Modificaciones al kernel Linux para redes de alta velocidad

Javier Díaz
jdiaz@info.unlp.edu.ar

Matías Zabaljáuregui
matiasz@info.unlp.edu.ar

Laboratorio de Investigación en Nuevas Tecnologías Informáticas
Universidad Nacional de La Plata

Resumen

La combinación de nuevas tecnologías de redes de alta velocidad (con capacidades de transmisión en el orden de Gigabits por segundo, como Ethernet 10Gbps) y la capacidad de cómputo en los procesadores de última generación ha impulsado el nacimiento de nuevos paradigmas de procesamiento distribuido. Sin embargo, en esta evolución, el diseño de los sistemas operativos de propósito general y las arquitecturas de hardware más utilizadas en servidores de red y estaciones de trabajo han comenzado a dar señales de limitaciones con respecto a la escalabilidad.

El presente artículo menciona los factores que motivan la búsqueda de soluciones alternativas para el procesamiento del tráfico de red y presenta algunos de los nuevos proyectos originados, tanto en entornos académicos como en la industria, para resolver los problemas mencionados.

Finalmente, este trabajo propone una modificación al diseño del subsistema de red del kernel Linux 2.6 basada en ideas publicadas recientemente. Se intenta evaluar el efecto de hacer un uso asimétrico de los procesadores en una máquina con arquitectura SMP (o de los núcleos en un procesador con varios nucleos de ejecución) asignando un procesador (o un subconjunto de ellos) de manera exclusiva al procesamiento de red. De esta forma se intenta reducir el overhead conocido como intrusión del sistema operativo.

1. Introducción

No caben dudas de que el tráfico de red asociado a los servicios actuales mantendrá el crecimiento exponencial que lo ha caracterizado en los últimos años. A esto, deben agregarse las nuevas formas de aprovechar las últimas tecnologías en telecomunicaciones, tanto en entornos locales como en redes distribuidas geográficamente. A modo de ejemplo se pueden mencionar: web services, streaming de audio y video, servicios de voip, la tecnología RFID, variantes de cómputo distribuido y *Network Centric Computing* como procesamiento sobre clusters y grids, soporte para data centers y *Storage Area Networks* como *Ip Storage*, etc. Las tecnologías de capas inferiores ya ofrecen capacidades de transmisión de datos lo suficientemente rápidas y a costos tan razonables que están dejando de ser de ser sólo tecnologías de comunicaciones para ser usada también como tecnologías de backplane.

Durante años, mientras crecía la disponibilidad de ancho de banda y mejoraba la performance del hardware de red, el poder de cómputo ofrecido por las máquinas conectadas a la red, tanto clientes como servidores, era suficiente para realizar el procesamiento necesario para enviar y recibir paquetes sin interferir en la ejecución de los procesos de usuario. Sin embargo, las arquitecturas de hardware y el diseño de software del subsistema de red de las maquinas actuales han comenzado a dar señales de ciertas limitaciones con respecto a la escalabilidad.

2. Los problemas actuales

2.1. Accesos a memoria

Las arquitecturas de hardware de los servidores actuales no fueron diseñadas para el volumen de tráfico esperado en los próximos años. Tanto el bus del sistema como la memoria principal representan importantes cuellos de botella, agregando latencia y generando ciclos ociosos de la CPU.

Los datos, encabezamiento y descriptor de cada paquete, asi como los parámetros de control de la transmisión, son mantenidos en la memoria principal. Esto provoca que la CPU acceda **cinco veces a la memoria RAM para procesar un único paquete** [8] y cada acceso implica una gran cantidad de ciclos de CPU desperdiciados debido a las diferencias de velocidad entre la CPU y el acceso a memoria [1].

Existe una regla empírica generalmente aceptada que indica que se necesita 1 MHz de procesamiento de CPU para manejar 1 Mbps de tráfico de red (esta relación empeora a medida que aumenta la frecuencia de la CPU) [25]. Hasta hace unos años, las mejoras en performance de CPU y memoria se habían mantenido alineadas con el incremento del ancho de banda disponible, sin embargo en éste último tiempo han comenzado a notarse las limitaciones de las arquitecturas actuales marcadas principalmente por la latencia en el acceso a la memoria principal. A 10 Gbps, los paquetes arriban más rápido de lo que un sistema puede procesarlos [25, 28].

2.2. Computación de protocolos

Las pilas de protocolos actuales se basan en algoritmos y código creados en los primeros años del desarrollo del networking, con funcionalidad agregada a lo largo del tiempo. Esto resultó en diseños e imlementaciones no del todo eficientes, incrementando el tiempo y recursos necesarios para el procesamiento de cada paquete.

Numerosas publicaciones plantean que las pilas de protocolos utilizadas en los actuales sistemas operativos han quedado obsoletas [25, 28, 8, 11, 20, 27, 16]. Los problemas estructurales que se presentan en estas implementaciones fueron heredados, en gran medida, por decisiones de diseño volcadas en las primeras versiones de tcp/ip, pensadas para redes soportadas por medios físicos mucho menos confiables y con velocidades de transmisión que representan sólo una pequeña fracción de los estándares actuales. Además, si bien es cierto que se han agregado nuevas funcionalidades a los protocolos originales, en general, estas modificaciones al código se han implementado como parches al diseño original, resultando en una mayor latencia en el procesamiento. Todo indica que es tiempo de un nuevo diseño.

2.3. Overhead del sistema

Para cada paquete enviado o recibido se realizan actividades tales como manejo de buffers, transiciones de modo usuario a modo kernel y viceversa, schedulling y manejo de interrupciones. Estas actividades se implementan como threads que consumen ciclos de CPU y que deben ser switcheadas entre sí y con las aplicaciones de usuario, la pila de protocolos y otras funciones del sistema, provocando un fenómeno conocido como *intrusión del sistema operativo*.

La intrusión es un nombre nuevo para un concepto que existe desde que las computadoras basan su funcionamiento en un proceso monitor o sistema operativo, y representa el overhead introducido por éste al realizar sus funciones principales de virtualización y protección de los recursos de la máquina. Las interrupciones por hardware y las *system calls* implementadas como traps son un ejemplo de intrusión por mecanismo (intrusión relacionada con la implementación de las funciones del sistema operativo) [2]. Los sistemas multiprocesadores tienen más probabilidades de sufrir intrusión por mecanismo, ya que la mayoría de los sistemas operativos de pro-

[1] Mientras una CPU moderna puede ejecutar varias instrucciones por nanosegundo, el acceso a la cache tardará aproximadamente 50ns. Si el acceso es a la RAM, la latencia es notablemente superior.

[2] En los modelos tradicionales de drivers para interfaces de red, el procesador es interrumpido por cada paquete recibido. Sin embargo, las interfaces de alta velocidad pueden recibir miles de paquetes por segundo, generando miles de interrupciones (y, en consecuencia, miles de intercambios de tareas) por segundo. Para mejorar la performance de Linux en sistemas high-end, los desarrolladores del subsistema de red del kernel linux han creado una nueva interface, basada en la técnica de *polling*, que fue bautizada como NAPI (new API) [1, 4]. Sólo unos pocos drivers del kernel oficial la utilizan.

pósito general utilizados en éstas máquinas, fueron adaptados de una base uniprocesador. Por lo tanto, además de coordinar el acceso de múltiples aplicaciones independientes a un único recurso físico, el sistema operativo debe propagar esta protección a lo largo de múltiples CPUs.

En resumen, múltiples interrupciones de hardware, pilas de protocolos de red implementadas de manera ineficiente, demasiados accesos a memoria y un overhead significativo agregado por el sistema operativo, consumen una cantidad de ciclos de CPU por paquete sorprendentemente alta. **En experimentos con servidores web sobrecargados, se comprobó que hasta un 70 % del tiempo de uso de CPU era consumido por el procesamiento de red, dejando apenas un 30 % para el proceso que implementa el servidor web** [5].

Los investigadores son concientes de que estos factores requerieren cambios importantes en el codiseño hardware/software de los servidores de red y han comenzado a proponer algunas soluciones para atacar diferentes aspectos de las implementaciones de estos protocolos.

3. Algunas Soluciones

Existe un conjunto de proyectos, tanto académicos como comerciales, que intentan mitigar algunos de los factores mencionados anteriormente. Algunas soluciones se basan exclusivamente en modificaciones orientadas al hardware y otras sólo proponen nuevos diseños de software. Sin embargo, en general se plantea una combinación de nueva tecnología de hardware con optimizaciones al sistema operativo y en particular al subsistema de red.

A continuación se describen algunos de ellos a modo de prueba confirmatoria de que la problemática mencionada es real y representa una de las mayores preocupaciones de los responsables del diseño de hardware y software de red.

La iniciativa denominada TCP Offload Engine [18, 14, 8], pero más conocida como TOE, propone el uso de hardware especializado en la propia placa de red para realizar algunas o todas las tareas asociadas con el procesamiento de los protocolos y de esta forma aliviar al procesador principal del sistema.

RDMA (Remote Direct Memory Access) [19, 8] es una tecnología que permite que el sistema que envía los datos, los ubique en una posición específica en la memoria del sistema que los recibe. De esta forma, se requiere menos intervención de la CPU del receptor para mover los datos entre buffers de su memoria.

Las soluciones conocidas como *Onloading* [5, 6, 7, 8, 9, 10, 16] mantienen el procesador (o procesadores) del sistema como el dispositivo de procesamiento principal para manejar el tráfico de red. La principal diferencia con los sistemas actuales es que estos proyectos proponen una partición del sistema, asignando de manera exclusiva un subconjunto de procesadores en una máquina SMP (o un subconjunto de *cores* en una máquina *multicore*) al procesamiento de red. Se logra una mayor eficiencia reduciendo el overhead generado por la instrusión del sistema operativo (interrupciones y context switches).

Se han planteado optimizaciones y versiones livianas del protocolo TCP [22, 23], modificaciones a la implementación de la capa de sockets [21, 16], optimizaciones a nivel de driver de dispositivos de red (como la alternativa NAPI, mencionada anteriormente) [17, 12] y algunas propuestas más integrales como los *Network Channels* [20] mencionados recientemente por la comunidad de desarrolladores del subsistema de red del kernel linux. Por último, se están desarrollando varias APIs (Microsoft, POSIX, Linux) [16] con una semántica asincrónicas para el acceso a sockets o archivos permitiendo un alto nivel de concurrencia sin el overhead del uso de threads, incrementando considerablemente la performance de las aplicaciones basadas en red.

4. Línea de trabajo actual

El incremento en la disponibilidad de las actuales máquinas SMP sumado al advenimiento de las futuras generaciones de procesadores que ofrecerán múltiples núcleos de procesamiento en un mismo chip nos brindan la oportunidad de utilizar y distribuir el poder de cómputo de nuevas formas. Queda abierto el desafío de encontrar la mejor forma de aprovechar estos recursos.

En la evolución de los sistemas operativos, desde el antiguo diseño de kernel monolítico, pasando por microkernels, exokernels (kernels verticales) y otras variantes, se ha llegado a lo que se conoce ac-

tualmente como kernel activo. Este nuevo diseño, presentado por Steve J. Muir en el año 2001 en su tesis de doctorado [6], plantea un kernel especializado en procesamiento de red, motivado principalmente por la poca eficiencia con que los sistemas operativos de propósito general realizan este tipo de tareas en máquinas SMP. Su trabajo propone dedicar, de manera exclusiva, uno o más procesadores a tareas específicas del kernel, permitiendo que las aplicaciones de usuario se ejecuten, con la menor cantidad de intrusión por parte del sistema operativo, en el resto de los procesadores. De esta forma, el kernel deja de ser un proveedor pasivo de servicios para convertirse en un componente activo del sistema, y la separación física de procesos de usuario con respecto a threads del kernel evita incurrir en el overhead necesario para implementar el modelo *usuario/kernel*.

A fines del año 2002, Kalpana Banerjee propone, como parte de su tesis de Master [7], una nueva arquitectura de software para servidores de Internet denominada *TCP Servers*. Tomando algunas de las ideas de Muir, esta publicación ofrece un nuevo diseño para el subsistema de red del kernel Linux y evalua una implementación en un cluster, usando un nodo como el hardware especializado para el procesamiento de red de todo el cluster. Más tarde, el mismo grupo de trabajo que participó en la tesis de Banerjee, miembros del *Laboratory for Network Centric Computing* [24], publica el diseño de una variante de *TCP Servers* para una maquina con arquitectura SMP [5] y una implementación para el kernel linux 2.4.

Nuestra propuesta consiste en continuar en esta línea de investigación, buscando la forma de adaptar el kernel linux a los altos requerimientos de procesamiento de red a los que estarán expuestos los servidores, dispositivos de red y estaciones de trabajo en un futuro cercano. Se pretende realizar una modificación al diseño del subsistema de red del kernel Linux 2.6[3], basada en las ideas publicadas en los trabajos mencionados anteriormente. Para evaluar los resultados de las modificaciones se realizará una implementación prototipo.

Linux es el kernel monolítico más utilizado en los sistemas operativos libres y por lo tanto representa una gran oportunidad para estudiar y proponer optimizaciones. Además es altamente modular y ofrece una API bien documentada, lo que facilita realizar modificaciones a su código. El tema expuesto en el presente trabajo representa una preocupación real de los desarrolladores del subsistema de red del kernel y es materia de discusión en las listas de correos pertinentes.

Complementando este trabajo, se hará un estudio exhaustivo de los problemas mencionados anteriormente y de algunas soluciones propuestas por diversos grupos académicos y del ámbito industrial. En éste último entorno, el caso que destaca del resto es la nueva tecnología que está siendo desarrollada por Intel®, denominada *I/O Acceleration Technology* [8, 9], y que utiliza varios de los conceptos mencionados en los trabajos citados anteriormente.

Con este trabajo se pretende estudiar en profundidad una problemática que está captando la atención de numerosos grupos de investigación y que aún no presenta una solución completamente definida. Las propuestas que han demostrado tener mayor éxito son aquellas que atacan el problema a nivel de sistema completo (hardware y software) y, desde esta perspectiva, el sistema operativo es un componente escencial.

[3] En este momento, el kernel linux 2.6 ha alcanzado un nivel de estabilidad, robustez y flexibilidad notables. Proveé soporte para más de veinte arquitecturas, incluyendo todo tipo de máquina NUMA, SMP y dispositivos embebidos. Mejoró considerablemente gran parte de los subsistemas con respecto al kernel 2.4 teniendo como principal objetivo la escalabilidad. Y finalmente, es el kernel que se distribuye por defecto en la gran mayoria de las distribuciones GNU / Linux.

Referencias

[1] The Linux TCP/IP Stack: Networking for Embedded Systems (Networking Series). Thomas Herbert. Charles River Media. May 2004.

[2] Understanding Linux Kernel 3rd Edition. Daniel P. Bonet, Marco Cesati. O'Reilly. Nov 2005.

[3] Linux Kernel Development 2 Edition. Robert Love. Novell Press. Jan 2005.

[4] Linux Device Drivers 3rd Edition. Jonathan Corbet, Alessandro Rubini, Greg Kroah-Hartman. O'Reilly & Associates. February 2005.

[5] M. Rangarajan, A. Bohra, K. Banerjee, E. V. Carrera, R. Bianchini. TCP Servers: Offloa-

ding TCP Processing in Internet Servers. Design, Implementation, and Performance.

[6] S. J. Muir. Piglet: An Operating System for Network Appliances.

[7] K. Banerjee. TCP Servers: A TCP/IP offloading architecture for internet servers, using memory-mapped communication.

[8] Intel. Accelerating High-Speed Networking with Intel I/O Acceleration Technology.

[9] Intel. Server Network I/O Acceleration. Fundamental to de Data Center of the Future.

[10] G. Regnier, S. Markineni, R. Illikkal, R. Iyer, D. Minturn, R. Huggahalli, D. Newell, L. Cline, A. Foong. TCP Onloading for Data Center Servers. Published by the IEEE Computer Society.

[11] A. Grover, C. Leech. Accelerating Network Receive Processing. Proceedings of the Linux Symposium.

[12] J.A. Ronciak, J. Brandeburg, G. Venkatesan, M. Willams. Networking Driver Performance and Measurement - e1000 A Case Study. Proceedings of the Linux Symposium.

[13] P. Pietikainen. Hardware-assisted Networking Using Scheduled Transfer Protocol on Linux. Department of Electrical Engineering, University of Oulu, Oulu, Finland.

[14] K. Hyong-Youb. Improving Networking Server Performance with Programmable Network Interfaces.

[15] S. P. Bhattacharya, V. Apte. A Measurement Study of the Linux TCP/IP Stack Performance and Scalability on SMP systems. Indian Institute of Technology, Bombay.

[16] Y. Turner, T. Brencht, G. Regnier, V. Saletore, G. Janakiraman, B. Lynn. Scalable Networking for Next-Generation Computing Platforms. Proceedings of the Third Annual Workshop on System Area Networks, Madrid, Spain, February 2004.

[17] L. Grossman. Large Receive Offload implementation in Neterion 10GbE Ethernet driver. Proceedings of the Linux Symposium.

[18] http://en.wikipedia.org/wiki/TCP/IP_offload_engine

[19] http://en.wikipedia.org/wiki/Rdma

[20] http://lwn.net/Articles/169961/

[21] L. Deri. Improving Passive Packet Capturing: Beyond Device Polling.

[22] C. Jin, D. Wei, S. H. Low, G. Buhrmaster, J. Bunn, D. H. Choe, R. L. A. Cottrel, J. C. Doyle, W. Feng, O. Martin, H. Newman, F. Paganini, S. Ravot, S. Singh. FAST TCP: From theory to experiments.

[23] A. Dunkels. Minimal TCP/IP implementation with proxy support.

[24] http://discolab.rutgers.edu/

[25] A. P. Fonng, T. R. Huff, H. H. Hum, J. P. Patwardhan, G. J. Regnier. Intel Corporation. Department of Computer Science. Duke University. TCP Performance Re-Visited.

[26] Ph.D Dissertation. Peter Druschel. Operating System Support for High-Speed Networking.

[27] P. Wan, Z. Liu. Computer and Information Science Department. Indiana University Purdue University Indianapolis. Operating System Support for High-Performance Networking, A Survey.

[28] N. L. Binkert, L. R. Hsu, A. G. Saidi, R. G. Dreslinski, A. L. Schultz, S. K. Reinhardt. Advanced Computer Architecture Lab. EECS Department, University of Michigan. Performance Analysis of System Overheads in TCP/IP Workloads.