Jvm系统定位问题，需要的数据包括：

运行日志、异常堆栈、GC日志、线程快照（threaddump、javacore文件）、堆转储快照（heapdump、hprof文件）等。

Jdk的bin目录包含很多的处理应用程序性能问题和定位故障的小程序，这些小程序之所以小，是因为它们仅仅是对jdk/lib/tools.jar类库的一层薄包装，它们的主要功能是在tools类库中实现的。

Jdk1.5需要在应用程序启动时开启jmx管理的功能，而jdk1.6是默认开启的。

Jdk监控和故障处理工具：

|  |  |
| --- | --- |
| 名称 | 主要作用 |
| Jps | Jvm process status tool,显示指定系统内所有的hotspot虚拟机进程 |
| Jstat | Jvm statistics monitoring tool,用于收集hotspot 虚拟机各方面的运行数据 |
| Jinfo | Configuration info for java,显示虚拟机配置信息 |
| Jmap | Memory map for java,生成虚拟机的内存转储快照（heapdump文件） |
| Jhat | Jvm heap dump browser,用于分析heapdump文件，它会建立一个http/html 服务器，让用户可以在浏览器上查看分析结果 |
| Jstack | Stack trace for java,显示虚拟机的线程快照 |

Jps:虚拟机进程状况工具，可以列出正在运行的虚拟机进程，并显示虚拟机执行主类名称以及这些进程的本地虚拟机唯一ID。

命令格式：jsp [options] [hostid]

C:\Users\UserName>jps -l

112 org.jetbrains.jps.cmdline.Launcher

7684 org.jetbrains.idea.maven.server.RemoteMavenServer

4408

3148 sun.tools.jps.Jps

6972 com.test.jvm\_study.Main

Jps可以通过RMI协议查询开启了RMI服务的远程虚拟机进程状态，hostid为RMI注册表中注册的主机名。

jps常用选项：

-q：只输出本地虚拟机唯一id，省略主类名称

C:\Users\UserName>jps -q

112

9024

7684

4408

6972

-m：输出虚拟机进程启动时传递给主类main函数的参数

112 Launcher D:/IntelliJ IDEA 2016.2.5/lib/jps-builders.jar;D:/IntelliJ IDEA 2016.2.5/lib/resources\_en.jar;D:/IntelliJ IDEA 2016.2.5/lib/trove4j.jar;D:/IntelliJ IDEA 2016.2.5/lib/jgoodies-forms.jar;D:/IntelliJ IDEA 2016.2.5/lib/util.jar;D:/IntelliJ IDEA 2016.2.5/lib/jdom.jar;D:/IntelliJ IDEA 2016.2.5/lib/jps-model.jar;D:/IntelliJ IDEA 2016.2.5/lib/annotations.jar;D:/IntelliJ IDEA 2016.2.5/lib/jna-platform.jar;D:/IntelliJ IDEA 2016.2.5/lib/asm-all.jar;D:/IntelliJ IDEA 2016.2.5/lib/jna.jar;D:/IntelliJ IDEA 2016.2.5/lib/forms\_rt.jar;D:/IntelliJ IDEA 2016.2.5/lib/snappy-in-java-0.5.1.jar;D:/IntelliJ IDEA 2016.2.5/lib/idea\_rt.jar;D:/IntelliJ IDEA 2016.2.5/lib/javac2.jar;D:/IntelliJ IDEA 2016.2.5/lib/openapi.jar;D:/IntelliJ IDEA 2016.2.5/lib/oromatcher.jar;D:/IntelliJ IDEA 2016.2.5/lib/nanoxml-2.2.3.jar;D:/IntelliJ IDEA 2016.2.5/lib/rt/jps-plugin-system.jar;D:/IntelliJ IDEA 2016.2.5/lib/protobuf-2.5.0.jar;D:/IntelliJ IDEA 2016.2.5/lib/netty-all-4.1.1.Final.jar;D:/IntelliJ IDEA 2016.2.5/lib/log

7008 Jps -m

7684 RemoteMavenServer

4408

6972 Main

-l：输出主类的全名，如果进程执行的是jar包，输出jar路径

112 org.jetbrains.jps.cmdline.Launcher

2672 sun.tools.jps.Jps

7684 org.jetbrains.idea.maven.server.RemoteMavenServer

4408

6972 com.test.jvm\_study.Main

-v：输出虚拟机进程启动时jvm参数

112 Launcher -Xmx700m -Djava.awt.headless=true -Djava.endorsed.dirs="" -Djdt.compiler.useSingleThread=true -Dpreload.project.path=F:/Html/mannual\_files\_all/jvm\_study/code/JVMDemo -Dpreload.config.path=C:/Users/DaiYan/.IntelliJIdea2016.2/config/options -Dcompile.parallel=false -Drebuild.on.dependency.change=true -Djava.net.preferIPv4Stack=true -Dio.netty.initialSeedUniquifier=-7041225670537940096 -Dfile.encoding=GBK -Djps.file.types.component.name=FileTypeManager -Duser.language=zh -Duser.country=CN -Didea.paths.selector=IntelliJIdea2016.2 -Didea.home.path=D:\IntelliJ IDEA 2016.2.5 -Didea.config.path=C:\Users\DaiYan\.IntelliJIdea2016.2\config -Didea.plugins.path=C:\Users\DaiYan\.IntelliJIdea2016.2\config\plugins -Djps.log.dir=C:/Users/DaiYan/.IntelliJIdea2016.2/system/log/build-log -Djps.fallback.jdk.home=D:/java/jre -Djps.fallback.jdk.version=1.8.0\_102 -Djava.io.tmpdir=C:/Users/DaiYan/.IntelliJIdea2016.2/system/compile-server/\_temp\_ -Dkotlin.incremental.compilation.experimental=true -Dkotlin.daemon.enabled -Dkotlin.daemon

7684 RemoteMavenServer -Djava.awt.headless=true -Didea.version==2016.2.5 -Xmx768m -Didea.maven.embedder.version=3.0.5 -Dfile.encoding=GBK

2632 Jps -Denv.class.path=.;D:\java\lib;D:\java\lib\dt.jar;D:\java\lib\tools.jar -Dapplication.home=D:\java -Xms8m

4408 -Xms128m -Xmx750m -XX:ReservedCodeCacheSize=240m -XX:+UseConcMarkSweepGC -XX:SoftRefLRUPolicyMSPerMB=50 -ea -Dsun.io.useCanonCaches=false -Djava.net.preferIPv4Stack=true -XX:+HeapDumpOnOutOfMemoryError -XX:-OmitStackTraceInFastThrow -Djb.vmOptionsFile=D:\IntelliJ IDEA 2016.2.5\bin\idea64.exe.vmoptions -Xbootclasspath/a:D:\IntelliJ IDEA 2016.2.5\lib\boot.jar -Didea.paths.selector=IntelliJIdea2016.2 -Didea.jre.check=true -XX:ErrorFile=C:\Users\DaiYan\java\_error\_in\_idea\_%p.log -XX:HeapDumpPath=C:\Users\DaiYan\java\_error\_in\_idea.hprof

6972 Main -agentlib:jdwp=transport=dt\_socket,address=127.0.0.1:13184,suspend=y,server=n -Dfile.encoding=UTF-8

Jstat:虚拟机统计信息监控工具，用于监控虚拟机各种运行状态信息的命令行工具。它可以显示本地或者远程虚拟机进程中的类装载、内存、垃圾收集、jit编译等运行数据。

命令格式：jstat [option vmid [interval [s|ms] [count] ] ]

如果是本地虚拟机进程，vmid与本地虚拟机进程id（lvmid）是一致的，如果是远程虚拟机进程，那vmid的格式应当是：

[protocol:][//]lvmid[@hostname[:port]/servername]

参数interval和count指明查询间隔和查询次数，如果省略这两个参数则只会查询一次。

选型option指明用户希望查询的虚拟机信息，主要分为3类：类装载，垃圾收集，运行期编译状况。

-class:监视类装载、卸载数量、总空间以及类装载所耗费的时间

-gc: 监视java堆状况，包括Eden区，两个survivor区，老年代，永久代等容量，已用空间，GC时间合计等信息

-gccapacity:监视内容与-gc基本相同，但输出主要关注java堆各个区域使用到的最大、最小空间

-gcutil:监视内容与-gc基本相同，但输出主要关注已使用空间占总空间的百分比

-gccause：与-gcutil功能一样，但是会额外输出导致上一次GC产生的原因

-gcnew:监视新生代GC状况

-gcnewcapacity:监视内容与-gcnew基本相同，输出主要关注使用到的最大、最小空间

-gcold:监视老年代GC状况

-gcoldcapacity:监视内容与-gcold基本相同，输出主要关注使用到的最大、最小空间

-gcpermcapacity:输出永久代使用到的最大、最小空间

-compiler:输出JIT编译器编译过的方法、耗时信息

-printcompilation:输出已被JIT编译的方法

Jinfo:java配置信息工具，作用是实时的查看和调整虚拟机各项参数。

命令格式：jinfo [option] pid

Jmap：java内存映像工具，用于生成堆转储快照（一般称为heapdump或dump），还可以查询finalize执行队列、java堆和永久代的向详细信息，如空间使用率、当前用的是哪种收集器等。

命令格式：jmap [option] vmid

选项option的合法值：

-dump:生成java堆转储快照，格式为dump:[live,] format=b,file=<filename>,其中live子参数说明是否只dump出存活的对象

-finalizeinfo：显示在F-Queue中等待Finalizer线程执行finalize方法的对象

-heap:显示java堆详细信息，如使用哪种回收器，参数配置，分代状况等

-histo:显示堆中对象统计信息，包括类、实例数量、合计容量

-permstat:以classloader为统计口径显示永久代内存状态

-F:当虚拟机进程对-dump选项没有响应时，可使用这个选项强制生成dump快照

Jhat:虚拟机堆转储快照分析工具，分析jmap生成的堆转储快照，jhat生成dump文件的分析结果后可以在浏览器中查看。因为分析工作时一个耗时而且消耗硬件资源的过程，所以一般不会直接使用jhat命令来分析dump文件。

Jstack:java堆栈跟踪工具，用于生成虚拟机当前时刻的线程快照（一般称为threaddump或者javacore文件）。

线程快照就是当前虚拟机内每一条线程正在执行的方法堆栈的集合，生成的快照的主要目的是定位线程出现长时间停顿的原因。

线程出现停顿的时候通过jstack来查看各个线程的调用堆栈，就可以知道没有响应的线程到底在后台做些什么事情，或者等待着什么资源。

命令格式：jstack [option] vmid

选项option的合法值：

-F:当正常输出的请求不被响应时，强制输出线程堆栈

-l:除堆栈外，显示关于锁的附加信息

-m:如果调用本地方法的话，可以显示C/C++的堆栈

Jdk1.5中，java.lang.Thread类新增加了一个getAllStackTraces()方法，这个方法用于获取虚拟机中所有线程的StackTraceElement对象，这个方法可以完成jstack的大部分功能，可以据此做一个web管理页面，随时在浏览器中查看状态信息。

JDK的可视化工具

两个功能强大的可视化工具：JConsole和VisualVM

JConsole：java监视与管理控制台，它是一种基于JMX的可视化监视、管理工具。

Jconsole可以用来监控本地虚拟机进程，也可一监控和管理远程虚拟机进程。

VisualVM:多合一故障处理工具，它除了运行监视、故障处理外，还提供了很多其他方面的功能，如性能分析