INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X

Impact factor: 4.295

(Volume 4, Issue 2)

Available online at: www.ijariit.com

Java Byte Code Instrumentation

Vipul Saini
vipul.saini32@gmail.com
Meerut Institute of Engineering and Technology, Meerut, Uttar Pradesh

Ajay Kumar Singh
ajay41274@gmail.com
Meerut Institute of Engineering and Technology, Meerut, Uttar Pradesh

ABSTRACT

Byte code Instrumentation (BCI) is a technique for adding bytecode to a Java class during "run time." It's not really during run time, but more during "load" time of the Java class.we write Java code—e.g., create a *.Java file—we compile the code—e.g., creating a *.class file, which is written in bytecode, and when we execute it, an interpreter—the Java.EXE—is responsible for actually executing the commands written in the bytecode format within the *.class file. As with any interpreter, since we are not dealing with real object code, one can manipulate the actual code written in the executed file.

If want to add functionality to a Perl/PHP/JSP/ASP code—that's easy. We could simply open the file in a text editor, change the code, and next time it was executed it would behave differently. We could easily write a program that changes the code back and forth as we wish as a result of some user interface activity.

With bytecode it's the same concept, only a bit trickier. Try to open bytecode in a text editor—not something we want to work with…but still possible .Anyhow, the way to manipulate the actual bytecode is by intervening during the class loading flow and changing code on the fly. Every JVM (Java Virtual Machine) will first load all the class files (sometime it will do it only when really required, but that doesn't change the following description) to its memory space, parsing the bytecode, and making it available for execution. The main () function, as it calls different classes, is actually accessing code which was prepared by the JVM's class loaders. There is a class loader hierarchy, and there is the issue of the classpath but all that is out of the scope of So the basic concept of bytecode instrumentation is to add lines of bytecode before and after specific method calls within a class, and this can be done by intervening with the class loader. Back in the good old days, with JDK <1.5, we needed to really mess with the class loader code to do that. From JDK 1.5 and above, Java introduced the Java agent interface, which allows writing Java code that will be executed by the class loader itself, thus allowing the manipulation of the bytecode within every specific class, and making the whole process pretty straightforward to implement, thus the zillion different products for Java profiling and "transaction management" for Java applications.

Keywords: *Java Agent, Application Monitoring, Profiler, BCI, Byte code, Agent, Transaction.*

1. INTRODUCTION

Bytecode instrumentation is a technique for changing the code of compiled Java applications, either before running them – on the disk, or "on the fly" as they are loaded into memory.

1.1 Need of bytecode instrumentation?

In bytecode instrumentation process, class code become modify before runtime, after adding functionality we can capture profiling data. Which is used for monitoring the business Application.

2. APPLICATION DOMAIN

ByteCode instrumentation is a technique, which can be add functionality before runtime. Which is used to capture some metrics of application.

2.1 Monitoring

Bytecode instrumentation technique is helpful to monitor application in real time or we can generate reports later for analysis.

2.1.1 Profiling

A typical application of bytecode instrumentation is profiling.Java profiling tools can be great for troubleshooting complex issues in Java applications. These tools range in cost and functionality from free Java profilers that ship with the JDK to commercial software like YourKit. The biggest limitation of Java profiling tools, however, is the overhead they create on the application. Java profilers introduce latency and overhead that are unacceptable for any production application. Netdiagnostics is an example of a (commercial) Java profiler that utilizes bytecode instrumentation.

3. PROPOSED WORK- INSTRUMENTING JAVA BYTE CODE

3.1 bytecode instrumentation Using ASM

A bytecode instruction is made of an opcode that identifies this instruction, and of a fixed number of arguments:

- The opcode is an unsigned byte value – hence the bytecode name – and is identified by a mnemonic symbol. For example the opcode value 0 is designed by the mnemonic symbol NOP, and corresponds to the instruction that does nothing.
- The arguments are static values that define the precise instruction behavior. They are given just after the opcode.

This is the adapter which I created , and it is capable to fetch so many information about application.

```
public class MethodAdapter extends ClassVisitor
{

  private final boolean useMethodNameInRecords = false;

  private String class_name;
  private String className;

  private NDAgentFlags ndAgentFlags ;

  private final static int INSTR_LEVEL_NO_EXIT_METHOD = 6; // All except method exit
  private final static int INSTR_LEVEL_NO_DUMP_REC = 5; // Above minus dumping of record of method entry
  private final static int INSTR_LEVEL_NO_WALL_TIME = 4; // Above minus taking of wall time
  private final static int INSTR_LEVEL_NO_CPU_TIME = 3; // Above minus taking of cpu time
  private final static int INSTR_LEVEL_NO_MANAGE_FACTORY = 2; // Above minus taking of cpu time
  private final static int INSTR_LEVEL_NO_THREAD_ID = 1; // Above minus taking thread info

  public MethodAdapter(int api, ClassVisitor cv)
  {
    super(api, cv);
  }

  @Override
  public void visit(int version, int access, String name, String signature, String superName, String[] interfaces)
  {
    cv.visit(version, access, name, signature, superName, interfaces);
    class_name = name.replace('/', '.');
    className = name;
  }

  @Override
  public MethodVisitor visitMethod(int access, String name, String desc, String signature, String[] exceptions)
  {
    MethodVisitor mv2 = cv.visitMethod(access, name, desc, signature, exceptions);

    if(!name.startsWith("<clinit>") && !name.startsWith("<init>"))
    {
      String fqname = class_name + "." + name + "" + desc;
      //mv2 = new CheckMethodAdapter(mv2);

      byte mthdInstrFlag;
      //if(NDHelper.isMethodinWhiteList(fqname))
      if(NDInstrumentManager.isMethodToInstrument(fqname))
```



```
    {
      mthdInstrFlag = NDMethodManager.genMethodFlag(fqname, true);
      mv2 = new MethodProcessor(fqname, name, access, desc, mv2, mthdInstrFlag);
    }
    else if(NDMethodMonitor.isMethodInCurrentMonitoringList(fqname))
    {
      mthdInstrFlag = NDMethodManager.genMethodFlag(fqname, false);
      mv2 = new MethodProcessor(fqname, name, access, desc, mv2, mthdInstrFlag);
    }
   }
   return mv2;
}

@Override
public void visitEnd()
{
  cv.visitEnd();
}

private class MethodProcessor extends LocalVariablesSorter
{
  private String fq_method_name;
  private String methodNameWithDesc;
//Adding lvar10 for wait Time and lvar11 for sync Time.
  private int lvar1, lvar2, lvar4, lvar5, lvar7, lvar9, startTime, cookieObj,
      cookieStr, playReqObj, playResObj ,lvar10 ,lvar11, entrySeq, callStackDepth;

  private int throwableObj = newLocal(Type.getObjectType("throwable"));
  private int methodId = 0;
  private long cpuTime = 0;


  public MethodProcessor(String fqmname, String method_name, int access, String desc, MethodVisitor mv, byte methodFlag)
  {
    super(ASM5, access, desc, mv);
    fq_method_name = fqmname;
    methodNameWithDesc = method_name + desc;
    methodId = NDMethodManager.addMethodAtMethodDiscovery(fq_method_name, methodFlag);
  }

  @Override
  public void visitCode()
  {

    super.visitCode();

    lvar1 = newLocal(Type.getType("java/lang/Thread")); // Thread
    lvar2 = newLocal(Type.LONG_TYPE); // long threadId
    //lvar3 = newLocal(Type.LONG_TYPE); // b_timestamp
    lvar4 = newLocal(Type.LONG_TYPE); // ThreadMXBean
    lvar5 = newLocal(Type.LONG_TYPE); // bCpu
    //adding new fields
    lvar10 = newLocal(Type.LONG_TYPE); // wait time
    lvar11 = newLocal(Type.LONG_TYPE); //Sync Time
    //lvar6 = newLocal(Type.LONG_TYPE); // e_timestamp
    lvar7 = newLocal(Type.LONG_TYPE); // eCpu
    lvar9 = newLocal(Type.LONG_TYPE); // thCpu
    startTime = newLocal(Type.LONG_TYPE); // start time of method executed by app. thread
    entrySeq = newLocal(Type.LONG_TYPE); // for getting sequencenumber at method entry
    callStackDepth = newLocal(Type.LONG_TYPE); // for manage callstack depth at method entry exit
    cookieObj = newLocal(Type.getType("javax/servlet/http/Cookie"));
    cookieStr = newLocal(Type.getType("java/lang/String"));
    playResObj = newLocal(Type.getType("play/mvc/Http$Response"));
    playReqObj = newLocal(Type.getType("play/mvc/Http$Request"));
    //                              lvar10 = newLocal(Type.LONG_TYPE); // trxIdG
```



```
    //keep the timestamp for executed method at start
    Label l99 = new Label();
    mv.visitLabel(l99);
    mv.visitMethodInsn(INVOKESTATIC, "java/lang/System", "currentTimeMillis", "()J");
    mv.visitVarInsn(LSTORE, startTime);

    //System.out.println("transforming method - " + fq_method_name);
    if(NDSys.getInstrLevel() > INSTR_LEVEL_NO_THREAD_ID)
    {

      Label l0 = new Label();
      mv.visitLabel(l0);
      mv.visitMethodInsn(INVOKESTATIC, "java/lang/Thread", "currentThread", "()Ljava/lang/Thread;");
      mv.visitVarInsn(ASTORE, lvar1);

      Label l1 = new Label();
      mv.visitLabel(l1);
      mv.visitVarInsn(ALOAD, lvar1);
      mv.visitMethodInsn(INVOKEVIRTUAL, "java/lang/Thread", "getId", "()J");
      mv.visitVarInsn(LSTORE, lvar2);
    }

    //This is the case where NDSys.getInstrLevel() IS 9 And INSTR_LEVEL_NO_DUMP_REC is 5 so always true. Here getInstrLevel has default value for all that is 9
    if(NDSys.getInstrLevel() > INSTR_LEVEL_NO_DUMP_REC)
    {

      // Method Entry Record Format:
      // 0,FPId,MethodId,CurTimeStampInMS(WallTimeStamp),ThreadId,SeqNum
      //
      // ThreadCpuTime(NanoSec) is stored in a variable and then in exit we can a diff
      // Note: ThreadCpuTime: total CPU time for the current thread in nanoseconds since start of the thread

      Label l13 = new Label();
      mv.visitLabel(l13);
      //                                  mv.visitLdcInsn((useMethodNameInRecords ? fq_method_name : methodId));//Method ID
      mv.visitLdcInsn(methodId);
      //mv.visitInsn(Opcodes.ICONST_0);

      ndAgentFlags = NDClassTypeConfig.isMethodEntry(fq_method_name);// restricting multple time check
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getMethodEntrySeqNum", "(I)J");
      mv.visitVarInsn(LSTORE, entrySeq);
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getCallStackDepth", "()J");
      mv.visitVarInsn(LSTORE, callStackDepth);

      mv.visitLdcInsn(methodId);
      mv.visitVarInsn(LLOAD, lvar2);//Thread ID

      mv.visitVarInsn(LLOAD, startTime);
      /*
       * Dumping Method entry :- That might be a service method or normal method
       */

      if(ndAgentFlags.isServiceEntryPoint && (!ndAgentFlags.genericEntry) && (!ndAgentFlags.jmsCall) && (!ndAgentFlags.playEntryPoint))
      {
        mv.visitVarInsn(ALOAD, 0);
        //mv.visitMethodInsn(INVOKEVIRTUAL, "java/lang/Object", "toString", "()Ljava/lang/String;");
        Label l2 = new Label();
        mv.visitLabel(l2);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "dumpServiceMethodEntryWithPageName", "(IJJLjava/lang/Object;)J");
        mv.visitVarInsn(LSTORE, lvar5);

        //changes done.
        //1. wait time
```



```
        mv.visitLdcInsn(methodId);//passing method Id to validate conditions .
        mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
        Label l3 = new Label();
        mv.visitLabel(l3);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadWaitTimeOnMethodStart", "(IJ)J");
        mv.visitVarInsn(LSTORE, lvar10);

        //2. sync time
        mv.visitLdcInsn(methodId);//passing method Id to validate conditions .
        mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
        Label l4 = new Label();
        mv.visitLabel(l4);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadSyncTimeOnMethodStart", "(IJ)J");
        mv.visitVarInsn(LSTORE, lvar11);

      }

      //For Generic Entry handling
      else if(ndAgentFlags.genericEntry || ndAgentFlags.playEntryPoint)
      {
        mv.visitLdcInsn(fq_method_name);
        Label l2 = new Label();
        mv.visitLabel(l2);
        mv.visitMethodInsn(INVOKESTATIC,  "com/cavisson/ndutils/NDSys",  "dumpServiceMethodEntryWithoutPageName",
"(IJJLjava/lang/String;)J");
        mv.visitVarInsn(LSTORE, lvar5);

        //changes done.
        //1. wait time
        mv.visitLdcInsn(methodId);//passing method Id to validate conditions .

        mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
        Label l3 = new Label();
        mv.visitLabel(l3);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadWaitTimeOnMethodStart", "(IJ)J");
        mv.visitVarInsn(LSTORE, lvar10);

        //2. sync time
        mv.visitLdcInsn(methodId);//passing method Id to validate conditions .

        mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
        Label l4 = new Label();
        mv.visitLabel(l4);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadSyncTimeOnMethodStart", "(IJ)J");
        mv.visitVarInsn(LSTORE, lvar11);

      }

      //commented as previous JMS Entry has been disabled.
      /*else if((isMethodInsideEntryPoint) && (!NDClassTypeConfig.httpCallOut) && (NDClassTypeConfig.jmsCall) &&
(!NDClassTypeConfig.customCallOut) && (!NDClassTypeConfig.genericEntry) && (!NDClassTypeConfig.playEntryPoint) &&
(!NDClassTypeConfig.customLogger) && (!NDClassTypeConfig.hessianCallOut))
      {
        //this we are expect for following method in jms call entry point

//org.springframework.jms.listener.AbstractMessageListenerContainer.invokeListener(Ljavax/jms/Session;Ljavax/jms/Message;)
V|JMSCall
        if(NDSys.getBCIURILogLevel() > 0)
                {
        mv.visitVarInsn(ALOAD, 1);//session object of jms call
        mv.visitVarInsn(ALOAD, 2);//message object of jms call

        Label l2 = new Label();
        mv.visitLabel(l2);
        mv.visitMethodInsn(INVOKESTATIC,  "com/cavisson/ndutils/NDSys",  "dumpJMSCallEntryWithMessage",
"(IJJLjavax/jms/Session;Ljavax/jms/Message;)J");
```

```
        mv.visitVarInsn(LSTORE, lvar5);
        }
                else
                 mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "dumpJMSCallEntry", "(IJJ)V");
       }*/

       //for JMS handling through consumer end need page name
       else if(ndAgentFlags.jmsCall)
       {
//         mv.visitLdcInsn(fq_method_name);

if(fq_method_name.startsWith("org.apache.activemq.ActiveMQMessageConsumer.dispatch(Lorg/apache/activemq/command/MessageDispatch;"))
        {
          mv.visitVarInsn(ALOAD, 1); // it loads the '1st argument' object.
          mv.visitMethodInsn(INVOKEVIRTUAL, "org/apache/activemq/command/MessageDispatch", "getMessage", "()Lorg/apache/activemq/command/Message;");
          mv.visitMethodInsn(INVOKEVIRTUAL, "org/apache/activemq/command/Message", "getDestination", "()Lorg/apache/activemq/command/ActiveMQDestination;");
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "dumpJMSServiceCallEntryWithPageName", "(IJJLjava/lang/Object;)J");
          mv.visitVarInsn(LSTORE, lvar5);

          //changes done.
          //1. wait time
          mv.visitLdcInsn(methodId);//passing method Id to validate conditions .

          mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
          Label l3 = new Label();
          mv.visitLabel(l3);
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadWaitTimeOnMethodStart", "(IJ)J");
          mv.visitVarInsn(LSTORE, lvar10);

          //2. sync time
          mv.visitLdcInsn(methodId);//passing method Id to validate conditions .

          mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
          Label l4 = new Label();
          mv.visitLabel(l4);
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadSyncTimeOnMethodStart", "(IJ)J");
          mv.visitVarInsn(LSTORE, lvar11);

        }
        else
if(fq_method_name.startsWith("com.ibm.mq.jms.MQMessageConsumer$FacadeMessageListener.onMessage(Ljavax/jms/Message;"))
        {
          mv.visitVarInsn(ALOAD, 1);
          mv.visitMethodInsn(INVOKEINTERFACE, "javax/jms/Message", "getJMSDestination", "()Ljavax/jms/Destination;");
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "dumpJMSServiceCallEntryWithPageName", "(IJJLjava/lang/Object;)J");
          mv.visitVarInsn(LSTORE, lvar5);

          //changes done.
          //1. wait time
          mv.visitLdcInsn(methodId);//passing method Id to validate conditions .
          mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
          Label l3 = new Label();
          mv.visitLabel(l3);
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadWaitTimeOnMethodStart", "(IJ)J");
          mv.visitVarInsn(LSTORE, lvar10);

          //2. sync time
          mv.visitLdcInsn(methodId);//passing method Id to validate conditions .
          mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
          Label l4 = new Label();
```



```
        mv.visitLabel(l4);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadSyncTimeOnMethodStart", "(IJ)J");
        mv.visitVarInsn(LSTORE, lvar11);
       }

      /* Label l2 = new Label();
       mv.visitLabel(l2);

       //again loading the variable
       mv.visitLdcInsn(methodId); //method ID
       mv.visitVarInsn(LLOAD, lvar2);//Thread ID
       mv.visitVarInsn(LLOAD, startTime); //start Time
       mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "dumpJMSCallEntryWithMessage", "(IJJ)J");
       mv.visitVarInsn(LSTORE, lvar5);*/
      }
      else
      {
       Label l2 = new Label();
       mv.visitLabel(l2);
       mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "dumpMethodEntry", "(IJJ)J");
       mv.visitVarInsn(LSTORE, lvar5);

       //changes done.
       //1. wait time
       mv.visitLdcInsn(methodId);//passing method Id to validate conditions .

       mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
       Label l3 = new Label();
       mv.visitLabel(l3);
       mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadWaitTimeOnMethodStart", "(IJ)J");
       mv.visitVarInsn(LSTORE, lvar10);

       //2. sync time
       mv.visitLdcInsn(methodId);//passing method Id to validate conditions .

       mv.visitVarInsn(LLOAD, lvar2);//Thread ID ,loading the thread Id to get info of current Thread.
       Label l4 = new Label();
       mv.visitLabel(l4);
       mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getThreadSyncTimeOnMethodStart", "(IJ)J");
       mv.visitVarInsn(LSTORE, lvar11);
      }

      /*
       * Dumping 7, 6 and 9 record
       * removing condition (NDSys.logNonNSFlowpath > 0) as it will always be instrumented.
       */
      /*if((isMethodInsideEntryPoint) && (!NDClassTypeConfig.httpCallOut) && (!NDClassTypeConfig.customCallOut) &&
(!NDClassTypeConfig.genericEntry) && (!NDClassTypeConfig.customLogger)  && (!NDClassTypeConfig.hessianCallOut))
      {
       mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture", "checkIsHeaderDumped", "()V");
      }*/

      //calling generate exception module to generate exception using file.


      if(NDSys.exceptionListFields.containsKey(fq_method_name))
      {
       //calling generateException method to throw exception by loading the method name using file
       mv.visitLdcInsn(fq_method_name);
       mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "generateException", "(Ljava/lang/String;)V");

      }

      if((!ndAgentFlags.jmsCall) && (!ndAgentFlags.genericEntry) && (ndAgentFlags.isServiceEntryPoint) &&
(!ndAgentFlags.jerseyCall) && (!ndAgentFlags.playEntryPoint))
      {
```



```
//          mv.visitVarInsn(ALOAD, 1);
        /*/if(NDClassTypeConfig.entryForWeblogicJSP || NDClassTypeConfig.hessianCallOut)
          mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletRequest");*/
        //setCookieInResponseObject(2);
        setCookieInResponseObject(2, IConstants.METHOD_ENTRY_MODE);
      }

      //calling generate exception module to generate exception using keyword.

      if(NDSys.getExceptionPct() > 0)
      {
       //calling generateException method to throw exception by loading the method name using keyword
       if(fq_method_name.contains(NDSys.getMethodForException()))
       {
        mv.visitLdcInsn(fq_method_name);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "generateException", "(Ljava/lang/String;)V");
       }
      }

      /*if(NDSys.getDelayTimeInMethod() > 0L)
      {
          if(fq_method_name.contains(NDSys.getMethodForDelay()))
          {
                  mv.visitLdcInsn(fq_method_name);
                  mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "putDelayInMethod",
"(Ljava/lang/String;)V");
          }
      }*/

      //Block for checking generating delay in the method
      if(AddDelay.isDelayMethodsAvailable())
      {
       if(AddDelay.putDelayInMethodmap.containsKey(methodId))
       {
        //new local created to store loop end time if cpu hogging is required
        int loopEndTime = newLocal(Type.LONG_TYPE);

        Label label2 = new Label();
        Label label3 = new Label();

        //retrieving delay time in ms
        //mv.visitLdcInsn(methodId);
        //mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/AddDelay", "getDelayTimeInMethod", "(I)J");

        //check if cpu hogging is enabled
        mv.visitLdcInsn(methodId);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/AddDelay", "ifDelayWithCpuHogging", "(I)Z");
        mv.visitJumpInsn(IFEQ, label2);

        //mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/AddDelay", "fromLoopBlock", "()V");

        //retrieving delay time in ms
        mv.visitLdcInsn(methodId);
        mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/AddDelay", "getDelayTimeInMethod", "(I)J");

        //calculating loop end time for cpu hoggging
        mv.visitMethodInsn(INVOKESTATIC, "java/lang/System", "currentTimeMillis", "()J");
        mv.visitInsn(Opcodes.LADD);
        mv.visitVarInsn(LSTORE, loopEndTime);

        //while loop
        Label label4 = new Label();
        mv.visitLabel(label4);
        mv.visitMethodInsn(INVOKESTATIC, "java/lang/System", "currentTimeMillis", "()J");
        mv.visitVarInsn(LLOAD, loopEndTime);
        mv.visitInsn(Opcodes.LCMP);
```



```
        mv.visitJumpInsn(Opcodes.IFLT, label4);

        //after loop jump to skip sleep logic
        mv.visitJumpInsn(Opcodes.GOTO, label3);

        //wait logic
        mv.visitLabel(label2);
        addWaitInByteCode();

        //sleep logic
        //mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/AddDelay", "fromSleepBlock", "()V");
        //mv.visitMethodInsn(INVOKESTATIC, "java/lang/Thread", "sleep", "(J)V", false);

        //end of put delay if else logic
        mv.visitLabel(label3);

      }
     }

    }
   }

   private void setCookieInResponseObjectForPlay()
   {
    mv.visitMethodInsn(INVOKESTATIC, "play/mvc/Http$Context", "current", "()Lplay/mvc/Http$Context;");
    mv.visitMethodInsn(INVOKEVIRTUAL, "play/mvc/Http$Context", "request", "()Lplay/mvc/Http$Request;");
    mv.visitVarInsn(ASTORE, playReqObj);
    mv.visitVarInsn(ALOAD, playReqObj);
    mv.visitMethodInsn(INVOKEINTERFACE, "play/mvc/Http$Request", "uri", "()Ljava/lang/String;");
    // testing

    mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture",
"checkConditionsForCookieSettingInResponse", "(Ljava/lang/String;)Z");
    Label l1 = new Label();
    mv.visitJumpInsn(IFEQ, l1);

    // making String builder and converting to String=========================================
    Label l2 = new Label();
    mv.visitLabel(l2);

    mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getCookieValueForHandledExceptionForPlay",
"()Ljava/lang/String;");

    mv.visitVarInsn(ASTORE, cookieStr);

    // completed ... String s2 = (new
StringBuilder(NDSys.getTrxId())).append(",").append(NDHttpCapture.getResponseStatus(httpservletresponse)).append(",").appe
nd(Server.TestRunIDValue).toString();
    // ================String s2 is made and inserted until here=========================================

    Label l3 = new Label();
    mv.visitLabel(l3);
    mv.visitMethodInsn(INVOKESTATIC, "play/mvc/Http$Context", "current", "()Lplay/mvc/Http$Context;");
    mv.visitMethodInsn(INVOKEVIRTUAL, "play/mvc/Http$Context", "response", "()Lplay/mvc/Http$Response;");
    mv.visitVarInsn(ASTORE, playResObj);
    mv.visitVarInsn(ALOAD, playResObj);
    // Making new Cookie object - Cookie cookie = new Cookie(NDSys.getFPHeaderName(), s2);
    mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getFPHeaderName", "()Ljava/lang/String;");
    mv.visitVarInsn(ALOAD, cookieStr);

    mv.visitMethodInsn(INVOKEVIRTUAL, "play/mvc/Http$Response", "setCookie",
"(Ljava/lang/String;Ljava/lang/String;)V");
    mv.visitLabel(l1);

   }
```



```
  private void captureResponseObjectForPlay()
  {
   Label l1 = new Label();
   mv.visitLabel(l1);

   mv.visitMethodInsn(INVOKESTATIC, "play/mvc/Http$Context", "current", "()Lplay/mvc/Http$Context;");
   mv.visitMethodInsn(INVOKEVIRTUAL, "play/mvc/Http$Context", "response", "()Lplay/mvc/Http$Response;");
   mv.visitMethodInsn(INVOKEVIRTUAL, "play/mvc/Http$Response", "getHeaders", "()Ljava/util/Map;");
   mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture", "dumpHttpResponseInNormalCaseForPlay",
"(Ljava/util/Map;)V");

  }

  @Override
  public void visitInsn(int opcode)
  {

   /*
    * Adding the code at the end of the method:
    */
   if(NDSys.getInstrLevel() > INSTR_LEVEL_NO_EXIT_METHOD)
   {
    // Method exit record format
    // 1,FPId,MethodId,CurTimeStampInMS(WallTimeStamp),ThreadId,SeqNum,CpuTimeTakenByMethod(NanoSec)
    if((opcode >= IRETURN && opcode <= RETURN) || opcode == ATHROW)
    {

     // capturing HttpServletResponse object, and passing it to NDHttpResponseCapture.dumpHttpResponse() method.
     if(ndAgentFlags.playEntryPoint)
     {
      if(opcode == ARETURN)
      {
       Label l19 = new Label();
       mv.visitLabel(l19);
       mv.visitInsn(DUP);
       mv.visitTypeInsn(CHECKCAST, "play/mvc/Result");
       if(NDClassTypeConfig.isPlayResultInterface)
         mv.visitMethodInsn(INVOKEINTERFACE, "play/mvc/Result", "status", "()I");
       else
        mv.visitMethodInsn(INVOKEVIRTUAL, "play/mvc/Result", "status", "()I");
       mv.visitMethodInsn(INVOKESTATIC,                                    "com/cavisson/ndutils/NDHttpCapture",
"dumpHttpResponseStatusCodeForPlay", "(I)V");
      }

      captureResponseObjectForPlay();
     }
     else if((!ndAgentFlags.jerseyCall) && (ndAgentFlags.isServiceEntryPoint) && (!ndAgentFlags.genericEntry) &&
(!ndAgentFlags.jmsCall) && (!ndAgentFlags.playEntryPoint))
     {
      captureResponseObject(2);
     }
     else if((ndAgentFlags.jerseyCall) && (ndAgentFlags.isServiceEntryPoint) && (!ndAgentFlags.playEntryPoint))
     {
      captureResponseObject(4);
     }

     if(ndAgentFlags.isServiceEntryPoint)
     {

      if(!ndAgentFlags.jmsCall && !ndAgentFlags.jerseyCall && !ndAgentFlags.genericEntry)
      /*{ JMS message capturing has been disabled because there multiple JMS providers now and for each provider capturing
logic need to be redefined.

       //mv.visitVarInsn(ALOAD, 1);//session object of jms call
       //mv.visitVarInsn(ALOAD, 2);//message object of jms call
```



```
          //mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDJMSCapture", "dumpJMSRequest",
"(Ljavax/jms/Session;Ljavax/jms/Message;)V");
         //          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "check", "()V");
          }
          else*/
          {
           mv.visitVarInsn(LLOAD, startTime);
           // loading httpServletRequest, which is the first parameter and calling getRequestURI() method.
           mv.visitVarInsn(ALOAD, 1);
           if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut)
            mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletRequest");
           mv.visitMethodInsn(INVOKEINTERFACE, "javax/servlet/http/HttpServletRequest", "getRequestURI",
"()Ljava/lang/String;");

           // loading httpServletRequest, which is the first parameter and calling getQueryString() method.
           mv.visitVarInsn(ALOAD, 1);
           if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut)
            mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletRequest");
           mv.visitMethodInsn(INVOKEINTERFACE, "javax/servlet/http/HttpServletRequest", "getQueryString",
"()Ljava/lang/String;");

           // loading httpServletRequest object, which is the first parameter.
           mv.visitVarInsn(ALOAD, 1);
           if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut)
            mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletRequest");

           // passing above 3 parameters (URI, query string and request object) to NDHttpCapture.dumpHttpRequest() method.
           mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture", "dumpHttpRequest",
"(JLjava/lang/String;Ljava/lang/String;Ljava/lang/Object;)V");
//           setCookieInResponseObject(2);
          setCookieInResponseObject(2, IConstants.METHOD_EXIT_MODE);
          }

        }

        else if((ndAgentFlags.jerseyCall) && (ndAgentFlags.isServiceEntryPoint))
        {
          mv.visitVarInsn(LLOAD, startTime);
          // loading httpServletRequest, which is the third parameter and calling getRequestURI() method.
          mv.visitVarInsn(ALOAD, 3);
          mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletRequest");
          mv.visitMethodInsn(INVOKEINTERFACE, "javax/servlet/http/HttpServletRequest", "getRequestURI",
"()Ljava/lang/String;");

          // loading httpServletRequest, which is the third parameter and calling getQueryString() method.
          mv.visitVarInsn(ALOAD, 3);
          mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletRequest");
          mv.visitMethodInsn(INVOKEINTERFACE, "javax/servlet/http/HttpServletRequest", "getQueryString",
"()Ljava/lang/String;");

          // loading httpServletRequest object, which is the third parameter.
          mv.visitVarInsn(ALOAD, 3);
          mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletRequest");

          // passing above 3 parameters (URI, query string and request object) to NDHttpCapture.dumpHttpRequest() method.
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture", "dumpHttpRequest",
"(JLjava/lang/String;Ljava/lang/String;Ljava/lang/Object;)V");
          setCookieInResponseObject(4,IConstants.METHOD_EXIT_MODE);
        }
        else if(ndAgentFlags.playEntryPoint)
        {
          mv.visitVarInsn(LLOAD, startTime);
          mv.visitMethodInsn(INVOKESTATIC, "play/mvc/Http$Context", "current", "()Lplay/mvc/Http$Context;");
          mv.visitMethodInsn(INVOKEVIRTUAL, "play/mvc/Http$Context", "request", "()Lplay/mvc/Http$Request;");
          mv.visitVarInsn(ASTORE, playReqObj);
          // uri
```

```
          mv.visitVarInsn(ALOAD, playReqObj);
          mv.visitMethodInsn(INVOKEINTERFACE, "play/mvc/Http$Request", "uri", "()Ljava/lang/String;");
          // query string
          mv.visitVarInsn(ALOAD, playReqObj);

          mv.visitMethodInsn(INVOKEINTERFACE, "play/mvc/Http$Request", "queryString", "()Ljava/util/Map;");
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "processPlayMap",
"(Ljava/util/Map;)Ljava/lang/String;");

          mv.visitVarInsn(ALOAD, playReqObj);
          mv.visitMethodInsn(INVOKEINTERFACE, "play/mvc/Http$Request", "headers", "()Ljava/util/Map;");

          mv.visitVarInsn(ALOAD, playReqObj);
          mv.visitMethodInsn(INVOKEINTERFACE, "play/mvc/Http$Request", "method", "()Ljava/lang/String;");

          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture", "dumpHttpRequestForPlay",
"(JLjava/lang/String;Ljava/lang/String;Ljava/util/Map;Ljava/lang/String;)V");
          // setCookieInResponseObjectForPlay();

        }

        //mv.visitLdcInsn((useMethodNameInRecords ? fq_method_name : methodId));
        mv.visitLdcInsn(methodId);
        //mv.visitInsn(Opcodes.ICONST_0);
        mv.visitVarInsn(LLOAD, lvar2);

        //getting cpu time of the method
        mv.visitVarInsn(LLOAD, lvar5);

        mv.visitVarInsn(LLOAD, startTime);

        //getting the wait time
        mv.visitVarInsn(LLOAD, lvar10);

        //getting the sync time
        mv.visitVarInsn(LLOAD, lvar11);
       //getting the callstackdepth
        mv.visitVarInsn(LLOAD, callStackDepth);
        mv.visitVarInsn(LLOAD, entrySeq);

       // isMethodInsideEntryPoint = NDClassTypeConfig.isMethodEntry(fq_method_name);// restricting multple time check
        // The check NDSys.getBCIURILogLevel() > 0 is removed as we need to dump service method exit record irrespective of
the value of bciUriLogLevel.
        if((ndAgentFlags.isServiceEntryPoint) || (ndAgentFlags.genericEntry) || (ndAgentFlags.playEntryPoint))
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "dumpServiceMethodExit", "(IJJJJJJJ)V");
        else
          mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "dumpMethodExit", "(IJJJJJJJ)V");

        mv.visitInsn(LCONST_0);
        mv.visitVarInsn(LSTORE, startTime);

        //System.out.println("transformed method - " + fq_method_name);
       }
     }

    super.visitInsn(opcode); // Must be done
   }


   //method used to use object.wait() mechanism to generate delay
   private void addWaitInByteCode()
   {
    int lockObj = newLocal(Type.getType("java/lang/String;"));
    Label l15 = new Label();
    Label l16 = new Label();
    Label l17 = new Label();
```

```
    mv.visitTryCatchBlock(l15, l16, l17, null);
    Label l18 = new Label();
    mv.visitTryCatchBlock(l17, l18, l17, null);
    Label l19 = new Label();
    Label l20 = new Label();
    mv.visitTryCatchBlock(l19, l20, l20, "java/lang/Exception");
    mv.visitLabel(l19);
    //                                  mv.visitLineNumber(30, l19);
    //                                  mv.visitVarInsn(ALOAD, 1);
    mv.visitLdcInsn(fq_method_name);
    mv.visitInsn(DUP);
    mv.visitVarInsn(ASTORE, lockObj);
    mv.visitInsn(Opcodes.MONITORENTER);
    mv.visitLabel(l15);
    //                                  mv.visitLineNumber(31, l15);
    //                                  mv.visitVarInsn(ALOAD, 1);
    mv.visitLdcInsn(fq_method_name);
    mv.visitLdcInsn(methodId);
    mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/AddDelay", "getDelayTimeInMethod", "(I)J");
    //                                  mv.visitLdcInsn(new Long(10000L));
    mv.visitMethodInsn(INVOKEVIRTUAL, "java/lang/Object", "wait", "(J)V");
    Label l21 = new Label();
    mv.visitLabel(l21);
    //                                  mv.visitLineNumber(30, l21);
    mv.visitVarInsn(ALOAD, lockObj);
    mv.visitInsn(Opcodes.MONITOREXIT);
    mv.visitLabel(l16);
    Label l22 = new Label();
    mv.visitJumpInsn(Opcodes.GOTO, l22);
    mv.visitLabel(l17);
    //                                  mv.visitFrame(Opcodes.F_FULL, 3, new Object[] {"pacUtil/WaitTest",
"java/lang/String", "java/lang/String"}, 1, new Object[] {"java/lang/Throwable"});
    mv.visitVarInsn(ALOAD, lockObj);
    mv.visitInsn(Opcodes.MONITOREXIT);
    mv.visitLabel(l18);
    mv.visitInsn(ATHROW);
    mv.visitLabel(l20);
    //                                  mv.visitLineNumber(34, l20);
    //                                  mv.visitFrame(Opcodes.F_FULL, 2, new Object[] {"pacUtil/WaitTest",
"java/lang/String"}, 1, new Object[] {"java/lang/Exception"});
    mv.visitVarInsn(ASTORE, lockObj);
    Label l23 = new Label();
    mv.visitLabel(l23);
    //                                  mv.visitLineNumber(36, l23);
    //                                  mv.visitVarInsn(ALOAD, 2);
    //                                  mv.visitMethodInsn(INVOKEVIRTUAL, "java/lang/Exception", "printStackTrace",
"()V", false);
    mv.visitLabel(l22);
    //                                  mv.visitLineNumber(38, l22);
    //                                  mv.visitFrame(Opcodes.F_SAME, 0, null, 0, null);
    //mv.visitInsn(RETURN);
   }

   private void setCookieInResponseObject(int parameterIndex, int setCookieMode)
   {
          Label l1;

      /*
      * if(NDHttpCapture.checkConditionsForCookieSettingInResponse(String uri))
         {
         String                    s2                    =                    (new
StringBuilder(NDSys.getTrxId())).append(",").append(NDHttpCapture.getResponseStatus(httpservletresponse)).append(",").appe
nd(Server.TestRunIDValue).toString();
         Cookie cookie = new Cookie(NDSys.getFPHeaderName(), s2);
         cookie.setPath("/");
         httpservletresponse.addCookie(cookie);
```

```
          NDHttpCapture.dumpHttpResponse(httpservletresponse);
          }
      */

      // below lines will insert --  if(NDHttpCapture.checkConditionsForCookieSettingInResponse(uri))
      mv.visitIntInsn(Opcodes.BIPUSH, setCookieMode);
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture",
"checkConditionsForCookieSettingInResponse", "(I)Z");
      l1 = new Label();
      mv.visitJumpInsn(IFEQ, l1);

      // making String builder and converting to String=======================================
      Label l2 = new Label();
      mv.visitLabel(l2);

      mv.visitVarInsn(ALOAD, parameterIndex);
      if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut)
        mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletResponse");

      mv.visitIntInsn(Opcodes.BIPUSH, setCookieMode);
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getCookieValueWithSessionId",
"(Ljava/lang/Object;I)Ljava/lang/String;");

//       mv.visitVarInsn(ALOAD, 1);
//       if(NDClassTypeConfig.entryForWeblogicJSP || NDClassTypeConfig.hessianCallOut)
//         mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletRequest");
////                      mv.visitMethodInsn(INVOKEINTERFACE, "javax/servlet/http/HttpServletRequest", "getCookies",
"()[Ljavax/servlet/http/Cookie;");
      //            mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getCookieValueForHandledException",
"(Ljava/lang/Object;[Ljava/lang/Object;)Ljava/lang/String;");

      //            mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getCookieValueForHandledException",
"(Ljava/lang/Object;)Ljava/lang/String;");

      mv.visitVarInsn(ASTORE, cookieStr);

      //completed ... String s2 = (new
StringBuilder(NDSys.getTrxId())).append(",").append(NDHttpCapture.getResponseStatus(httpservletresponse)).append(",").appe
nd(Server.TestRunIDValue).toString();
      //================String s2 is made and inserted until here=======================================

      Label l3 = new Label();
      mv.visitLabel(l3);

      // Making new Cookie object - Cookie cookie = new Cookie(NDSys.getFPHeaderName(), s2);
      mv.visitTypeInsn(NEW, "javax/servlet/http/Cookie");
      mv.visitInsn(DUP);
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getFPHeaderName", "()Ljava/lang/String;");
      mv.visitVarInsn(ALOAD, cookieStr); // loading String s2
      mv.visitMethodInsn(INVOKESPECIAL, "javax/servlet/http/Cookie", "<init>", "(Ljava/lang/String;Ljava/lang/String;)V");
      mv.visitVarInsn(ASTORE, cookieObj);

      // Setting path "/" to cookie object - cookie.setPath("/");
      Label l4 = new Label();
      mv.visitLabel(l4);
      mv.visitVarInsn(ALOAD, cookieObj);
      mv.visitLdcInsn("/");
      mv.visitMethodInsn(INVOKEVIRTUAL, "javax/servlet/http/Cookie", "setPath", "(Ljava/lang/String;)V");

      //setting domain
      Label l5 = new Label();
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getCookieDomainName", "()Ljava/lang/String;");
      //if domain name is null then don't set it
      mv.visitJumpInsn(Opcodes.IFNULL, l5);
      mv.visitVarInsn(ALOAD, cookieObj);
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getCookieDomainName", "()Ljava/lang/String;");
```



```
      mv.visitMethodInsn(INVOKEVIRTUAL, "javax/servlet/http/Cookie", "setDomain", "(Ljava/lang/String;)V");
      mv.visitLabel(l5);

      // Setting cookie object to response object - httpservletresponse.addCookie(cookie);
      mv.visitVarInsn(ALOAD, parameterIndex);
      if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut)
        mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletResponse");

      mv.visitVarInsn(ALOAD, cookieObj);
      mv.visitMethodInsn(INVOKEINTERFACE, "javax/servlet/http/HttpServletResponse", "addCookie",
"(Ljavax/servlet/http/Cookie;)V");

      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "isXheaderSet", "()Z");
      mv.visitJumpInsn(IFEQ, l1);

      mv.visitVarInsn(ALOAD, parameterIndex);
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getXHeaderName", "()Ljava/lang/String;");

      mv.visitVarInsn(ALOAD, parameterIndex);
      if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut)
        mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletResponse");

      mv.visitIntInsn(Opcodes.BIPUSH, setCookieMode);
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDSys", "getXHeaderValue",
"(Ljava/lang/Object;I)Ljava/lang/String;");
      mv.visitMethodInsn(INVOKEINTERFACE, "javax/servlet/http/HttpServletResponse", "setHeader",
"(Ljava/lang/String;Ljava/lang/String;)V");

      mv.visitLabel(l1); // if condition ends here.
    }

    private void captureResponseObject(int parameterIndex)
    {
      Label l1 = new Label();
      mv.visitLabel(l1);

      mv.visitVarInsn(ALOAD, parameterIndex);
      if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut || ndAgentFlags.jerseyCall)
        mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletResponse");

      // passing Response object to NDHttpResponseCapture.dumpHttpResponse() method.
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture", "dumpHttpResponseInNormalCase",
"(Ljava/lang/Object;)V");
    }

    /*private void captureResponseObject(int parameterIndex)
    {
      Label l1 = new Label();
      mv.visitLabel(l1);
      //capture status code from http response object using ASM instead of reflection

      mv.visitVarInsn(ALOAD, parameterIndex);
      if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut)
        mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletResponse");

      mv.visitMethodInsn(INVOKEINTERFACE, "javax/servlet/http/HttpServletResponse", "getStatus", "()I");
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/flowpath/NDFlowpathGenerator", "setTloclStatusCode",
"(I)V");

      //passing the response object to get headers
      mv.visitVarInsn(ALOAD, parameterIndex);
      if(ndAgentFlags.entryForWeblogicJSP || ndAgentFlags.hessianCallOut || ndAgentFlags.jerseyCall)
        mv.visitTypeInsn(CHECKCAST, "javax/servlet/http/HttpServletResponse");
      // passing Response object to NDHttpResponseCapture.dumpHttpResponse() method.
      mv.visitMethodInsn(INVOKESTATIC, "com/cavisson/ndutils/NDHttpCapture", "dumpHttpResponseInNormalCase",
"(Ljava/lang/Object;)V");
```

```
   }   */


   @Override
   public void visitMaxs(int maxStack, int maxLocals)
   {
    // Increases the stack size to accommodate local variables.
    //System.out.println("visitMaxs start for - " + fq_method_name);
    //mv.visitMaxs(0, 0);
    //System.out.println("visitMaxs end for - " + fq_method_name);
    super.visitMaxs(maxStack + 40, maxLocals + 10);
    //System.out.println("maxStack: " + maxStack + " maxLocals: " + maxLocals);
   }
  }
}
```

4. EXPERIMENT RESULT

4.1 After (Adding functionality) Instrumentation Java bytecode-

Let's suppose we want to measure the time spent in each class of a program. We need to add a static timer field in each class, and we need to add the execution time of each method of this class to this timer field. In other words we want to transform a class such as C:

public class C {

public void m() throws Exception {

Thread.sleep(100);

}

}

into this:

public class C {

public static long timer;

public void m() throws Exception {

timer -= System.currentTimeMillis();

Thread.sleep(100);

timer += System.currentTimeMillis();

}

}

In order to have an idea of how this can be implemented in ASM, we can compile these two classes and compare the output of TraceClassVisitor on these two versions (either with the default Textifier backend, or with an ASMifier backend). With the default backend we get the following differences

(in bold):

GETSTATIC C.timer : J

INVOKESTATIC java/lang/System.currentTimeMillis()J

LSUB

PUTSTATIC C.timer : J

LDC 100

INVOKESTATIC java/lang/Thread.sleep(J)V

GETSTATIC C.timer : J

INVOKESTATIC java/lang/System.currentTimeMillis()J

LADD

PUTSTATIC C.timer : J

RETURN

MAXSTACK = 4

MAXLOCALS = 1

We see that we must add four instructions at the beginning of the method,

and four other instructions before the return instruction. We also need to

update the maximum operand stack size. The beginning of the method's code

is visited with the visitCode method. We can therefore add the first four

instructions by overriding this method in our method adapter:

```
public void visitCode() {
mv.visitCode();
mv.visitFieldInsn(GETSTATIC, owner, "timer", "J");
mv.visitMethodInsn(INVOKESTATIC, "java/lang/System",
"currentTimeMillis", "()J");
mv.visitInsn(LSUB);
mv.visitFieldInsn(PUTSTATIC, owner, "timer", "J");
}
```

where owner must be set to the name of the class that is being transformed. We must now add four other instructions before any RETURN, but also before any xRETURN or before ATHROW, which are all the instructions that terminate the method's execution. These instructions do not have any argument, and are therefore visited in the visitInsn method. We can then override this

method in order to add our instructions:

```
public void visitInsn(int opcode) {
if ((opcode >= IRETURN && opcode <= RETURN) || opcode == ATHROW) {
mv.visitFieldInsn(GETSTATIC, owner, "timer", "J");
mv.visitMethodInsn(INVOKESTATIC, "java/lang/System",
"currentTimeMillis", "()J");
mv.visitInsn(LADD);
mv.visitFieldInsn(PUTSTATIC, owner, "timer", "J");
}
mv.visitInsn(opcode);
}
```

Finally we must update the maximum operand stack size. The instructions that we add push two long values, and therefore require four slots on the operand stack. At the beginning of the method the operand stack is initially empty, so we know that the four instructions added at the beginning require a stack of size 4. We also know that our inserted code leaves the stack state

unchanged (because it pops as many values as it pushes). As a consequence,if the original code requires a stack of size s, the maximum stack size needed

by the transformed method is max(4, s). Unfortunately we also add four instructions before the return instructions, and here we do not know the size of the operand stack just before these instructions. We just know that it is less than or equal to s. As a consequence we can just say that the code added before the return instructions may require an operand stack of size up to s+ 4.

This worst case scenario rarely happens in practice: with common compilers the operand stack before a RETURN contains only the return value, i.e. it has a size of 0, 1 or 2 at most. But if we want to handle all possible cases, we need to use the worst case scenario2 We must then override the visitMaxs

method as follows:

```
public void visitMaxs(int maxStack, int maxLocals) {
```



```
mv.visitMaxs(maxStack + 4, maxLocals);
}
```

Of course it is possible to not bother about the maximum stack size and rely on the COMPUTE_MAXS option that, in addition, will compute the optimal value and not a worst case value. But for such simple transformations it does not cost much effort to update maxStack manually.hopefully it is not necessary to give the optimal operand stack size. Giving any value greater than or equal to this optimal value is possible, although it may waste memory on

the thread's execution stack.

Now an interesting question is: what about stack map frames? The original

code did not contain any frame, nor the transformed one, but is this due to the

specific code we used as example? are there some situations where frames must

be updated? The answer is no because

1) the inserted code leaves the operand stack unchanged,

2) the inserted code does not contain jump instructions and

3) the jump instructions – or, more formally, the control flow graph – of the

original code is not modified. This means that the original frames do not change, and since no new frames must be stored for the inserted code, the compressed original frames do not change either.

We can now put all the elements together in associated ClassVisitor and

MethodVisitor subclasses:

```
public class AddTimerAdapter extends ClassVisitor {
private String owner;
private boolean isInterface;
public AddTimerAdapter(ClassVisitor cv) {
super(ASM4, cv);
}
@Override public void visit(int version, int access, String name,
String signature, String superName, String[] interfaces) {
cv.visit(version, access, name, signature, superName, interfaces);
owner = name;
isInterface = (access & ACC_INTERFACE) != 0;
}
@Override public MethodVisitor visitMethod(int access, String name,
String desc, String signature, String[] exceptions) {
MethodVisitor mv = cv.visitMethod(access, name, desc, signature,
exceptions);
if (!isInterface && mv != null && !name.equals("<init>")) {
mv = new AddTimerMethodAdapter(mv);
}
return mv;
}
@Override public void visitEnd() {
if (!isInterface) {
FieldVisitor fv = cv.visitField(ACC_PUBLIC + ACC_STATIC, "timer",
"J", null, null);
if (fv != null) {
fv.visitEnd();
```



```
}
}
cv.visitEnd();
}
class AddTimerMethodAdapter extends MethodVisitor {
public AddTimerMethodAdapter(MethodVisitor mv) {
super(ASM4, mv);
}
@Override public void visitCode() {
mv.visitCode();
mv.visitFieldInsn(GETSTATIC, owner, "timer", "J");
mv.visitMethodInsn(INVOKESTATIC, "java/lang/System",
"currentTimeMillis", "()J");
mv.visitInsn(LSUB);
mv.visitFieldInsn(PUTSTATIC, owner, "timer", "J");
}
@Override public void visitInsn(int opcode) {
if ((opcode >= IRETURN && opcode <= RETURN) || opcode == ATHROW) {
mv.visitFieldInsn(GETSTATIC, owner, "timer", "J");
mv.visitMethodInsn(INVOKESTATIC, "java/lang/System",
"currentTimeMillis", "()J");
mv.visitInsn(LADD);
mv.visitFieldInsn(PUTSTATIC, owner, "timer", "J");
}
mv.visitInsn(opcode);
}
@Override public void visitMaxs(int maxStack, int maxLocals) {
mv.visitMaxs(maxStack + 4, maxLocals);
}
}
}
```

The class adapter is used to instantiate the method adapter (except for constructors), but also to add the timer field and to store the name of the class that is being transformed in a field that can be accessed from the method adapter.

Result output-

```
99,NewConnection,FPGVersion:1.0
11,133267436535,All
Meta record-
5,Fully Qualified method name,method Id
5,javax.servlet.http.HttpServlet.service(Ljavax/servlet/http/HttpServletRequest;Ljavax/servlet/http/HttpServletResponse;)V,1
5,weblogic.wsee.persistence.LogicalStore$1.run()V,2
5,com.mysql.jdbc.NonRegisteringDriver$1.run()V,3
```


5,weblogic.platform.GCMonitorThread.run()V,4

5,weblogic.servlet.logging.FormatStringBuffer.getBytes()[B,5

5,weblogic.kernel.ExecuteThread.run()V,6

5,com.octetstring.vde.DoSManager.run()V,7

5,com.octetstring.vde.frontend.LDAP.run()V,8

5,com.octetstring.vde.backend.standard.TransactionProcessor.run()V,9

5,com.octetstring.vde.util.TimedActivityThread.run()V,10

5,com.octetstring.vde.replication.Replication.run()V,11

5,com.octetstring.vde.LDAPServer.run()V,12

5,weblogic.t3.srvr.ServerStartupTimer.run()V,13

5,weblogic.timers.internal.TimerThread$Thread.run()V,14

5,weblogic.work.ExecuteThread.run()V,15

5,weblogic.t3.srvr.T3Srvr$2.run()V,16

5,com.cavisson.ndmain.NDMainHelper$1.run()V,17

5,java.util.concurrent.ThreadPoolExecutor.execute(Ljava/lang/Runnable;)V,18

5,java.util.concurrent.ThreadPoolExecutor.beforeExecute(Ljava/lang/Thread;Ljava/lang/Runnable;)V,19

5,java.util.concurrent.ThreadPoolExecutor.afterExecute(Ljava/lang/Runnable;Ljava/lang/Throwable;)V,20

5,java.util.logging.LogManager$Cleaner.run()V,21

5,java.util.TimerThread.run()V,22

5,java.lang.ref.Finalizer$FinalizerThread.run()V,23

5,java.lang.ref.Reference$ReferenceHandler.run()V,24

5,java.net.URL.openConnection()Ljava/net/URLConnection;,25

5,java.lang.Thread.run()V,26

5,java.lang.Thread.start()V,27

5,java.lang.Throwable.printStackTrace()V,28

5,com.cavisson.nsecom.main()V,30

Business Transaction-

7,btid,business Transaction

7,112,NsecomSearchProduct

7,13,NsecomShippingAddress

7,23,NsecomCheckOut

7,1,Others

7,12,NsecomAddToBag

7,9,NsecomHome

7,5,Callout

7,15,NsecomProductPage

2,46119312412147456 29,133267792,30,30.1,1,/console/console.portal?_nfpb=true,,4611931241214745629,0,0,[ACTIVE] ExecuteThread: '0' for queue: 'weblogic.kernel.Default (self-tuning)',1,-1

4,4611931241214745629,0,12,0,9991,1,30_0_0_1_30_0_0_1_30_0_0_1_25_0_35__25_1_0_0___25_0_576__25_0_576__25_1_0_0___25_1_0_0___25_0_810__25_1_0_0___25_0_915__25_1_0_0___25_0_1619__25_1_0_0___25_0_1619__25_1_0_0___25_0_6212__25_1_0_0___25_0_6231__25_1_0_0___25_0_6250__25_1_0_0___25_0_6251__25_1_0_0___25_0_6252__25_1_0_0___25_0_6300__25_1_0_0___25_0_6309__25_1_0_0___25_0_6310__25_1_0_0___25_0_6311__25_1_0_0___25_0_6312__25_1_0_0___25_0_6322__25_1_0_0___25_0_6324__25_1_0_0___25_0_6324__25_1_0_0___25_0_6325__25_1_0_0___25_0_6326__25_1_0_0___25_0_6410__25_1_0_0___25_0_6412__25_1_0_0___25_0_6413__25_1_0_0___25_0_6414__25_1_0_0___25_0_6415__25_1_0_0___25_0_6415__25_1_0_0___25_0_6417__25_1_0_0___25_0_6418__25_1_0_0___25_0_8199__25_0

_8199__25_1_0_0___25_1_0_0___25_0_8412__25_1_0_0___29_0_9990__29_1_0_0___30_1_9991_0___30_1_9991_0___30_1_9991_0___

Total time taken by method - com.cavisson.nsecom.main()V

Method id -30,

Time Taken= 9991milisecond .

5. CONCLUSION

Bytecode instrumentation technique make generic to application monitoring .with the help of bytecode instrumentation it is easy to monitor taken time by application methods, without modifying application methods.

6. REFERENCES

[1] Apache Ant. http://ant.apache.org/.
[2] ASM Java bytecode manipulation framework. http://asm.objectweb.org/.
[3] AspectJ. http://aspectj.org/.
[4] AspectWerkz AOP framework. http://aspectwerkz.codehaus.org/.
[5] BCEL: The Byte Code Engineering Library. http://jakarta.apache.org/bcel/.
[6] Cobertura code coverage tool. http://cobertura.sourceforge.net/.
[7] JAdvise AOP framework. http://crazybob.org/downloads/jadvise-javadoc/.
[8] Dynamic java proxy classes http://java.sun.com/j2se/1.3/docs/guide/reflection/proxy.html.
[9] Java 1.5.0 API documentation. http://java.sun.com/j2se/1.5.0/docs/api/.

