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

Un artículo por Hand *et al.* en el reciente taller HotOS re-examina los microkernels y los contrasta con monitores de máquina virtual(*virtual machine monitors* - VMM). Se encuentra que los dos tipos de sistemas comparten criterios en arquitectura pero también tienen un número de diferencias técnicas las cuáles se examinan en el artículo. Se concluye que los VMMs son un tipo especial de microkernels, "microkernels bien hechos". Cuando se hace una mirada más cercana a los argumentos principales hechos por Hand *et al*, se encuentra que estos son difíciles de justificar o estan en desacuerdo con la literatura disponible.

Background. *History Revisited:* Ambos, Microkernels y VMMs tiene una larga historia que data de los años 70s. Golberg define un VMM como "*software que transforma la interfaz de una sola máquina en una ilusión de muchas. Cada una de etas interfaces (máquinas virtuales) es una replica eficiente del sistema original de la computadora completa con todas las instruciones de procesador ...*". Liedtke describe el enfoque microkernel como "*minimizar el kernel e implementar todo lo posible fuera del kernel*". Examinando los objetivos de los dos enfoques se muestra que hay más similitud que la evidente a partir de las definiciones: Goldber lista confiabilidad del software, seguridad de datos, APIs de sistema alternativas y nuevos mejoras y mecanismos como beneficios. Liedtke menciona flexibilidad, mantenibilidad e interdependencia restringuida. Parece qeu mientras los VMMs y microkerneles comparten un conjunto de objetivos, ellos toman un enfoque diferente hacia la solución. Ambos consideran la minimalización como algo importante. Mientras que para los microkernels este es un objetivo clave, Goldberg lo reporta como un resultado de la estructura del sistema: "*un principio clave en el análisis de la confiabilidad del software es que el VMM problemente esté correcto - la probabilidad de error es cercana a cero. Esta suposición es razonable porque el VMM probablemente sea un programa pequeño ...*". *Core primitives:* en un esfuerzo por minimizar la funcionalidad del kernel, microkernesl ofrecen un conjunto mínimo de abstracciones con una primitiva central para extensibilidad: comunicación inter-proceso(IPC). En un microkernel, IPC provee tres propósitos primarios: (1) IPC es el mecanismo para cambio controlado por kernel de flujos de ejecución entre dominios de protección. (2) IPC es el mecanismo para transferencia de datos controlado por kernel entre dominios de protección. (3) IPC es el mecanismo para delegación de recursos entre dominios de protección que requiere acuerdo mutuo entre múltiples partes. Combinando estas tres operaciones ortogonales en una sola primitiva se reduce el número de mecanismos de seguridad, se reduce la complejidad del códiso y se reduce el tamaño del software. Un código más pequeño reduce el número de errores en el kernel privilegiado y también reduce la huella de caché. Un requerimiento obvio clave para cualquier microkernel es por tanto una primitiva de IPC de bajo *overhead*. Todas las otras tres operationces que requieren una combinación de los tres mecanismos puede ser implementados a través de una sola primitiva de IPC. VMMs, en comparación, se parecen mucho al hardware del procesador y ofrece una rica variedad de primitivas. Cada primitiva requiere de un conjunto dedicado de mecanismos de seguridad, recursos y código de kernel. A continuación una lista del subconjunto común de

primitivas que pueden encontrarse en la mayoría de VMMs: (1) cambio sincrónico de dominio de protección de usuario invitado(*user guest*) a kernel invitado(*guest kernel*). (2) cambio sincrónico de dominio de protección de *guest kernel* a *guest user*. (3) Canales de comunicación asincrónica a través de dominios (de máquina virtual(VM) a máquina virtual). (4) Asignación de recursos por VM a través de la interfaz*hypercall* de VMM. (5) Asignación de recursos dentro de la VM (via virtualización de *page-table* en hardware). (6) Re-asignación de recursos (via *page flipping*). (7) *Page-fault* y manejo de excepciones a través de virtualización de excepciones. (8) Notificación asincrónica de eventos a través de dominios via mecanismos de señales interrupcion-virtual. (9) Notificación de interrupción de hardware a través de un controlador de interrupciones virtualizadas. (10) Un conjunto de dispositivos comunes, tales como NIC y disco.

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?