

JVM笔记无特殊说明，基于jdk8&14

[零、参考资料 2](#_Toc39500651)

[一、常用JVM参数 3](#_Toc39500652)

[1. GC参数 3](#_Toc39500653)

[2. 堆参数 4](#_Toc39500654)

[3. 非堆参数 4](#_Toc39500655)

[4. 其他参数 5](#_Toc39500656)

[二、JDK分析工具 5](#_Toc39500657)

[1jps 5](#_Toc39500658)

[2jstat 6](#_Toc39500659)

[3jinfo 6](#_Toc39500660)

[4jmap 7](#_Toc39500661)

[5jhat 7](#_Toc39500662)

[6jstack 7](#_Toc39500663)

[7jstatd 8](#_Toc39500664)

[8jcmd 8](#_Toc39500665)

[三、JDK可视化工具 9](#_Toc39500666)

[1、OQL(对象查询语言) 9](#_Toc39500667)

[1.1简介 9](#_Toc39500668)

[1.2、SELECT 9](#_Toc39500669)

[1.3内置对象 9](#_Toc39500670)

[1.4对象函数 10](#_Toc39500671)

[1.5统计集合 11](#_Toc39500672)

[2、Visual VM 12](#_Toc39500673)

[2.1启动： 12](#_Toc39500674)

[2.2dump当前JVM 13](#_Toc39500675)

[2.3插件 15](#_Toc39500676)

[2.4BTrace 16](#_Toc39500677)

[3、MAT 17](#_Toc39500678)

[四、LINUX命令(centos7) 17](#_Toc39500679)

[1、top 17](#_Toc39500680)

[2、ps 18](#_Toc39500681)

[3、vmstat 20](#_Toc39500682)

[4、iostat 21](#_Toc39500683)

[5、uptime 22](#_Toc39500684)

[6、dmesg 22](#_Toc39500685)

[7、mpstat 23](#_Toc39500686)

[8、sar 24](#_Toc39500687)

[9、free 24](#_Toc39500688)

[10、tcpdump 25](#_Toc39500689)

[11、netstat 26](#_Toc39500690)

[12、pidstat 28](#_Toc39500691)

[13、grep 28](#_Toc39500692)

[14、cat 30](#_Toc39500693)

[15、tail 31](#_Toc39500694)

[16、curl 31](#_Toc39500695)

[17、ping 36](#_Toc39500696)

[18、vim 37](#_Toc39500697)

[五、Windows工具 38](#_Toc39500698)

[1.1. Perfmon 38](#_Toc39500699)

[1.2. Process Explorer 38](#_Toc39500700)

[1.3. pslist 38](#_Toc39500701)

[六、JVM常用执行指令 38](#_Toc39500702)

[1、java 38](#_Toc39500703)

[2、javac 38](#_Toc39500704)

[3、jar 39](#_Toc39500705)

[4、javap 39](#_Toc39500706)

[附录 40](#_Toc39500707)

# 零、参考资料

Java虚拟机规范(Java SE 8版)

Java语言规范基于Java SE 8

深入理解Java虚拟机——JVM高级特性与最佳实践

实战Java虚拟机——JVM故障诊断与性能优化

垃圾回收的算法与实现

深入理解计算机系统

Stackoverflow:

https://stackoverflow.com/questions/16549066/java-major-and-minor-garbage-collections

OpenJdk：

https://openjdk.java.net/jeps/122

Oracle官方DOC：

https://docs.oracle.com/javase/8/javase-books.htm

https://docs.oracle.com/en/java/javase/14/

https://docs.oracle.com/javase/specs/jvms/se8/jvms8.pdf

https://www.oracle.com/webfolder/technetwork/tutorials/mooc/JVM\_Troubleshooting/week1/lesson1.pdf

https://www.oracle.com/webfolder/technetwork/tutorials/obe/java/gc01/index.html

http://cr.openjdk.java.net/~sundar/8022483/webrev.01/raw\_files/new/src/share/classes/com/sun/tools/hat/resources/oqlhelp.html

补充说明：

1 因为我的私服是centos7，所以Linux命令是基于centos7的linux内核

# 一、常用JVM参数

## GC参数

* 1. 打印GC基本日志

JDK8: -XX:PrintGC JDK9+: -Xlog:gc

* 1. 打印GC详细日志

JDK8: -XX:PrintGCDetails JDK9+: -Xlog:gc\*

-XX:PrintGCDetails 还会使JVM在退出前打印堆详细信息

* 1. 打印GC时堆的全面信息

JDK8: -XX:PrintHeapAtGC JDK9+:移除

* 1. 打印GC发生时间

JDK8: -XX:PrintGCTimeStamps JDK9+: -Xlog:gc\*已包含

* 1. 打印程序执行时间

-XX:+PrintGCApplicationConcurrentTime

* 1. 打印程序因STW而停顿的时间

-XX:+PrintGCApplicationStoppedTime

* 1. 打印软引用、弱引用、虚引用和Finallize队列信息

JDK8: +PrintReferenceGC JDK9+移除

* 1. 输出GC信息至文本

JDK8:-Xloggc:log/${filePath} JDK9+:-Xlog:gc:log/${filePath}

* 1. 打印类加载日志

JDK8: -XX:+TraceClassLoading JDK9+: -Xlog:class+load=info

* 1. 打印类卸载日志

JDK8: -XX:+TraceClassUnLoading JDK9+: -Xlog:class+unload=info

* 1. 显示当前的类信息柱状图

-XX:+PrintClassHistogram

* 1. 查看JVM运行显示参数

-XX:+PrintVMOptions

* 1. 打印JVM显示和隐式(默认)参数

-XX:+PrintCommandLineFlags

* 1. 打印所有系统参数

-XX:+PrintFlagsFinal

* 1. 在新生代和老年代使用串行回收器

-XX:UseSerialGC

* 1. 在新生代使用并行回收器

-XX:+UseParNewGC

* 1. 在老年代使用并行回收器

-XX:+UseParallelOldGC

* 1. 设置用于垃圾回收的线程数

-XX:ParallelGCThreads

* 1. 设置最大GC停顿时间

-XX:MaxGCPauseMillis

* 1. 设置吞吐量大小

-XX:GCTimeRatio

* 1. 打开开自适应GC策略

-XX:+UseAdaptiveSizePolicy

* 1. 使用G1

-XX:+UseG1GC

* 1. 设置停顿间隔

-XX:GCPauseIntervalMillis

* 1. 使用CMS

-XX:+UseConcMarkSweepGC

* 1. 设置CMS线程数

-XX:ParallelCMSThreads

* 1. 设置CMS触发时，老年代比例

-XX:CMSInitiatingOccupancyFraction

* 1. 设置CMS在FullGC后是否进行碎片整理

-XX:+UseCMSCompactAtFullCollection

* 1. 设置CMS进行内存压缩前，FullGC次数

-XX:CMSFullGCsBeforeCompaction

* 1. 允许对类元数据回收

-XX:+CMSClassUnloadingEnabled

* 1. 设置CMS触发时，永久区比例,仅在激活CMSClassUnloadingEnabled时有效

-XX:CMSInitiatingPermOccupancyFraction

* 1. 只有达到阈值才执行CMS GC

-XX:UseCMSInitiatingOccupancyOnly

## 堆参数

* 1. 初始堆空间

-Xms

* 1. 最大堆空间

-Xmx

* 1. 设置新生代大小

-Xmn

* 1. 设置新生代中eden区和from/to区的比例

-XX:SurvivorRatio

SurvivorRatio=eden/from=eden/to

* 1. 设置新生代与老年代的比例

-XX:NewRatio

NewRatio=老年代/新生代

* 1. 设置在内存溢出时到处整个堆信息

-XX:+HeapDumpOnOutOfMemoryError

* 1. 设置导出堆文件存放路径

-XX:HeapDumpPath

与-XX:+HeapDumpOnOutOfMemoryError配合

* 1. 设置大对象直接进入老年代的阈值

-XX:MaxTenuringThreshold

## 非堆参数

* 1. 最大直接内存

-XX:MaxDirectMemorySize

如果不设置，则默认与-Xmx相同

* 1. 每个线程的栈大小

-Xss

* 1. 打开TLAB

-XX:+UseTLAB

* 1. 打印TLAB相关信息

-XX:+PrintTLAB

JDK9+不支持

* 1. TLAB大小设置

-XX:TLABSize

* 1. 设置TLAB大小自动调整

-XX:+ResizeTLAB

* 1. 扩容时触发FullGC的初始化阈值

-XX:MetaspaceSize

## 其他参数

* 1. 设置工作模式为客户端

-client

* 1. 设置工作模式为服务端

-server

* 1. 禁用System.gc()（显示GC）

-XX:+DisableExplicitGC

* 1. 设置并发处理显示GC

-XX:+ExplicitGCInvokesConcurrent

* 1. 后台编译

-XX:+BackgroundCompilation

* 1. 开启逃逸分析

-XX:+DoEscapeAnalysis

仅在server模式下有效

* 1. 打开标量替换

-XX:+EliminateAllocation

* 1. 开启解释执行模式

-Xint

* 1. 开启编译模式

-Xcomp

* 1. 开启混合模式民，

-Xmixed

# 二、JDK分析工具

## 1jps

a说明

类似Linux的ps命令，只列出当前用户的有权限查看的java程序的进程

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jps.html

c命令

jps [-q] [-mlvV] [hostid]

d参数说明

|  |  |
| --- | --- |
| -q | 只输出进程ID |
| **mlvV** | |
| -m | 列出传递给主函数的参数 |
| -l | 主函数完整路径 |
| -v | 列出传递给jvm的参数 |
| -V | 列出通过flags文件传递给jvm的参数 |

## 2jstat

a说明

查看java运行时信息

b文档

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jstat.html#BEHHGFAE

c命令

jstat [Options] ${pid} [ ${interval}[s|ms] [ ${count} ] ]

d参数

|  |  |
| --- | --- |
| -help | 帮助信息 |
| -class | 显示classLoader信息 |
| -compiler | 显示jit(即时编译器)相关信息 |
| -gc | 监视gc信息 |
| -gccapacity | 显示各个区域信息 |
| -gccause | 跟gcutil一样，显示垃圾回收信息，但额外显示上一次GC原因 |
| -gcnew | 新生代信息 |
| -gcnewcapacity | 新生代大小 |
| -gcold | 老年代信息和元数据区信息 |
| -gcoldcapacity | 老年代大小 |
| -gcmetacapacity | 显示元数据区大小 |
| -gcutil | 与-gc相同，显示垃圾回收信息，着重于已使用空间的百分比 |
| -printcompilation | 显示jit编译的方法信息 |
| -t | 显示程序运行时间 |
| -h | 指定周期性输出数据， |
| pid | 进程ID |
| interval | 周期时间 |
| count | 循环次数 |

## 3jinfo

a说明

查看java的参数，支持在运行时修改部分参数，这个命令在jdk8中处于实验性质

b文档

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/jinfo.html#BCGEBFDD

c命令

jinfo [option] pid

d参数

|  |  |
| --- | --- |
| no-option | 输出全部的参数和系统属性 |
| -flag<name> | 打印指定参数 |
| -flag [+|-]<name> | 开启或关闭指定参数, Boolean |
| -flag <name>=<value> | 设置指定jvm参数值 |
| - sysprops | 输出系统属性 |

## 4jmap

ashuoming

可以生产Java程序的堆dump文件，也可以查看堆统计信息、classLoader信息、finalizer队列

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jmap.html

c命令

jmap [options] pid

d参数

|  |  |
| --- | --- |
| no option |  |
| -histo[:live] | 打印每个class的实例数目,内存占用,类全名信息. VM的内部类名字开头会加上前缀”\*”. 如果live子参数加上后,只统计活的对象数量 |
| -dump:[live,] format=b, file=filename | 使用hprof二进制形式,输出jvm的heap内容到文件，live参数可选，如果指定，则只输出活对象 |
| -heap | 打印堆摘要、配置、gc算法，打印字符串的数量和大小。 |
| -clstats | 打印classLoader信息 |
| -F | 在没有pid时，强制使用-histo和-dump，同时不支持live |
| -h | 帮助信息 |
| -help | 帮助信息 |
| -Jflag | 传递参数给jmap启动的jvm |

## 5jhat

在jdk9+被移除，官方建议用visualVm代替

## 6jstack

a说明

导出线程栈信息，进行死锁检查

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jstack.html

c命令

d参数

|  |  |
| --- | --- |
| -F | 强制打印栈信息 |
| -l | 长列表，打印锁的附加信息 |
| -m | 打印Java 和 native C/C++ frames的所有栈信息 |
| -h ｜-help | 打印帮助信息 |

## 7jstatd

a说明

启用远程监控，需要配置java的安全策略，并保存于jstatd.all.policy文件中

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jstatd.html

c命令

jstatd J-Djava.security.policy=jstatd.all.policy [ options ]

d参数

|  |  |
| --- | --- |
| -nr | 找不到RMI注册表时，不尝试创建 |
| -p | 指定端口 |
| -n | RMI名称，默认JstatRemoteHost，如果本地有多个jstatd服务，需要保证唯一 |

## 8jcmd

a说明

将诊断命令请求发送到正在运行的JVM，用来导出堆、查看Java进程、导出线程信息、执行GC、还可以进行采样分析

b文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jcmd.html

c命令

jcmd <pid | main-class> <command ... | PerfCounter.print | option>

d参数

|  |  |
| --- | --- |
| -l | 列出所有JVM |
| -h ｜-help | 列出JVM支持的命令 |
| -f filename | 从文件中读取命令 |
| PerfCounter.print | 打印目标Java进程上可用的性能计数器 |

# 三、JDK可视化工具

## 1、OQL(对象查询语言)

### 1.1简介

OQL是用于查询Java堆的类SQL查询语言。OQL允许过滤/选择从Java堆中获取的信息。虽然HAT已经支持预定义的查询，例如“显示类X的所有实例”，但OQL增加了更多的灵活性。OQL基于JavaScript表达式语言。

### 1.2、SELECT

select <JavaScript expression to select>

[ from [instanceof] <class name> <identifier>

[ where <JavaScript boolean expression to filter> ] ]

说明：

class name：java类的完全限定名，如：java.lang.String, java.util.ArrayList

instanceof：表示也查询某一个类的子类，如果不明确instanceof，则只精确查询class name指定的类

可以使用obj.field\_name语法访问Java字段，并且可以使用array [index]语法访问数组元素

例子：

1.查询长度大于等于100的字符串:

select s from java.lang.String s where s.value.length >= 100

2.查询长度大于等于256的int数组:

select a from [I a where a.length >= 256

3.查询长度大于等于256的int数组:

select a from int[] a where a.length >= 256

4.显示所有File对象的文件路径:

select file.path.value.toString() from java.io.File file

5.选取所有的ClassLoader，包括子类:

select classof(cl).name from instanceof java.lang.ClassLoader cl

6. 由给定id字符串标识的Class的实例

select o from instanceof 0x741012748 o(0x741012748是类的ID)

7.表示两位数整数的字符串:

select {instance: s, content: s.toString()} from java.lang.String s where /^\d{2}$/(s.toString())

### 

### 1.3内置对象

|  |  |  |
| --- | --- | --- |
| 对象 | 说明 | 例子 |
| heap.forEachClass(callback) | 对每一个Class对象执行一个回调操作。其中 callback 为 Javascript 函数 | heap.forEachClass(function(it) {  if (classes[it.name] != null) {  if (multipleLoadedClasses[it.name] != null) {  multipleLoadedClasses[it.name] = multipleLoadedClasses[it.name] + 1;  } else {  multipleLoadedClasses[it.name] = 1;  }  } else {  classes[it.name] = it;  }  }); |
| heap.forEachObject (callback, clazz, includeSubtypes) | clazz:指定实例对象，默认为Java.lang.Object  includeSubtypes：是否包含子类，默认true |  |
| heap.findClass(className) | 查找给定名称的Java类, 生成的Class对象具有以下**属性**:  **name** - name of the class.  **superclass** - Class object for super class (or null if java.lang.Object).  **statics** - name, value pairs for static fields of the Class.  **fields** - array of field objects. field object has name, signature properties.  **loader** - ClassLoader object that loaded this class.  **signers** - signers that signed this class.  **protectionDomain** - protection domain to which this class belongs  同时具有一下**方法**：  **isSubclassOf** - tests whether given class is direct or indirect subclass of this class or not.  **isSuperclassOf** - tests whether given Class is direct or indirect superclass of this class or not.  **subclasses** - returns array of direct and indirect subclasses.  **superclasses** - returns array of direct and indirect superclasses. | select heap.findClass("java.util.Vector")  select heap.findClass("java.util.Vector").superclasses() |
| head.findObject(objID) | 根据对象ID找对象 |  |
| heap.classes() | 返回堆快照中所有的类的集合 |  |
| heap.objects(clazz, [includeSubtypes], [filter]) | 返回堆快照中所有的对象的集合  clazz:指定类名称，默认java.lang.Object  includeSubtypes：是否包含子类，true  filter:过滤规则 | select heap.objects ("java.io.File",true, 'it.value !=null') |
| head.livepaths(obj) | 查找对象活的引用链 | select heap.livepaths(s) from java.lang.String s  输出例子：  java.lang.String#1600-> geym.zbase.ch7.heap.WebPage#57-> java.lang.Object[]#341-> java.util.Vector#11-> geym.zbase.ch7.heap.Student#3 |
| heap.roots() | 获取堆的根对象 |  |
| heap.finalizables() | 等待垃圾收集的java对象的枚举 |  |

### 1.4对象函数

|  |  |  |
| --- | --- | --- |
| 函数 | 说明 | 例子 |
| classof(objname) | 返回指定对象的类  返回对象有以下属性：  **name** - name of the class.  **superclass** - Class object for super class (or null if java.lang.Object).  **statics** - name, value pairs for static fields of the Class.  **fields** - array of field objects. Field objects have name, signature properties.  **loader** - ClassLoader object that loaded this class.  **signers** - signers that signed this class.  **protectionDomain** - protection domain to which this class belongs.  有以下方法：  **isSubclassOf()** - tests whether given class is direct or indirect subclass of this class or not.  **isSuperclassOf()** - tests whether given Class is direct or indirect superclass of this class or not.  **subclasses()** - returns array of direct and indirect subclasses.  **superclasses()** - returns array of direct and indirect superclasses. | select classof(o).name from instanceof java.lang.ref.Reference o |
| allocTrace(objName) | This returns allocation site trace of a given Java object if available.  返回对象的属性有：  **className** - name of the Java class whose method is running in the frame.  **methodName** - name of the Java method running in the frame.  **methodSignature** - signature of the Java method running in the frame.  **sourceFileName** - name of source file of the Java class running in the frame.  **lineNumber** - source line number within the method. |  |
| objectid(objName) | 获取对象的ID | select objectid(o) from java.lang.Object o |
| forEachReferrer(callback,obj) | 遍历指定对象的引用者 |  |
| reachables(obj) | 获取指定对象的可达对象集合 | select reachables(p) from java.util.Properties p |
| referrers(obj) | 返回指定对象的引用者集合 | select count(referrers(o)) from java.lang.Object o |
| referees(obj) | 返回指定对象的直接引用者集合 |  |
| refers(firstobj,secondObj) | 判断firstobj是否引用了secondObj |  |
| root(obj) | 判断是否根对象，如果是则返回描述，否则返回null |  |
| sizeof(obj) | 对象大小，但不包含其引用对象 |  |
| rsizeof(obj) | 对象大小，包含其引用对象,即堆深。其不仅与对象有关，还有当前对象的数据内容有关 |  |
| toHtml(obj) | 将对象转为html显示 | select "<b>" + toHtml(o) + "</b>" from java.lang.Object o |
|  |  |  |
|  |  |  |

### 1.5统计集合

|  |  |  |
| --- | --- | --- |
| 函数 | 说明 | 例子 |
| contains(objSet, booleanExpression) | 判断当前集合是否包含指定表达式的对象，  内置对象有：  **it** -> currently visited element  **index** -> index of the current element  **array** -> array/enumeration that is being iterated | select count(heap.classes(), "/java.io./.test(it.name)") |
| count(objSet, booleanExpression) | 当前集合包含指定表达式的对象的总数，  内置对象有：  **it** -> currently visited element  **index** -> index of the current element  **array** -> array/enumeration that is being iterated |  |
| filter(objSet, booleanExpression) | 获得当前集合包含指定表达式的对象的子集，  内置对象有：  **it** -> currently visited element  **index** -> index of the current element  **array** -> array/enumeration that is being iterated |  |
| length(objSet) | 返回集合中元素数量 |  |
| concat(objSet1,ObjSet2) | 链接两个集合 |  |
| map(objSet,transerRule) | 将集合中元素按特定规则转换  内置对象：  **it** -> currently visited element  **index** -> index of the current element  **array** -> array/enumeration that is being iterated  **result** -> result array/enumeration | select map(heap.findClass("java.io.File").statics, "index + '=' + toHtml(it)") |
| max(setObj,[express]) | 获得集合中最大的元素，默认进行数值比较  内置对象：  lhs -> left side element for comparison  rhs -> right side element for comparison | select max(map(heap.objects('java.lang.String', false), 'it.value.length'))  select max(heap.objects('java.lang.String'), 'lhs.value.length > rhs.value.length') |
| min(setObj,[express]) | 最小值，其他同max |  |
| sort(setObj,[express]) | 排序，其他同max |  |
| top(set,expression,num) | 获取集合中，指定规则的头几个对象  内置对象：  lhs -> left side element for comparison  rhs -> right side element for comparison |  |
| sum(setObj,[express]) | 计算集合的累计值，默认进行数值计算 | select sum(map(reachables(p), 'sizeof(it)')) from java.util.Properties p  select sum(reachables(p), 'sizeof(it)') from java.util.Properties p |
| unique(objSet) | 返回无重复的对象集合 |  |

## 2、Visual VM

### 2.1启动：

A screenshot of a social media post

Description automatically generated

### 2.2dump当前JVM

A screenshot of a social media post

Description automatically generated

A screenshot of a social media post

Description automatically generated

A screenshot of a cell phone

Description automatically generated

### 2.3插件

A screenshot of a computer

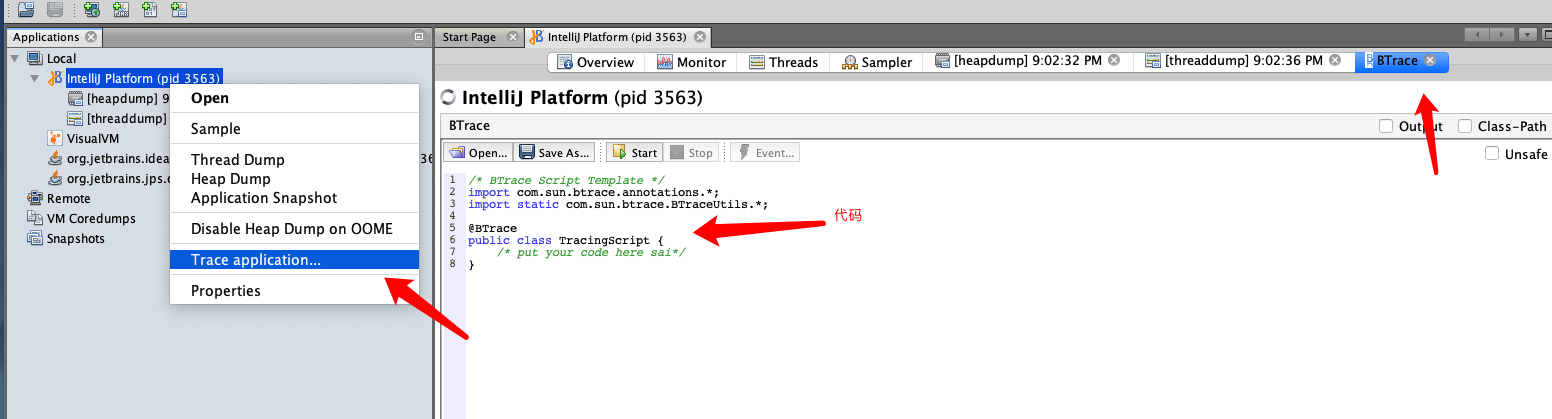
Description automatically generated

A screenshot of a social media post

Description automatically generated

### 2.4BTrace

通过字节码注入，动态监控系统的运行情况。它可以跟踪指定的方法调用、构造函数调用和系统内存等信息。在Visual VM中安装插件BTrace后，右击Java程序打开Trece application



例子脚本

@TLS

private static long startTime = 0;

@OnMethod(clazz="/.+/", //监控任意类

method="/slowMethod/") //监控slowMethod方法

public static void startMethod(){

startTime = timeMillis();

}

@OnMethod(clazz="/.+/",method="/slowMethod/",location=@Location(Kind.RETURN))//方法返回时触发

public static void endMethod(){

long time = timeMillis() - startTime;

println(strcat("execute time(nanos): ", str(time)));

}

3、JConsole

4、Mission Control

5、JHSDB

## 3、MAT

# 四、LINUX命令(centos7)

## 1、top

a、说明

实时显示系统各个进程的资源占用情况

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -a |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

c、显示参数说明



|  |  |  |
| --- | --- | --- |
| 第一部分 | | |
| 位置 | 参数 | 说明 |
| top | days | 系统已运行的时间 |
|  | users | 登陆用户数 |
|  | load average | 平均负载，1分钟，5分钟，15分钟 |
| Task | total | 进程总数 |
|  | running | 运行总数 |
|  | sleeping | 睡眠总数 |
|  | stopped | 停止总数 |
|  | zombie | 僵尸进程数(父进程已经退出,而该进程dead之后没有进程接受,就成为僵尸进程.(zombie)进程) |
| %Cpu | us | 用户占用率 |
|  | sy | 内核占用率 |
|  | ni | 用户改变过优先级的进程占用率 |
|  | id | 空闲CPU占用率 |
|  | wa | 等待中的CUP比例 |
|  | hi | 硬件中断请求占用率 |
|  | si | 软件中断请求占用率 |
|  | st | 被偷走百分比 |
| KiB Mem | total | 物理总量 |
|  | free | 空闲 |
|  | used | 使用 |
|  | buff/cache | 作为内核缓存的总量 |
| Kib Swap | total | 虚拟内存总量 |
|  | free | 空闲虚拟内存 |
|  | used | 虚拟内存使用量 |
|  | avail Mem | 可以用总量 |
| 第二部分 | | |
| head | PID | 进程号 |
|  | USER | 进程所有者用户 |
|  | PR | 优先级 |
|  | NI | nice值，负值表示高优先级，正值表示低优先级 |
|  | VIRT | 进程使用虚拟内存总量 |
|  | RES | 进程使用的、未被换出的物理内存大小 |
|  | SHR | 共享内存大小 |
|  | S | 进程状态。  D=不可中断的睡眠状态  R=运行  S=睡眠  T=跟踪/停止  Z=僵尸进程 |
|  | %CPU | CPU占比 |
|  | %MEM | 物理内存占比 |
|  | TIME+ | 使用CPU时间，单位 1/100秒 |
|  | COMMEND | 命令名/命令行 |

## 2、ps

a、说明

查看进程状态

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -A  -e |  |
| -a |  |
| a |  |
| -d |  |
| -N  --deselect |  |
| r |  |
| T |  |
| x |  |
| -C |  |
| -G <GID>  --GROUP <GID> |  |
| -g <group>  --group <group> |  |
| -p <PID>  p <PID>  --pid <PID> |  |
| --ppid <PID> |  |
| -q <PID>  q  --quick-pid <PID> |  |
| -s  --sid <session> |  |
| -t <tty>  t <tty>  --tty <tty> |  |
| -u, U, --user <UID> |  |
| -U, --User <UID> |  |
| -F |  |
| -f |  |
| f  --forest |  |
| -H |  |
| -j |  |
| j |  |
| -l |  |
| l |  |
| -M  Z |  |
| -O <format> |  |
| O |  |
| -o <format>  o <format>  --format <format> |  |
| s |  |
| u |  |
| v |  |
| X |  |
| -y |  |
| --context |  |
| --headers |  |
| --no-headers |  |
| --cols <num>  --columns <num>  width <num> |  |
| --rows <num>  --lines <num> |  |
| H |  |
| -L |  |
| -m  m |  |
| -T |  |
| -c |  |
| c |  |
| e |  |
| k  --sort |  |
| L |  |
| n |  |
| S  --cumulative |  |
| -y |  |
| -V  V  --version |  |
| -w  w |  |

c、显示参数说明

## 3、vmstat

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -a  --active |  |
| -f  --forks |  |
| -m  --slabs |  |
| -n  --one-header |  |
| -s  --stats |  |
| -d  --disk |  |
| -D  --disk-sum |  |
| -p <dev>  --partition <dev> |  |
| -S <char>  --unit <char> |  |
| -w  --wide |  |
| -t  --timestamp |  |
| -V  --version |  |

c、显示参数说明

## 4、iostat

a、说明

通过iostat方便查看CPU、网卡、tty设备、磁盘、CD-ROM 等等设备的活动情况, 负载信息。它的特点是汇报磁盘活动统计情况，同时也会汇报出CPU使用情况。同vmstat一样，iostat也有一个弱点，就是它不能对某个进程进行深入分析，仅对系统的整体情况进行分析

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -c |  |
| -d |  |
| -h |  |
| -k |  |
| -m |  |
| -N |  |
| -t |  |
| -V |  |
| -x |  |
| -y |  |
| -z |  |
| -j { ID | LABEL | PATH | UUID | ... } |  |
| -T |  |
| -g <group> |  |
| -p |  |

c、显示参数说明

## 5、uptime

a、说明

打印系统总共运行了多长时间和系统的平均负载

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -p  --pretty |  |
| -s  --since |  |
| -V  -vesion |  |

c、显示参数说明

## 6、dmesg

a、说明

显示内核缓冲区系统控制信息的工具, 有时候屏幕上的启动信息一闪而过，我们无法查看到具体信息，又或者服务器在电信机房，更看不到开机启动信息。这时候linux提供了dmesg这条命令

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -C  --clear |  |
| -c  --read-clear |  |
| -D  --console-off |  |
| -d  --show-delta |  |
| -e  --reltime |  |
| -E  --console-on |  |
| -F <file>  --file <file> |  |
| -f <list>  --facility <list> |  |
| -H  --human |  |
| -k  --kernel |  |
| -L  --color |  |
| -l <list>  --level <list> |  |
| -n <level>  --console-level <level> |  |
| -P  --nopager |  |
| -r  --raw |  |
| -S  --syslog |  |
| -s  --buffer-size <size> |  |
| -T  --ctime |  |
| -t  --notime |  |
| -u  --userspace |  |
| -w  --follow |  |
| -x  --decode |  |
| -V  --version |  |

c、显示参数说明

## 7、mpstat

a、说明

实时系统监控工具。其报告与CPU的一些统计信息，这些信息存放在/proc/stat文件中。在多CPUs系统里，其不但能查看所有CPU的平均状况信息，而且能够查看特定CPU的信息。mpstat最大的特点是：可以查看多核心cpu中每个计算核心的统计数据

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -A |  |
| -u |  |
| -V |  |
| -I |  |
| -P <cpu> |  |

c、显示参数说明

## 8、sar

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -A |  |
| -B |  |
| -b |  |
| -C |  |
| -d |  |
| -F |  |
| -H |  |
| -h |  |
| -p |  |
| -q |  |
| -R |  |
| -r |  |
| -s |  |
| -t |  |
| -u |  |
| -V |  |
| -v |  |
| -W |  |
| -w |  |
| -y |  |
| -I |  |
| -P |  |
| -m |  |
| -n |  |
| -j |  |
| -f |  |
| -o |  |
| -i |  |
| -s <hh:mm:ss> |  |
| -e <hh:mm:ss> |  |

c、显示参数说明

## 9、free

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -b  -bytes |  |
| -k  --kilo |  |
| -g  --giga |  |
| --tera |  |
| --peta |  |
| -h  --human |  |
| --si |  |
| -l  --lohi |  |
| -t  -total |  |
| -s N  --seconds N |  |
| -c N  -count N |  |
| -w  --width |  |

c、显示参数说明

## 10、tcpdump

a、说明

根据使用者的定义对网络上的数据包进行截获的包分析工具

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -B <size> |  |
| -c <count> |  |
| -C <file\_size> |  |
| -E |  |
| -F <file> |  |
| -G <seconds> |  |
| -i <interface> |  |
| -j <tstamptype> |  |
| -M |  |
| --number |  |
| -Q  -P |  |
| -r <file> |  |
| -s |  |
| --time-stamp-precision |  |
| --immediate-mode |  |
| -T |  |
| -V <file> |  |
| -w <file> |  |
| -W |  |
| -y |  |
| -z |  |
| -Z |  |

c、显示参数说明

## 11、netstat

a、说明

显示各种网络相关信息，如网络连接，路由表，接口状态 (Interface Statistics)，masquerade 连接，多播成员 (Multicast Memberships) 等等

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -r  --routec |  |
| -I  --interfaces=<Iface> |  |
| -i  --interfaces |  |
| -g  --groups |  |
| -s  --statistics |  |
| -M  --masquerade |  |
|  |  |
| -v  --verbose |  |
| -W  --wide |  |
| -n  --numeric |  |
| --numeric-hosts |  |
| --numeric-ports |  |
| --numeric-users |  |
| -N  --symbolic |  |
| -e  --extend |  |
| -p  --programs |  |
| -o  --timers |  |
| -c  --continuous |  |
| -l  --listening |  |
| -a  --all |  |
| -F  --fib |  |
| -C  --cache |  |
| -Z  --context |  |

c、显示参数说明

## 12、pidstat

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -d |  |
| -h |  |
| -I |  |
| -l |  |
| -r |  |
| -s |  |
| -t |  |
| -U |  |
| -u |  |
| -V |  |
| -w |  |
| -C |  |
| -p |  |
| -T |  |

c、显示参数说明

## 13、grep

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -E  --extended-regexp |  |
| -F  --fixed-strings |  |
| -G  --basic-regexp |  |
| -P  --perl-regexp |  |
| -e  --regexp=PATTERN |  |
| -f  --file=FILE |  |
| -i  --ignore-case |  |
| -w  --word-regexp |  |
| -x  --line-regexp |  |
| -z  --null-data |  |
| -s  --no-messages |  |
| -v  --invert-match |  |
| -m  --max-count=NUM |  |
| -b  --byte-offset |  |
| -n  --line-number |  |
| --line-buffered |  |
| -H  --with-filename |  |
| -h  --no-filename |  |
| --label=label |  |
| -o  --only-matching |  |
| -q  --quiet  --silent |  |
| --binary-files=TYPE |  |
| -a  --text |  |
| -I |  |
| -d  --directories=ACTION |  |
| -D  --devices=ACTION |  |
| -r  --recursive |  |
| -R  --dereference-recursive |  |
| --include=FILE\_PATTERN |  |
| --exclude=FILE\_PATTERN |  |
| --exclude-from=FILE |  |
| --exclude-dir=PATTERN |  |
| -L  --files-without-match |  |
| -l  --files-with-matches |  |
| -c  --count |  |
| -T  --initial-tab |  |
| -Z  --null |  |
| -B  --before-context=NUM |  |
| -A  --after-context=NUM |  |
| -C  --context=NUM |  |
| -NUM |  |
| --group-separator=SEP use SEP as a group separator |  |
| --no-group-separator |  |
| --color[=WHEN] |  |
| --colour[=WHEN] |  |
| -U  --binary |  |
| -u  --unix-byte-offsets |  |

c、显示参数说明

## 14、cat

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -A  --show-all |  |
| -b  --number-nonblank |  |
| -e |  |
| -E  --show-ends |  |
| -n  --number |  |
| -s  --squeeze-blank |  |
| -t |  |
| -T  --show-tabs |  |
| -u |  |
| -v  --show-nonprinting |  |

## 15、tail

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -c  --bytes |  |
| -f  --follow |  |
| -F |  |
| -n  --lines |  |
| --max-unchanged-stats |  |
| --pid |  |
| -q  --quiet  --silent |  |
| --retry |  |
| -s  --sleep-interval |  |
| -v --verbose |  |

## 16、curl

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| --anyauth |  |
| -a --append |  |
| --basic |  |
| --cacert |  |
| --capath |  |
| -E  --cert |  |
| --cert-type |  |
| --ciphers |  |
| --compressed |  |
| -K  --config |  |
| --connect-timeout |  |
| -C  --continue-at |  |
| -b  --cookie |  |
| -c  --cookie-jar |  |
| --create-dirs |  |
| --crlf |  |
| --crlfile |  |
| -d  --data |  |
| --data-ascii |  |
| --data-binary |  |
| --data-urlencode |  |
| --delegation |  |
| --digest |  |
| --disable-eprt |  |
| --disable-epsv |  |
| -D  --dump-header |  |
| --egd-file |  |
| --engine |  |
| -f  --fail |  |
| -F  --form |  |
| --form-string |  |
| --ftp-account |  |
| --ftp-alternative-to-user |  |
| --ftp-create-dirs |  |
| --ftp-method |  |
| --ftp-pasv |  |
| -P  --ftp-port |  |
| --ftp-skip-pasv-ip |  |
| --ftp-pret |  |
| --ftp-ssl-ccc |  |
| --ftp-ssl-ccc-mode |  |
| --ftp-ssl-control |  |
| -G  --get |  |
| -g  --globoff |  |
| -H  --header |  |
| -I  --head |  |
| -h  --help |  |
| --hostpubmd5 |  |
| -0  --http1.0 |  |
| --ignore-content-length |  |
| -i  --include |  |
| -k  --insecure |  |
| --interface |  |
| -4  --ipv4 |  |
| -6  --ipv6 |  |
| -j  --junk-session-cookies |  |
| --keepalive-time |  |
| --key |  |
| --key-type |  |
| --krb |  |
| --libcurl |  |
| --limit-rate |  |
| -l  --list-only |  |
| --local-port |  |
| -L  --location |  |
| --location-trusted |  |
| -M  --manual |  |
| --mail-from |  |
| --mail-rcpt |  |
| --mail-auth |  |
| --max-filesize |  |
| --max-redirs |  |
| -m  --max-time |  |
| --metalink |  |
| --negotiate |  |
| -n  --netrc |  |
| --netrc-optional |  |
| --netrc-file |  |
| -N  --no-buffer |  |
| --no-keepalive |  |
| --no-sessionid |  |
| --noproxy |  |
| --ntlm |  |
| -o  --output |  |
| --pass |  |
| --post301 |  |
| --post302 |  |
| --post303 |  |
| -#  --progress-bar |  |
| --proto |  |
| --proto-redir |  |
| -x  --proxy |  |
| --proxy-anyauth |  |
| --proxy-basic |  |
| --proxy-digest |  |
| --proxy-negotiate |  |
| --proxy-ntlm |  |
| -U  --proxy-user |  |
| --proxy1.0 |  |
| -p  --proxytunnel |  |
| --pubkey |  |
| -Q  --quote |  |
| --random-file |  |
| -r  --range |  |
| --raw |  |
| -e  --referer |  |
| -J  --remote-header-name |  |
| -O  --remote-name |  |
| --remote-name-all |  |
| -R  --remote-time |  |
| -X  --request |  |
| --resolve |  |
| --retry |  |
| --retry-delay |  |
| --retry-max-time |  |
| -S  --show-error |  |
| -s  --silent |  |
| --socks4 |  |
| --socks4a |  |
| --socks5 |  |
| --socks5-basic |  |
| --socks5-gssapi |  |
| --socks5-hostname |  |
| --socks5-gssapi-service |  |
| --socks5-gssapi-nec |  |
| -Y  --speed-limit |  |
| -y  --speed-time |  |
| --ssl |  |
| --ssl-reqd |  |
| -2  --sslv2 |  |
| -3  --sslv3 |  |
| --ssl-allow-beast |  |
| --stderr |  |
| --tcp-nodelay |  |
| -t  --telnet-option |  |
| --tftp-blksize |  |
| -z  --time-cond |  |
| -1  --tlsv1 |  |
| --tlsv1.0 |  |
| --tlsv1.1 |  |
| --tlsv1.2 |  |
| --tlsv1.3 |  |
| --tls-max |  |
| --trace |  |
| --trace-ascii |  |
| --trace-time |  |
| --tr-encoding |  |
| -T  --upload-file |  |
| --url |  |
| -B  --use-ascii |  |
| -u  --user |  |
| --tlsuser |  |
| --tlspassword |  |
| --tlsauthtype |  |
| --unix-socket |  |
| -A  --user-agent |  |
| -v  --verbose |  |
| -V  --version |  |
| -w  --write-out |  |
| --xattr |  |
| -q |  |

## 17、ping

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -6 |  |
| -c |  |
| -i |  |
| -I |  |
| -m |  |
| -M |  |
| -l |  |
| -p |  |
| -Q |  |
| -s |  |
| -S |  |
| -t |  |
| -T |  |
| -w |  |
| -W |  |

c、显示结果说明

## 18、vim

a、说明

b、参数

|  |  |
| --- | --- |
| 参数 | 说明 |
| -v |  |
| -e |  |
| -E |  |
| -s |  |
| -d |  |
| -y |  |
| -R |  |
| -Z |  |
| -m |  |
| -M |  |
| -b |  |
| -l |  |
| -C |  |
| -N |  |
| -V[N][fname] |  |
| -D |  |
| -n |  |
| -r |  |
| -r |  |
| -L |  |
| -A |  |
| -H |  |
| -F |  |
| -T |  |
| -u |  |
| --noplugin |  |
| -p[N] |  |
| -o[N] |  |
| -O[N] |  |
| --cmd |  |
| -c |  |
| -S |  |
| -s |  |
| -w |  |
| -W |  |
| -x |  |
| --startuptime |  |
| -i |  |

e、编辑时快捷键

|  |  |
| --- | --- |
| 快捷键 | 说明 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 五、Windows工具

### Perfmon

### Process Explorer

### pslist

# 六、JVM常用执行指令

## 1、java

1.1基础指令

java [options] [-jar filename][ -m| --module module[/mainclass]][mainClassname] [args]

https://docs.oracle.com/en/java/javase/14/docs/specs/man/java.html

https://docs.oracle.com/javase/8/docs/technotes/tools/unix/java.html

|  |  |
| --- | --- |
| classname | 运行的class文件 |
| args | main方法入参 |
| options | |
| -version[:release] |  |
| -jar | 执行一个jar包 |
| -m  --module | 执行一个模块，jdk9开始有模块概念 |
|  |  |
|  |  |

## 2、javac

2.1文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/javac.html

2.2基础指令

2.3主要参数

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 3、jar

3.1文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/jar.html

3.2基础指令

3.3主要参数

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

## 4、javap

4.1文档

https://docs.oracle.com/en/java/javase/14/docs/specs/man/javap.html

4.2基础指令

4.3主要参数

|  |  |
| --- | --- |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 附录

a堆分配参数示意图

