

Máquina Virtual de Java

Diego Antonio Lucena Pumar

7 de septiembre de 2008

Resumen

El presente artículo recoge diversas notas sobre la máquina virtual de Java, también conocida como JVM. Se presenta una descripción de los elementos principales que componen a la misma. No se trata de un estudio exhaustivo sino de una aproximación al mundo del ambiente de ejecución de Java a través de la plataforma de SUN.

1. Componentes

La JVM emula sobre nuestro sistema operativo una máquina Java. Una máquina con un chip llamado picoJava, que junto con un API provee al usuario/programador de una ambiente de ejecución para los binarios de Java. Este chip es capaz de interpretar los “bytecodes” y mandar las ordenes a la computadora sobre la que subyace. El código Java “bytecodes” se compilar para el chip Java a través del compilador `javac`¹ (el compilador de Sun para Java que se distribuye gratuitamente con la JDK, Java SE, versión estándar de Java y en su defecto para otras más profesionales como SDK, Java EE). Podríamos dividir el chip Java en cinco partes:

1. *Snack* (Pila): Usa un esquema FIFO de recuperación y almacenamiento, está dividida en tres partes:
 - a) **Región de variable global**: Provee el acceso a las variables locales.
 - b) **Región ambiente de ejecución**: Usada para proveer código de operación (opcode) para mantener los métodos del marco de pila. También mantiene punteros a variables locales, el marco de la pila previa y el inicio y el final de la región de operandos.
 - c) **Región de operandos**: Contiene los operandos para el método en ejecución.
2. *Registers* (Registros): Para trabajar el chip Java usa cinco registros, los más importantes:
 - a) **Pc**: Contador de programas.
 - b) **Optop**: Mantiene la referencia de memoria al tope de la pila.
 - c) **Frame**: Provee un puntero al marco actual de pila.

¹Existen disponibles compiladores libres de Java como es el de GNU, *Gcj*.

- d) **Vars**: Provee el valor de desplazamiento para variables locales en relación con el puntero a la pila.
3. *Garbage Collection Heap* (Recolector de basura): Limpia la memoria de “basura” en ejecución, por ejemplo, objetos en desuso.
4. *Instruction Set* (Instrucciones): Corresponde a la instrucción que maneja la JVM.
5. *Methods Area* (Área de Métodos): Dispositivo de almacenamiento primario para todos los métodos en ejecución de las distintas clases del programa Java ejecutable.

2. Protección

La protección de la máquina se realiza por medio del verificador de “bytecodes”. La JVM aísla el binario Java para conferir una mayor seguridad al sistema operativo nativo.

3. Compilador: *Just in Time* (Justo a Tiempo)

Java tiene tal vez el inconveniente de que es interpretado, digo tal vez porque sino fuera así no sería tan portable. Para aumentar la velocidad de ejecución de nuestros programas Java podemos utilizar el compilador *Just in Time* que compila los “bytecodes” a nuestra máquina virtual nativa de manera dinámica con lo que conseguimos un 10% más de velocidad.

4. Referencias y Bibliografía

- *Java 2 Curso de Programación*, Fco. Javier Ceballos Sierra.
- <http://www.dcc.uchile.cl/~rbaeza/cursos/proyarq/lbastias/JVM.html>