# TProfiler介绍文档

**工具介绍**

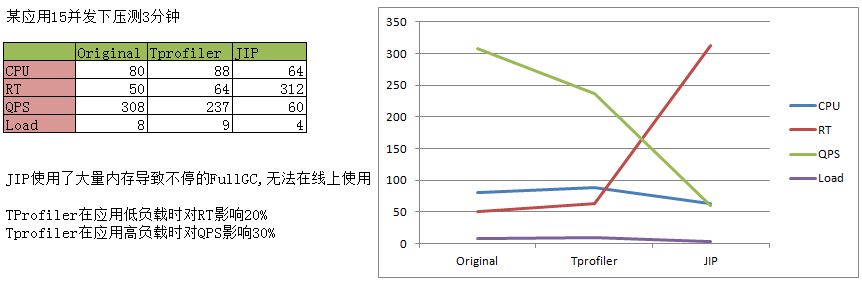
TProfiler是一个可以在生产环境长期使用的性能分析工具.它同时支持剖析和采样两种方式,记录方法执行的时间和次数,生成方法热点 对象创建热点 线程状态分析等数据,为查找系统性能瓶颈提供数据支持.

TProfiler在JVM启动时把时间采集程序注入到字节码中,整个过程无需修改应用源码.运行时会把数据写到日志文件,一般情况下每小时输出的日志小于50M.

业界同类开源产品都不是针对大型Web应用设计的,对性能消耗较大不能长期使用,TProfiler解决了这个问题.目前TProfiler已应用于淘宝的核心Java前端系统.

部署后低峰期对应用响应时间影响20% 高峰期对吞吐量大约有30%的降低(高峰期可以远程关闭此工具)

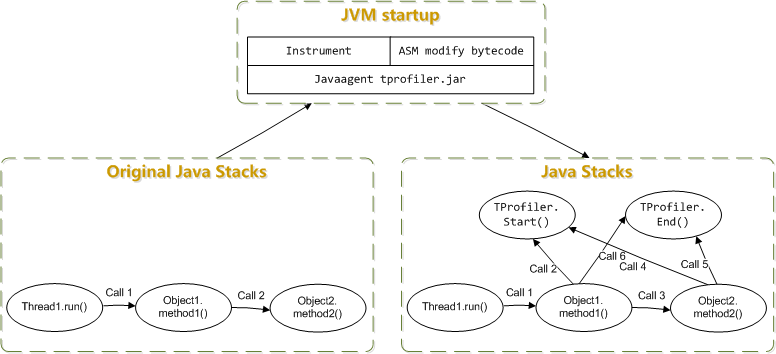
**同类对比**



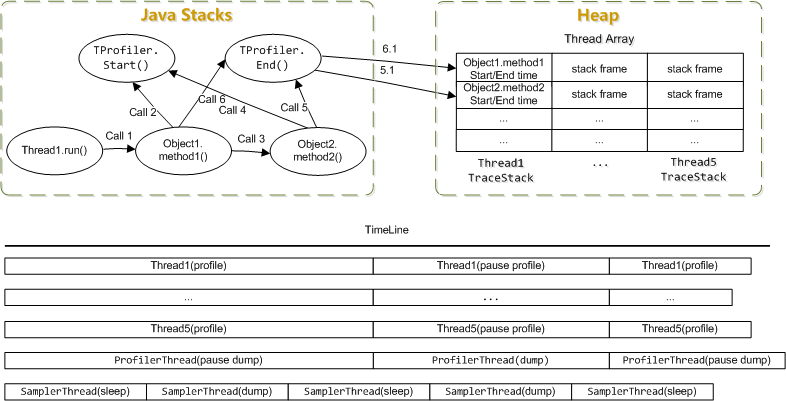
|  |  |  |
| --- | --- | --- |
| **与同类开源工具jip对比** | | |
| **项目** | **TProfiler** | **JIP** |
| 交互控制 | 支持远程开关和状态查看 | 支持远程开关等多种操作 |
| 过滤包和类名 | 支持包和类的过滤 | 支持包和类的过滤 |
| 低消耗 | 响应时间延长20% QPS降低30%(详细对比看上图) | 同等条件下资源消耗较多,使JVM不断的FullGC;Profile时会阻塞其他线程 |
| 无本地代码 | 未使用JVMTI,纯Java开发 | 未使用JVMTI,纯Java开发 |
| 易用性 | 只有一个jar包,使用简单 | 模块多,配置使用相对复杂 |
| 日志文件 | 对日志进行优化,每小时一般小于50M | 同等条件下日志大约是TProfiler的8倍,不能自动dump需要客户端触发 |
| 日志分析 | 目前只提供文本展示 | 可以利用客户端分析展示日志 |
| 使用场景 | 大型应用/小型应用 长期使用 | 小型应用 短期使用 |

# TProfiler实现原理

字节码修改



运行实现原理



# TProfiler日志分析

**tprofiler.log文件格式说明:**

线程ID 线程栈深度 方法ID 方法执行时间

13 2 14558 6

13 2 14554 2

13 3 14576 2

13 4 14567 2

13 2 14554 4

13 2 14556 13

**分析sampler log命令:** java -cp tprofiler.jar com.taobao.profile.analysis.SamplerLogAnalysis d:/tsampler.log d:/method.log d:/thread.log,会生成method.log和thread.log

**method.log文件格式说明:**

方法信息 采样过程中方法出现次数

org.quartz.simpl.SimpleThreadPool.getNextRunnable(SimpleThreadPool.java:428) 19728

org.quartz.simpl.SimpleThreadPool$WorkerThread.run(SimpleThreadPool.java:518) 19728

org.quartz.simpl.SimpleThreadPool.access$000(SimpleThreadPool.java:47) 19728

org.apache.mina.util.NamePreservingRunnable.run(NamePreservingRunnable.java:51) 17558

**thread.log文件格式说明:**

线程信息 采样过程中线程出现次数

56 DefaultQuartzScheduler\_Worker-3 TIMED\_WAITING 661

55 DefaultQuartzScheduler\_Worker-2 TIMED\_WAITING 661

60 DefaultQuartzScheduler\_Worker-7 TIMED\_WAITING 661

**分析profiler log命令:** java -cp tprofiler.jar com.taobao.profile.analysis.ProfilerLogAnalysis d:/tprofiler.log d:/tmethod.log d:/topmethod.log d:/topobject.log,会生成topmethod.log和topobject.log

**topmethod.log文件格式说明:**

方法信息 执行次数 平均执行时间 全部执行时间

org/apache/velocity/runtime/parser/node/ASTBlock:render:74 38954 28 1101570

org/apache/velocity/runtime/parser/node/SimpleNode:render:338 39202 21 820064

**topobject.log文件格式说明:**

方法信息 执行次数 平均执行时间 全部执行时间

sketch/compile/parser/node/PropertyExecutor:<init>:32 573 1 636

sketch/util/introspection/UberspectImpl:<init>:282 34 7 241

# TProfiler配置部署

## 打包

下载源码,在根目录运行package.bat脚本或者执行mvn assembly:assembly命令,将生成的包含依赖的jar包重命名为tprofiler.jar即可

## 配置

profile.properties作为tprofiler.jar的配置文件，需要和Jar包放在相同的目录(例如/home/admin/tprofiler/),根据实际情况调整配置;如果不显式配置这个文件,将使用默认配置

其中

startProfTime = 9:00:00 开始profile的时间点

endProfTime = 11:00:00 结束profile的时间点

eachProfUseTime = 5 profile时间长度(单位秒)

eachProfIntervalTime = 50 两次profile的时间间隔(单位秒)

samplerIntervalTime = 20 两次采样的时间间隔(单位秒)

debugMode = false 是否进入调试模式

needNanoTime = false 是否需要用纳秒记录时间

ignoreGetSetMethod = true 是否忽略采集get/set方法

logFilePath = /home/admin/logs/tprofiler.log profile log的存放位置

methodFilePath = /home/admin/logs/tmethod.log method log的存放位置

samplerFilePath = /home/admin/logs/tsampler.log sampler log的存放位置

excludeClassLoader = org.eclipse.osgi.internal.baseadaptor.DefaultClassLoader 不注入的类加载器(分号分割)

includePackageStartsWith = com.taobao 进行profile的类包名(分号分割)

excludePackageStartsWith = com.alibaba;com.taobao.forest.domain.dataobject 不进行profile的类包名(分号分割)

## 部署

在 jvm 启动参数中添加-javaagent:/path/tprofiler.jar -Dprofile.properties=/path/profile.properties，然后重启应用

重启后就会在logFilePath设置的路径下出现log文件,共有3个:

1. tsampler.log
2. tprofiler.log
3. tmethod.log

为了提升性能将method单独输出,此一项改进使log减小为原来的1/8

endProfTime时间到了后才会输出tmethod.log(因为在这个时间之前有些类和方法没有被load,所以选择profile结束后一次性输出method)

## 远程操作

远程查看状态操作:

java -cp tprofiler.jar com.taobao.profile.client.TProfilerClient 10.232.\*.\* 端口号 status

远程开始操作:

java -cp tprofiler.jar com.taobao.profile.client.TProfilerClient 10.232.\*.\* 端口号 start

远程停止操作:

java -cp tprofiler.jar com.taobao.profile.client.TProfilerClient 10.232.\*.\* 端口号 stop

远程刷出方法数据:

java -cp tprofiler.jar com.taobao.profile.client.TProfilerClient 10.232.\*.\* 端口号 flushmethod

## 卸载

在 jvm 启动参数中去掉相关参数 -javaagent:/path/tprofiler.jar -Dprofile.properties=/path/profile.properties, 重启应用即可卸载此工具

## 注意事项

在采集时间区间内重启应用, 需要手动清空当天生成的日志; 采集时间过去以后, 修改时间区间重新启动应用, 需要手动清空当天生成的日志

因为当天日志会续写, **两次采集的日志写到一个文件中, 会导致分析结果不准! 线上长期运行的应用无此问题**