Aprendiendo Multiprocesamiento Simétrico con Minix

Álvarez Llorente, J.M.; Díaz Martín, J.C; Rodríguez García, J.M.; Rico Gallego, J.A.

Dpto. Informática, Universidad de Extremadura, Cáceres, Spain,
llorente@unex.es, juancarl@unex.es, jmrodri@unex.es, jarico@unex.es
http://gsd.unex.es

Resumen. Minix SMP es una extensión del sistema operativo Minix 2.0.0 sobre una arquitectura Intel de multiprocesamiento simétrico. Minix SMP constituye una herramienta docente, un ejemplo real, que facilita el aprendizaje de las nuevas arquitecturas de computadores y los sistemas operativos que las explotan. La experiencia adquirida en su desarrollo demuestra que la mayor dificultad estriba en la familiarización con el hardware multiprocesador subyacente ya que éste se integra con naturalidad en el microkernel Minix. Proponemos, pues, una asignatura de diseño de sistemas operativos multiprocesador basada en Minix SMP.

1 Motivación y objetivos

Los sistemas operativos modernos deben dar soporte a las nuevas arquitecturas formadas por múltiples procesadores. En los últimos 5 o 6 años se han venido popularizando y abaratando las arquitecturas SMP (Symmetric Multiprocessing, Multiprocesador Simétrico), basadas en múltiples procesadores con memoria compartida. Si bien podemos encontrar referencias a esta arquitectura en cualquier bibliografía actual sobre sistemas operativos, lo cierto es que es difícil dar con textos que, como el de Schimmel [1], abarquen con cierta profundidad y criterio didáctico lo referente al diseño de sistemas operativos que soporten tales arquitecturas, y los manuales técnicos ayudan poco.

Es cierto que el estudio de Linux SMP puede ayudar en este sentido, pero consideramos que no es el caso de estudio idóneo en un curriculum universitario. No es la herramienta docente más adecuada porque que no es ese el objetivo con el que fue creado. Es innegable el amplio despliegue de páginas de documentación, foros de discusión, etc., que hay disponibles para todos los aspectos de Linux y en todos los idiomas, pero quizá el carácter didáctico de Linux se pierde, víctima de su propio éxito como sistema operativo a la altura de los sistemas comerciales.

Dentro del código de Linux encontramos la solución software a cada reto que plantean las nuevas arquitecturas del hardware. El problema es que cada una de ellas forma parte de un todo difícil de dividir, fuertemente acoplado, y con unas interrelaciones complejas, las propias de un sistema monolítico y multiplataforma. Multiprocesamiento, interrupciones memoria cache, memoria virtual, coprocesador matemático, extensiones multimedia, contro-

ladores de dispositivos, mecanismos de ahorro de energía, etc. se entremezclan en el núcleo de Linux, rodeados de complejas directivas de compilación condicional, que dan lugar a un sistema adecuado para su explotación comercial, pero poco adecuado para ser estudiado.

La distribución estándar de Minix es un sistema básico y, por tanto extensible: no proporciona soporte multiprocesador, ni de memoria virtual, ni de extensiones multimedia, etc. Muchas otras iniciativas han abordado algunas ampliaciones, entre las que podemos destacar RT-Minix, una extensión para tiempo real sobre Minix [3] y Minix-VMD, una extensión sobre Minix con soporte para memoria virtual [4]. Todo ello nos ha llevado a plantear Minix como un entorno de aprendizaje perfectamente válido para el estudio de la arquitectura de multiprocesamiento simétrico.

Abordamos en este trabajo la extensión de Minix a la arquitectura multiprocesador de Intel [5], lo que llamaremos Minix SMP. En su desarrollo hemos perseguido como principal objetivo obtener una herramienta docente de calidad con las siguientes premisas:

- Mantener la estética y la estructura original de Minix lo más intacta posible, de manera que la frontera entre Minix y la extensión SMP quede perfectamente definida.
- Documentar adecuadamente todas las modificaciones introducidas, así como el trasfondo de las mismas.
- Construir un sistema multiprocesador funcional a la par que básico, abierto a modificaciones, ampliaciones y mejoras.
- Diseñar el sistema de manera que sea escalable con el número de procesadores y pueda adaptarse fácilmente a diferentes variaciones del hardware, manteniendo la compatibilidad con el sistema original, de manera que cualquier programa que funcione en Minix, también lo haga en Minix SMP.

2 La experiencia adquirida en la construcción de Minix SMP

La especificación multiprocesador de Intel [5] es muy adecuada para iniciarse en el terreno del multiprocesamiento, debido a que la mayor parte de los problemas que plantean estas arquitecturas están resueltos por parte del hardware. La gestión de memoria compartida y la coherencia de cache, por ejemplo, son prácticamente transparentes al software.

A grandes rasgos, podemos enunciar que la única diferencia entre un monoprocesador de Intel y un multiprocesador es la presencia de un sistema avanzado de gestión de interrupciones. Éste es el encargado de repartir las señales de interrupción entre los distintos procesadores del sistema, a través de un bus dedicado, gestionado por uno o más controladores avanzados (I/O APIC) junto con los controladores locales presentes en cada CPU (Local APIC), todo ello complementado por la labor del antiguo controlador de interrupciones (PIC) de la arquitectura tradicional Intel. La figura 1 ilustra este esquema [5].

Minix es un sistema operativo micro-núcleo, por lo que los procesos del sistema se gestionan de la misma forma que los de usuario. Los procesos se organizan en tres niveles de prioridad: las tareas de E/S, servidores y procesos de usuario. El núcleo sólo soporta el paso de mensajes entre procesos, la planificación, la gestión de interrupciones, y, por supuesto, todo lo referido al arranque y detención del sistema.

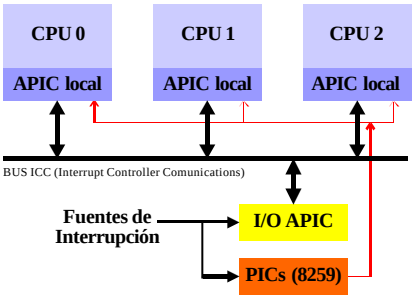

Fig. 1. Sistema de gestión de interrupciones en la arquitectura multiprocesador de Intel

Minix tiene un núcleo reentrante. Esto quiere decir que la elevación de una interrupción hardware durante la ejecución del código del núcleo provoca una reentrada. En ella se anota el evento para ser atendido tan pronto finalicen la entrada y posibles reentradas anteriores. Las secciones críticas del núcleo son los métodos del planificador y la programación del controlador de interrupciones. Están protegidas en Minix de los efectos de la reentrada mediante la inhabilitación de interrupciones. Lograr que el núcleo multiprocesador siga siendo reentrante ha requerido algunas de las modificaciones más complejas.

Para cumplir los objetivos propuestos, la implementación del núcleo multiprocesador se ha realizado procurando siempre la mínima modificación de la estructura y organización del núcleo original de Minix (figura 2).

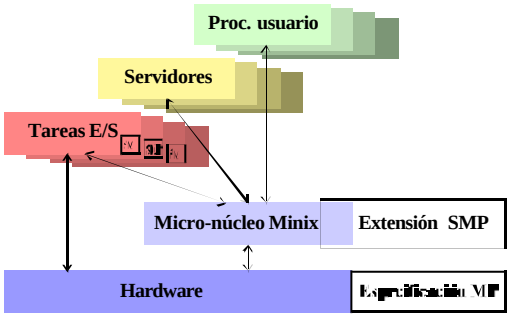

Fig. 2. Arquitectura de la extensión multiprocesador. Únicamente afecta al micro-núcleo y a algunas de las tareas de E/S.

Hemos introducido pequeñas modificaciones sobre el código fuente, aunque también ha sido necesario escribir código nuevo para añadir la interfaz con el hardware multiprocesador. No es necesaria modificación alguna sobre lo procesos de usuario ni los servidores, y tan sólo algunos pequeños ajustes son requeridos en las tareas de E/S más ligadas al núcleo. El punto de partida ha sido Minix 2.0.0, primera versión conforme POSIX 1003.1a ([2]).

Existe una descripción completa en castellano de todas las modificaciones y ampliaciones. La mayor parte del nuevo código implementado es el necesario para realizar las funciones de obtención de información sobre el sistema multiprocesador, manipulación del sistema de interrupciones, arranque y detención del modo multiprocesador, y las primitivas de sin-

cronización global entre procesadores. Los cambios en el núcleo se reducen a la solución de dos problemas: la necesidad de replicar algunos recursos para los distintos procesadores (pila, variables globales, etc.) y la necesidad de sincronización en dos puntos críticos: la entrada al núcleo y el planificador.

En multiprocesamiento simétrico el reparto de tareas entre procesadores debe ser distribuido, de manera que ningún procesador se dedique a asignar trabajo a los demás [5]. La idea aplicada en Minix SMP es que cada procesador trabaje en el sistema sin tener conocimiento del estado de los demás procesadores, compitiendo con ellos por realizar su trabajo: ejecutar procesos. Así, únicamente nos preocuparemos, primero, de que varios procesadores no intenten ejecutar las mismas tareas al mismo tiempo, para lo cual adaptamos el planificador; y segundo, de que los procesadores no ejecuten código crítico a la vez, para lo cual estableceremos unas primitivas de sincronización basadas en cerrojos globales (*spinlock* [1], [6]) alrededor de las secciones críticas antes identificadas.

La entrada al núcleo es uno de los puntos más conflictivos, al complicarse el control de reentradas debido a la concurrencia de los múltiples procesadores. La solución adoptada consiste en permitir múltiples reentradas por parte de un mismo procesador, bloqueando los intentos de entrada por parte del resto de procesadores, hasta que el primero abandone el núcleo. Esto ocasiona una detención de los procesadores que esperan, pero consideramos que éstas serán presumiblemente cortas debido al diseño micro-núcleo, y por lo tanto, tolerables.

El arranque del sistema se realiza en dos fases. En primer lugar se produce el arranque normal del sistema por parte de un procesador (BSP, *bootstrap processor*), estando los demás (AP, *application processors*) inhabilitados. Cuando el BSP se encuentra en condiciones de comenzar a ejecutar tareas, habilita consecutivamente cada uno de los AP hasta dejarlos en el mismo estado que el primero. Entonces, todos comienzan a competir por ejecutar tareas.

Finalmente, el sistema de interrupciones se programa de manera que cualquier procesador puede ser el destino de una interrupción hardware.

El desarrollo del nuevo sistema se han realizado sobre una arquitectura con 2 procesadores Pentium III MMX a 500 MHz (figura 3).

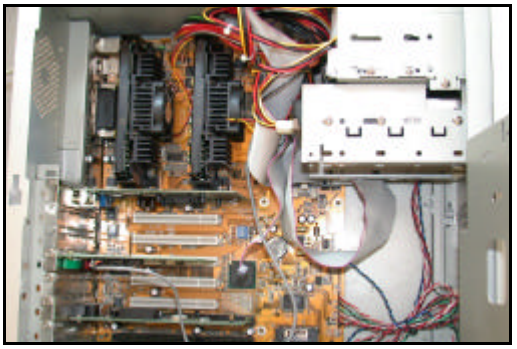

Fig. 3. Placa Gigabyte GA-6BXD con 2 procesadores Intel Pentium III MMX.

Todos estos detalles de la estructura de Minix SMP se desprenden de la experiencia en la construcción real del mismo, y por esa razón constituyen una descripción precisa de cómo

construir sistemas operativos multiprocesador. Por consiguiente, creemos que son un fundamento válido para el planteamiento de una asignatura de diseño de sistemas operativos multiprocesador, de la que trataremos en el siguiente apartado.

3 Aplicación docente

Hace ya más de una quincena de años de la publicación de la primera edición de *Operating Systems, Design and Implementation* [2], sin duda alguna, el referente de estudio por excelencia para la docencia de Sistemas Operativos. En él, A.S. Tanenbaum nos demuestra que el caso real de estudio es la mejor vía para el aprendizaje. Aparte de la herramienta ideal para el estudio de los contenidos del libro, Minix es un excelente punto de partida para abordar otros contenidos más avanzados.

El abaratamiento y popularidad de que vienen gozando durante los últimos años las arquitecturas multiprocesador nos invitan a ir incluyendo su estudio en los programas de las asignaturas de sistemas operativos. A pesar de ello, no es fácil encontrar referencias bibliográficas adecuadas sobre estas arquitecturas, a excepción, por supuesto, de los nada didácticos manuales proporcionados por los fabricantes.

Esto representa un notable inconveniente para la docencia de asignaturas de diseño de sistemas operativos. Analizando los programas actuales de estas asignaturas en distintas universidades, observamos que gran parte de ellas optan por sumergirse en el estudio del sistema operativo Linux, una opción interesante, dada la enorme y creciente popularidad del mismo. Sin embargo, pensamos que Linux como herramienta docente se aleja del planteamiento de simplicidad de A.S. Tanenbaum. Es cierto que en el núcleo monolítico de Linux están resueltos todos las cuestiones que afectan a las nuevas posibilidades del hardware, y que podemos "aprender cómo está hecho", pero la complejidad del sistema, por su concepción monolítica, y por el propio hecho de que "lo comprende todo, todo está resuelto", es muy alta. Creemos que es más adecuada una herramienta que permita centrarnos en problemas concretos. Además, según nuestra experiencia, es más didáctico resolver problemas "creando la solución" que limitarse a "aprender de las soluciones de otros". Estas consideraciones nos llevan a seguir apostando por Minix como herramienta para la docencia de diseño de sistemas operativos, y Minix SMP para abordar las arquitecturas multiprocesador.

Al igual que ocurre en Minix, la versión SMP es una solución básica y lo más simple posible. Puesto que se ha mantenido la estructura original de Minix, el paso a SMP no requiere empezar de cero. La documentación que se ha generado durante el desarrollo de Minix SMP es la guía ideal para abordar ese paso sin invertir mucho tiempo ni esfuerzo, que se puede aprovechar para practicar todas aquellas cuestiones que nos brinda la teoría sobre sistemas multiprocesador. Por ejemplo, podemos considerar las siguientes actividades, partiendo de Minix SMP:

– Aplicación de nuevas políticas de planificación, introduciendo diversas mejoras sobre el código del planificador. Minix SMP hereda el planificador de Minix prácticamente sin modificación: cada procesador compite con los demás, sin más, por ejecutar procesos. Resulta de especial interés la implementación y evaluación de políticas estudiadas en teoría, como la afinidad de procesadores. El incremento en el rendimiento que puede obtenerse con esta mejora podría verse empañada por la sobrecarga que se introduce en un punto

tan crítico como el planificador. Otros casos de estudio pueden ser la implantación de un planificador que localice procesadores ociosos, que realice un reparto no simétrico de tareas, que reserve procesadores para tareas determinadas, etc.

- Evaluación del impacto de la cache en el rendimiento, teniendo en cuenta las posibles modificaciones de planificación propuestas anteriormente, especialmente la afinidad de procesadores.
- Evaluación de la escalabilidad del sistema, una vez más, habida cuenta de las distintas posibilidades de planificación que se plantean. Se plantea también la evaluación de la sobrecarga introducida por la gestión de multiprocesamiento, comparando el rendimiento del sistema Minix original y una versión SMP en una arquitectura con sólo un procesador.

Quizá el mayor inconveniente que se pueda aducir para la utilización de Minix SMP sea la dificultad para disponer de un laboratorio con hardware multiprocesador real. En este sentido, el coste actual y el abaratamiento previsible de este tipo de computadores, hacen que esta idea parezca sensata: la adquisición de equipos sencillos con 2 procesadores (de última generación) frente a sus equivalentes con 1 procesador puede suponer un incremento en el presupuesto de unos 500 euros por equipo (aproximadamente un 40-50% sobre el coste de un equipo normal). Esta inversión puede ser rentabilizada compatibilizando el aprovechamiento de los laboratorios con otras disciplinas donde estas arquitecturas pueden ser especialmente explotadas, como pueden ser diseño CAD, montaje audiovisual, etc.

No obstante, Minix SMP se ha probado con éxito en el simulador de arquitecturas Bochs [10], por lo que, en el peor de los casos, es posible emplearlo sin la necesidad de invertir en un laboratorio de computadores multiprocesador.

Otra opción ampliamente extendida en la docencia de sistemas multiprocesador es la utilización de simuladores, como el LIMES [7]. No cabe duda de que los simuladores son una herramienta versátil, pero los casos reales siempre resultan más didácticos y atractivos.

3. 1 Propuesta de un programa para una asignatura de diseño de sistemas operativos multiprocesador

Gracias a la experiencia adquirida en la construcción de Minix SMP, hemos identificado los aspectos fundamentales en el diseño de los sistemas operativos multiprocesador, los cuales proponemos como base para el desarrollo de una asignatura para un curriculum de ingeniería informática, cuyos contenidos se recogen en la tabla 1. Se trata de una asignatura de 6 créditos (3 teóricos y 3 prácticos) que se cursaría como ampliación de otra previa de diseño de sistemas operativos.

Tema y contenidos (teoría)	Horas
Tema 1. La arquitectura de Minix 2.0 (Organización general; planificación; entradas y salidas al núcleo; arranque y detención del sistema).	4
Tema 2. Introducción a la arquitectura Intel MP (Introducción; tabla de configuración MP; APIC local; I/O APIC; instrucciones atómicas; el sistema de cache; procedimiento de arranque de procesadores de aplicación)	4

Tema 3. Introducción a la arquitectura de Minix SMP (protección del núcleo; planificador SMP; asignación de procesos a procesadores; protección del planificador; manipulación del hardware multiprocesador: APIC local, I/O APIC, CMOS; tabla de configuración MP, identificación de procesadores).	6
Tema 4. Gestión de interrupciones (Entrada y salida del núcleo: cambio de contexto; pila del núcleo; implementación de cerrojos; replicación de estructuras de datos; comunicación entre procesadores).	6
Tema 5. Planificación (Planificador multiprocesador; sincronización de las funciones del planificador; la tarea del reloj).	6
Tema 6. Arranque y detención del sistema (arranque de procesadores; habilitación e inhabilitación de procesadores; detención del sistema multiprocesador).	4
Prácticas propuestas	
Práctica 1. Recompilación del sistema multiprocesador (configuración para 1 procesador y medición de rendimiento; configuración para 2 procesadores y medición de rendimiento; comparación con el rendimiento usando el núcleo original monoprocesador).	4
Práctica 2. Creación de herramientas para la monitorización del sistema (adición de contadores de entradas y reentradas al núcleo, accesos al planificador, carga de CPU).	6
Práctica 3. Creación de un planificador con afinidad de procesadores (asignación estática de procesos a procesadores y medición de rendimiento; asignación dinámica a procesadores ociosos y medición de rendimiento; comparación del rendimiento frente a un sistema con un único procesador y frente al sistema original).	12
Práctica 4. Creación de un planificador no simétrico (creación de un planificador no simétrico y medición de rendimiento; comparación del rendimiento frente a un sistema con un único procesador y frente al sistema original).	6
Práctica 5. Impacto de la cache (Medición del rendimiento de todas las versiones de las prácticas anteriores con la cache inhabilitada).	2

Tabla 1. Programa para una asignatura de Diseño de Sistemas Operativos Multiprocesador

4 Conclusiones

Una vez construido Minix SMP, podemos afirmar que la extensión de un sistema micro núcleo como Minix a una arquitectura SMP ha resultado relativamente sencilla: bastan un conjunto de cambios sobre el código original de Minix y algo de código nuevo para tener el sistema funcionando. Un sistema sencillo, con modificaciones sencillas da como resultado otro sistema sencillo, fácil de entender, extender, modificar y aprender: una herramienta ideal para utilizar en docencia.

Las principales dificultadas se han debido al desconocimiento del funcionamiento preciso de la arquitectura hardware subyacente, lo que nos conduce a afirmar el interés de este

trabajo por la documentación que al respecto ha generado [8]. Así, el esfuerzo invertido se ha traducido en una inestimable experiencia que además podemos trasladar a otras arquitecturas (por ejemplo las basadas en múltiples procesadores DSP). Parecidas modificaciones pueden aplicarse a otros sistemas operativos para lograr multiprocesamiento. Basta compararlas con las efectuadas en su día sobre el núcleo de Linux para darse cuenta de ello. Los principios de la extensión de Minix SMP se están aplicando con éxito a la extensión del sistema VSTa [9] por parte de investigadores de la Bond University en Australia.

El sistema construido no es cerrado: sólo es una base sobre la que seguir trabajando. Sobre Minix SMP se pueden practicar, por ejemplo, diversas políticas de planificación con múltiples procesadores. Pero también es un sistema real, que funciona, sobre el que se pueden ensayar algoritmos a nivel de usuario. Minix SMP se ha probado en varias arquitecturas Intel y en el simulador Bochs [10], lo que posibilita, en último extremo, trabajar con Minix SMP sin hacer una inversión en hardware multiprocesador.

Por todas estas razones, proponemos una asignatura de diseño de sistemas operativos multiprocesador, basada en el estudio de Minix SMP como herramienta fundamental, cuyos contenidos se han derivado de la experiencia adquirida en la construcción de Minix SMP.

El código fuente de Minix SMP está disponible en Internet [11] para libre descarga, acompañado de una documentación completa en castellano [8].

Agradecimientos

Este trabajo ha sido financiado por el proyecto CICYT nº TIC99-0960.

Referencias

1. Schimmel, C. UNIX Systems for Modern Architectutes. Addison-Wesley, 1994.
2. Tanenbaum, A.S., Operating Systems, Design and Implementation, 2nd Edition. Prentice Hall, 1997.
3. Minix RT, http://www.sce.carleton.ca/faculty/ wainer/rt-Minix/rt-Minix.html.
4. Minix VMD, http://www.Minix-vmd.org.
5. Intel Corporation, Multiprocessor Specification Version 1.4. Intel Corporation, 1997.
6. Akl, S.G., Parallel Computation. Models and Methods, Prentice Hall, 1997.
7. Ikodinovic, I., Magdic, D., Milenkovic, A., Milutinovic, V., Limes: A Multiprocessor Simulation Environment for PC Platforms, Proceedings of Third International Conference on Parallel Processing and Applied Mathematics (PPAM), Kazimierz Dolny, Poland, September, 1999
8. http://webepcc.unex.es/~jalvarez/Minixsmp/Minixsmp.pdf
9. http://www.vsta.org.
10. http://bochs.sourceforge.net.
11. http://webepcc.unex.es/~jalvarez/Minixsmp