BTrace动态追踪工具

— 黄阳全

这个返回值是啥?

线上不能debug!

貌似是依赖jar包
出问题?

这个参数是啥?

卧槽！！
这地方为什么没日志

到底是哪里耗时高了

到底是谁在调这个函数?

BTrace

A safe, dynamic tracing tool for the Java platform

1. BTrace介绍

2. BTrace基本使用及案例

3. BTrace原理浅析

1. BTrace介绍

BTrace 介绍

SUN Kenai云计算开发平台下的一个开源项目

https://github.com/btraceio/btrace

直接attach应用JVM，不用重启应用进程，可快速方便地定位问题

Client（Java compile api + attach api）

Agent（脚本解析引擎 + ASM + JDK6 Instumentation）

Socket

2. BTrace基本使用及案例

BTrace基本使用及案例

btrace [-p <port>] [-cp <classpath>] <pid> <btrace-script> [<args>]

BTrace基本使用及案例

btrace <pid> xxx.java

BTrace基本使用及案例

安装BTrace

BTrace基本使用及案例

1）下载地址：https://github.com/btraceio/btrace/releases/tag/v1.3.11.1

2）解压缩

3）设置环境变量

```
BTRACE_HOME=/data1/workspace_yangquan1/btrace/bin
export BTRACE_HOME
export PATH=$PATH:$BTRACE_HOME/bin
```

BTrace基本使用及案例

1. 获取方法的参数和返回值
2. 获取方法的执行时间
3. 打印当前线程堆栈信息
4. 获取类的属性值
5. 获取方法的调用次数

DEMO应用

```
public class TargetVM {

    private String name;

    public static void main(String[] args) {
        TargetVM main = new TargetVM();
        main.name = "just test";
        for (int i = 0; i < 20000; i++) {
            try {
                Thread.sleep(1000);
                main.invokeMethod("hello" + i);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }

    private int invokeMethod(String arg) {
        return new Random().nextInt();
    }

}
```

获取方法的参数和返回值

获取方法的参数和返回值

```
import com.sun.btrace.AnyType;
import com.sun.btrace.annotations.*;

import static com.sun.btrace.BTraceUtils.printArray;
import static com.sun.btrace.BTraceUtils.println;

@BTrace
public class PrintArgs {

    @OnMethod(clazz = "/application\\..*/", method = "invokeMethod", location = @Location(Kind.RETURN))
    public static void run(@ProbeClassName String className, @ProbeMethodName String methodName,
                           AnyType[] args, @Return AnyType result) {
        println("class name: " + className);
        println("method name: " + methodName);
        println("args: ");
        printArray(args);
        println("result: " + result);
    }

}
```

BTrace基本使用及案例

@OnMethod

1、使用全限定名：clazz="com.weibo.common.BtraceCase", method="add"

2、使用正则表达式：clazz="/javax.swing../", method="/./"

3、使用接口：clazz="+com.weibo.demo.Filter", method="doFilter"

4、使用注解：clazz="@javax.jws.WebService", method=""@javax.jws.WebMethod"

5、如果需要分析构造方法，需要指定method="<init>"

BTrace基本使用及案例

@Location

1、Kind.ENTRY

2、Kind.RETURN

3、Kind.CALL

4、Kind.LINE

5、Kind.ERROR, Kind.THROW, Kind.CATCH

BTrace基本使用及案例

Method Annotations

1. @OnMethod

2. @OnTimer

3. @OnError

4. @OnExit

5. @OnEvent

获取方法的参数和返回值

```
import com.sun.btrace.AnyType;
import com.sun.btrace.annotations.*;

import static com.sun.btrace.BTraceUtils.printArray;
import static com.sun.btrace.BTraceUtils.println;

@BTrace
public class PrintArgs {

    @OnMethod(clazz = "/application\\..*/", method = "invokeMethod", location = @Location(Kind.RETURN))
    public static void run(@ProbeClassName String className, @ProbeMethodName String methodName,
                           AnyType[] args, @Return AnyType result) {
        println("class name: " + className);
        println("method name: " + methodName);
        println("args: ");
        printArray(args);
        println("result: " + result);
    }

}
```

使用限制

不能新建类, 新建数组, 抛异常, 捕获异常,

不能调用实例方法以及静态方法(com.sun.btrace.BTraceUtils除外)

不能将目标程序和对象赋值给BTrace的实例和静态field

不能定义外部, 内部, 匿名, 本地类

不能有同步块和方法

不能有循环

不能实现接口, 不能扩展类

不能使用assert语句, 不能使用class字面值

获取方法的参数和返回值

run

获取类的属性值

```
import com.sun.btrace.BTraceUtils;
import com.sun.btrace.annotations.BTrace;
import com.sun.btrace.annotations.OnMethod;
import com.sun.btrace.annotations.Self;

import java.lang.reflect.Field;

import static com.sun.btrace.BTraceUtils.println;
```

```
@BTrace
public class PrintField {

    @OnMethod(clazz = "/application\\..*/", method = "invokeMethod")
    public static void run(@Self Object obj) {
        Field field = BTraceUtils.field("application.Test", "name");
        String name = (String) BTraceUtils.get(field, obj);
        println("test name: " + name);
    }

}
```

```
[root@box028 java]# btrace 228368 trace/PrintField.java
test name: just test
test name: just test
test name: just test
test name: just test
test name: just test
```

BTrace基本使用及案例

获取方法的执行时间

```
import com.sun.btrace.annotations.*;
import static com.sun.btrace.BTraceUtils.println;

@BTrace
public class PrintCost {
    @OnMethod(clazz = "/application\\..*/", method = "invokeMethod", location =
@Location(Kind.RETURN))
    public static void run(@ProbeClassName String className, @ProbeMethodName String methodName,
@Duration long duration) {
        println("class name: " + className);
        println("method name: " + methodName);
        println("cost(nanos): " + duration);
    }

}
```

@鹏飞 解决story Discover_card_list 接口探测报警超时问题，
统计TextEncoderHelper.encodeText方法耗时

BTrace基本使用及案例

打印当前线程堆栈信息

```
import com.sun.btrace.annotations.*;

import static com.sun.btrace.BTraceUtils.jstack;
import static com.sun.btrace.BTraceUtils.println;

@BTrace
public class printJstack {

    @OnMethod(clazz = "/application\\..*/", method = "/invokeMethod/", location = @Location(Kind.RETURN))
    public static void run(@ProbeClassName String className, @ProbeMethodName String methodName) {
        println("class name: " + className);
        println("method name: " + methodName);
        println("Invoke stack: ");
        jstack();
    }

}
```

@子龙 解决Tomcat NIO导致大文件上传hang住的问题，
了解到一个底层调用是哪些方法在调用

3. BTrace原理浅析

Btrace原理浅析

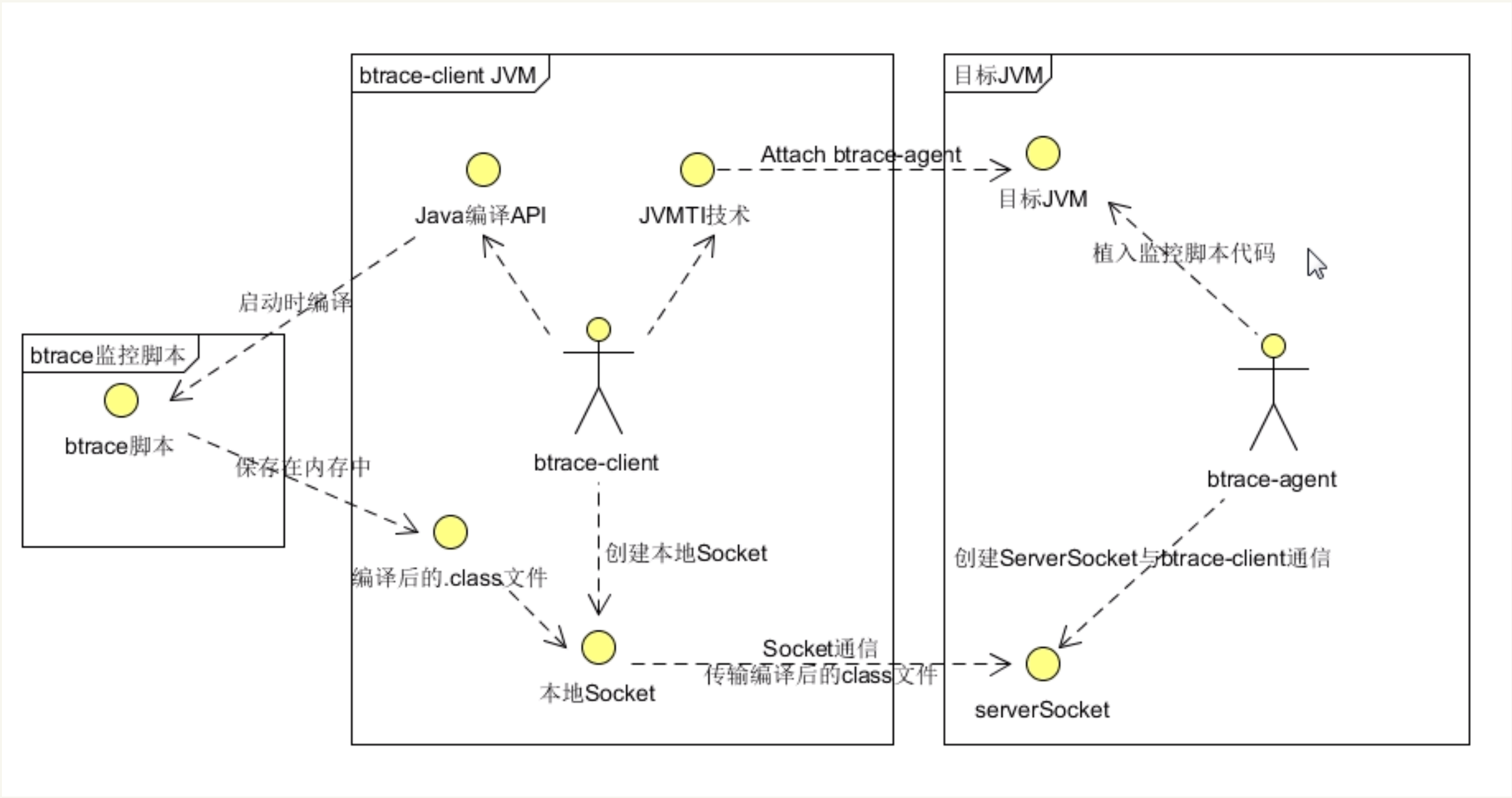
Client（Java compile api + attach api）

Agent（脚本解析引擎 + ASM + JDK6 Instumentation）

Socket

Btrace原理浅析

btrace流程图



btrace源码分析

```
try {
    Client client = new Client(port, OUTPUT_FILE, PROBE_DESC_PATH,
        DEBUG, TRACK_RETRANSFORM, TRUSTED, DUMP_CLASSES, DUMP_DIR, statsdDef);
    if (! new File(fileName).exists()) {
        errorExit("File not found: " + fileName, 1);
    }

    // 1. 编译btrace脚本
    byte[] code = client.compile(fileName, classPath, includePath);
    if (code == null) {
        errorExit("BTrace compilation failed", 1);
    }

    // 2. attach到目标VM
    client.attach(pid, null, classPath);
    registerExitHook(client);
    if (con != null) {
        registerSignalHandler(client);
    }

    // 3. 提交btrace请求
    if (isDebug()) debugPrint("submitting the BTrace program");
    client.submit(fileName, code, btraceArgs,
        createCommandListener(client));
} catch (IOException exp) {
    errorExit(exp.getMessage(), 1);
}
```

btrace源码分析

```
public void attach(String pid, String sysCp, String bootCp) throws IOException {
    try {
        String agentPath = "/btrace-agent.jar";
        String tmp = Client.class.getClassLoader().getResource("com/sun/btrace").toString();
        tmp = tmp.substring(0, tmp.indexOf('!'));
        tmp = tmp.substring("jar:".length(), tmp.lastIndexOf('/'));
        agentPath = tmp + agentPath;
        agentPath = new File(new URI(agentPath)).getAbsolutePath();
        attach(pid, agentPath, sysCp, bootCp);
    } catch (RuntimeException re) {
        throw re;
    } catch (IOException ioexp) {
        throw ioexp;
    } catch (Exception exp) {
        throw new IOException(exp.getMessage());
    }
}
```

btrace源码分析

```
try {
    // 参数解析
    loadArgs(args);

    // do something

    // 启动server
    Thread agentThread = new Thread(new Runnable() {
        @Override
        public void run() {
            BTraceRuntime.enter();
            try {
                startServer();
            } finally {
                BTraceRuntime.leave();
            }
        }
    });
} finally {
    inst.addTransformer(transformer, true);
    Main.debugPrint("Agent init took: " + (System.nanoTime() - ts) + "ns");
}
```

btrace源码分析

```
public void submit(String fileName, byte[] code, String[] args,
        CommandListener listener) throws IOException {
    if (sock != null) {
        throw new IllegalStateException();
    }
    submitDTrace(fileName, code, args, listener);
    try {
        long timeout = System.currentTimeMillis() + 5000;
        // 与目标VM进行通信
        while (sock == null && System.currentTimeMillis() <= timeout) {
            try {
                sock = new Socket("localhost", port);
            } catch (ConnectException e) {
                Thread.sleep(20);
            }
        }
        oos = new ObjectOutputStream(sock.getOutputStream());

        // do something

        // 给目标VM发送InstrumentCommand
        WireIO.write(oos, new InstrumentCommand(code, args, new
DebugSupport(sSettings)));
        ois = new ObjectInputStream(sock.getInputStream());
        commandLoop(listener);
    } catch (UnknownHostException uhe) {
        throw new IOException(uhe);
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }
}
```

Btrace原理浅析

自己动手实现一个

Btrace原理浅析

对正在运行的程序加一段代码，动态改变其方法

Btrace原理浅析

1. java attach api附加agent.jar

2. 使用asm来重写指定类的字节码

3. 使用instrument实现对原有类的替换

Btrace原理浅析

准备agent.jar

Btrace原理浅析

```java
public class Transformer implements ClassFileTransformer {

    private static final String methodName = "invokeMethod";

    public byte[] transform(ClassLoader loader, String className, Class<?> classBeingRedefined,
                            ProtectionDomain protectionDomain, byte[] classfileBuffer) throws IllegalClassFormatException {
        if (!className.equals("application/TargetVM")) {
            return null;
        }
        return getRewriteClassBytes(classfileBuffer, methodName);
    }

    /**
     * ASM 动态改变方法字节码
     *
     * @param classfileBuffer
     * @param methodName
     * @return
     */
    private byte[] getRewriteClassBytes(byte[] classfileBuffer, String methodName) {
        ClassReader cr = new ClassReader(classfileBuffer);
        ClassNode cn = new ClassNode();
        cr.accept(cn, 0);
        for (Object obj : cn.methods) {
            MethodNode md = (MethodNode) obj;
            if (md.name.equals(methodName)) {
                InsnList insns = md.instructions;
                InsnList il = new InsnList();
                il.add(new FieldInsnNode(Opcodes.GETSTATIC, "java/lang/System", "out", "Ljava/io/PrintStream;"));
                il.add(new LdcInsnNode("Hi, I'm Here !!! " + cn.name + "#" + md.name));
                il.add(new MethodInsnNode(Opcodes.INVOKEVIRTUAL, "java/io/PrintStream", "println", "(Ljava/lang/String;)V"));
                insns.insert(il);
                md.maxStack += 3;
            }
        }
        ClassWriter cw = new ClassWriter(0);
        cn.accept(cw);
        return cw.toByteArray();
    }

}
```

```
public class TestAgent {

    public static void agentmain(String args, Instrumentation inst) {
        inst.addTransformer(new Transformer(), true);
        inst.retransformClasses(TargetVM.class);
        System.out.println("Agent Main Done");
    }

}
```

```
Manifest-Version: 1.0
Can-Redefine-Classes: true
Can-Retransform-Classes: true
Agent-Class: attach.agent.TestAgent
Permissions: all-permissions
```

Btrace原理浅析

准备attach

```
public class TestMain {

    public static void main(String[] args) throws AttachNotSupportedException,
            IOException, AgentLoadException, AgentInitializationException {
        VirtualMachine vm = VirtualMachine.attach(args[0]);
        vm.loadAgent("attach/agent/testAgent.jar");
    }

}
```

```
java attach.TestMain <pid>
```

Btrace原理浅析

Btrace原理浅析

回顾

THX