System Calls no bloqueantes para threads a nivel de usuario versus Scheduler Activations en un Sistema Operativo Didáctico.

Nicanor Casas ncasas@unlam.edu.ar
Graciela De Luca gdeluca@unlam.edu.ar
Martín Cortina mcortina@unlam.edu.ar
Hugo Ryckeboer h_ryckeboer@yahoo.com.ar

Universidad Nacional de la Matanza
Departamento de Ingeniería e Investigaciones Tecnológicas
Dirección: *Florencio Varela 1703* - Código Postal: 1754 - Teléfono: *4480-8900/8835*

CONTEXTO

El Sistema Operativo SODIUM tiene como uno de sus objetivos de desarrollo permitir a los alumnos de las materias de sistemas operativos observar y sacar conclusiones del funcionamiento de los distintos algoritmos que componen un sistema operativo, uno de estos análisis está enfocado hacia la ejecución de procesos pesados y livianos. En una primera etapa se desarrollaron las funciones para la gestión y administración de procesos pesados. Luego en segunda instancia nuestro objetivo se centró en el desarrollo de threads o hilos de ejecución, comenzando primeramente con los hilos a nivel de usuario. Para ello se programó una biblioteca de hilos de usuario STL (SODIUM Thread Library) con las funciones mínimas para la gestión y administración de hilos que ejecuta en el nivel de usuario. Para poder aprovechar las ventajas de los hilos de usuario se inició un análisis de las distintas implementaciones propuesta para System Calls no bloqueantes.

De tal manera que el Sistema Operativo pueda mostrar la ejecución del proceso y dentro del proceso la ejecución de los hilos, permitiendo diferenciar a los alumnos planificación y estado de proceso vs planificación y estado de hilos de usuarios mediante los mecanismos que posee el SODIUM para mostrar el estado del sistema y guardar un archivo histórico para su análisis posterior.

RESUMEN

La necesidad de dotar al sistema operativo SODIUM de elementos comparativos, como los realizados con los diferentes algoritmos de planificación y los diferentes administradores de memoria, nos obligó a montar más de una biblioteca de hilos. Estás fueron en principio las correspondientes a POSIX Pth, utilizándose una codificación similar a la realizada para Linux y agregar una biblioteca de hilos definida como es la Scheduler Activations de la Universidad de Washington. De ésta no se tienen los programas fuentes, pero elabora conceptos que debieron ser actualizados pero que tiene un buen fundamento. Esta nueva biblioteca es programada por los alumnos de la Universidad de la Matanza y es el primer elemento para comparar comportamientos.

Palabras Claves:

Blocking, Upcall, Downcall, STL, Scheduler Activations – Psyche

1.- INTRODUCCIÓN

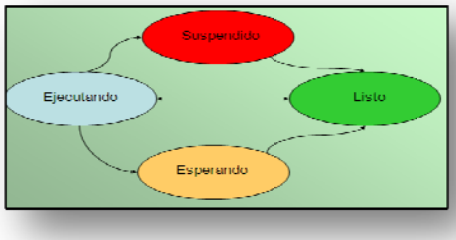
Se consideró que un sistema operativo didáctico debe poseer más de de un tipo de Hilos de Ejecución, ya que le permite al alumno aprender a evaluar cual sería su mejor opción ante un tipo de necesidad especifica en la ejecución de un sistema. Para esto el sistema operativo SODIUM debe poseer por lo menos dos tipos de implementaciones de hilos de ejecución. Para poder realizar un paralelismo entre las diferentes bibliotecas, en

las mismas condiciones, estos hilos necesitan dos tipos de System Calls: las que bloquean el proceso cuando se solicita una System Call bloqueante y las que implementan alguna técnica para salvar este bloqueo y permitir que el hilo que se bloquee mientras que otro hilo del mismo proceso pase a ejecución ya que esta System Call no bloquea al proceso permitiendo a la biblioteca STH bloquear al hilo y pasar otro a ejecución. Nuestra propuesta posee una diferencia fundamental entre los sistemas operativos de uso general (tales como Linux o Windows) ya que este trata de generar estructuras sencillas para los procesos tal como aparecen en los libros de Sistemas Operativos intercambiar los planificadores del procesador y administradores de memoria mediante una parametrización, ahora agregamos a los procesos, hilos de usuario, pero debemos encontrar algún sistema que impida el bloqueo de los procesos por parte de los System Calls bloqueantes, para esto estudiamos las propuestas de diferentes sistemas operativos y bibliotecas threads a nivel de usuario, para intentar encontrar una implementación efectiva y sencilla a los efectos didácticos, sin importar la mejora de la perfomance de ejecución, teniendo en cuenta siempre el objetivo didáctico de este sistema operativo.

2. LINEAS DE INVESTIGACION y DESARROLLO.

Cuando un proceso crea **hilos de usuario**, su existencia no es reconocida por el sistema operativo ya que para su creación, asignación de recursos, planificación, cambios de contexto y eliminación no realiza el proceso ninguna llamada al sistema, todo lo hace con sus propios recursos. Por lo tanto para el sistema operativo cuando hay una solicitud de servicios, éste la ve como un pedido del proceso y lo bloquea si es necesario. Para estas estructuras de biblioteca se definieron las funciones elementales para su funcionamiento tomando en parte como base las funcionalidades de la biblioteca de threads a nivel de usuario del proyecto GNU Pth [06] ya que esta posee una gran documentación y generando nuestra propia biblioteca SLT con una cantidad de funciones limitada.

Ciclo de Vida de losThreads de Usuario



Análisis de diferentes tratamientos para las System Call Bloqueantes

Cuando una aplicación requiere el servicio del kernel, lo realiza mediante una System Call. Las System Call proporcionan la interface entre las aplicaciones de nivel de Usuario y el Kernel. Son utilizadas normalmente cuando la aplicación requiere un servicio del hardware subyacente del sistema, como una E/S o una interfaz de red. Los servicios del Sistema operativo la respuesta y retorna un resultado dependiendo de la definición de la System call. A veces es imposible para el kernel atender una petición inmediatamente debido a que el hardware subyacente no está listo para proporcionar el servicio requerido en forma instantánea. En estos casos el kernel configura al proceso removiéndolo de la cola de ejecutando y sirviendo otros procesos. Solo cuando la petición es realizada, entonces el proceso es desbloqueado y puesto en la cola de listos nuevamente. En cualquier aplicación que use un sistema en el que el número de procesos o hilos de kernel sea el mismo que el número de procesadores del sistema, puede darse una situación en que el número de hilos de kernel ejecutando sea menor que el número de procesadores disponibles. Este desperdicio de recursos indica que podría haber hilos de usuario que esperan para ser ejecutados y procesadores que no realizan nada, mientras esperan para desbloquear hilos de kernel. Esto es una condición que surge debido a la inhabilidad del kernel de reconocer hilos de usuario y la carencia de comunicación entre el

kernel de sistema operativo y el planificador de la biblioteca a nivel de usuario.

La alternativa: Scheduler Activations

Los scheduler activations [02] fueron propuestos originalmente por Anderson (Universidad de Washington). Sus autores implementaron este mecanismo en la cima de la biblioteca de hilos. Este sistema desafortunadamente esta fuera de uso y su fuente nunca fueron publicados. Sin embargo, esto sirvió como base para determinar una nueva codificación para bibliotecas de hilos. Así surge la biblioteca SLT_SA reiseñada y desarrollada en forma experimental para el SODIUM..

Descripción de los SODIUM Scheduler Activations

Permiten al kernel notificar a la aplicación cuando ejecuta una decisión de planificación sobre uno de sus hilos. Al igual que Anderson usamos el término "Scheduler Activation" porque cada evento en el kernel causa que el sistema de hilos de Usuario reconsidere su decisión de planificación. Esto se logra con un mecanismo que introduce un tipo especial de System Call llamada *upcall* [04] a diferencia de una System Call tradicional que desde una aplicación hacia otra aplicación es definida como una *downcall*. Una upcall es una llamada desde el kernel hacia la aplicación. Un Scheduler Activation es un contexto de ejecución prácticamente similar al modo de manejo que realiza un hilo de kernel, utilizando para su implementación de los hilos de kernel nativos del sistema y agregándose simplemente ciertas funcionalidades que son necesarias para upcalls. Cuando una aplicación usa un hilo de kernel clásico, crea dicho hilo y designa una función para el sistema operativo ejecute. Lo opuesto ocurre con los Scheduler Activations. El sistema operativo decide cuando un Activation es necesitado. Luego lo crea y comienza a ejecutar una función de Usuario especifica.

Soporte del Kernel para Scheduler Activations

Pedir soporte de kernel es una solución utilizada para permitir a los hilos de Usuario que sean ejecutados mientras otros hilos esperan por un servicio del kernel y permanecen bloqueados. Los **Scheduler Activation** provén este Soporte al crear un nuevo hilo de kernel y notificar a la **biblioteca de hilos de usuario** cuando un hilo de usuario se bloquea. En este sentido otro hilo de usuario puede ejecutar en vez de tener la aplicación bloqueada en el kernel.

De esta manera la eficiencia de los hilos de usuario se mantiene intacta.

Los hilos de Kernel se bloquean y resumen sin notificar a los hilos de usuario

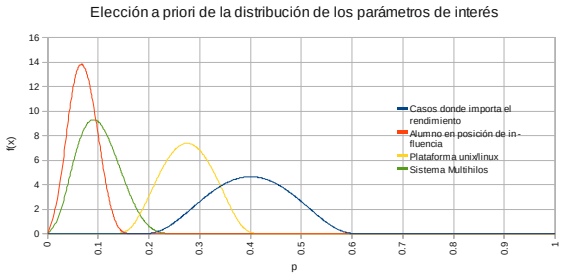
3. RESULTADOS OBTENIDOS/ ESPERADOS

Los resultados obtenidos en esta primera etapa son hasta el momento satisfactorios en términos de ejecución y en términos de desarrollo.

La perfomance de la nueva biblioteca es pobre referente a las bibliotecas utilizadas por Linux, tales como la GNU Pth, contra la cual se realizan los estudios comparativos hasta el momento.

Está en estudio ampliar las funciones de biblioteca para realizar estudios más completos, nuevas funcionalidades al Kernel y agregar la biblioteca para manejos de hilos generada por el sistema operativo PSYCHE, como un elemento más de comparación, a pesar de que ésta no permite el uso de hilos de kernel.

Los primeros informes sobre el comportamiento entre diferentes elementos se muestran a continuación



4. FORMACION DE RECURSOS HUMANOS

Se realizó:

- la primera transferencia de los conocimientos obtenidos a los alumnos que cursan Sistemas Operativos, ya que realizaron el análisis de la arquitectura y las distintos formatos de ejecutables conjuntamente con el análisis del SODIUM e intervinieron en el desarrollo de los administradores.
- Transferencia de conocimientos a los alumnos de Sistemas de Computación II de la Universidad de La Matanza y a los alumnos de Sistemas Operativos de la Universidad Tecnológica Nacional, Regional Buenos Aires.
- Publicación de los avances en la investigación en dos congresos internacionales.
- Se prevé continuar con las publicaciones en otros congresos internacionales

Se está estudiando:

- el realizar convenios de colaboración con otras universidades nacionales estatales y privadas de las cuales recibimos ofrecimientos de colaboración, con el objetivo de intercambiar conocimientos y ampliar los alcances del sistema.

En esta línea de investigación tenemos:

- dos trabajos de la Maestría en informática en curso.
- colaboran también en el desarrollo un becario, un alumno de Ingeniería electrónica y un alumno que cursa los primeros años de la carrera de Ingeniería en informática.
- Recibimos la colaboración además de alumnos que ya han cursado la materia y han desarrollado parte del SODIUM en los años anteriores.

5. BIBLIOGRAFÍA

Publicaciones

[ANG98] Angulo José M. y Funke Enrique – Microprocesadores avanzados 386 y 486 – Introducción al Pentium y Pentium – Pro Editorial Paraninfo – Cuarta Edición

[BRE00] BRE00- Brey Barry B. – Los Microprocesadores Intel – Editorial Prentice Hall – Quinta Edición.

[CAR01] Card Rémy, Dumas Eric, Mével Franck - Programación Linux 2.0 API del sistema y funcionamiento del núcleo – Enrolles y Ediciones Gestión 2000 S.A.

[DEI90] Deitel Harvey M. – Introducción a los Sistemas Operativos - Addison-Wesley Iberoamericana – Segunda Edición

[INTEL] Manual de microprocesadores 386 y 486 y Pentium.

[KNUT] Donald E. Knuth. Fundamental Algorithms, The Art of Computer Programming Vol. 1, Addison Wesley, Second Edition,

[MCD07] MacDougall, Mauro, Solaris Internals (Solaris 10 and opensolaris kernel architecture) –Editorial Prentice Hall – Second Edition

[MIL94] Milenkovic Milan – Sistemas Operativos Conceptos y diseño – Mc Graw Hill – Segunda edición

[SIL97] Silverschatz, Avi; Galvin, Peter – Operating System Concepts – Addison-Wesley Longman – Fifth Edition

[SMC00] Standard Microsystems Corporation – Application note 6.12

[STA98] Stallings Willams – Operating Systems Internals and design principles – Prentice Hall International – Third Edition

[TAN97] Tanenbaum Andrew S., Woodhull Albert S. – Operating Systems Design and Implementation – Prentice Hall – Second Edition

[TAN03] Tanenbaum Andrew S.– Sistemas Operativos Modernos – Pearson Education – Segunda Edición

[TISELF] Executable and Linking Format (ELF) Specification – Tool Interface Standard (TIS) Committee

[TUR03] Turley James L. – Advanced 80386 programming techniques – Osborne McGraw Hill

[VHA06] William Von Hagen – The Definitive Guide to GCC – Apress – Segunda Edición

[RUS00] Russinovich, Salomon – WINDOWS INTERNALS Windows Server 2003, Windows XP, Windows 2000 – Microsoft Press

Revistas , Publicaciones e Internet

[01] “First-Class User-Level Threads”. Brian D. Marsh, Michael L. Scott, Thomas J. LeBlanc, Evangelos P. Markatos. Computer Science Department. University of Rochester.

[02] “Avoiding Blocking System Calls in a User-Level Threads Scheduler for Shared Memory Multiprocessors”. Andrew Borg. University of Malta.

[03] “Scheduler Activations: Effective Kernel Support for the User-Level Management of Parallelism”. THOMAS E. ANDERSON, BRIAN N. BERSHAD, EDWARD D. LAZOWSKA, and HENRY M. LEVY. University of Washington.

[04] “Integrating Kernel Activations in a Multithreaded Runtime System on top of Linux”. Vincent Danjean, Raymond Namyst and Robert D. Russell. University of New Hampshire.

[05] “Page-based optimistic concurrency control for memory-mappedpersistent object systems”. Inohara, S. Shigehata, Y. Uehara, K. Miyazawa, H. Yamamoto, K. Masuda, T. University of Tokyo.

[06] Manual Document GNU Pth http://www.gnu.org/software/pth/pth-manual.html Ralph Engelschalt