as-boot.jar -h` 打印更多参数信息。

执行前需要 有 正在运行的Java程序，然后我们去选择监听那个服务

本次我为大家提供了基础的测试java服务jar文件，如若获取请 `程序员Bug终结者`公众号 回复 `arthas` 即可获取

先运行提供 的 java服务 jar文件

![arthas1.png](https://p1-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/96e8a420d95848bda5018b2e5d6240b0~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1535&h=684&s=371702&e=jpg&b=000000)

然后另开一个窗口去运行 arthas java jar文件

![arthas2.png](https://p6-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/8912e2c9b9274f7288a417326ed36936~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1716&h=806&s=295776&e=jpg&b=000000)

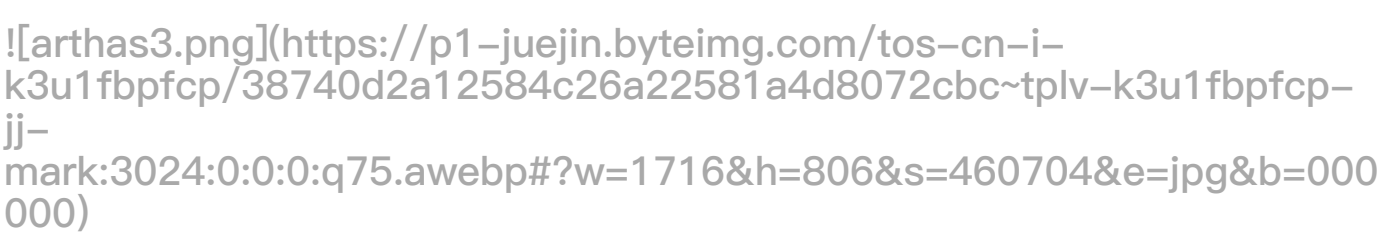
三、Arthas 基础命令
==========

常用命令列表

* dashboard - 当前系统的实时数据面板
* getstatic - 查看类的静态属性
* heapdump - dump java heap, 类似 jmap 命令的 heap dump 功能
* jvm - 查看当前 JVM 的信息
* logger - 查看和修改 logger
* mbean - 查看 Mbean 的信息
* memory - 查看 JVM 的内存信息
* ognl - 执行 ognl 表达式
* perfcounter - 查看当前 JVM 的 Perf Counter 信息

* sysenv – 查看 JVM 的环境变量
* sysprop – 查看和修改 JVM 的系统属性
* thread – 查看当前 JVM 的线程堆栈信息
* vmoption – 查看和修改 JVM 里诊断相关的 option
* vmtool – 从 jvm 里查询对象，执行 forceG

查看当前系统的实时数据面板

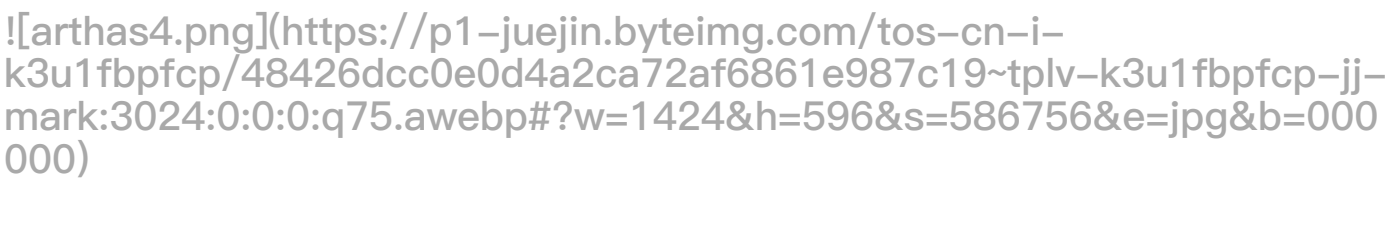


分为三大模块

* thread模块
* 内存模块
* 实时运行信息模块

进行详细查看 主线程的 详细信息

```
thread 1
```



打印最繁忙的3个线程

```
thread -n 3
```

![arthas5.png](https://p1-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/93578619d57b4659960a87a2677fa121~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1170&h=490&s=347934&e=jpg&b=000000 可能在中间被修改导致前后不一致，除了 `-b` 事件点 `params` 代表函数入参外，其余事件都代表函数出参
* 当使用 `-b` 时，由于观察事件点是在函数调用前，此时返回值或异常均不存在
* 在 watch 命令的结果里，会打印出`location`信息。`location`有三种可能值：`AtEnter`，`AtExit`，`AtExceptionExit`。对应函数入口，函数正常return，函数抛出异常。

cpu飙升演示

```
thread
```

执行cpu命令后，会迅速沾满内存，执行 thread命令查看线程信息

![arthas16.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/4a498a4ac4d54a55bde66f113163480d~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1646&h=526&s=374889&e=jpg&b=000000)
cpu 飙升 100%，查看该线程详细信息

```
thread 1
```

![arthas17.png](https://p3-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/5eb54bc4567043418f1b4d1d282f970b~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=1564&h=555&s=625176&e=jpg&b=000000)

反编译 cpu 方法代码分析具体问题

```
jad *ArthasServiceImpl cpu

```

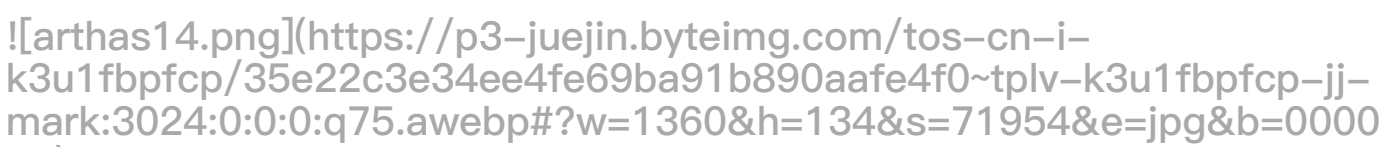
发现是 代码中存在死循环，解决该问题，再次运行即可。

方法演示

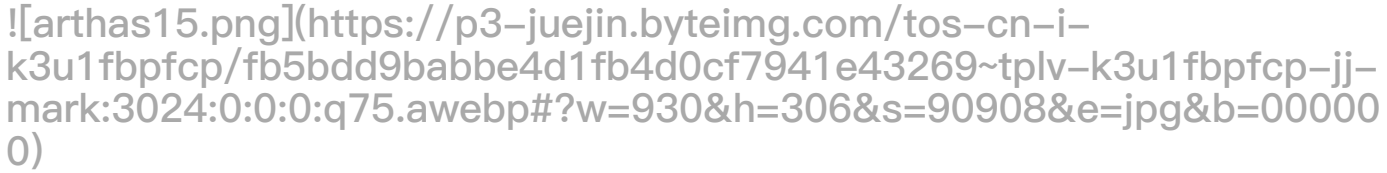
```
# 监听指定方法，查看方法执行时间
trace *ArthasServiceImpl method

```

先运行以上监听命令，再运行 method方法，即可查看方法返回的具体信息，消耗时间等。



反编译method方法



可以查看出问题的代码进行修复即可。

小结
==

以上就是【**Bug 终结者**】对 **都2024年了，线上部署你不会只会log 调试吧，Arthas了解下！** 的简单介绍， **通过本文已了解Arthas的基础使用**`技术改变世界！！！`** **下篇文章将为大家分享Arthas更高级的操作**

原文链接: https://juejin.cn/post/7357546247849230374