Java字节码技术

程显峰@OneAPM

我

- 伪技术人员 @OneAPM
- chengxianfeng@oneapm.com

提纲

- 什么是Bytecode
- Bytecode有哪些用处
- JVM如何运行Bytecode
- 几个有趣的示例

Bytecode是什么

- Java程序的归宿，但从规范上讲和Java已没有任何关系了
- JVM能够解释执行（JIT也有编译）
- JVM上的汇编语言

Bytecode的一些特点

- 标准JVM使用的堆栈（区别于寄存器）
- 一个字节的指令
- 理论上256个指令（已经用了200+）
- 近二十年，貌似只增加了一个指令
- 有类型的，虽然有点残

Bytecode的用途

- 静态检查
- 调试/热切换/诊断工具
- 在JVM上的新语言
- AOP，ORM
- Mock, 尤其是Fault Injection

chronon
DVR for Java

Clojure

Plumbr

byteman

ONE APM

Groovy

mockito

AppDynamics

New Relic.

Scala

UNIVERSITY OF MARYLAND 18 56

FindBugs™
because it's easy

HIBERNATE

dyn js

JRebel

为什么折腾字节码

- 语言无关
- 执行效率高
- 不用修改源代码
- 增加语言的特性

肉眼看懂Bytecode

- 诊断性能问题
- 逆向工程
- 安全审计
- 调试遗留代码
- 给FindBugs贡献个插件

JVM如何執行Bytecode

```java
class Add{
    int add(int a, int b){
        return a + b;
    }
}
```

```
[mars@r2d2:~/demo/java-bytecode]
% javac Add.java
[mars@r2d2:~/demo/java-bytecode]
% javap -c Add
```

```
Add();
    Code:
       0: aload_0
       1: invokespecial #1                  // Method java/lang/Object."<init>":()V
       4: return

int add(int, int);
  Code:
     0: iload_1
     1: iload_2
     2: iadd
     3: ireturn
```

```java
class Add{
    public Add(){
        super();
    }
    int add(int a,
int b){
        return a + b;
    }
}
```

```
int add(int, int);
  Code:
    0: iload_1
    1: iload_2
    2: iadd
    3: ireturn
```

```java
class Add{
    int add(int a, int b){
        return a + b;
    }
}
```

local vars

0	this
1	<a>
2	<b>
3	
4	

stack

0	<a+b>
1	
2	
3	
4	

constants

#1	
#2	
#3	
#4	
#5	

TYPE OPERATION

- `<TYPE> ::= b, s, c, i, l, f, d, a`
- 常量 (`ldc, iconst_1`)
- 本地变量和堆栈互操作 (`load/store`)
- 数组操作 (`aload, astore`)
- 算数运算 (`add, sub, mul, div`)
- 逻辑和位运算 (`iand, ixor`)
- 比较和分支 (`cmpl, ifeq, jsr`)
- 转换 (`l2d, i2l`)

```java
import java.util.*;

class Main{
    public static void main(String[] args){
        (new ArrayList<String>()).add("Hello");
    }
}
```

```
  public static void main(java.lang.String[]);
    Code:
       0: new           #2                  // class java/util/ArrayList
       3: dup
       4: invokespecial #3                  // Method java/util/
ArrayList."<init>":()V
       7: ldc           #4                  // String Hello
       9: invokevirtual #5                  // Method java/util/ArrayList.add:
(Ljava/lang/Object;)Z
      12: pop
      13: return
```

```
  public static void main(java.lang.String[]);
    Code:
       0: new           #2                  // class java/util/ArrayList
       3: dup
       4: invokespecial #3                  // Method java/util/
ArrayList."<init>":()V
       7: ldc           #4                  // String Hello
       9: invokevirtual #5                  // Method java/util/ArrayList.add:
(Ljava/lang/Object;)Z
      12: pop
      13: return
```

local vars

0	ArrayList
1	
2	
3	
4	

stack

0	<ArrayList>
1	
2	
3	
4	

constants

#1	java/lang/Object."<init>":()V
#2	java/util/ArrayList
#3	java/util/ArrayList."<init>":()V
#4	Hello
#5	java/util/ArrayList.add:(Ljava/lang/Object;)Z

描述符

- Ljava/util/List;
- ([Ljava/lang/String;)V

调用

- invokespecial
  - 初始化，私有，父类
- invokeinterface
- invokestatic
- invokevirtual (通常)
- invokedynamic (大坑，勿入)

```java
import java.util.*;

public class Items{
    private List<Integer> ids = new ArrayList<Integer>();
    {
            ids.add(1);
            ids.add(100);
            ids.add(100000);

    }

    public int getId(int i){
        return ids.get(i);
    }
}
```

```
 0: aload_0
 1: getfield      #4
 4: aload_0
 5: invokestatic  #5
 8: dup
 9: invokespecial #6
12: putfield      #4
15: return
```

```
 0: aload_0
 1: invokespecial #1
 4: aload_0
 5: new           #2
 8: dup
 9: invokespecial #3
12: putfield      #4
15: pop
```

```
// Fields:Ljava/util/List;
// Method java/lang/Object."<init>":()V
// int 100000
// Method java/lang/Integer.valueOf:(I)Ljava/
// class java/util/ArrayList
// Method java/util/ArrayList."<init>":()V
// Field ids:Ljava/util/List;
```

```
aload_0
invokestatic #5,  2
(I)Ljava/lang/Integer;
invokeinterface #6,  2
(Ljava/lang/Object;)Z
```

```
// Interface Method java/util/List.add:
// Method java/lang/Integer.intValue:()I
```

增加日志


byteman

```java
class AppMain
{
     public static void main(String[] args)
     {
          for (int i = 0; i < args.length; i++) {
               System.out.println(args[i]);
          }
     }
}
```

```
[mars@r2d2:~/demo/java-bytecode]
% javac AppMain.java
[mars@r2d2:~/demo/java-bytecode]
% java AppMain foo bar bah
foo
bar
bah
```

```
RULE trace main exit
CLASS AppMain
METHOD main
AT EXIT
IF TRUE
DO traceln("exiting main")
ENDRULE

RULE trace main entry
CLASS AppMain
METHOD main
AT ENTRY
IF TRUE
DO traceln("entering main")
ENDRULE
```

```
[mars@r2d2:~/demo/java-bytecode]
% java -javaagent:byteman.jar=script:appmain.btm AppMain foo bar
entering main
foo
bar
exiting main
```

Brainfuck 编译器

https://github.com/jkutner/jipsy

- p 打印变量
- > 将0压入堆栈
- < 弹栈pop
- + 变量加一
- - 变量减一
- [循环开始
-] 循环结束
- s 交换堆栈顶的两个值

0
0

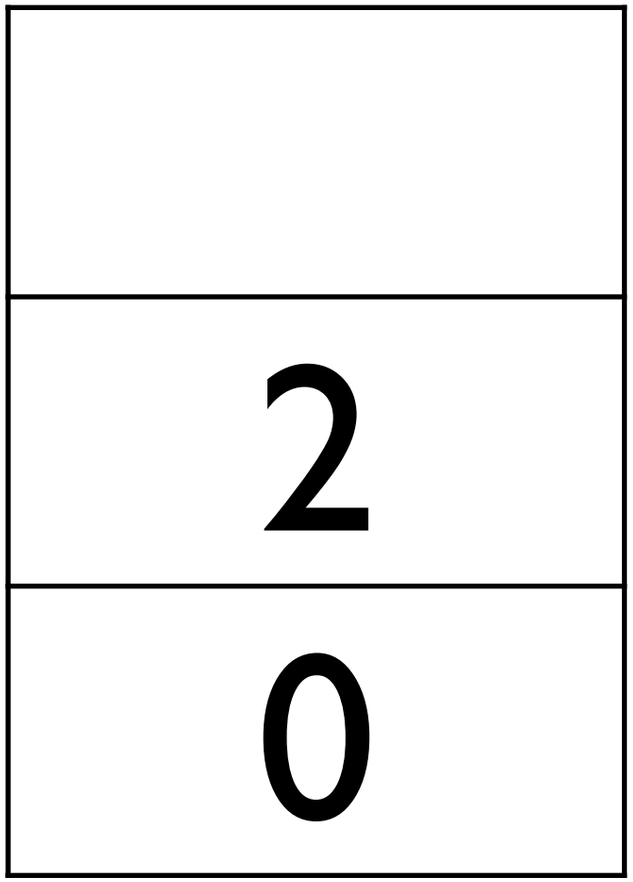
```
> > + + + [ - s + + + + + s ] < p
```

3	
0	

```
> > + + + [ - s + + + + + s ] < p
```

3
0

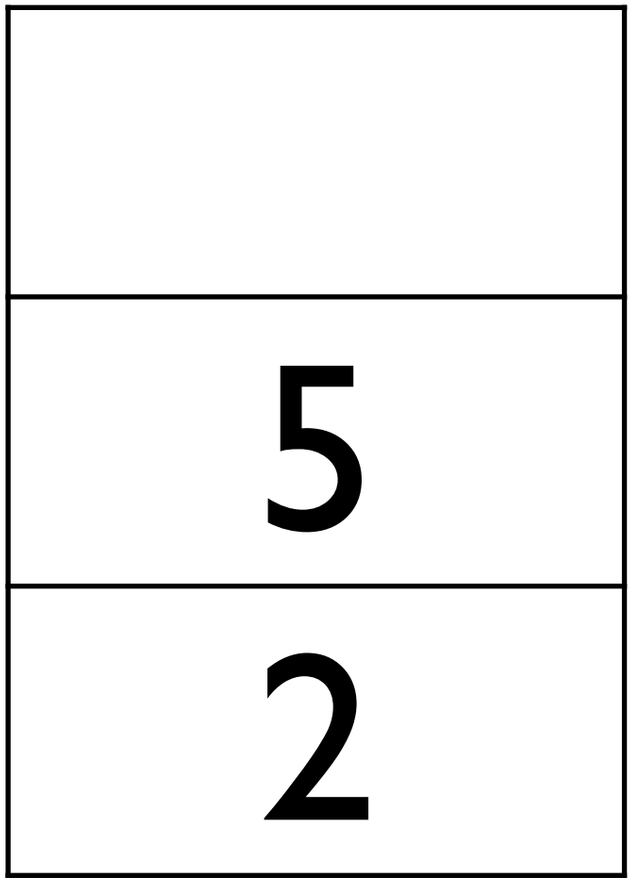
```
> > + + + [ - s + + + + + s ] < p
```




```
> > + + + [ - s + + + + + s ] < p
```

0
2

```
> > + + + [ - s + + + + + s ] < p
```




```
> > + + + [ - s + + + + + s ] < p
```

2
5

```
> > + + + [ - s + + + + + s ] < p
```

0
15

```
> > + + + [ - s + + + + + s ] < p
```

15

```
> > + + + [ - s + + + + + s ] < p
```

15

```
> > + + + [ - s + + + + + s ] < p
```

```
[mars@r2d2:~/demo/java-bytecode/jipsy on master]
% mvn exec:java -Dexec.mainClass="Compiler" -Dexec.args="> > + + + [ - s + + +
+ + s ] < p"
[INFO] Scanning for projects...
[INFO]
[INFO] Using the builder
org.apache.maven.lifecycle.internal.builder.singlethreaded.SingleThreadedBuild
er with a thread count of 1
[INFO]
[INFO]
------------------------------------------------------------------------
[INFO] Building jipsy 0.0.1-SNAPSHOT
[INFO]
------------------------------------------------------------------------
[INFO]
[INFO] --- exec-maven-plugin:1.4.0:java (default-cli) @ jipsy ---
[INFO]
------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO]
------------------------------------------------------------------------
[INFO] Total time: 4.259 s
[INFO] Finished at: 2015-04-19T17:51:10+08:00
[INFO] Final Memory: 9M/115M
[INFO]
------------------------------------------------------------------------
[mars@r2d2:~/demo/java-bytecode/jipsy on master]
% java Main
★
15
```

```java
import java.util.*;

class Main{
    public static void main(String[] args){
        (new ArrayList<String>()).add("Hello");
    }
}
```

```
public static void main(java.lang.String[]);
  Code:
     0: new           #2
     3: dup
     4: invokespecial #3
     7: ldc           #4
     9: invokevirtual #5
    12: pop
    13: return
```

```
new JiteClass("Main") {{
 defineMethod("main", ACC_PUBLIC |
ACC_STATIC,
 sig(void.class, String[].class),
 new CodeBlock() {{
 new(p(ArrayList.class));
 dup();
 invokestatic(
 p(ArrayList.class), "<init>",
sig(void.class));
 ldc("Hello")
 invokevirtual(
 p(ArrayList.class), "add",
 sig(boolean.class, Object.class));
 pop();
 voidreturn();
 }});
}};
```

```
for (String arg : tokens) {
    if ("p".equals(arg)) {
        jPrintInt();
    } else if ("+".equals(arg)) {
        jInc();
    } else if ("-".equals(arg)) {
        jDec();
    } else if (">".equals(arg)) {
        iconst_0();
    } else if ("<".equals(arg)) {
        pop();
    } else if ("s".equals(arg)) {
        swap();
    } else if ("[".equals(arg)) {
        LabelNode[] labelNodes = jBeingLoop();
        loopStack.push(labelNodes);
    } else if ("]".equals(arg)) {
        LabelNode[] labelNodes = loopStack.pop();
        jEndLoop(labelNodes);
    }
}
```

```
public void jPrintInt() {
    dup();
    invokestatic(p(String.class), "valueOf", sig(String.class, int.class));
    getstatic(p(System.class), "out", ci(PrintStream.class));
    swap();
    invokevirtual(p(PrintStream.class), "print", sig(void.class, Object.class));
}

public void jInc() {
    iconst_1();
    iadd();
}

public LabelNode[] jBeingLoop() {
    LabelNode begin = new LabelNode();
    LabelNode end = new LabelNode();
    label(begin);
    dup();
    ifeq(end);
    return new LabelNode[] {begin, end};
}

public void jEndLoop(LabelNode[] beginAndEnd) {
    LabelNode begin = beginAndEnd[0];
    LabelNode end = beginAndEnd[1];
    go_to(begin);
    label(end);
}
```

参考资料

- http://docs.oracle.com/javase/specs/jvms/se8/html/index.html
- http://zeroturnaround.com/rebellabs/

程显峰
@程显峰-Mars

local vars

0	
1	
2	
3	
4	

stack

0	
1	
2	
3	
4	

constants

#1	
#2	
#3	
#4	
#5	