TRABAJO PRÁCTICO ESPECIAL

Compilador para configuración de VMs

GRUPO Nº 7

Integrantes del grupo:

61413	Alasia, Gastón
61016	Della Torre, Matías
61156	Escudeiro, Patricio
60459	Matilla, Juan Ignacio

Fecha de entrega del informe: 22 de junio de 2023

Introducción

Este proyecto consiste en la implementación de un compilador que permite, a través de un determinado lenguaje, generar un archivo de configuración para levantar una máquina virtual (VM). Es decir, se busca resolver el problema de lidiar con complejos archivos de configuración de máquinas virtuales en XML. Esto es, a través de un lenguaje sencillo, amigable y natural al lenguaje humano.

Particularidades del lenguaje

El lenguaje permite definir variables para representar máquinas virtuales, y dentro se agrega un bloque de código con los recursos de dicha VM. Entre ellas se encuentran nombre, cantidad de cores del procesador, memoria RAM, tamaño de disco, path a una ISO y configuraciones de red. Para estas últimas se utiliza un bloque de código adicional que incluye los siguientes recursos: dirección MAC y tipo de placa de red.

En cuanto a la sintaxis, los recursos de la VM dentro de cada bloque se pueden escribir en cualquier orden. Obligatoriamente deben aparecer el nombre y el path al SO. Los demás se pueden omitir, y se tomarán los siguientes valores por default:

- RAM: 2GB
- Cores: 2
- Disk: 15GB
- Network: tipo NAT, con una dirección MAC aleatoria

En cuanto a los recursos que refieren a capacidades de computadora, se soportan las siguientes unidades: KB, MB, GB, TB. Se permiten las siguientes operaciones aritméticas entre dichos valores: suma, resta y multiplicación.

El lenguaje permite además poder asignar el valor de un recurso de una VM referenciando a otra mediante su nombre de variable. Los valores referenciables son RAM, cores y disk. Cabe destacar que la referencia debe ser sobre una VM declarada por encima de la línea actual, no por debajo.

El path a la imagen ISO debe ser absoluto, sin / adelante y debe apuntar a un archivo ISO cuya extensión sea .iso. No hay que escribir “.iso”, sino que debe terminar en el nombre de archivo sin la extensión. Tampoco puede contener guiones.

Por último, la dirección MAC debe comenzar con la letra 'm' y luego se puede escribir normalmente.

Output del compilador

El output del compilador consiste en uno o más archivos de configuración por cada VM descripta en el código en la carpeta output en la raíz del proyecto. En caso de que no exista dicha carpeta, dará error. Estos archivos pueden luego ser utilizados directamente sin

modificaciones para levantar máquinas virtuales con la librería *libvirt* y el entorno de programas de Qemu-system.

Librerías utilizadas

Además de la utilización de Flex y Bison para la construcción del parser y el AST, se utilizó una librería externa de Sets pero que fue modificada para cumplir con las necesidades concretas de este proyecto.

Limitaciones

En primer lugar, ciertas funcionalidades anunciadas en un principio debieron ser eliminadas:

- Especificar el nombre de un sistema operativo en vez de un path a su ISO. Esto se debe a que descargar un archivo ISO demanda mucho tiempo y no tendría sentido hacer un script que espere ese tiempo para luego poder testear si el archivo de configuración generado es correcto o no. Por lo tanto, se decidió que el usuario deberá buscar una ISO por su cuenta.

- Establecer una dirección IP específica para la configuración de red de una VM. Esto se debe a que no se encontró la manera de cambiar eso en un archivo de configuración XML de *libvirt*.

- Las particularidades previamente mencionadas del lenguaje sobre el formato del path a la ISO y de la dirección MAC. Normalizarlo significaría hacer cambios en la estructura central de la gramática y el AST, lo que excedía los tiempos de entrega del proyecto.

Otras, en cambio, fueron repensadas teniendo en cuenta los objetivos de la materia y la comodidad del uso del proyecto por parte de los usuarios:

- Se decidió no entregar un script de creación de la VM sino que se detallan los pasos y comandos a ejecutar en el archivo *README.md* del proyecto. La idea es que se ejecute esta serie de comandos por cada VM a crear. Esto se hizo ya que es poco probable que un usuario quiera definir y levantar varias máquinas virtuales automática y simultáneamente, sobre todo porque en cada una de ellas deberá instalar el SO en el disco duro de la VM (salvo que sea un SO portable). Es más probable que el usuario primero quiera generar archivos de configuración para una o más VMs y luego ejecutarlas de a una y manualmente cuando lo requiera.

Por último, cabe destacar que al crear el archivo de configuración de una VM se ejecuta sin problemas en libvirt. Sin embargo, hay un error al momento de levantar el disco de booteo (“error no bootable device”). No hubo tiempo suficiente para solucionar el problema, pero el archivo XML obtenido como output del compilador es correcto.

Posibles mejoras al proyecto

En primer lugar se podría implementar chequeos luego de creado el archivo de configuración de una VM para evitar que se ingresen recursos poco realistas, que excedan los recursos de la computadora host o directamente cadenas sin sentido. Actualmente queda a cargo del usuario tener criterio sobre los recursos que le asigna a una VM.

Además, se podría trabajar en la posibilidad de extender el lenguaje para dejar configurar algunas características más específicas de la máquina virtual como podría ser el CPU mode y dispositivos virtuales conectados.

Por último, se podría crear un script con los comandos a ejecutar para levantar la VM y solucionar el problema del booteo.

Posibles mejoras al proyecto desde el punto de vista de la materia

En líneas generales el proyecto se desarrolla de manera bien distribuida durante todo el cuatrimestre gracias a las tres entregas, por lo que se considera correcto. Los tiempos para cada entrega son acordes al trabajo que hay que hacer.

Una sugerencia considerada es cambiar la primera entrega para que requiera algo de programación, de esta manera se puede ir comprendiendo de primera mano la potencialidad y las limitaciones de Flex y Bison. Así, se evita pensar proyectos demasiado ambiciosos que luego deben ser simplificados por la dificultad de su implementación.