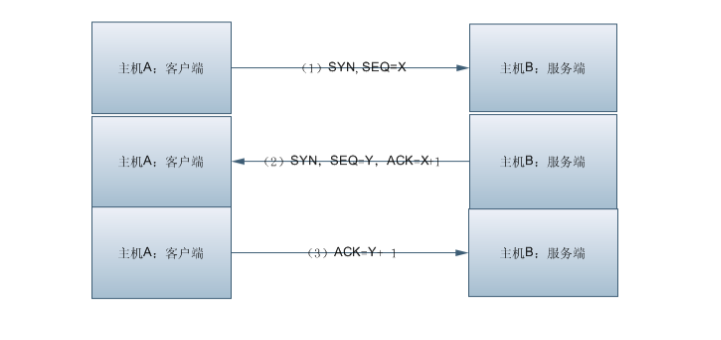
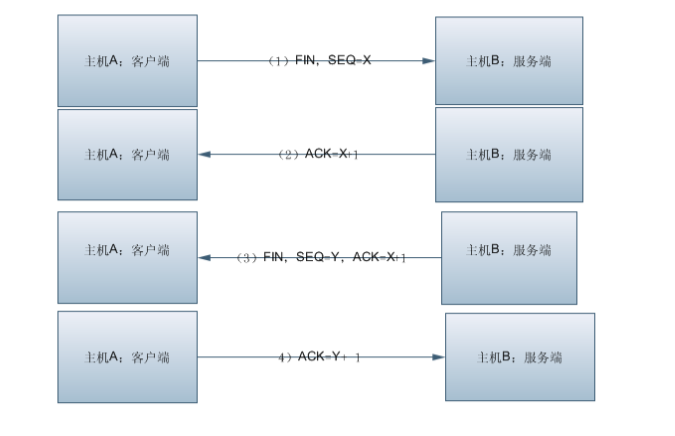
1. Java内存溢出和内存泄漏的区别
   1. Java内存溢出就是指要分配的内存超出了系统能给你的；例如你申请了一个int，但给它赋值long，就会内存溢出；
   2. Java内存泄漏是指程序的逻辑问题造成内存无法释放，这样的话内存会被占用光，例如无限死循环，会导致系统越来越慢，甚至当掉；
2. Java虚拟机的垃圾回收机制
3. Java的Error和Exception的区别和联系
4. NFS——网络文件系统（network file system）
5. 查看线程状态
   1. ps –ef | grep java
   2. jstack 进程号
6. 性能测试
   1. 并发用户数
   2. 虚拟机数
   3. 平均响应时间
   4. 请求丢包率
   5. CPU使用率
   6. 内存使用率
7. java.lang.Thread.State: RUNNABLE
   1. 代码中有循环或者递归造成程序死循环，线程的堆栈会越来越高，一会儿就会内存溢出，导致服务挂机；
   2. 死锁或者阻塞；例如静态变量的死锁或者流未关闭等等；
   3. 在while循环中加入延时函数，释放系统控制权；
   4. HashMap的使用会在多线程高并发中发生死锁问题，建议使用ConcurrentMap
8. CPU100%问题
   1. 服务器CPU一般在75%以下为正常
   2. 造成的原因可能是程序有死循环、IO没释放、静态变量、线程不安全的Java类
9. TCP/IP的三次握手
   1. 第一次握手——客户端首先给服务器端发送一个特殊的TCP数据段，该数据段不包含应用层数据，并将头部中的SYN位设置为1，所以该数据段被称为SYN数据段；另外客户端选择一个初始化序列号SEQ，设SEQ=X，并将这个编号放到初始的TCP SYN数据段的序列号字段中，该数据段被封装到一个IP数据报中，并发送给服务端
   2. 第二次握手——一旦装有TCP SYN数据段的IP数据报到达了服务器主机，服务器将以该数据报中提取出的TCP SYN，给该链接分配TCP缓冲区和变量，并给客户端TCP发送一个允许连接的数据段；这个允许连接的数据段也不包含任何应用层数据；但是，他的头部中含有3个重要信息；首先，SYN被设置为1，其次，TCP数据头部的确认字段被设置为X+1；最后服务端选择自己的初始顺序号，SEQ=y，并将该值改道TCP数据段头部的序列号字段中；
   3. 第三次握手——在接受到允许连接数据段之后，客户端也会给服务端连接分配缓冲区和变量；客户端也会给服务器发送另一个数据段，对服务器的允许连接数据段给出确认；



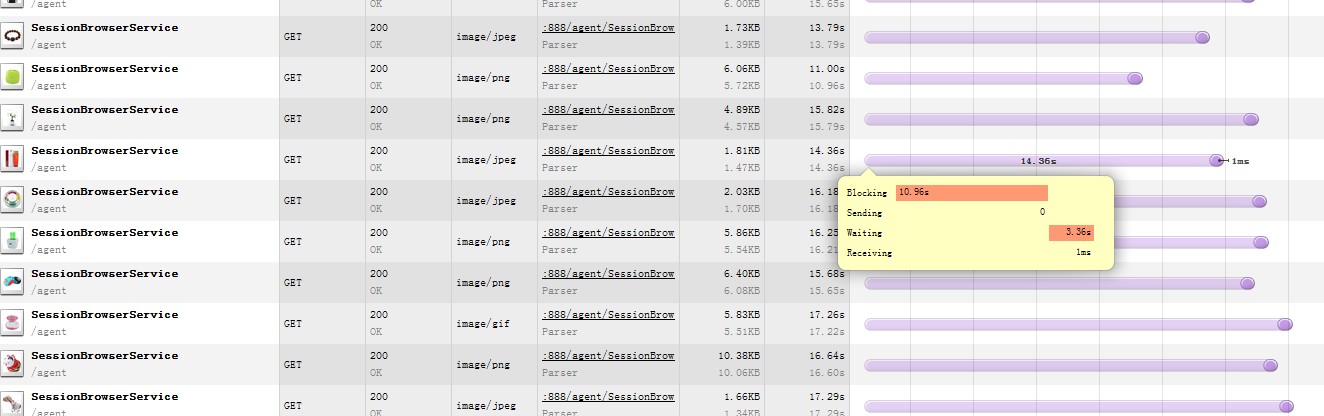
连接的释放过程：

1. 第一次握手——由进行数据通信的任意一方提出要求释放连接的请求报文段
2. 第二次握手——接收端收到请求后，会发送确认报文段，同时当接收端的所有数据数据也都已经发送完毕后，接收端会向发送端发送一个带有自己序号的报文段
3. 第三次握手：发送端收到接收端的要求释放连接的报文段后，发送反向确认



1. SYN——同步（初始）比特，当SYN=1时，表示这是一个连接请求或者连接接受报文
2. FIN——终止比特，用来释放一个连接；当FIN=1时，表示此报文段发送的数据已经发送完毕，并要求释放连接
3. ACK——确认比特，只有当ACK=1时，确认字段才有效，当ACK=0时，确定号无效
4. RST——复位比特，当RST=1时，表明TCP连接中出现严重错误（比如主机崩溃等原因）必须释放连接，通知一下对方；
5. URG——紧急比特，当URG=1时，表明紧急指针字段有效，它告诉系统此报文段中有紧急数据，应尽快传送（相当于优先级高的数据）
6. PSH——推送比特，接收端收到推送比特置1的报文段，就尽快地交付给接收应用进程，而不再等到整个缓存都填满了后再向上交付
7. 移动网络
   1. CMNET——提供了NAT服务，能够直接访问Internet终端服务器
   2. CMWAP——只提供了WAP代理和Http代理，只能用支持这两种协议的软件

1. 对于图片比较多的网站需要对图片域名进行分发，一个域名只能同时支持有限数量的请求；列如淘宝等图片较多的网站。



1. Java反射机制总结
   1. Class类
      1. Class clazz1 = String.class;
      2. String str = “abc”; Class clazz = str.getClass();//对实体类运用的方法
      3. Class clazz3 = Class.forName(“java.lang.String”);
   2. Field类
      1. Field[] fields1 = clazz1.getDeclaredFields();//获取所有申明的字段
      2. Field[] fields2 = clazz1.getFields();//获取所有的公共类字段
      3. Field fields1 = clazz1.getDeclaredFields(“value”);//获取指定字段名（所有申明的字段都可以获取）
      4. Field field2 = clazz1.getField(“CASE\_INSENSITIVE\_ORDER”);
   3. Method类
      1. Method[] method1 = clazz1.getDeclaredMethods();
      2. Method[] method2 = clazz1.getMethods();
      3. Method method1 = clazz1.getDeclaredMethod(“checkBounds”,new Class[]{byte[].class,int.class,int,class});
      4. Method method2 = clazz1.getMethod(“hashCode”);
   4. Constructor构造方法类
      1. Constructor[] cons1 = clazz1.getDeclaredConstructors();
      2. Constructor[] cons2 = class1.getConstructors();
      3. Constructor con1 = clazz1.getContructor(char[].class);
      4. Constructor con2 = clazz1.getDeclaredConstructor(new Class[]{int.class,int.class,char[].class});
   5. 由一个Class得到一个实体类
      1. String obj = (String)clazz1.newInstence();//类必须有默认构造函
   6. 由一个Constructor对象，得到类的实体
      1. String str2 = (String) con1.newInstance(new char[]{‘a’,’b’,’c’});