VNC

# 概述

根据以下网址提供的资料，在是否开源，是否提供针对Web的远程控制接口，是否知名应用范围广，是否持续完善和更新几方面，筛选出几款软件展开详细调研。

主要参考文献：

<http://en.wikipedia.org/wiki/Comparison_of_Java_Remote_Desktop_projects>

<http://en.wikipedia.org/wiki/Comparison_of_remote_desktop_software>

其中大多数软件基于realVNC，tightVNC，UltraVNC开发。因此针对其进行详细调研。

# VNC配置

**Windows：**

直接双击VNC-5.0.1-Windows.exe，选择安装server和viewer，设置用户名和密码，端口为5800（web），5900（viewer）

**Linux：**

CentOS自带的应用程序->附件->vnc viewer

可使用linux自带的vnc server，无法通过浏览器访问。

**安装VNC Server：**

下载vnc-4\_1\_3-x86\_linux.tar.gz

[root@centos ~]# tar jxvf vnc-4\_1\_3-x86\_linux.tar.gz

[root@centos ~]# cd vnc-4\_1\_3-x86\_linux

[root@centos vnc-4\_1\_3-x86\_linux]# ./vncinstall

[root@centos vnc-4\_1\_3-x86\_linux]# mkdir -p /usr/local/vnc/classes

[root@centos vnc-4\_1\_3-x86\_linux]# cp java/\* /usr/local/vnc/classes

**安装tightVNC Server：**

下载tightvnc-1.3.10\_unixsrc.tar.bz2

[root@centos ~]# tar zxvf tightvnc-1.3.10\_unixsrc.tar.bz2

[root@centos ~]# cd vnc\_unixsrc

[root@centos vnc\_unixsrc]# xmkmf #生成makefile

[root@centos vnc\_unixsrc]# make World #检测是否安装JPEG和zlib

[root@centos vnc\_unixsrc]# cd Xvnc

[root@centos Xvnc]# ./configure

[root@centos Xvnc]# make

[root@centos Xvnc]# cd ..

[root@centos vnc\_unixsrc]# ./vncinstall /usr/local/bin /usr/local/man

[root@centos vnc\_unixsrc]# mkdir -p /usr/local/vnc/classes

[root@centos vnc\_unixsrc]# cp classes/\* /usr/local/vnc/classes

**配置Server：**

[root@centos ~]# vi /etc/sysconfig/vncservers #该文件不用修改

# VNCSERVERS="2:myusername"

# VNCSERVERARGS[2]="-geometry 800x600 -nolisten tcp -nohttpd -localhost"

在VNCSERVERS中修改 dwt 为你希望运行VNC桌面的用户。VNCSERVERS中的1表示VNC以桌面1运行，如果希望添加其他的桌面，可以修改配置如下：

-----------------------------------------------

VNCSERVERS="1:user1 2:user2 3:user3"

-----------------------------------------------

-geometry 桌面分辨率；

-nohttpd 不监听HTTP端口(58xx端口)；

-nolisten tcp 不监听X端口(60xx端口)；

-localhost 只允许从本机访问；

-AlwaysShared 默认只同时允许一个vncviewer连接，此参数允许同时连多个vncviewer；

-SecurityTypes None 登录不需要密码认证

[dwt@centos ~]$ rpm -q vnc vnc-server #CentOS自带vnc-server，否则yum install vnc-server

[dwt@centos ~]$ vncpasswd #设置当前用户的vnc密码

[dwt@centos ~]$ vncserver #先运行一次vnc，生成~/.vnc/xstartup

New 'X' desktop is centos:1 #dwt在1号端口，再运行一次就在n+1号端口，以此类推

[dwt@centos ~]$ vi .vnc/xstartup #将文本显示改为gnome显示

#!/bin/sh

unset SESSION\_MANAGER #root不用增加这两行

exec /etc/X11/xinit/xinitrc

xrdb $HOME/.Xresources

xsetroot -solid grey

#xterm -geometry 80x24+10+10 -ls -title "$VNCDESKTOP Desktop" &

#twm &

gnome-session &

[dwt@centos ~]$ vncserver -kill :1 #关闭1号端口上的vnc

[dwt@centos ~]$ vncserver #重启

New 'X' desktop is centos:1

**问题1：VNC Viewer无法连接**

关闭防火墙，/etc/init.d/iptables stop chkconfig -level 2345 iptabls off

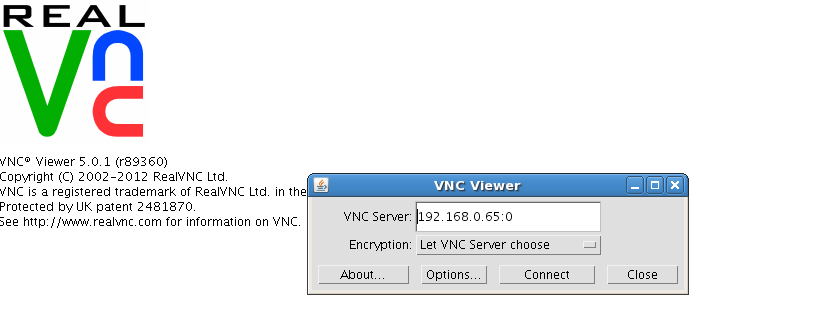
**问题2：VNC Viewer连接后，灰屏**

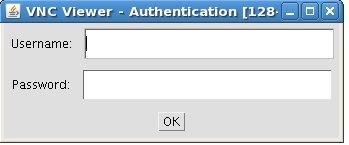
Root启动的vnc界面不会出现这个问题，一般用户需在~/.vnc/xstartup中添加unset SESSION\_MANAGER exec /etc/X11/xinit/xinitrc两行

[root@centos ~]# chmod a+x /etc/X11/xinit/xinitrc

**问题3：浏览器无法通过580x端口访问VNC SERVER**

Window作为server，可通过IE，firefox（必须安装java插件）等浏览器访问，键入<http://ip:5800>，弹出使用java插件提示，允许使用，输入server设置的用户名密码。





Linux作为server，直接用IE访问无法连接，用其他浏览器则报File Not Found。

查看~/.vnc下的日志可知vncserver会调用/usr/local/vnc/classes里的jar，找不到

解决方法：找到源码，将java（for VNC）或classes（for tightVNC）下的内容拷贝到/usr/local/vnc/classes中。

# 开源远程控制软件

## RealVNC

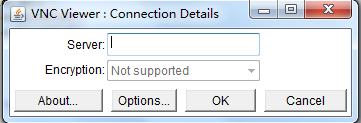
RealVNC半商业性质。

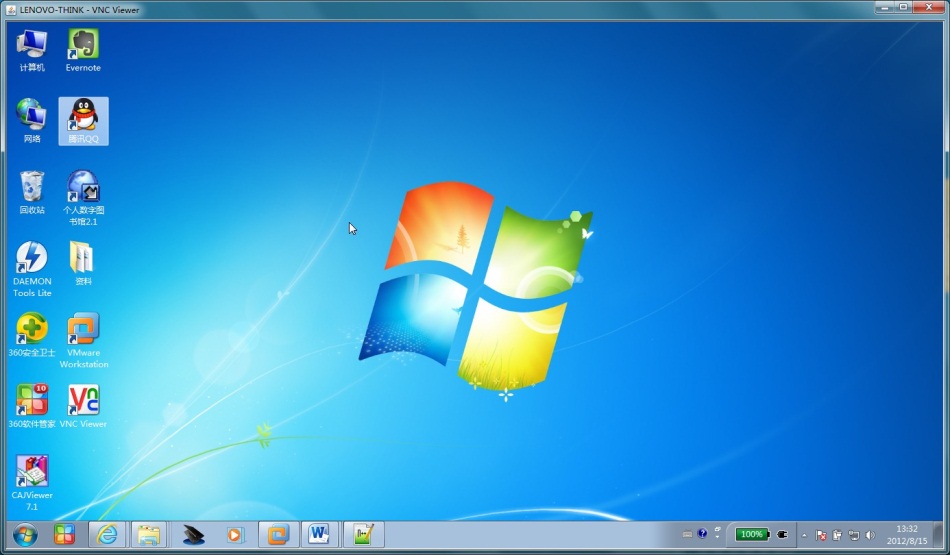
应用平台： Win9x/NT/2000/XP/2003/win7 Linux/unix

开发语言：C++

优点：提供的源码用eclipse运行成功。

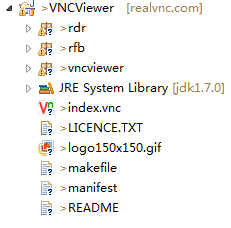
缺点：弹出界面。



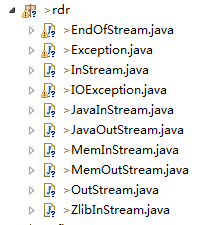


Win7启动VNC Server，通过IE访问弹出的界面

VNCViewer利用applet和awt实现，主要包含三个包，rdr，rfb，vncviewer



1. **Rdr包为读写类**



InStream

JavaInStream

读socket

MemInStream

未使用

ZlibInStream

读压缩包

OutStream

JavaOutStream

写socket

MemOutStream

未使用

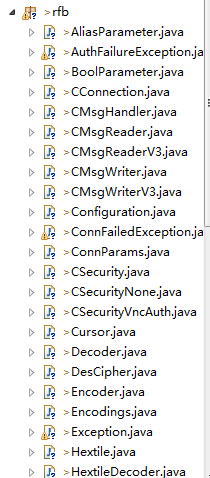
Exception

EndOfStream

IOException

RuntimeException

1. **Rfb包为系统类**



CSecurity

加密父类

CSecurityNone

无加密

CSecurityVncAuth

VNC加密

VncAuth

调用加密算法

DesCipher

Des加密算法

CMsgReaderer

CMsgReaderV3

CMsgHandler

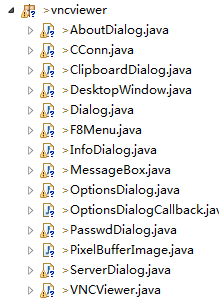
消息处理句柄

CConnection

CConn

CConn：主控制，包含以上对话框的引用。

1. **Vncviewer包为界面类**



Frame

Dialog

AboutDialog

关于对话框

ClipboardDialog

剪切对话框

InfoDialog

信息对话框

MessageBox

消息盒

OptionsDialog

参数对话框

PasswdDialog

用户名密码对话框

ServerDialog

服务器端口对话框

最重要的界面为：

VNCViewer：包含main函数，继承自Applet；

DesktopWindow：远程桌面显示界面，继承自Canvas；

## tightVNC

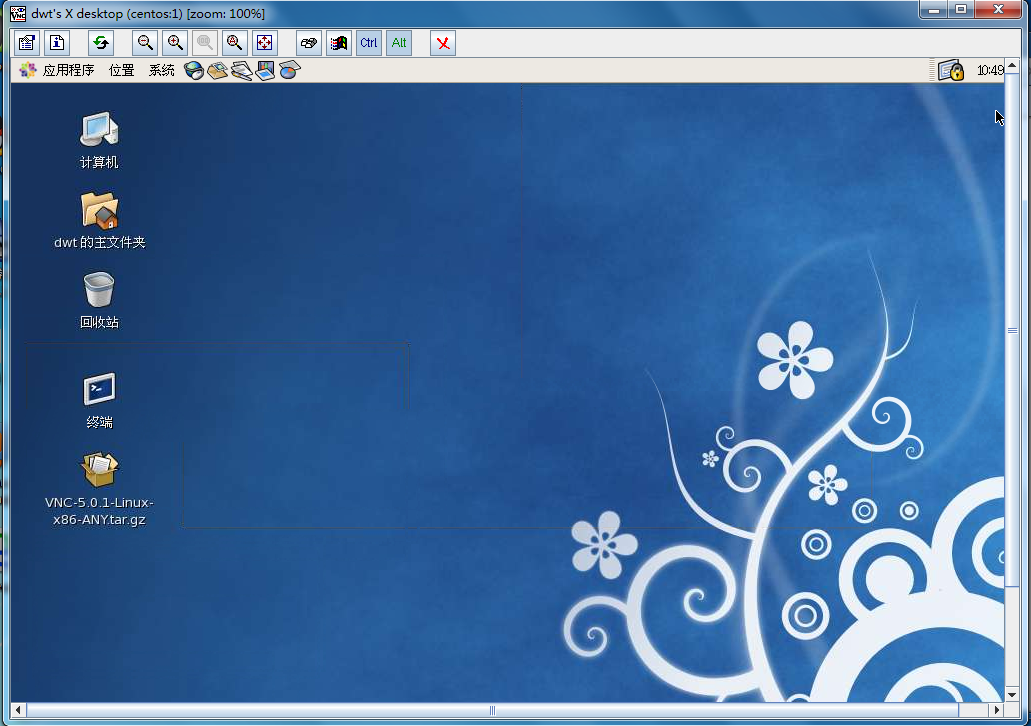
TightVNC是自由软件，遵循GPL条款，源代码开源，个人、企业使用均无任何限制。

应用平台： Win9x/NT/2000/XP/2003/win7 Linux/unix

开发语言：C++

优点：提供jar包作为viewer，能直接应用到Web页面开发中，提供放大，缩小等按钮。

缺点：其源码尚未知如何运行，界面是弹出式的，不能内嵌入网页。



CentOS启动tightVNC Server，通过IE访问的弹出界面

Win7启动tightVNC Server，通过firefox访问的弹出界面相同

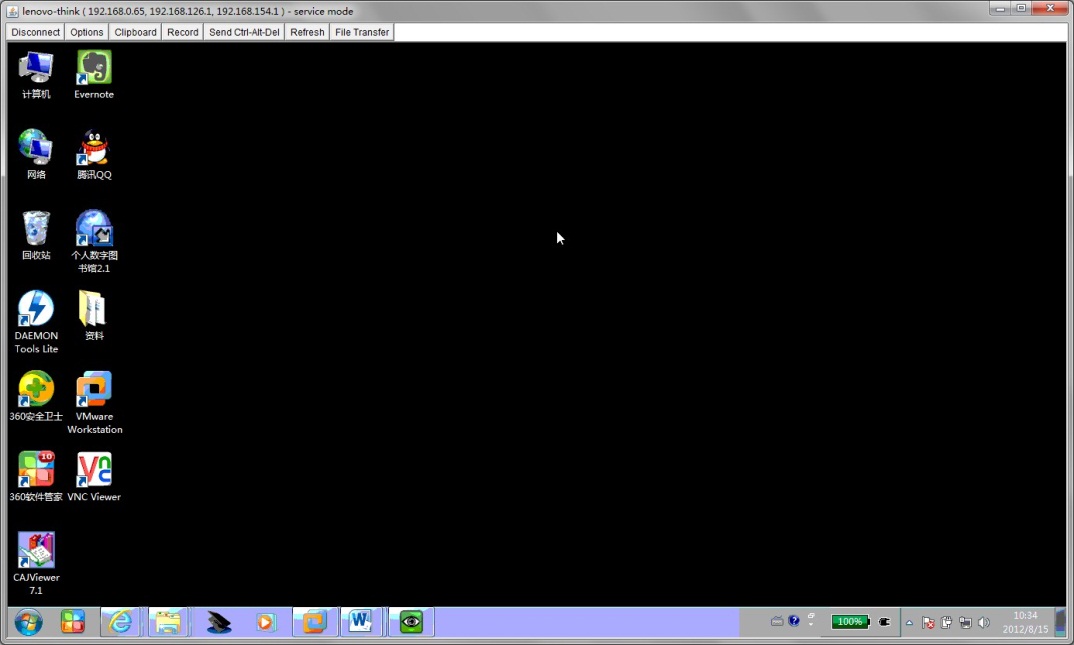
## UltraVNC

应用平台： 95, 98, Me, NT4, 2000, XP, 2003, Vista, Windows 7

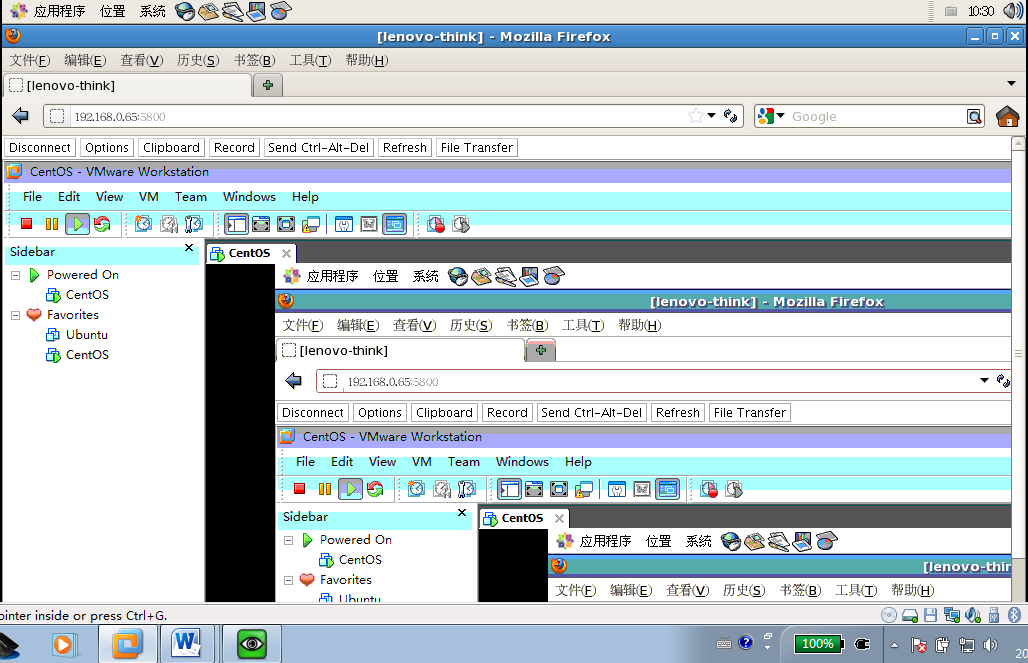
开发语言：C++，内置的java viewer提供web访问功能

优点：通过IE访问时仍为弹出式界面，firefox访问是内嵌式界面，界面顶部提供文件传输等按钮，如图所示。

缺点：没有提供完整的jar包，server和viewer只支持windows，或linux下用浏览器作为viewer。



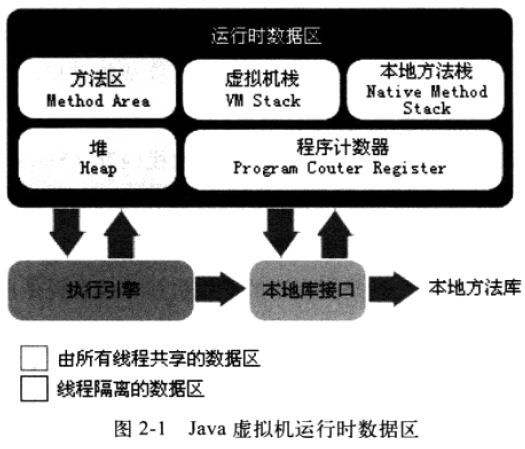
Win7启动UltraVNC Server，通过IE访问的弹出式界面



Win7启动UltraVNC Server，通过Firefox访问的嵌入式界面

# Java

## 内存区设置



1. **程序计数器**

**线程私有，**当前线程所执行的字节码的行号指示器。不会抛异常。

1. **Java虚拟机栈**

**线程私有**，**虚拟机执行的Java方法**被执行的时候都会同时创建一个栈帧用于存储局部变量表、操作栈、动态链接、方法出口等信息。会抛StackOverflowError（无法分配新栈帧）和OutOfMemoryError（无法建立新线程）异常。

1. **本地方法栈**

**线程私有**，与Java虚拟机栈类似，区别是**虚拟机使用的Native方法**，会抛StackOverflowError和OutOfMemoryError异常。

HotSpot不区分虚拟机栈和本地方法栈，-Xss设置栈大小。

1. **Java堆**

**线程共享**，所有**对象实例**以及数组都要在堆上分配。会抛OutOfMemoryError异常。通过-Xmx，-Xms设置。

1. **方法区**

**线程共享**，用于存储已被虚拟机加载的**类信息**、常量、静态变量、即时编译器编译后的代码等数据。很少垃圾回收。会抛OutOfMemoryError异常。-XX:PermSize和-XX:MaxPermSize限制方法区大小。

1. **运行时常量池**

方法区的一部分，用于存放编译期生成的各种字面量和符号引用。会抛OutOfMemoryError异常。

1. **Direct Memory直接内存**

NIO(New Input/Output)类，直接分配堆外内存，通过DirectByteBuffer引用该内存，以提高性能，由于忽略直接内存，会导致内存区域综合大于物理内存限制抛出OutOfMemoryError异常。-XX:MaxDirectMemorySize指定大小。CometD中使用大量NIO操作需要用到Direct Memory。

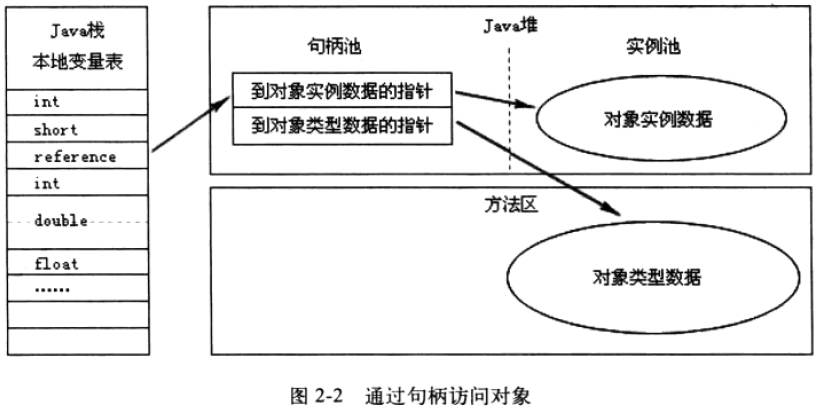
1. **JNI代码**

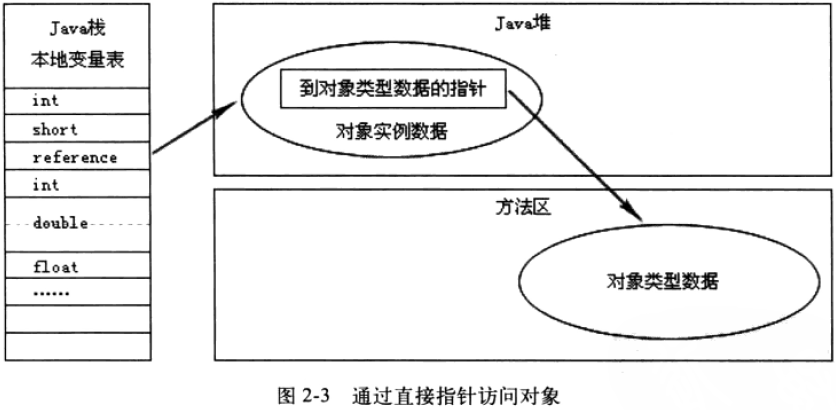
代码中使用JNI调用本地库，本地库使用的内存也不在堆中。

1. **Socket缓存区**

每个Socket连接都Receive和Send两个缓存区，分别占大约37KB和25KB内存，连接过多会抛出IOException:Too many open files。

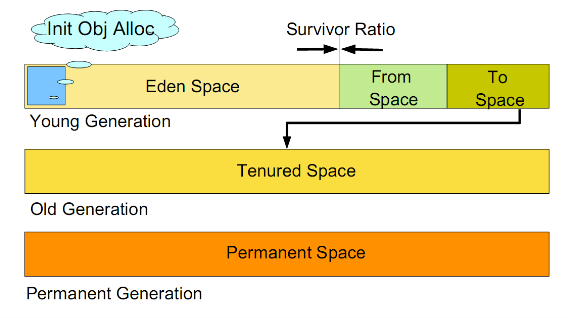
对象访问方式：使用句柄和直接指针。





## 垃圾收集器

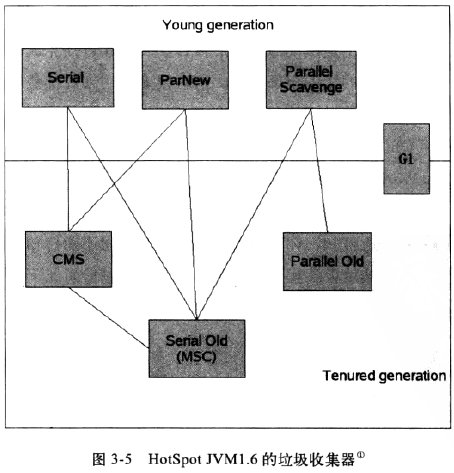
对象已死？引用计数算法，根搜索算法



新生代（Young Generation）：分为8份Eden，1份From，1份To，GC(Minor GC)采用复制算法

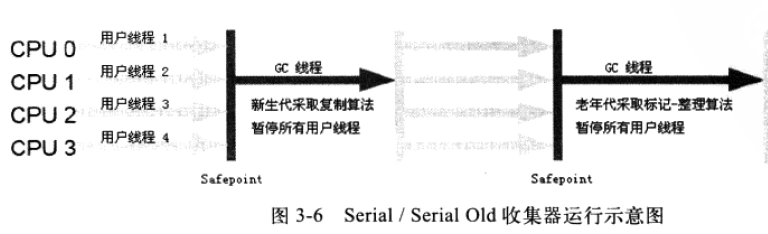
老年代（Old Generation）：担保，GC(Major GC/Full GC)采用标记-清除算法（碎片问题），标记-整理算法。

永久代（Permanent Space）：方法区



Serial收集器：单线程，垃圾收集时暂停所有工作线程，Client模式下默认新生代收集器。

-XX:SurvivorRatio,-XX:PretenureSizeThreshold,-XX:HandlePromotionFailue

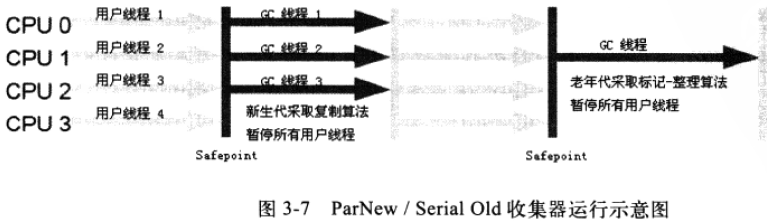


ParNew收集器： Serial的多线程版本，Server模式下的首选新生代收集器，与CMS收集器配合

-XX:+UseConcMarkSweepGC:默认GC

-XX:+UseParNewGC:使用ParNewGC

-XX:ParallelGCThreads：限制GC线程数



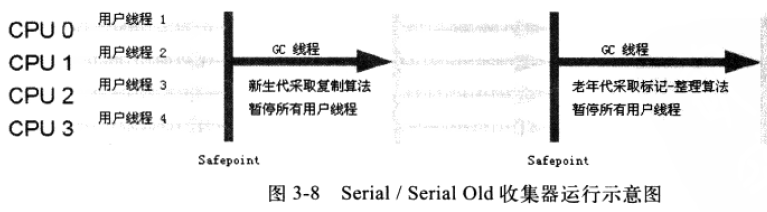
Parallel Scavenge收集器：可控制的吞吐量

-XX:MaxGCPauseMillis：最大GC停顿时间

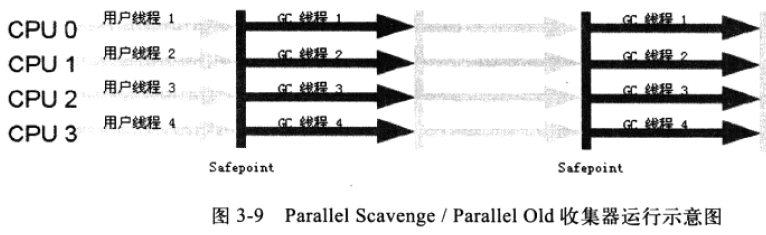
-XX:GCTimeRatio:吞吐量大小

-XX:+UseAdaptiveSizePolicy:自适应参数

Serial Old收集器：Serial的老年代版本，单线程，标记-整理。与Parallel Scavenge配合



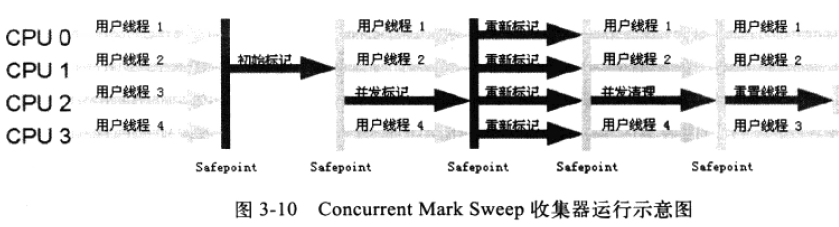
Parallel Old收集器：Parallel Scavenge的老年代版本，多线程，标记-整理。



CMS收集器：最短停顿时间，4个步骤：初始标记、并发标记、重新标记、并发清除。标记-清除。

-XX:+UseCMSCompactAtFullCollection:Full GC后进行一次碎片整理。

-XX:CMSFullGCsBeforeCompaction:多少次Full GC后进行一次碎片整理。



G1收集器：标记-整理，精确控制停顿。将整个Java堆划分为多个大小固定的独立区域，并且跟踪这些区域的垃圾堆积程度，维护一个优先列表，优先回收垃圾最多的区域。

+XX:+PrintGCDetails：打印内存回收日志

-Xms，-Xmx：最小/最大堆大小

-Xmn：新生代堆大小

-XX:SurvivorRatio：新生代Eden区与Survivor区的空间比例

-XX:PretenueSizeThreshold：大于该值的对象直接在老年代中分配。

-XX:MaxTenuringThreshold：晋升老年代的年龄阈值

-XX:+HeapDumpOnOutOfMemoryError：OOM时生成Dump

64位JDK的性能测试结果普遍低于32位JDK。由于指针膨胀及数据补齐导致程序在64位JDK中消耗的内存一般比32位大。32位机器中每个进程只能使用2GB内存，堆最大1.5GB，并受到最高4GB内存限制。

# noVNC

yum groupinstall Desktop

yum install gnome-core firefox

yum install tigervnc-server

useradd vncuser

passwd vncuser

su - vncuser

vncpasswd

vi /etc/sysconfig/vncservers

VNCSERVERS="1:vncuser"

VNCSERVERARGS[1]="-geometry 1024x768"

service vncserver start

chkconfig vncserver on

VNC Viewer连接ip:5901，输入密码登陆桌面。

git clone <https://github.com/kanaka/noVNC>

cd noVNC/utils

./websocket.py --8000 localhost:5901

打开noVNC网站的demo，输入host：ip，port：8000，password：vncpasswd

# JIRA

<https://confluence.atlassian.com/jira/jira-installation-and-upgrade-guide-100303124.html>

http://www.confluence.cn/pages/viewpage.action?pageId=1671276

## 安装

安装jdk、mysql-server

tar zxvf atlassian-jira-6.3.6.tar.gz

sudo mkdir -p /home/jira

sudo chown -R dwt:dwt /home/jira

vi atlassian-jira-6.3.6-standalone/atlassian-jira/WEB-INF/classes/jira-application.properties

jira.home = /home/jira

mv mysql-connector-java-5.1.36.jar atlassian-jira-6.3.6-standalone/atlassian-jira/WEB-INF/lib

mysql -u root -p

mysql> create database jira character set utf8 ;

sudo netstat -utlp | grep 8080

lsof -i:8080 查看8080端口是否被占用，如果占用

vi atlassian-jira-6.3.6-standalone/conf/server.xml

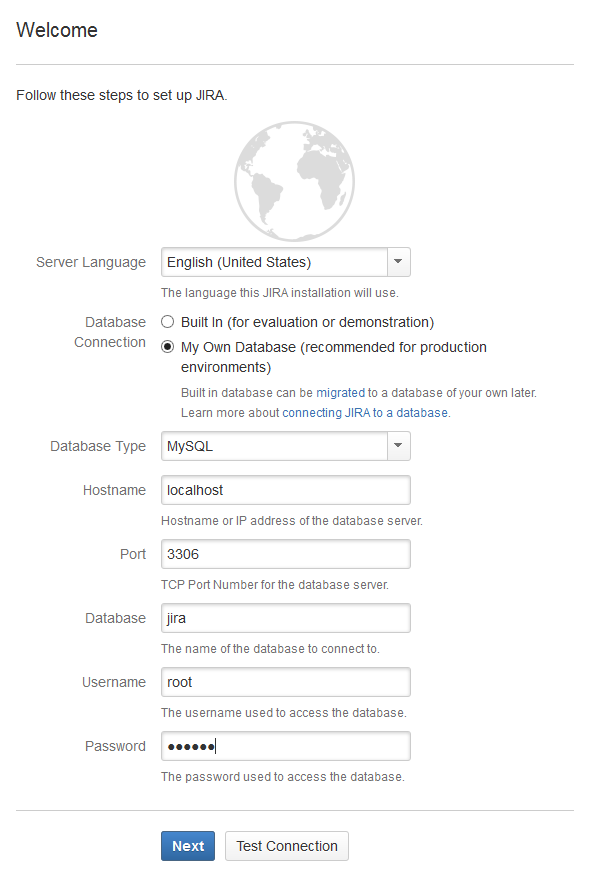
#<Connector port="8080"，

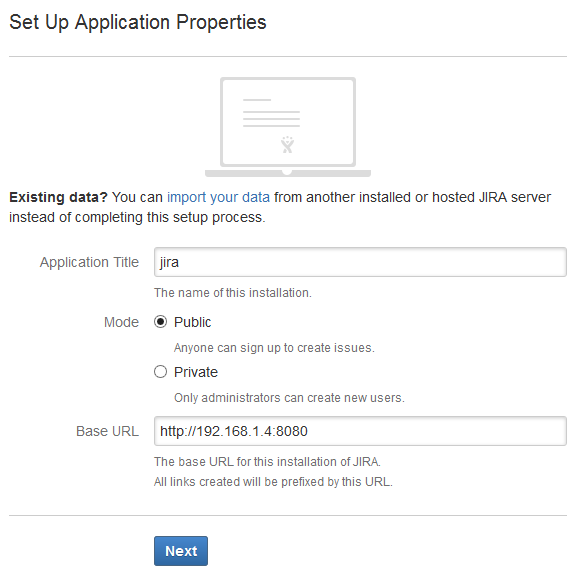
<Connector port="8081"

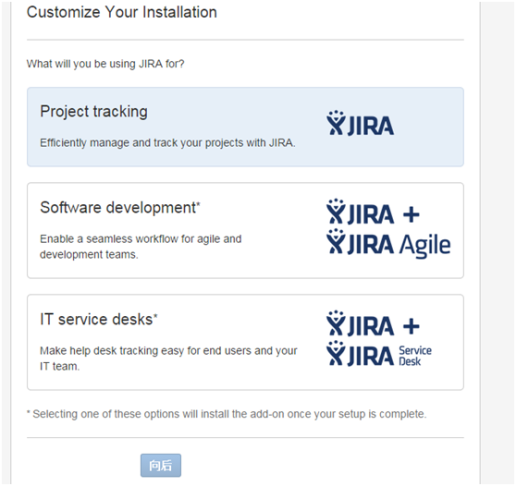
cd atlassian-jira-6.3.6-standalone/bin

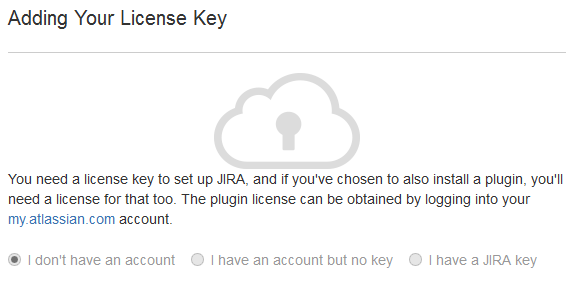
./start-jira.sh

登陆192.168.1.4:8080，进入设置页面，配置MySQL位置，test connection，Next









申请账号和key，Next-Now

填写SMTP和账户-Finsh

破解不成功：

下载破解包，拷贝到指定目录

mv ~/atlassian-extras-2.2.2.jar ~/atlassian-jira-6.3.6-standalone/atlassian-jira/WEB-INF/lib/

./start-jira.sh

## 使用

基本概念：project，component，issue，version（affects versions、fix versions）

