Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería

SISTEMAS OPERATIVOS

Kernel Monolítico vs Micro Kernel

Docente: Álvarez Salgado Carlos Francisco
Alumno: Gómez Cárdenas Emmanuel Alberto
Matricula: 1261509

El kernel es la parte dispensable más importante de un sistema operativo. Normalmente un sistema operativo consiste de dos partes, el espacio del kernel (modo privilegiado) y el espacio del usuario (modo no privilegiado), esto permite la protección entre procesos. Existen dos conceptos de kernel diferentes: kernel monolítico (el enfoque más antiguo del cual se encuentra en Unix, MS-DOS y en los primeros Mac Os) y el µ-kernel o microkernel.

Actualmente existen dos tipos de microkernel, la primera generación era más o menos un kernel monolítico despojado de algunas capacidades, debido a inconvenientes con el rendimiento, bastantes capacidades tuvieron que ser puestas de vuelta dentro del kernel, lo que resultó en un kernel incluso más grande que el monolítico. Investigaciones en el campo del microkernel mostraron que la mejor solución es crear un microkernel puro. La segunda generación de µ-kernel como el L4 se encuentran altamente optimizados, lo cual resulta en un buen rendimiento de E/S.

Comparación entre los conceptos básicos de ambos enfoques.

Los kernels monolíticos utilizan señales y “socktes” para asegurar la comunicación entre procesos mientras que los µ-kernel utilizan el enfoque de colas de mensajes la cual permite que todas las partes del sistema sean intercambiables.

Los kernels monolíticos implementan todo lo necesario para una gestión de memoria en el espacio del kernel (estrategias de asignación de memoria, gestión de memoria virtual y algoritmos de reemplazamiento de página). La primera generación de µ-kernels delega la gestión de memoria al espacio de usuario y se encarga de controlar solo los accesos de registros básicos, cada que ocurre una falla en la gestión de páginas, se tiene que entrar en modo privilegiado para obtener acceso a la memoria y regresar al modo de usuario, después envía el resultado de regreso al proceso. Esto hace que el proceso de reservar memoria consuma bastante tiempo, además de ser tedioso. Para resolver este problema la segunda generación de µ-kernels tienen estrategias más refinadas de gestión de memoria.

En el tema de la seguridad y estabilidad del kernel excluir los procesos del sistema del espacio del kernel es una manera de evitar bastantes problemas como lo son modificación de procesos del sistema por parte del usuario u otros procesos, detención repentina de un programa causando una detención de más programas causando una detención del sistema, entre otros. Es más sencillo asegurar un funcionamiento correcto de un kernel pequeño que el de uno grande, con este enfoque los problemas de estabilidad son más sencillos de resolver.

La comunicación E/S es trabajada mediante interrupciones emitidas por el hardware o enviadas a él, los kernels monolíticos y la mayoría de µ-kernels de primera generación ejecutan controladores de dispositivos dentro del espacio del kernel. El concepto de los módulos fue implementado para obtener mas independencia y separación del kernel, estos solo son cargados cuando son necesitados por el sistema. El enfoque del µ-kernel para la comunicación E/S no es manejado directamente, solo asegura la comunicación.

En cuanto al tema de portabilidad y extensibilidad estos son las partes, además del tamaño, en las que más destaca el µ-kernel en comparación con el kernel monolítico. Agregar nuevas características a un kernel monolítico significa recompilar el kernel completo, en cambio gracias a que en el µ-kernel los servicios son aislados unos de los otros a través del sistema de mensajes basta con re implementar el servicio. Los µ-kernels se mantienen independientes a la máquina y pueden ser fácilmente porteados.

El kernel monolítico es implementado en GNU/Linux en los cuales se encuentran todas las funciones del sistema (incluyendo los procesos enteros, gestión de memoria, funcionalidad E/S, entre otros). La comunicación de E/S es proveída por los módulos, los cuales pueden ser insertados y removidos en el tiempo de ejecución.

El kernel hibrido (µ-kernel de primera generación) representa la base de Next, Mac OS X y otros. Fue pensado como un µ-kernel de tamaño pequeño altamente portable el cual incluye la más mínima cantidad de funciones del set de kernel.

QNX (Quick Unix) es el más puro sistema operativo para aplicaciones de tiempo real basado en µ-kernel. Las aplicaciones de tiempo real enfatizan predictibilidad y estabilidad.

El µ-kernel L4 probó junto con QNX que los µ-kernels pueden ser tan veloces como la contraparte monolítica permitiendo extensibilidad. El rendimiento alcanzado por el pequeño tamaño (12KB de código) y la optimizada comunicación entre procesos. Todas las demás funciones, como gestión de memoria, controladores de dispositivos, manejo de interrupciones, protocolos de pila, etc… Se encuentran en el espacio del usuario.

En conclusión, L4 y QNX han demostrado que la velocidad ya no es un problema entre los µ-kernels y su portabilidad y extensibilidad adicional solamente demuestran el potencial que pueden llegar a alcanzar, son más fáciles de dar mantenimiento (a comparación con su contraparte monolítica), la comunicación entre procesos puede ser aún más rápida con la utilización de algoritmos de comunicación más inteligentes. Para alcanzar mejor rendimiento los µ-kernels deben ser directamente optimizados para el procesador en el cual se tiene como objetivo correrlos, de esa manera puede ser utilizado el potencial completo del µ-kernel.