	2.19

可以看出，增加了对象池的优化技术后，系统平均短信处理时间从 6ms 下降到 2ms。主要的原因是连接对象池大幅度缩短了建立 socket 连接的时间，同时多个 socket 对象可以在其生命周期内被多个连接所共用。这使得系统吞吐率大幅度提高，同时也大幅度提高了整个平台的运行效率。

4　结束语

针对综合业务平台高性能运行的需求，提出用对象池模式对业务系统和 AAA 服务器之间的通信性能进行优化。运营商实验网上测试表明，该优化模型适合 ISP 业务请求特点，减少了业务平均处理时间，有较大应用前景。

（下转第 99 页）

(上接第 96 页)

参考文献

1 王志军、唐雄燕，移动综合业务平台关键问题研究，电信科学，2006，6：17－20.
Wang Zhijun，Tang Xiongyan. Research on the Mobile Integrated Service Platform. Telecommunications Science，2006，6：17－6.

2 徐萌、孟祥武、陈俊亮、梅翔，综合业务平台负载均衡的研究，北京邮电大学学报，2006，S1：94－97.
Xu Meng，Meng Xiang－wu，Chen Jun－liang，Mei Xiang. Research of Load Balancing for Integrated Service Platform. Journal of Beijing University of Posts and Telecommunications，2006，S1：94－97.

3 水超、李慧，对象池模式的扩展与设计，计算机工程，2004，9：26－27.
Shui Chao，Li Hui. Extension and Implementation of Object pool pattern. Computer Engineering，2004，9：26－27

4 Apache Jakarta Project，Commons Pool http://jakarta.apache.org/commons/pool/guide/index.html.