Desarrollo de un framework para la enseñanza de los Sistemas Operativos usando Lguest

Lic Javier Díaz, Mg. Lía Molinari, Lic. Matías Zabaljauregui
LINTI, Laboratorio de Investigación en Nuevas Tecnologías Informáticas
Universidad Nacional de La Plata
La Plata, Argentina
{javierd, lmolinari, matiasz}@info.unlp.edu.ar

Abstract— La virtualización no es un concepto nuevo en Sistemas Operativos. Pero varios factores, entre ellos la evolución del hardware, la multiplicidad de plataformas y la corriente de "green IT", permiten que hoy sea revalorizada y promovida.
Este documento analiza una alternativa de enseñanza para introducir este tema en las materias relacionadas con los Sistemas Operativos y para fomentar su uso orientado a lograr mejores servicios a menor costo, más seguros, además de la ventaja básica de poder convivir distintos sistemas operativos en una sola computadora
Keywords: virtualización, Sistemas operativos, lguest

Introducción

Hoy las organizaciones ofrecen sus servicios de información a través de diferentes plataformas y ambientes. En algunos casos se cuenta con una computadora para la implementación de cada servicio (Web, mail, aplicaciones) conectadas por redes de alta velocidad.

Los motivos por los cuales se opta por este esquema pueden ser desde la implementación gradual de estos servicios a través de los años, bajo políticas tradicionales, hasta la necesidad de "aislar" cada servicio con el objetivo de garantizar que, ante la caída de uno o varios de ellos, el resto puede mantener la disponibilidad, es decir fomentar la tolerancia a fallas.

En algunos casos, el administrador opta por este esquema por que intuye que garantiza la seguridad de los servicios por el hecho de residir en computadoras diferentes.

Sumémosle a este esquema la convivencia de diferentes modelos y paradigmas, que sobreviven transformándose en la "herencia" (legacy) que deben seguir prestando servicio sobre plataformas que ya están discontinuadas, o que no justifican asignarles un equipamiento para su ejecución.

Otro punto a considerar es la transición desde esquemas propietarios a libres. Seguramente en esa etapa desde mantenerse el servicio que pueden ofrecer distintas aplicaciones que se ejecutan en los distintos ambientes. Tomemos como ejemplo una organización que contaba con ambiente Windows y sus aplicaciones pero inicia una etapa de adquisición de la cultura open source, bajo Linux. Hasta que pueda garantizarse la puesta a punto de las aplicaciones en esta transición, será necesario contar con mecanismos que permitan el uso de ambos modelos.

Es innegable que este esquema de mantener los servicios/aplicaciones en máquinas separadas ofrece ventajas tales como el rendimiento, la escalabilidad, o una distribución de la carga de trabajo.

Pero también es innegable la dificultad en la administración de esta multitud de ambientes, la cantidad de especialistas que se deben contratar, y las complicaciones ante migraciones o cambios de equipamiento.

Es por eso que en los últimos años se ha "revitalizado" el concepto de virtualización. Hablamos de revitalización porque la

virtualización no es un concepto nuevo. En los 60's, en los mainframes se trabajaba con la metodología por lotes, pero se iniciaba la implementación de sistemas interactivos, lo que hizo necesario definir un ambiente donde pudieran convivir diferentes modelos1.

Son múltiples las referencias y artículos que hoy existen en la Web acerca de la implementación de las maquinas virtuales, y los primeros datan de los 60's.

Vale destacar el trabajo de Popek y Goldberg acerca de las condiciones que una arquitectura debe cumplir para soportar virtualización eficientemente, y que data del 1974 [1].

La virtualización es una metodología que permite compartir los recursos de una computadora en múltiples (y diferentes) ambientes de ejecución.

De acuerdo al grado de abstracción que ofrezcan o metodología utilizada, hay diferentes técnicas de virtualización.

Con la gran cantidad de técnicas de virtualización emergentes, sus posibles combinaciones (paravirtualización y virtualización asistida por harware) y distintos desarrolladores que las implementan, hace tiempo que resulta evidente una imperiosa necesidad de estandarizar, al menos, algunas de las interfaces intervinientes. El universo técnico conformado por ingenieros, investigadores, desarrolladores del kernel Linux ha sido precursor en intentar resolver la complejidad que se genera al combinar las distintas implementaciones de guests, hypervisors, drivers y dispositivos virtuales.

En este esfuerzo por la estandarización y la reducción de complejidad se destacan dos aportes fundamentales en la arquitectura de virtualización de Linux, principalmente orientados a implementar técnicas de paravirtualización:

- paravirt_ops: una interfase que permite conectar distintos guests a distintos hypervisors. [2]
- virtio: abstracción de un mecanismo de transporte que permite que los guests accedan a los dispositivos virtuales. [3]

[1] Ver z/VM, http://www.vm.ibm.com/overview/

La enseñanza de los Sistemas Operativos

Los libros habituales de consulta para los profesores de Sistemas Operativos incluyen el concepto de máquinas virtuales con menor o mayor profundidad. El concepto está habitualmente relacionado con IBM y el uso que hizo de esta tecnología en sus mainframes.

La última versión del libro de Tanenbaum [4] dedica un importante lugar a la virtualización en el capítulo dedicado a Sistemas de múltiples procesadores.

Llegar a incluir virtualización en la materia Sistemas Operativos, depende de la forma en que se estructure, como así también la metodología en cuanto a los trabajos prácticos (taller, laboratorio).

El docente se enfrenta al desafío de definir un temario donde deben convivir conceptos básicos imprescindibles (conceptos de interrupciones y system calls, kernel y sus estructuras, memoria y su administración, proceso-hilos y su administración, entrada-salida, kernel) y las tendencias actuales.

Existen en la actualidad diferentes experiencias para la enseñanza de sistemas operativos, a partir del desarrollo de un sistema operativo propio. Entre varios podemos citar, PINTOS [pintos] (Ben Pfaff, Stanford University), NACHOS [nachos] (Thomas Anderson, University of California, Berkeley), Minix (Andrew S. Tanenbaum, Vrije Universiteit, Amsterdam).

En la Argentina, es de destacar la experiencia de SODIUM, sistema operativo que se desarrolló en la Universidad Nacional de La Matanza, (Ryckeboer, Casas, De Luca, Valiente, Cortina, Puyo) que nació como un proyecto para la enseñanza en la cátedra, donde los alumnos implementaban algoritmos básicos, lo que

posteriormente permitía realizar comparaciones de rendimiento o uso de recursos.

Generalmente un guest es un sistema operativo, originalmente diseñado e implementado para ser ejecutado sobre hardware, que luego es modificado para poder ser ejecutado en un entorno virtual.

Nuestro aporte consiste en construir, sobre ambas abstracciones mencionadas anteriormente, un sistema operativo tipo framework, un template, implementado como un guest concebido desde el principio para ser ejecutado en un entorno paravirtual. Esto simplifica enormemente el desarrollo del prototipo y permite que nuestro guest se ejecute en tantas arquitecturas como soporte el hypervisor.

Sobre el prototipo

El conjunto de drivers básico, de consola, red y disco, serán drivers compatibles con virtio. Es decir, no es necesario implementar un driver distinto para cada posible modelo de dispositivos de hardware.

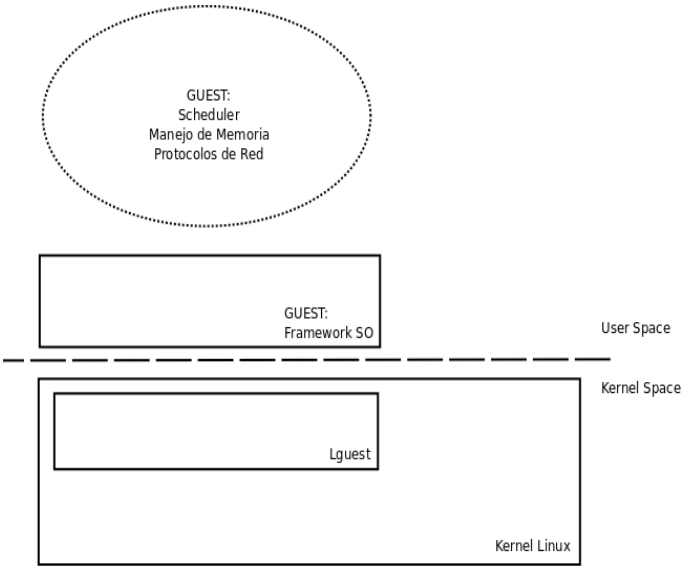
Nuestro prototipo se construye como un guest del hypervisor embebido en el kernel Linux, llamado Lguest [5].

Lguest es un proyecto creado por Rusty Russell [6] con el cuál se intenta demostrar las ventajas de implementar las funcionalidades de un hypervisor como un módulo del kernel. Lguest transforma al kernel Linux en un hypervisor y a los guests en procesos ordinarios del host. Esto plantea una diferencia sustancial con respecto a la aproximación que ofrecía Xen [7] o VMWare hace unos años.

El Lguest de Russel tiene 6000 líneas, incluyendo el código en el espacio de usuario.

El núcleo de Lguest es el módulo lg. El hypervisor es cargado en un área de memoria que reserva este módulo en el momento de la inicialización. Su tarea principal es realizar el “switch” entre el kernel y el guest virtualizado.

Pero, principalmente, Lguest es un proyecto que tiene como finalidad facilitar el estudio, investigación y modificación de un hypervisor real. Es por esto que parece alinearse perfectamente con los objetivos de nuestro proyecto. En algunos cursos de sistemas operativos, utilizamos Lguest como material de estudio [8]. Además parece importante mencionar que uno de los autores, el Lic. Zabaljáuregui, colabora con el proyecto Lguest desde el año 2007, corrigiendo errores y aportando nuevas funcionalidades al código del hypervisor [9].



Al facilitarse el desarrollo de los sistemas operativos, se nos plantea la posibilidad real de construir sistemas operativos guests especializados.

El primer caso de uso en el que pensamos fue la enseñanza/investigación de temas afines. Dado que el alumno no debe lidiar con los detalles del acceso al hardware, se hace viable la opción de aprender sistemas operativos escribiendo

módulos de un guest: scheduler, memoria, protocolos, etc.

No obstante vale aclarar que no es el objetivo abstraerse de las capas inferiores, si no empezar a analizar y ejercitar la idea de poder contar con diferentes ambientes dentro de la misma máquina, y como un Framework puede cumplir con el objetivo de unificar la visión de las capas subyacentes.

El análisis relacionado con la tolerancia a fallas, se basa en asumir que gran parte de los errores están en el software. En las máquinas virtuales el único software que se ejecuta en modo supervisor es el hypervisor.

Un punto a destacar es además, el ahorro de energía y electricidad que se logra al reemplazar varios servidores por uno con diferentes máquinas virtuales.

Por último se nos presenta la posibilidad de enseñar las nuevas técnicas de virtualización a través de una implementación real de un hypervisor y un guest.

Referencias

[1] Popek G. and Goldberg R. Formal Requirements for Virtualizable Third Generation Architectures. 1974

[2] http://lwn.net/Articles/194543/

[3] http://portal.acm.org/citation.cfm?id=1400108

[4] Modern Operating Systems, 3/E, Andrew S. Tanenbaum, ISBN-10: 0136006639. Publisher: Prentice Hall, 2008.

[5] http://www.linux.com/archive/feature/126293

[6] http://en.wikipedia.org/wiki/Rusty_Russell

[7] http://portal.acm.org/citation.cfm?id=945462

[8] http://linux.linti.unlp.edu.ar/

[9] http://git.kernel.org/?p=linux%2Fkernel%2Fgit%2Ftorvalds%2Flinux-2.6.git&a=search&h=HEAD&st=commit&s=zabaljauregui