Diseño e Implementación de múltiples planificadores de disco para el sistema operativo Linux

J. Fernández, F. García, J. Carretero
Depto. de Arquitectura y Tecnología de Sistemas Informáticos
Universidad Politécnica de Madrid,
28660 Madrid, España
e-mail: {jfernand,fgarcia,jcarrete}@laurel.datsi.fi.upm.es

Octubre 1999

Resumen

El uso creciente de sistemas operativos de propósito general, como Linux, para dar soporte a aplicaciones de propósito específico, como las de multimedia, vídeo bajo demanda o tiempo real no crítico, hace necesario introducir en dichos sistemas operativos mecanismos de control de la entrada/salida que permitan asignar los recursos de forma adecuada. Entre estos mecanismos, el sistema de ficheros y los planificadores de disco son elementos clave a modificar.

En este articulo se presenta el diseño e implementación de mecanismos para cambiar de forma fácil y rápida el planificador de disco del S.O. Linux. Dichos mecanismos se acompañan de una propuesta de modificación del interfaz de E/S para permitir peticiones al disco con parámetros adicionales que puedan usar los nuevos planificadores de disco. Para demostrar la viabilidad de las soluciones propuestas, se presenta también una evaluación comparativa de algunas políticas de planificación implementadas a modo de ejemplo.

Palabras Clave: E/S, planificación, disco, Linux.

1 Introducción

En los últimos años se se ha despertado un gran interés en el campo de la arquitectura de los computadores por los métodos de planificación [Ghandeharizadeh95] de los dispositivos de entrada/salida. Las técnicas genéricas de planificación usadas y desarrolladas dentro de los sistemas de tiempo real [Lehoczky89][Liu73] funcionan adjudicando con antelación las prioridades de los distintos procesos. Sin embargo en el caso de la planificación en los dispositivos de entrada/salida esta prioridad no puede ser conocida con antelación, así la prioridad de cada flujo de datos debe ser calculada de forma dinámica. Los métodos de planificación más usados incluyen las técnicas FIFO, SCAN y CSCAN. Ninguno de ellos permite especificar parámetros adicionales procedentes de los programas de usuario, puesto que sólo dependen de la geometría del disco. Ejemplos de técnicas de planificación basadas en parámetros adicionales procedentes de los programas de usuario pueden ser la planificación basada en prioridades, en plazos de respuesta o en políticas mixtas de planificación [Carretero99-2]. El

paso de parámetros adicionales desde los programas de usuario hasta el planificador de disco implica, en muchos casos, modificar el interfaz de entradas/salida del sistema.

En este articulo se describe una implementación de Linux [Beck98, Rusling98] que permite al usuario cambiar de forma dinámica el planificador de disco del sistema y que, además, expande el interfaz de entrada/salida para incluir parámetros adicionales en las peticiones al disco de los programas de usuario. En la sección 2 se describe el diseño del sistema. La sección 3 muestra someramente la implementación realizada. La sección 4 muestra una evaluación comparativa de algunos algoritmos de planificación de disco. Por ultimo, en la sección 5 se exponen las conclusiones extraídas y los futuros trabajos a realizar.

2 Diseño del sistema

Para poder describir adecuadamente los puntos de que consta el diseño del sistema es importante empezar con una breve introducción al funcionamiento del subsistema de E/S en Linux [Beck98], para después comentar los cambios a realizar para la consecución del sistema final.

A modo de reseña, indicar que Linux es una sistema operativo similar a UNIX desarrollado por miles de voluntarios en todo el mundo conectados vía Internet. La primera versión de este sistema operativo fue publicada en la red por Linus Torvald en 1991. Actualmente hay millones de computadores con Linux instalado, con una previsión de crecimiento anual del 25% durante los próximos años. Su principal ventaja es que su código fuente es de libre distribución, tanto para estudiarlo, como para modificarlo o distribuirlo sin limitaciones [Card97].

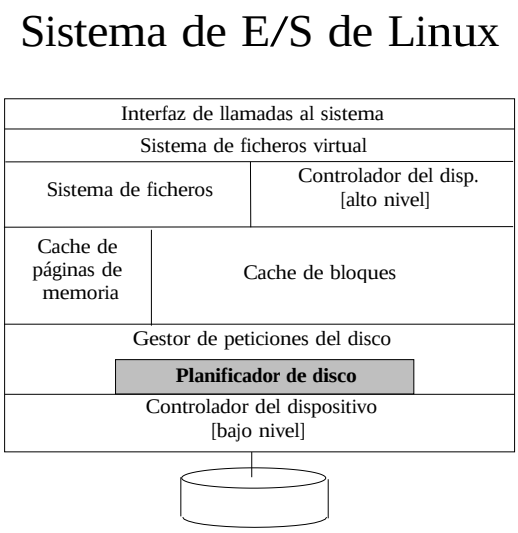

Figura 1: Arquitectura del subsistema de E/S en Linux

La Figura 1 expone el esquema básico del subsistema de E/S de Linux. En ella se pueden distinguir las distintas partes que lo componen. Entre estas partes hay que

destacar el gestor de peticiones al disco y el planificador del disco. El gestor de peticiones al disco se encarga de recoger las peticiones de lectura o escritura en forma de bloques de cache listos para transferir datos desde o hacia el disco. Además se encarga de juntar las transferencias del mismo tipo, cuyos bloques sean contiguos, para que se realicen como una sola petición. Cuando tiene una nueva petición, llama al planificador de disco para que éste la introduzca de forma ordenada en la cola de peticiones. En Linux existe una cola de peticiones por cada tipo de dispositivo. El planificador instalado en Linux por defecto implementa una política de planificación de tipo CSCAN.

Para poder diseñar un sistema que cumpla los objetivos previamente marcados hay que resolver dos problemas fundamentales:

- Descubrir un método que permita cambiar de planificador de disco de la forma menos traumática posible.
- Modificar la interfaz de E/S de Linux para permitir que las peticiones al disco incluyan parámetros adicionales a los habituales.

Para resolver el problema planteado en el primer punto existen dos posibles soluciones.

- Incorporar el nuevo planificador dentro del código fuente que compone el núcleo principal de Linux.

  Esta solución es la más sencilla de implementar. Su principal desventaja, sin embargo, es la dificultad que impone a la hora de incorporar nuevos planificadores de disco al sistema. Para realizar nuevas incorporaciones es necesario recompilar todo el núcleo y reiniciar el sistema completamente.

- Modificar el núcleo de Linux para permitir la incorporación de nuevos planificadores de disco en forma de módulos externos [Pomerantz98].

  Para conseguir esta funcionalidad nos basamos en la capacidad que tiene Linux desde su versión 2.0.X para incluir parte del código del núcleo en módulos compilados de forma independiente y que luego son instalados en el sistema mientras este está en funcionamiento. Su principal ventaja consiste en poder efectuar todo el proceso de desarrollo de los nuevos planificadores de disco de forma independiente del resto del código de Linux e incluirlos después en el sistema sin tener que reiniciarlo. Para que esto sea posible es necesario modificar el núcleo de Linux para que admita la inclusión de nuevos planificadores a partir de módulos externos.

La opción que se ha elegido para implementar el sistema ha sido la segunda. Tal como muestra la figura 2 de esta forma es posible tener los planificadores de disco desarrollados fuera del sistema e incluirlos sólo cuando sea necesario. Además, y dado que la planificación de las peticiones de E/S es una tarea indispensable del sistema, es necesario incluir en el sistema un planificador de disco por defecto para cuando no se haya especificado ningún otro planificador. Este planificador sera el mismo que incluye Linux por defecto, que usa la política CSCAN.

En cuanto al segundo problema a tratar, modificar la interfaz de E/S para permitir peticiones al disco con parámetros adicionales, hay que tener en cuenta los siguientes puntos.

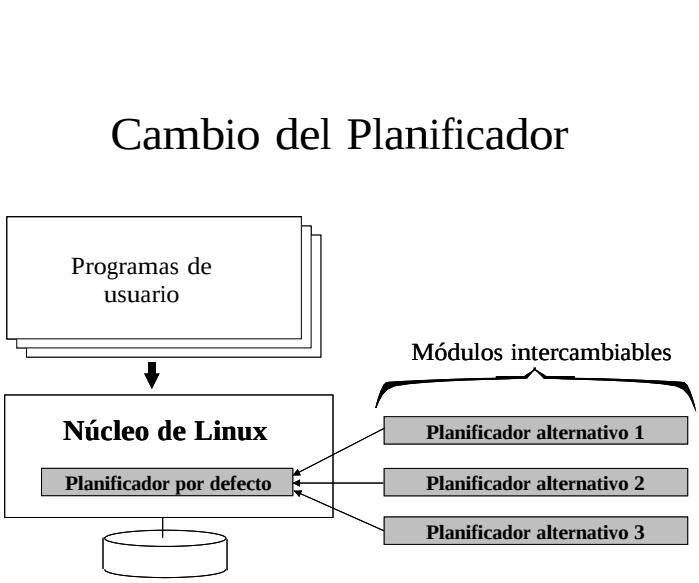

Figura 2: Esquema de inclusión de planificadores alternativos en el sistema

- En primer lugar, las peticiones de E/S de los programas de usuario deben ser realizadas a través de llamadas al sistema. Para la realización de peticiones con parámetros adicionales se ha optado por sobrecargar la llamada al sistema `ioctl`. Esta llamada se usa generalmente para operaciones de control sobre los dispositivos físicos. La interfaz de que dispone la llamada ioctl es muy versátil, pudiendo implementarse con ella cualquier tipo de llamada. Nuestra idea es implementar dos nuevas funcionalidades a la llamada `ioctl`: una de lectura y otra de escritura. Además de los parámetros de lectura y escritura habituales, estas funciones incorporan los parámetros adicionales necesarios para controlar las nuevas políticas de planificación de disco.

- Otro aspecto a tener en cuenta es cómo pasar los parámetros extra a través de la cache del sistema. La forma más fácil es incluir estos parámetros en la estructura de control de cada bloque de cache. De esta forma, cuando un bloque de cache sea enviado al gestor de peticiones al disco y éste lo pase al planificador de disco, se podrán consultar todos estos parámetros adicionales y realizar la planificación en función de estos datos. Es importante destacar que cuando un bloque de cache está asociado a una petición de E/S sin parámetros adicionales, su estructura por defecto debe incluir un valor por defecto para estos parámetros adicionales.

3 Implementación del sistema

A la hora de realizar la implementación de un sistema como este hay que considerar que partes es necesario incluir dentro del fichero principal del núcleo de Linux y cuáles se pueden implementar como módulos independientes.

En el primer grupo se encuentran los siguientes cambios:

- Modificación de la estructura de control de los buffers de la cache de Linux para que incluyan parámetros extra. Además se han incluido funciones para que la cache pueda tratar con estos parámetros adicionales. Estos cambios se han realizado tanto para la cache de bloques como para la cache de páginas de memoria.
- Modificación del gestor de peticiones al disco para que pueda tratar adecuadamente los bloques de cache con parámetros adicionales, sobre todo a la hora de agruparlos en peticiones más grandes.

```
Planificador_de_disco(petición_de_E/S)
{
        planificador_nuevo = get_module_symbol("modulo, "función");
        if (planificador == NULL) {
                planificador_por_defecto (petición_de_E/S);
        } else {
                planificador_nuevo (petición_de_E/S);
        }
}
```

Figura 3: Algoritmo para la detección de un planificador alternativo de disco.

- Modificación del planificador de peticiones al disco para que detecte si existe otro planificador cargado en el sistema y en caso afirmativo pasarle el control. La figura 3 muestra el algoritmo de detección. Cuando llega una nueva petición se comprueba si la función del planificador alternativo esta en el sistemas. Si está, se usa ese planificador. En caso contrario se usa el planificador por defecto.

En cuanto al segundo grupo, se han realizado los siguientes módulos:

- Modulo del sistema de ficheros modificado. Este módulo implementa un sistema de ficheros basado en el `ext2` de Linux que, además de las funciones normales, permite realizar peticiones de E/S con parámetros adicionales por medio de la llamada al sistema `ioctl`.
- Módulo del dispositivo físico genérico. Este módulo permite realizar peticiones con parámetros adicionales a cualquier controlador de dispositivo físico sobre el cual se coloque.
- Módulo del planificador de disco. Se han construido 4 módulos de planificación de disco para los métodos de planificación FIFO, SCAN, CSCAN y un nuevo método basado en prioridades. Como se muestra en la figura 4, el nuevo método de planificación es asimilable al uso de múltiples colas: una para cada valor posible de la prioridad. Estas colas se ordenadas internamente según el método

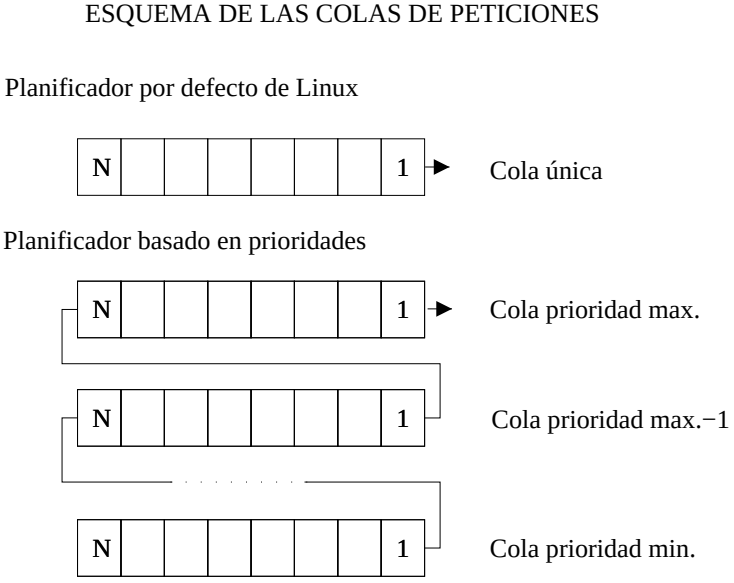

Figura 4: Esquema para las cola de peticiones según el planificador.

CSCAN. Además, las peticiones de una cola sólo serán realizadas cuando las colas de mayor prioridad estén vacías.

4 Evaluación del sistema

Una vez construido el sistema, y como forma de probar su funcionamiento, se ha realizado una pequeña comparativa entre los métodos de planificación implementados como ejemplo. Para realizar esta comparativa se han realizado dos pruebas distintas:

- En primer lugar se realizó una prueba consistente en múltiples procesos que realizan simultáneamente, y de forma exhaustiva, lecturas y escrituras al disco variando cada vez el método de planificación usado y se comprobó cual era el ancho de banda total que se conseguía extraer del disco. La gráfica de la figura 5 muestra los resultados obtenidos. Como se puede comprobar el método que peor resultado arroja es el método FIFO. Esto es así porque este método no realiza ninguna optimización del movimiento de cabezas del disco. Después siguen los métodos SCAN y CSCAN. El que mejor rendimiento proporciona es el método basado en prioridades, debido a que los proceso con prioridad más elevada copan por completo el ancho de banda del disco. Así, al ser muy pocos los procesos efectivos, el hardware del disco SCSI es capaz de optimizar mejor el ancho de banda del dispositivo. La figura 6 muestra la diferencia en el ancho de banda de los procesos con mayor prioridad comparándolo con los de menor prioridad, los cuales casi no acceden al disco.
- Además de la prueba anterior, se ha realizado una prueba con clientes de vídeo, situados en distintos ordenadores, que acceden al mismo servidor de películas

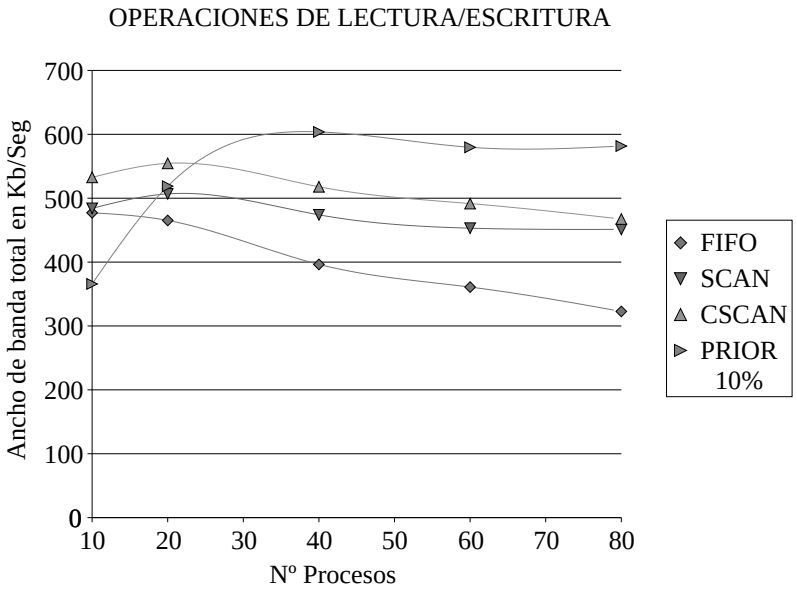

Figura 5: Resultados de la prueba de lecturas y escrituras exhaustivas en disco para cada una de las políticas de planificación.

en formato MPEG mediante una capa de comunicaciones basada en sockets. Esta prueba se realizó para cada uno de los métodos de planificación de disco implementados. En la figura 7 se ven los resultados obtenidos con el planificador basado en prioridades. En ella se puede comprobar como el cliente con mayor prioridad consigue un mayor número de imágenes por segundo que el resto, si bien el resultado no es tan espectacular como en la prueba anterior debido al tiempo de cálculo que consumen los clientes de vídeo y a las interferencias que produce la red de comunicaciones.

5 Conclusiones

En este artículo se ha descrito un sistema que permite el desarrollo y la inclusión de nuevos planificadores de disco dentro del sistema operativo Linux de una forma fácil y rápida, con vistas a la creación y experimentación de nuevos planificadores de disco. Además se ha modificado la interfaz de entrada/salida en Linux para permitir peticiones de E/S con parámetros adicionales que puedan ser usados por el planificador de disco y así poder realizar planificadores de disco basados en prioridades, en plazos de respuesta, etc. A modo de ejemplo se implementado distintas políticas de planificación y se han evaluado para compararlas entre si. Los resultados de la evaluación demuestran que una política de prioridad de mejor rendimiento y permite distinguir entre distintos tipos de aplicaciones, dando los recursos a quien más los necesita.

A partir de ahora los objetivos se centran en la realización de nuevos planificadores de disco que puedan ser eficaces para entorno tan específicos como los sistemas multimedia y los sistemas de vídeo bajo demanda [Carretero99-1]. Estos nuevos planifi-

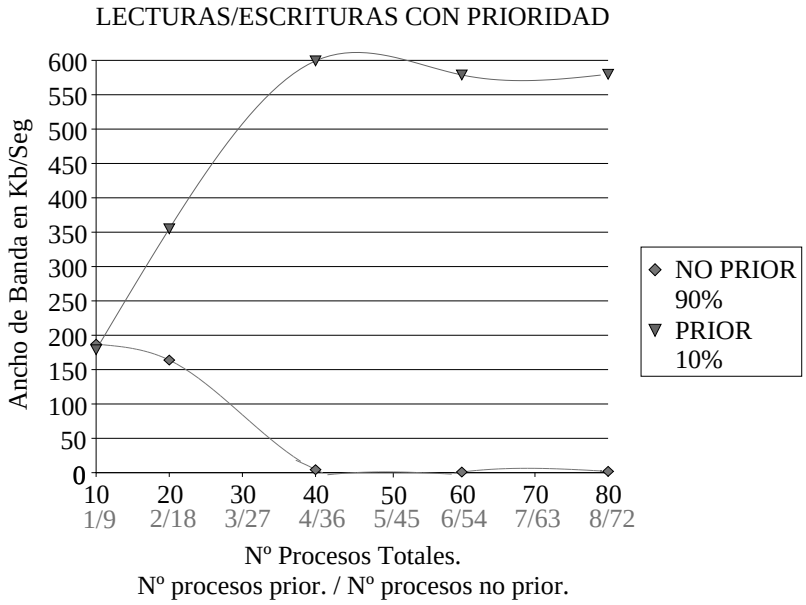

Figura 6: Resultados de la prueba de lecturas y escrituras exhaustivas en disco usando la política de planificación basada en prioridades.

cadores se besarán en los plazos de respuesta de las peticiones y la prioridad de las mismas.

Referencias

[Beck98] Beck M., Böhme H., Dziadzaka M., Kunitz U., Magnus R., Verworner D. Linux Kernel Internals. Second Edition. Ed: Addison-Wesley, 1998

[Card97] Card R., Dumas E., Mével F. Programación Linux 2.0, API del sistema y funcionamiento del núcleo. Ed: Eyrolles-Gestión 2000, 1997

[Ghandeharizadeh95] Ghandeharizadeh S., Kim S.H., Shahabi C. On configuring a single disk continuous media server. ACM SIGMETRICS '95, pages 37-46, 1995

[Lehoczky89] Lehoczky J., Sha L., Ding Y. The rate monotonic scheduling algorithm exact caracterization and average case behavior. Proc. of the IEEE RealTime System Symposium, Pages 166-171, 1989

[Liu73] Liu C., Layland, J. Scheduling algorithms for multiprogramming in a hard real time enviroment. Journal of the ACM, 20(1):46-61, Enero 1973

[Pomerantz98] Pomerantz O. Linux Kernel Module Programming Guide. Linux Documentation Project, 1998

[Rusling98] Rusling David A. The Linux Kernel. Linux Documentation Project, 1998

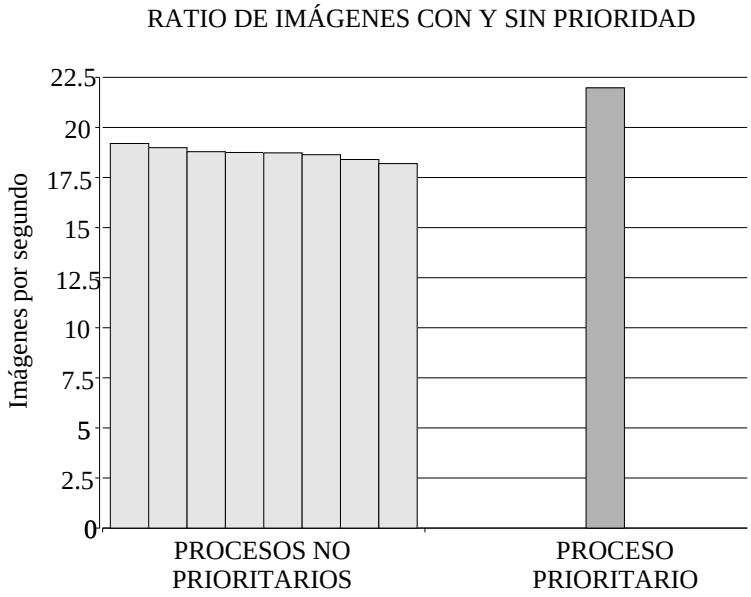

Figura 7: Resultados de cada uno de los clientes de vídeo ejecutados con la política de planificación basada en prioridades.

[Carretero99-2] Carretero J., Zhu W., and A. Choudhary A. Design and Evaluation of a Multimedia Integrated Parallel File System IEEE International Conference on Multimedia Computing and Systems ICMCS'99, Florence, Italy, June 7-11, 1999.

[Carretero99-1] Carretero J., Zhu W., and Choudhary A. Hierarchical Scheduling for Disk I/O in an Integrated Environment ISCA 14th International Conference on Computers and Their Applications, Cancún, Mexico, April 7-9 1999.