软件复用讨论课二

徐锦程 1353012

参考业界架构，讨论程序的拓展

* 负载均衡

<http://www.docin.com/p-896261693.html>

* ID分配

http://blog.csdn.net/oldworm/article/details/5372202

* 通信可靠高效

http://www.docin.com/p-1083798283.html

* 协议

http://blog.csdn.net/xnwyd/article/details/7668742

* 消息队列

http://www.360doc.com/content/13/0304/10/10384031\_269182397.shtml

* 垃圾消息过滤

http://www.doc88.com/p-0913739833086.html

* 安全

http://bbs.csdn.net/topics/270042374

负载均衡

可以通过在客户端进行负载均衡的处理，比如自己实现一定的连接策略。通俗点，就是按照自定义的方式获取对服务器的连接。这个可以参考某些jms中间件的实现。一般需要在客户端维护服务器的列表，每次连接时调用策略的来选择服务器。

基于客户端的解决方法需要每个客户程序都有一定的服务器集群的知识，进而把以负载均衡的方式将请求发到不同的服务器。例如，[Netscape Navigator](http://www.baike.com/sowiki/NetscapeNavigator?prd=content_doc_search" \o "Netscape Navigator)浏览器访问[Netscape](http://www.baike.com/sowiki/Netscape?prd=content_doc_search" \o "Netscape)的主页时，它会随机地从一百多台服务器中挑选第N台，最后将请求送往wwwN.netscape.com。然而，这不是很好的解决方法，Netscape只是利用它的Navigator避免了RR-DNS解析的麻烦，当使用IE等其他浏览器不可避免的要进行RR-DNS解析。

[Smart Client](http://www.baike.com/sowiki/SmartClient?prd=content_doc_search" \o "Smart Client)是[Berkeley](http://www.baike.com/sowiki/Berkeley?prd=content_doc_search" \o "Berkeley)做的另一种基于客户端的解决方法。服务提供一个[Java Applet](http://www.baike.com/sowiki/JavaApplet?prd=content_doc_search" \o "Java Applet)在客户方浏览器中运行，Applet向各个服务器发请求来收集服务器的负载等信息，再根据这些信息将客户的请求发到相应的服务器。高可用性也在Applet中实现，当服务器没有响应时，Applet向另一个服务器转发请求。这种方法的透明性不好，Applet向各服务器查询来收集信息会增加额外的网络流量，不具有普遍的适用性。

ID分配

有些时候需要给资源分配一个唯一id(32bit or 64bit or CHAR[N])，首先我们有个基本前提，如果是单线程分配，那么我们无需下面的方法，直接++value即可(CHAR型无论几线程都可使用GUID)顺序产生不重复序列，下面讨论的方法都是多线程下的分配策略：

方法1、 win下做简单的方法莫过于使用InterlocckedIncrement(or InterlockedIncrement64)了，这个调用也很简单，每次递增一个，多线程间保证顺序递增绝无重复。此方法只可在单一进程上使用。

方法2、区间法，每个线程一次申请一个id区间[m, n]，用完了再申请下一个区段，申请的时候锁一次，其他时间都不用锁，效率比3略低，比1高。此方法也只可在一个进程上使用，当然如果申请的策略修改一下也可实现多个进程甚至不同机器上的进程之间独立分配id。

方法3、方法1虽然简单但毕竟InterlockedXXX系列函数调用还是有些耗时的，大概50cpu周期级别，更简单的方法可以使用线程切分原理，如有3个线程参与id分配，我们这样分配：

线程1  base=1, step =3，序列1,4,7,10,…

线程2  base=2, step=3，序列2,5,8,11,…

线程3  base=3, step=3，序列3,6,9,12,…

绝无重复，调用非常简单每个线程id = base; base += step;即可。

此方法在单进程上使用很简单，如果要拓展到多个进程上使用要通过配置来实现，但也是不难的。

方法4、如果id可用GUID表示那么方法要简单一点，生成id直接调用guid生成算法，这个id生成算法即使在多个进程之间甚至不同机器之间也可以保证唯一，也有其价值。

通信可靠高效

考虑到各种网络故障,在基于socket的网络通信应用上，,可以使用可达性测试算法,使得服务器可靠地辨别哪些客户端无法被服务器访问。算法主要包括以下步骤：

1、客户端自身启动一个socket监听线程，然后向服务器请求登录。登录消息中包括客户端的监听端口号，但不包括自己的IP地址。

2、服务端检验客户端的合法性，如果合法，则发送验证成功消息。

3、服务器通过客户端发起的链接获取客户端的IP地址（链接对等端IP地址）以及登录消息中的监听端口号，向客户端发起链接。如果连接不成功，服务器将客户端标记为不可达；如果连接成功，则发送可达性测试报文，然后等待客户端回复。

4、如果客户端收到可达性测试报文，则进行确认回复，服务器收到确认后标记客户端为可达。如果服务器在一定时间后还未收到客户端的回复确认，则标记客户端为不可达。

协议

当用socket进行进程通信，传输数据的时候，会出现以下一些情况：

（1）完整的一条消息被系统拆分成几条发送，例如要发送一条消息：Hello World ，却被系统分成两条消息发送，分别为：Hello 和 World。

（2）几条独立的消息被系统合成一条消息发送，例如要发送两条消息分别为：a memory from my past和it’s been a year，却被系统和成一条消息发送：a memory from my pastit’s been a year。

这个时候，需要为socket通信设计一种通信协议，以保证数据的准确性。

这是网上找到的某种通信协议设计如下：

C:\Users\Administrator\Desktop\1339822494_2865.JPG

Head：帧头，2个字节，此处为0xa5a5

Type：通信类型，1个字节，范围0x00~0xff

Data Length：数据长度，1个字节，即Data的字节总数，

Data：实际传输的数据，长度不定

CS：校验值，1个字节，type、data length、data三个域所有字节的异或值，实际中并没用到校验

End：帧尾，2个字节，此处为0xbeef

假设socket客户端C和服务端S通信，C向S发送消息M1。

1、  S收到消息M1。S把消息M1拷贝到缓存Q中，Q为循环队列。假如M1的长度大于Q的剩余空间，则只拷贝剩余空间大小的字节到Q。

2、  从Q的当前指针开始，查找帧头<Head>。如果找到，则当前指针向后移2个字节位置，继续查找<Type>；如果没找到，则删除前1个字节，当前指针向后移1个字节位置，继续查找<Head>

3、  从Q的当前指针开始，查找<Type>。如果Q中至少还剩一个字节，则表示找到，当前指针向后移1个字节位置，否则退出解析。

4、  从Q的当前指针开始，查找<DataLength>。如果Q中至少还剩一个字节，则表示找到，当前指针向后移1个字节位置，否则退出解析。

5、  从Q的当前指针开始，向后移DataLength个字节位置，查找<End>。如果找到，则从Q中取出一条完整的消息P1，并从Q中删除此消息空间，调用外部的回调函数；否则删除帧头的第一个字节a5，当前指针指向帧头第二个字节a5位置，从步骤2开始，重新一轮解析。

消息队列

[MQ](http://baike.baidu.com/view/732119.htm" \t "_blank)是消费-生产者模型的一个典型的代表，一端往[消息队列](http://baike.baidu.com/view/262473.htm" \t "_blank)中不断写入消息，而另一端则可以读取或者订阅队列中的消息。MQ和[JMS](http://baike.baidu.com/view/157103.htm" \t "_blank)类似，但不同的是JMS是SUN JAVA[消息中间件](http://baike.baidu.com/view/3118541.htm" \t "_blank)服务的一个标准和API定义，而MQ则是遵循了AMQP协议的具体实现和产品。

消息队列上，我们组使用了RabbitMQ。消息队列的使用过程大概如下：

1、客户端连接到消息队列服务器，打开一个channel。  
2、客户端声明一个exchange，并设置相关属性。  
3、客户端声明一个queue，并设置相关属性。  
4、客户端使用routing key，在exchange和queue之间建立好绑定关系。  
5、客户端投递消息到exchange。

6、exchange接收到消息后，就根据消息的key和已经设置的binding，进行消息路由，将消息投递到一个或多个队列里。

rabbitMQ则具有以下优点

1. 基于erlang语言开发具有高可用高并发的优点，适合集群服务器。

2.健壮、稳定、易用、跨平台、支持多种语言、文档齐全。

3.有消息确认机制和持久化机制，可靠性高。

4. RabbitMQ支持消息的持久化，也就是数据写在磁盘上

5.开源

垃圾消息过滤

垃圾消息过滤的方法常常基于以下几个方面

1、垃圾关键词精确匹配（黑名单）

通过分析预料，总结一下反映垃圾消息特征的关键词或者关键短语，例如：“订购热线”、“欢迎咨询”等明显的垃圾广告词语。一旦在评论正文中匹配出黑名单中的词汇或者短语，即判为垃圾评论。

2、重复消息

同一个客户端所发送的消息中，经常会有大量的同一消息，或者在不同客户端发送的消息中，存在大量相似或完全相同的消息。

3、包含字符

消息中包含的文字很少，却含有大量的链接，或者在消息中包含大量的字符或者特殊字符。

垃圾消息过滤的算法包括了许多数据挖掘的内容，如k近邻、朴素贝叶斯、支持向量机、boosting方法等等

安全

Socket在通信的过程中：  
1：不验证用户，无法确定通信对方究竟是谁，只能知道一个IP地址和端口。   
  
2：不加密数据，完全是明文；只要在通信路径上截取到数据报文，就可以解读。

针对以一个问题，通常在服务端指定允许链接的IP地址。当收到一个连接请求时，解析出其IP。如果是指定的IP，就允许链接，否则就refuse掉。但是这样的方法也存在一定的问题，因为IP是完全可以伪造的。

针对第二个问题，可以使用SSL。它是Secure Socket Layer的，即安全的套接层。它就是用于给普通Socket提供安装功能，实现安全的几个主要功能，包括认证，加密数据使数据不可读，防止数据被修改，维持数据完整性等等。具体的实现技术，大体上就是在通信之前先通过数字证书认证身份，并且商定HASH协议，然后在通信时通过对方的公钥来加密发送的数据。只有拥有私钥的人才能解密这个数据。