# 前言

记录学习笔记，请从官网重下载Arthas压缩包。

参考博客:https://blog.csdn.net/lydms/article/details/125238249

## 目录前言

解压其中包含以下文件：

arthas-agent.jar

arthas-core.jar

arthas.properties

as-service.bat

install-local.sh

**math-game.jar --官方demo**

**arthas-boot.jar --启动类**

arthas-output

arthas-spy.jar

as.sh

lib

arthas-client.jar

arthas-packaging-3.7.1-bin.zip

as.bat

async-profiler

logback.xml

## **下载地址(最新版)**

<https://arthas.aliyun.com/download/latest_version?mirror=aliyun>

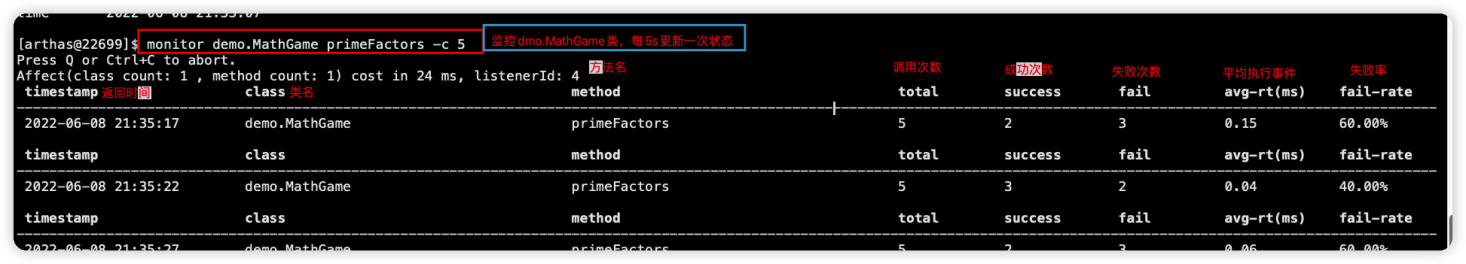
# 命令笔记

## 监控方法monitor -C

### 命令与实例

命令格式：monitor 包名+类 方法名 -c 5 (5S更新一次)

**monitor demo.MathGame primeFactors -c 5**



### **出参说明**

|  |  |
| --- | --- |
| 监控项 | 说明 |
| timestamp | 时间戳 |
| class | Java类 |
| method | 方法（构造方法、普通方法） |
| total | 调用次数 |
| success | 成功次数 |
| fail | 失败次数 |
| rt | 平均耗时 |
| fail-rate | 失败率 |

## 监控方法出入参watch

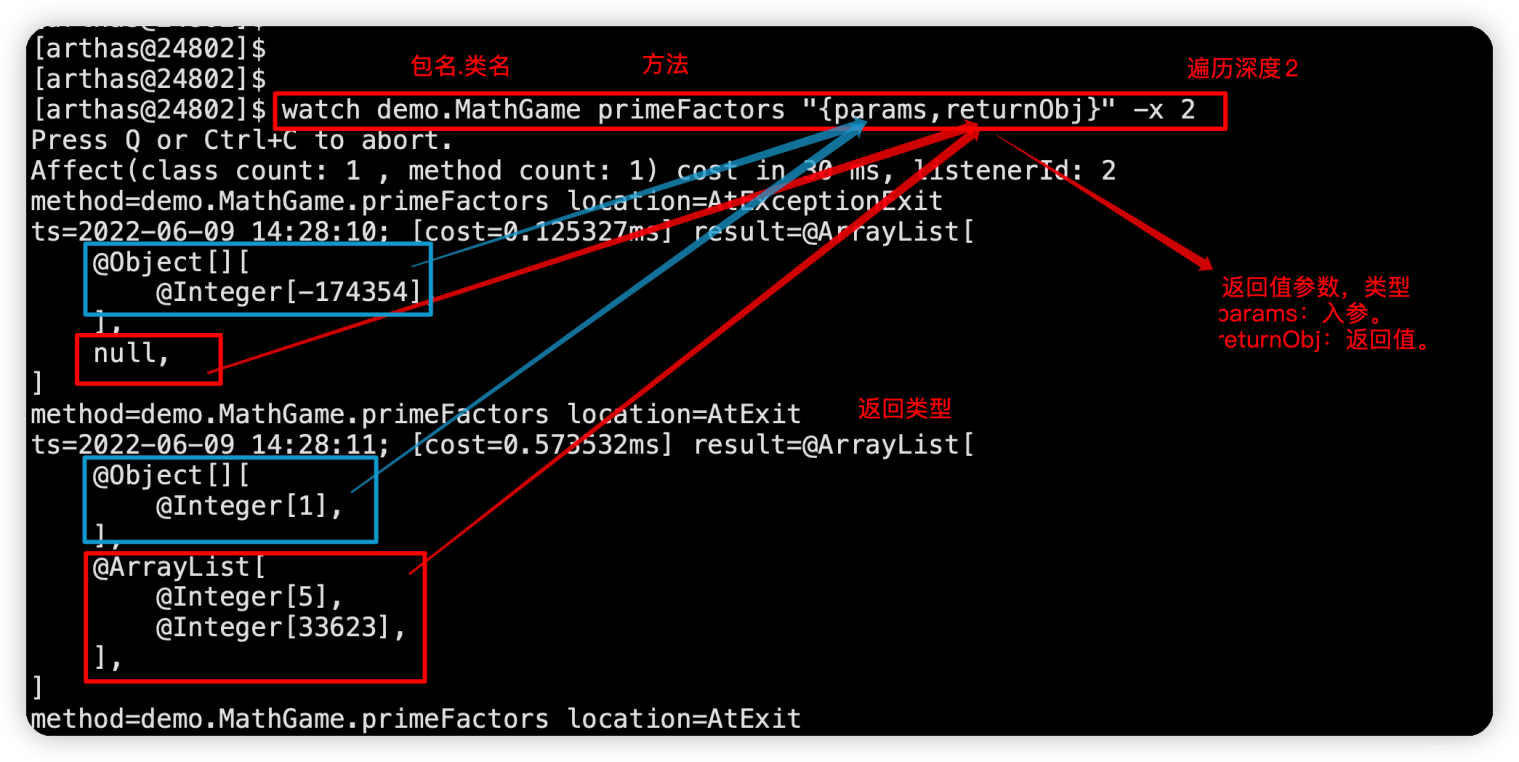
### **入参说明**

|  |  |  |
| --- | --- | --- |
| 参数名称 | | 参数说明 |
| class-pattern | | 类名表达式匹配 |
| method-pattern | | 方法名表达式匹配 |
| express | | 观察表达式 |
| condition-express | | |
| 可选参数 | 入参 | 说明 |
| [b] | 在方法调用之前观察**before** |
| [e] | 在方法异常之后观察 **exception** |
| [s] | 在方法返回之后观察 **success** |
| [f] | 在方法结束之后(正常返回和异常返回)观察 **finish** |
| [E] | 开启正则表达式匹配，默认为通配符匹配 |
| [x:] | 指定输出结果的属性遍历深度，默认为 1 |

### 实例(查看出入参)

命令格式：**watch** 包名+类 方法名 观察表达式 -x 2 遍历深度2

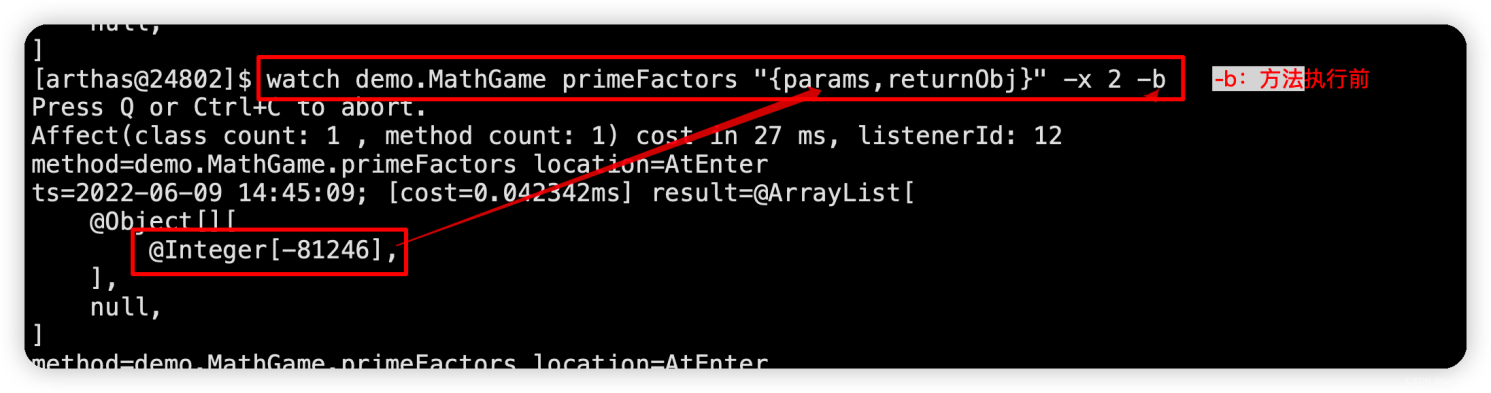
**watch demo.MathGame primeFactors "{params,returnObj}" -x 2**



### 实例(查看入参)

方法执行前的参数

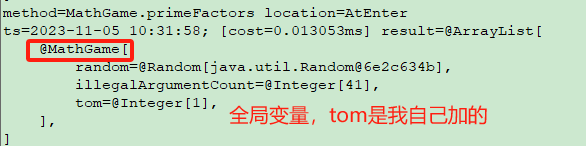
watch demo.MathGame primeFactors "{params,returnObj}" -x 2 -b



### 实例(查看全局变量)

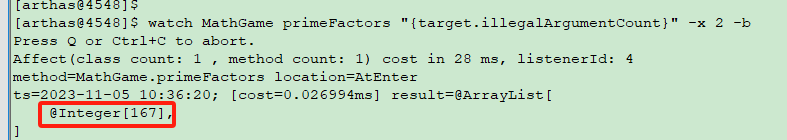
查看类的全局变量(感觉没什么用)

watch demo.MathGame primeFactors "{target}" -x 2 -b



### 实例(查看全局变量值)

watch MathGame primeFactors "{target.illegalArgumentCount}" -x 2 -b



### 实例(出参)

watch MathGame primeFactors "{params,target,returnObj}" -x 2 -b -s -n 2

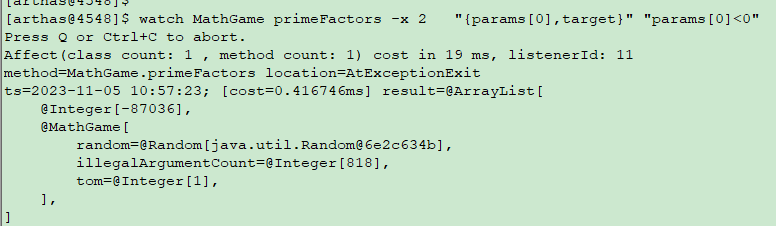
备注：-n 2执行两次



### 实例(判断)

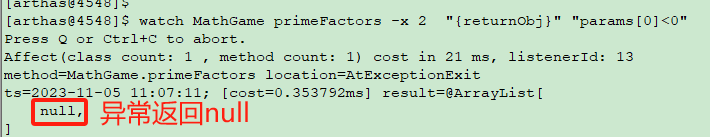
当执行某方法时第一入参是小于0，执行输出第一个入参和类的全局变量

watch MathGame primeFactors -x 2 "{params[0],target}" "params[0]<0"



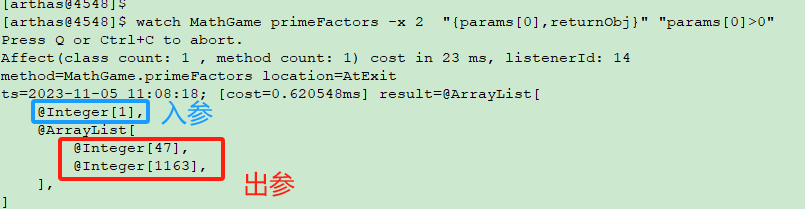
当执行某方法时第一入参是小于0，执行输出第一个入参和方法返回值

watch MathGame primeFactors -x 2 "{params[0],returnObj}" "params[0]<0"



当执行某方法时第一入参是大于0，执行输出第一个入参和方法返回值

watch MathGame primeFactors -x 2 "{params[0],returnObj}" "params[0]>0"



### 实例(查看异常堆栈)

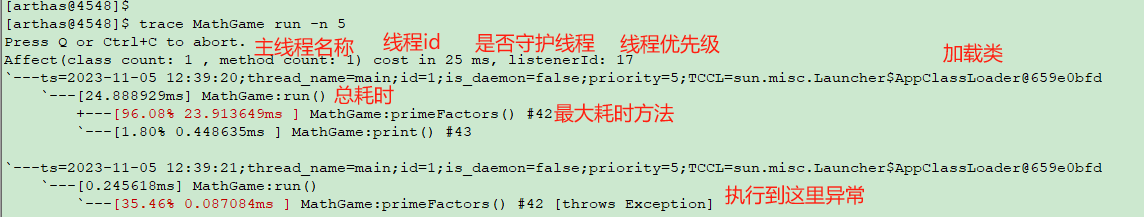
watch demo.MathGame primeFactors "{params[0], throwExp, @java.lang.Thread@currentThread().getStackTrace()}" -e -x 2



## 监控方法链路耗时trace

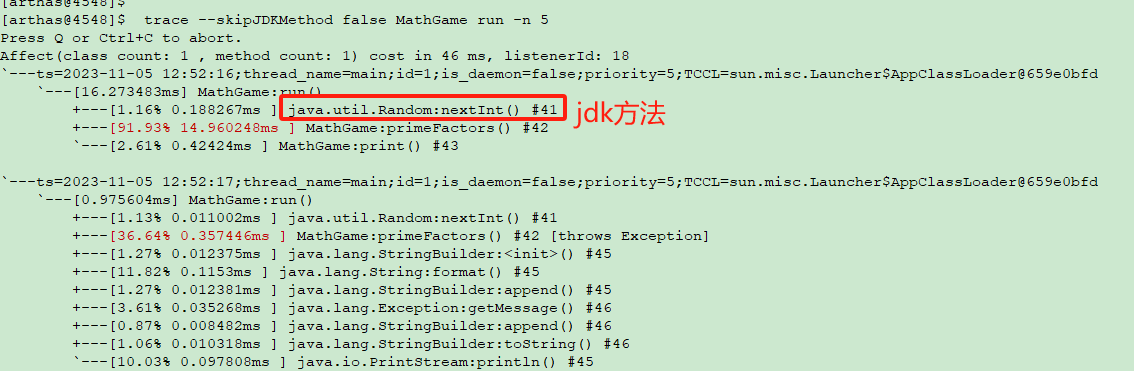
### 实例一

trace MathGame run -n 5(执行5次)



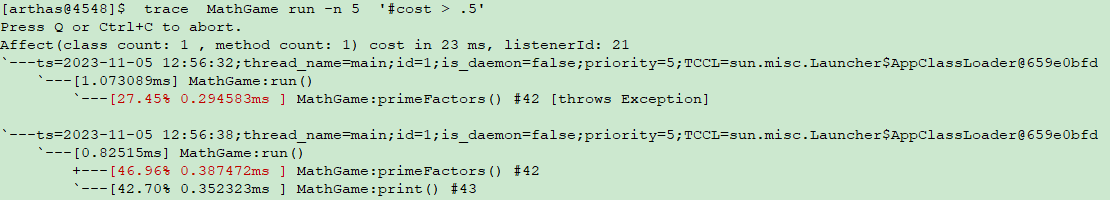
### 实例二(含jdk方法)

trace --skipJDKMethod false MathGame run -n 5



### 实例三(筛选时间)

trace MathGame run -n 5 '#cost > 1' (大于1ms)



### 实例四(正则)

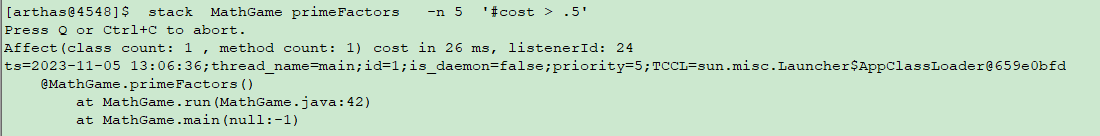
可以用正则表匹配路径上的多个类和函数，一定程度上达到多层trace的效果。

**trace -E com.test.ClassA|org.test.ClassB method1|method2|method3**

## 监控方法调用链路stack

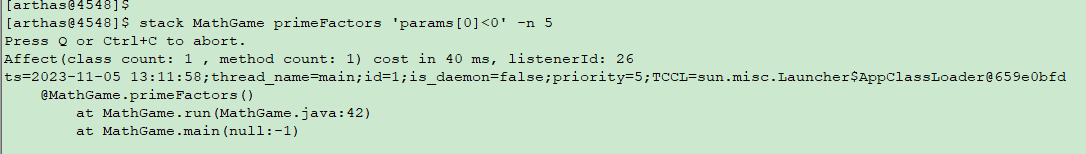
### 实例一(筛选时间)

stack MathGame primeFactors -n 5 '#cost > .5'



### 实例二(筛选参数)

stack MathGame primeFactors 'params[0]<0' -n 5



## 列表形式查看多个方法调用情况tt

### 实例

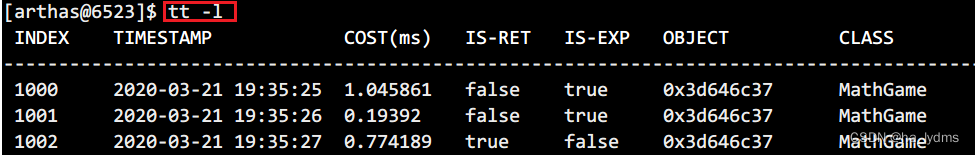
1. 打印调用情况

tt -t MathGame primeFactors -n 5(5次)



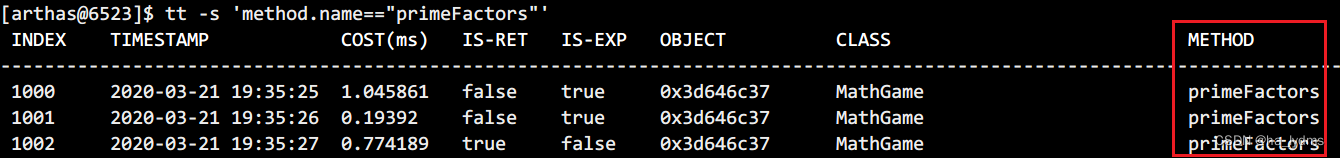
备注:OBJECT是执行对象的hashCode()，注意，曾经有人误认为是对象在JVM中的内存地址，但很遗憾他不是。但他能帮助你简单的标记当前执行方法的类实体。

1. 复查刚刚的记录 tt -l



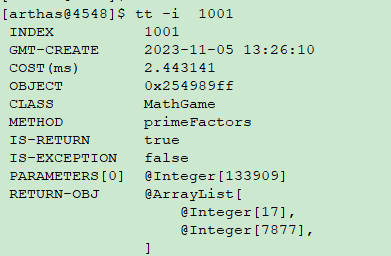
1. 需要筛选出 `primeFactors` 方法的调用信息

tt -s 'method.name=="primeFactors"'



1. 查看记录详情

tt -i 1000



1. 指定耗时(未实践)

tt -t \*Game primeFactors #cost>0.5

1. 指定异常(未实践)

tt -t \*Game primeFactors ‘throwExp instanceof NullpointException’

1. 指定异常(未实践)

tt -t \*Game primeFactors isThrow=true

### 解决重载问题

遇到多个重载方法，可以通过筛选参数实现。

1. 指定入参的长度=1

tt -t \*Game primeFactors params.length==1

1. 指定入参的类型

tt -t \*Game primeFactors 'params[0] instanceof Integer'

1. 指定固定的入参值

tt -t \*Game primeFactors '"params[0]==13989838402"'

1. 指定固定的入参值(未实践)

tt -t \*Game primeFactors '"params[0].name==abc"'

### 重复调用(不明白)

tt 命令由于保存了当时调用的所有现场信息，所以我们可以自己主动对一个 INDEX 编号的时间片自主发起一次调用，从而解放你的沟通成本。此时你需要 -p 参数。通过 --replay-times 指定 调用次数，通过 --replay-interval 指定多次调用间隔(单位ms, 默认1000ms)。

tt -i 1002 -p

#再重新调用3次

tt -i 1002 -p --replay-interval 3

#再重新调用3次,并且间隔2S

tt -i 1008 -p --replay-times 3 --replay-interval 2000

