Are Virtual Machine Monitors Microkernels Done Right?

CARLOS MARTÍN FLORES GONZÁLEZ, Carné: 2015183528

Instituto Tecnológico de Costa Rica
Maestría en Computación
Sistemas Operativos Avanzados
Profesor: Francisco Torres Rojas, Ph.D

Motivation and μHistory: Microkernels y monitores de máquina virtual (*virtual machine monitors* - VMM) has sido dos áreas muy exploradas por la investigación en sistemas operativos(SO). Ambas áreas se han concentrado en convertir sistemas en componentes aislados qeu se comunican a través de interfaces bien definidas. Los microkernels han recibido mayor atención en investigaciones académicas mientras que la investigación en VMM ha sido más bien llevado a cabo por el sector industrial.

Microkernels: Noble Idealism: El más profilifico microkernel que ha desarrollado fue probablemente Mach, un proyecto de investigación del CMU, sus inicios fuer en el Rochester Inteligent Gateway(RIG). El término "microkernel" fue acuñado en respuesta a los kernels monolíticos predominantes del momento. Los partidarios de microkernel argumentan que un núcleo de SO más pequeño es más fácil de mantener, validar y portar a nuevas arquitecturas. Un tema recurrente en muchos de los trabajos en microkernels es que el que dice que los microkernels son arquitecturalmente mejores que los kernels monolíticos. Aparte de Mach, otros sistemas microkernels fueron construidos y muchos de ellos evolucionaron para demostrar que los microkernels, los cuales muchas veces han sido criticados por tener un rendimiento pobre, podían igualar e inclusive superar variantes Unix.

VMMs: Rough Pragmatism. El trabajo previo en máquinas virtuales(VM) fue motivado por la necesidad de mejorar la utilización del hardware facilitando el tiempo compartido de la máquina. Típicamente, las VMs en el modelo de IBM eran "copias" idénticas del hardware subyacente en donde cada instancia corría en su propio SO. Múltiples VMs pueden ser creadas y gestionadas por medio de interfaces exportadas por la VMM, un componente que corre máquina física. Dado que las VMs pueden ser propiedad de múltiples usuarios, mecanimos para proveer un aislamiento fuerte son necesarios en el VMM. Otra característica importante provista por los VMMs es la de compartir el hardware, multiplexando de forma segura varias VMs en un solo conjunto de recursos físicos. Una cantidad significativa de investigación en VMMs se ha llevado a cabo con el fin de mantener el *overhead* en el rendimiento del VMM bajo, el cual ha tenido gran éxito. Aunque las arquitecturas de VMM difieren en el grado de modificación requerido por el SO invitado(*guest*) que ellos alojan, estas modificaciones típicamente son muy pocas o ninguna. Xen y Denali alojan SO *guest* con ligeras modificaciones para mejorar el rendimiento mientras que VMware provee virtualización total del hardware y de esta forma no se necesita ningún cambio. Una característica importante de la mayoría de VMMs es su habilidad para soportar la ejecución de aplicaciones. Los usuarios pueden correr código que es ejecutable en sus máquinas. Investigación previa ha combinado conceptos de microkernel y VM para proveer VMs recursivas que corren SO basado en microkernel. El modo usuario de Linux logra virtualización a nivel de software al correr una VMM como una aplicación dentro de un sistema Linux que lo hospeda(*host*).

Architectural Lessons. Ambos, VMMs y microkernels tiene una gran similitud en su arquitectura.

Avoid Liability Inversion: Uno de las propiedades fundamentales de los microkernesl es la división de un sistema en componentes de espacio de usuario separados. Mientras que el kernel resultante es más pequeño, esta reducción funcional relaja las dependabilidad dentro del sistema: aplicacioens tiene que depender en otros componentes de nivel de usuario para correr. Más importante, el microkernel como tal depende de componentes de nivel de aplicación, como por ejemplo un paginador*pagers*, para trabajar. Al depender de componentes arbitrarios de nivel de usuario con el fin de continuar su ejecución, el kernel renuncia a su responsabilidad por mantener vivo el sistema. A esto se le llama inversión de responsabilidad(*liability inversion*). Una de las principales guías de diseño en Xen ha sido precisamente el evitar estas situaciones. La gestión de memoria de Xen no tiene la noción de paginación, en su lugar se particiona memoria de forma estricta entre las VMs y se permite compartir solo un pequeño conjunto de características. Las VMs son responsables por cualquier trabajo de paginación dentro de las aplicaciones. Decisiones como estas son desarrolladas para asegurar que el fallo de una VM sea aislado y no pueda degrada la estabilidad del sistema como un todo.

Make IPC Performance Irrelevant: En la experiencia en el desarrollo de VMMs se ha notado de que IPC rápido no es una preocupación de diseño crítica en la construcción de VMMs de alto rendimiento. Dado que los VMMs mantienen el aislamiento como un objetivo clave, IPC entre VMs es considerablemente menos común en general. Esta es una consecuencia natural del hecho que el diseño de VMM consideran el SO entero como la unidad de planificación y protección: de esta forma sincronización y transferencia protegida de control son únicamente necesarios cuando dos VMs desean establecer comunicación explícitamente. También se ha determinado que una separación clara entre rutas de operación de control y datos permitió optimizaciones. Se observó, que al configurar explícitamente los canales de comunicación se pueden realizar la inicialización costosa de permisos y verificaciones de seguridad en tiempo de inicialización y asi eludir la validación durante operaciones de datos frecuentes. Este desacoplamiento permite a los mecanismos de comunicación de alto nivel mayor libertad en cómo son implementados. Los diseñadores de microkernel ven los sistemas como un conjunto de componentes que interactúan sobre interfaces basadas en IPC: ellos consideran estas interacciones como llamadas a procedimientos, en donde el sistema entero es una colección de componentes bien aislados. Comunicaciones dentro de un VMM típicamente luce como interacciones entre dispositivos: una ruta de control asincrónica simple combinada con transferencia de datos de formato fijo transparentes.

Treat the OS as a Component. La importante diferencia final entre VMMs y microkernels es que el nivel de granularidad de componentización. Al posicionarse ellos mismos como una respuesta a kernels monolíticos, microkernels se concentran en dividir las unidades funcionales de un SO en partes discretas. Un problema práctico que enfrentan los desarrolladores de microkernels es el mismo que se enfrenta en cualquier esfuerzo de SO nuevo: al cambiar la API visible a las aplicaciones, un SO pierde el conjunto completo de software disponible en los sistemas existentes. De esta forma, la mayoría de proyectos microkernel se abandonaron gastando un esfuerzo considerable para implementar capas de interface de emulación para SO existentes.

VMMs difieren significativamente con esto. Su implementación *a priori* es la de soportar SO existentes. Por ejemplo, el código puede ser compilado para qeu sea ejecutando en una variedad de SO existentes. Puede ser ejecutado en SO *guest* encima de Xen. Esto reduce el costo de usuarios y aplicaciones de usar esta tecnologoía y resuelve dos de los principales problemas de los sistemas de microkernel: la dificultada de atraer una base de usuarios substancial y el reto de mantener SO microkernel actualizados con las caracterísiticas de los SO existentes. Al suportar SO existentes, VMMs únicamente necesitan justificar el potencial *overhead* en su rendimiento con el fin de ser una opción atractiva. Las VMMs atraen a los desarrolladores porque presentan un ambiente de desarrollo familiar. Usando SO existentes como los bloques de componentización básicos se permite que los desarrolladores que continuen usando el mismo conjunto de herramientas que ellos tienen en sus sistemas existentes. El tamaño de los componentes que corren en VMMs puede ser ajustado dependiendo de la funcionalidad requerida por

ellos. Un ejemplo es *ttylinux*, una distribución minimalista de Linux que provee multi-tarea, multi-usuario y capacidades de red en menos de 4Mb de tamaño de SO.

The future of VMMs. La interfaces delgadas(*narrow*) entre los componentes del sistema son cruciales para facilitar la extención. La interface limpia de IPC provista por microkernels permitió a los investigadores la habilidad de enfocarse en componentes específicos del sistema sin necesidad de lidiar con código no relacionado. De forma similar, las interfaces delgadas presentes en Xen permiten que los dispositivos y el SO sea fácilmente extendido. La arquitectura de los dispositivos de Xen permite a los controladores de dispositivos estar aislados en una VM separada, para mejorar la confiabilidad y permitir que interfaces de bajo nivel sean extendidas sin necesidad de modificar el SO o el VMM. La exploración de cómo los servicios son administrados en un sistema multi-SO continuará presentando muchas oportunidades de investigación en el futuro. Otra ventaja de interfaces delgadas acopladas con un kernel de privilegio mínimo, es el lograr un alto nivel de seguridad en el sistema. La innovación reciente en VMMs no se ha dedicado a explorar aspectos centrados en el rendimiento sino al hecho de que en esta plataforma promisoria se le puede seguir desarrollando e integrando características de los sistemas existentes.

1. ¿CUÁL ES EL PROBLEMA QUE PLANTEA EL *PAPER*?
2. ¿POR QUÉ EL PROBLEMA ES INTERESANTE O IMPORTANTE?
3. ¿QUÉ OTRAS SOLUCIONES SE HAN INTENTADO PARA RESOLVER ESTE PROBLEMA?
4. ¿CUÁL ES LA SOLUCIÓN PROPUESTA POR LOS AUTORES?
5. ¿QUÉ TAN EXITOSA ES ESTA SOLUCIÓN?