

¿Por qué los ordenadores simplemente no pueden funcionar todo el tiempo?

MICRO VS. MONOLÍTICO

La trifulca entre el inventor de Minix Andrew W. Tanenbaum, y el entonces novato Linus Torvalds ya es materia de leyenda. Torvalds era usuario de Minix antes de montar su primer Linux basándose en el sistema de Tanenbaum. El Doctor Tanenbaum ha sido tan gentil como para escribir para nosotros su opinión... una opinión que en 15 años no ha cambiado: tanto Linux como Windows son poco fiables.

POR ANDREW W. TANENBAUM

Los usuarios de ordenadores están cambiando. Hace diez años, la mayoría era gente joven con muchos conocimientos técnicos. Cuando algo iba mal, cosa que ocurría a menudo, sabían arreglarlo. Hoy en día, el usuario medio es mucho menos sofisticado, puede que sea una niña de 12 años o su abuelo. Tal vez sepan tanto de arreglar ordenadores como sabe el hacker típico de arreglar su coche. Lo que más desean es un ordenador que funcione todo el tiempo, sin excentricidades ni fallos.

Muchos usuarios comparan sus ordenadores con su televisor. Ambos están llenos de mágicos componentes electrónicos y tienen grandes pantallas. La mayoría tienen un concepto implícito de televisor: (1) compras uno; (2) lo enchufas; (3) funciona perfectamente sin fallos durante los siguientes 10 años. Esperan que ocurra lo mismo con los ordenadores y, cuando no es así, se frustran. Cuando un experto en informática les dice que "si Dios hubiera querido que los ordenadores funcionaran todo el tiempo, no habría inventado el botón RESET", no le ven la gracia.

Ante la ausencia de una definición de fiabilidad mejor, adoptemos la siguiente: Un dispositivo se considera un 99% fiable si los usuarios nunca experimentan ningún fallo a lo largo de todo el periodo en el que poseen el dispositivo. En función de esta definición, casi ningún ordenador es fiable, mientras que casi todos los TVs, iPods, cámaras digitales,

vídeo cámaras, etc. sí lo son. La gente con conocimientos técnicos está dispuesta a perdonar a un ordenador que se cuelga un par de veces al año, pero los usuarios normales no.

Los usuarios domésticos no son los únicos a quienes frustra la pobre fiabilidad de los ordenadores. Incluso en entornos altamente técnicos es un problema. Las compañías como Google y Amazon, con centenares de miles de servidores, experimentan miles de fallos cada día. Han aprendido a vivir con ello, pero preferirían sistemas que simplemente funcionaran todo el tiempo. Desafortunadamente, el software actual les falla.

El problema de fondo es que el software contiene errores, y cuanto más software hay, más errores contiene. Varios estudios han demostrado que el número de bugs por mil líneas de código (KLoC) cae en un horquilla de entre 1 y 10 en sistemas de producción de gran tamaño. Un software bien escrito puede tener 2 bugs por KLoC, pero nunca menos. Según esto, un sistema operativo con, digamos, 4 millones de líneas de código puede contener entonces hasta 8000 bugs. No todos son fatales, pero algunos lo serán. Un estudio de la Universidad de Stanford muestra que los controladores de dispositivos, que componen el 70% de la base de código de un sistema operativo típico, tiene índices de errores de entre 3 a 7 veces superiores al resto del sistema. Y tienen unos índices tan altos porque

(1) son más complicados y (2) se inspeccionan menos. Mientras que mucha gente inspecciona el scheduler, pocos se ocupan del controlador de una impresora.

Kernels más Pequeños

La solución a este problema es mover el código fuera del kernel, donde puede hacer el máximo daño, y depositarlo en procesos del espacio de usuario, donde los bugs no pueden colgar el sistema. Así es como se ha diseñado MINIX 3. El sistema MINIX actual es la (segunda) encarnación del MINIX original que se lanzó en 1987 como sistema operativo educativo, pero que, desde entonces, ha sido revisado radicalmente hasta convertirse en un sistema de alta fiabilidad y con capacidades de autoreparación. Lo que sigue es una breve descripción de MINIX 3. Puede el lector aprender más sobre el sistema en [1].

MINIX 3 está diseñado para correr la menor cantidad de código posible en modo kernel, donde los errores pueden ser fácilmente fatales. En vez de 3 ó 4 millones de líneas de código, MINIX 3 cuenta con unas 5000 líneas de código de kernel. A veces, a los kernels de este tamaño se les conoce como microkernels. Manejan la administración de procesos de bajo nivel, scheduling, interrupciones y el reloj, y proveen algunos servicios de bajo nivel a componentes del espacio de usuario.

El bloque mayor del sistema operativo se ejecuta como una colección de controladores de dispositivos y servidores, cada uno ejecutándose como un proceso ordinario en espa-



cio de usuario y con permisos restringidos. Ninguno de estos controladores ni servidores corren como superusuario ni nada que se le parezca. Ni siquiera pueden acceder a dispositivos de entrada y salida o al hardware MMU directamente. Han de utilizar los servicios del kernel para escribir en el hardware. La capa de procesos que se ejecuta en modo usuario directamente sobre el kernel se compone de controladores de dispositivos, con el controlador de disco, el controlador de Ethernet y todos los otros controladores ejecutándose en procesos separados, protegidos por el hardware MMU para que no puedan ejecutar ninguna instrucción privilegiada y para que no puedan escribir en ninguna dirección de memoria que no les pertenezca.

Por encima de la capa de controladores viene la capa de servidores, con un servidor de ficheros, de procesos y otros servidores. Los servidores utilizan los controladores así como los servicios del kernel. Por ejemplo, para leer de un fichero, un proceso de usuario envía un mensaje al servidor de ficheros que, a su vez, envía un mensaje al controlador del disco para que recupere los bloques que necesita. Cuando el sistema de ficheros los tiene en el búfer de la caché, llama al kernel para que los mueva al espacio de direcciones del usuario.

Además de los servidores mencionados existe otro servidor llamado "servidor de reencarnación". El servidor de reencarnación es el padre de todos los procesos de controladores y servidores y monitoriza su comportamiento. Si descubre alguno que no responda a sus pings, arranca una nueva copia desde el disco (a excepción del controlador de disco, que se copia a RAM). El sistema se ha dise-

ñado para que muchos (aunque no todos) los controladores y servidores puedan ser reemplazados automáticamente mientras el sistema esté en funcionamiento y sin molestar a los procesos de usuarios o siquiera notificar al usuario. De esta manera, el sistema es capaz de autorepararse.

Para probar que estas ideas funcionaban en la práctica, condujimos el siguiente experimento: Empezamos un proceso de inyección de fallos que sobrescribía 100 instrucciones de máquina en el controlador binario de Ethernet que en ese momento estaba en funcionamiento para ver lo que pasaba si alguno de ellos se ejecutaba. Si no pasaba nada durante unos segundos, se inyectaban otros 100 y así sucesivamente. En total, inyectamos 800.000 fallos en cada uno de los tres controladores Ethernet y provocamos 18.000 cuelgues del controlador. En todos los casos, el controlador fue restituido por el servidor de reencarnación. A pesar de inyectarle en total más de 2,4 millones de fallos, el sistema operativo no se colgó una sola vez. Huelga decir que si ocurre un error fatal en un controlador de Linux o Windows que se ejecute en el kernel, todo el sistema operativo se vendría abajo instantáneamente.

¿Hay algún problema con esta manera de hacer las cosas? Sí. El rendimiento se resiente. No lo hemos medido extensivamente, pero un grupo de investigación en Karlsruhe, que ha desarrollado su propio microkernel, L4, sobre el cual luego han ejecutado Linux como un proceso monousuario, han conseguido bajar el impacto a un 5%. Creemos, que de dedicarle el suficiente esfuerzo, podríamos reducir el impacto entre el 5 y el 10%. Sin embargo, el rendimiento no ha sido una de nuestras prioridades. Como mucho, los usua-

rios que estén leyendo su correo o mirando sus páginas de Facebook no se ven limitados por el rendimiento de su CPU. Lo que sí quieren, sin embargo, es un sistema que funcione todo el tiempo.

¿Por qué Nadie Usa Micro kernels?

De hecho sí lo hacen. Probablemente estés ejecutando muchos. En tu móvil, por ejemplo, hay un pequeño pero muy convencional ordenador. Y es muy probable que esté ejecutando L4 o Symbian, otro microkernel. El router de alto rendimiento de Cisco utiliza otro. En los mercados militares y aeroespaciales, donde la fiabilidad es primordial, se utiliza ampliamente Green Hills Integrity, otro microkernel. PikeOS y QNX son también ejemplos de microkernels utilizados en sistemas industriales y embebidos. En otras palabras, cuando lo que realmente importa es que "el sistema simplemente funcione todo el tiempo", la gente opta por los microkernels. Para más sobre este tema, aconsejo leer [2].

Resumiendo, creemos que, en base a nuestras muchas conversaciones con usuarios no-técnicos, lo que más desean es un sistema que funcione perfectamente todo el tiempo. Tienen una baja tolerancia para sistemas poco fiables, pero no cuentan en la actualidad con la posibilidad de elegir. Creemos que un sistema basado en microkernel es el camino hacia sistemas más fiables. ■

RECURSOS

[1] Sitio de MINIX: www.minix3.org

[2] Sobre microkernels y más cosas: www.cs.vu.nl/~ast/reliable-os/

PERDIÓ EL BUS, DOCTOR

◀ Viene de la página 3

xes, BSDs, Solarises, o incluso si me apuras, Windowses, alegando que son inferiores sin haber puesto a prueba en el mundo real tu propia creación, queda un poco de perdedor (de bus) resentido.

Uno de los puntos de debate más intensos entre Tanenbaum y Torvalds versaba sobre el hecho de que MINIX no era libre. Aparte de lo cuestionable moralmente de que un profesor universitario cobrara a sus alumnos por un SO didáctico de su propia invención, ahí estuvo su gran error:

sin ser libre jamás consiguió una gran comunidad, no al menos teniendo a Linux y BSD por ahí fuera. Tanenbaum tropezó con la misma piedra que los del laboratorio Bell y su Plan 9: liberaron demasiado tarde, y perdieron de nuevo el bus – en este caso, el de la popularidad.

MINIX funcionará perfectamente mientras no se le toque. Pero un SO no puede evolucionar en un vacío, y en el momento que se le saque (si es que se saca alguna vez) de su campana de cristal, le empezarán a crecer las verrugas. Algunas se

podrán curar. A otras tendrán que conformarse con ponerles un parche.

Igualito, igualito que lo que ocurre con los SOs del mundo real. ■



Paul C. Brown
Director