

Análisis del comportamiento y rendimiento del subsistema de comunicaciones del sistema operativo Linux bajo entornos distribuidos

Javier Mendiara Cañardo

Ingeniería Técnica en Informática de Sistemas

Proyecto dirigido por: Francesc Giné de Solà y Josep Lluís Llérida Monsó

Departamento de Informática e Ingeniería Industrial

Escola Universitària Politècnica – Universitat de Lleida

tfn:656 27 59 17

e-mail:jmendiara@hotmail.com

Resumen

En un entorno distribuido cluster son muchos los factores que influyen en su rendimiento, siendo uno de los más importantes tanto el sistema operativo como la red de comunicaciones subyacente. Este proyecto de fin de carrera realiza un estudio para conocer cómo afectan al rendimiento de las aplicaciones distribuidas el subsistema de comunicaciones del sistema operativo. Esto ha comportado la necesidad de conocer cómo se comportan cada uno de los elementos que intervienen en el proceso de comunicación. Asimismo y para este fin, se han creado herramientas que nos permitan monitorizar el subsistema de comunicaciones al nivel de detalle que se pretenda, causando la mínima intrusión posible en el sistema.

1. Introducción

Las exigencias de cálculo de las aplicaciones comerciales y científicas crecen día a día. Para hacer frente a esta demanda es necesario aumentar constantemente la capacidad de cálculo de los sistemas informáticos. Bajo esta necesidad nació el procesamiento paralelo. De la necesidad de reducir costes y mantener prestaciones que ofrecen los MPP (Massively Parallel Processor) nacieron los NOW's o clusters. Estos son un conjunto de ordenadores heterogéneos de bajo costo interconectados en red. En esta interconexión y en su gestión residen algunos cuellos de botella de su rendimiento.

El sistema operativo preferido por las distintas comunidades para realizar computación distribuida es Linux. Se trata de sistema operativo de libre distribución que se distribuye con el código fuente. Bajo Linux, funcionan varias librerías distribuidas, siendo una de las más aceptadas PVM.

Entre los objetivos de este proyecto está la creación de unas herramientas capaces de monitorizar el estado de las colas, tanto de los sockets como del dispositivo lógico que usa el sistema operativo Linux[1][2]. Conocer esta información, junto con otra que se considere necesaria, nos permitirá realizar un estudio de los distintos cuellos de botella que

afectan a las comunicaciones, y por lo tanto a las aplicaciones distribuidas que usen las librerías de PVM[3]. El siguiente objetivo ha sido minimizar la intrusión que produzca la monitorización, mediante optimización de código y adquisición de datos bajo demanda a una velocidad elevada. Para concluir se ha realizado una comparativa de distintos núcleos del sistema operativo Linux, en concreto las versiones 2.0, 2.2 y 2.4, para determinar cuál de ellos responde mejor a nuestras necesidades.

Así pues, los pasos que se han seguido para realizar este proyecto han sido:

- Realizar un estudio exhaustivo de los elementos de los núcleos 2.0.36, 2.2.15 y 2.4.16 del sistema operativo Linux que intervienen en las comunicaciones, para conocer todo el proceso de comunicaciones, entender su funcionamiento y sus diferencias.
- Implementar las herramientas necesarias que nos ayuden a evaluar el comportamiento de los elementos implicados en el sistema de comunicaciones, con el objetivo de encontrar cuellos de botella y poder identificar los elementos que se podrían rectificar para aumentar el rendimiento.
- Minimizar la intrusión de estas herramientas, con el objetivo de obtener una medición objetiva que dependa del sistema de comunicaciones.

2. Las comunicaciones en Linux

Un paquete de datos atraviesa el kernel pasando por distintas colas. La implementación de estas colas varía para cada versión estudiada, aunque finalmente, el concepto sea el mismo. La fig. 1 muestra un resumen de estas estructuras de datos para las tres versiones analizadas.

A nivel de socket existe una cola de recepción (`receive_queue`) y otra de envío (`write_queue`) para cada canal de comunicación. En el dispositivo lógico se observa que en la versión 2.4, no existe una sola cola de entrada como en sus predecesoras, sino tantas como CPU's disponga la máquina. En la versión 2.2 y sucesivas se

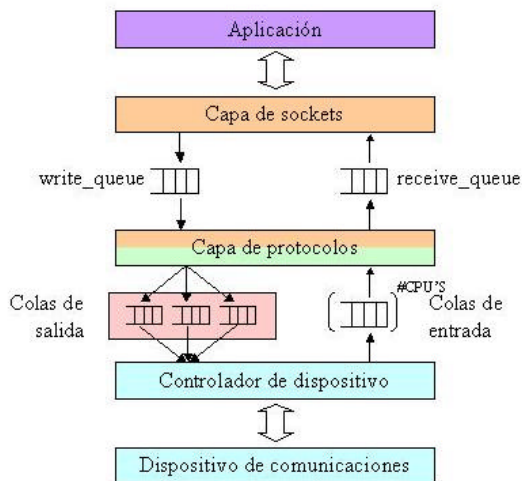


Fig. 1. Resumen de estructuras de datos implicadas en la comunicación

introduce el concepto de QoS para la cola de salida. Esto permite que tanto la estructura como la política de gestión de las mismas pueda ser modificada en tiempo real. Por defecto, el QoS está implementado con tres colas, donde los paquetes se ordenan por prioridad. En la versión 2.0 el concepto era el mismo, aunque la implementación era estática y única.

3. Herramientas para el análisis

Se han creado un conjunto de herramientas que nos permiten monitorizar el estado de cada una de las capas ilustradas en la fig.1. Las herramientas de monitorización realizadas crean una entrada en sistema de archivos. Para poder monitorizar la información de las colas de los sockets se ha implementado un módulo de carga dinámica denominado *stadsoc*. El módulo *staddev* muestra la información estadística sobre las comunicaciones mantenidas por el kernel y el controlador del dispositivo. Finalmente, se ha creado dentro del mismo kernel otra entrada denominada *stadque* que informa sobre las colas del dispositivo lógico.

netmon (*Network Monitor*) es la herramienta de monitorización que conjuga las tres herramientas anteriores junto con otras específicas de PVM. Ha sido desarrollado después del estudio de varios algoritmos de adquisición rápida de datos y uso de memoria dinámica con el objetivo de producir la menor intrusión y obtener la mayor precisión posible. Los resultados proporcionados por *netmon* permiten el dibujo de gráficas que muestran gran cantidad de información hasta ahora no contemplada por los programas de su familia.

4. Experimentación

Los primeras mediciones experimentales se han hecho mediante cuatro programas sintéticos con el objetivo de medir la intrusión de *netmon*. El slowdown obtenido se mantiene por debajo de 1.01 en el peor de los casos.

A continuación se ejecutaron varios benchmarks distribuidos de la familia NPB con *netmon* monitorizando los nodos implicados, con el objetivo de medir estas

aplicaciones distribuidas con respecto a sus requerimientos de comunicación. Los resultados gráficos mostrados como ejemplo en las figuras fig.2 y fig.3 pueden ayudar al programador a decidir cómo mejorar su aplicación para que esta tenga un mejor rendimiento.

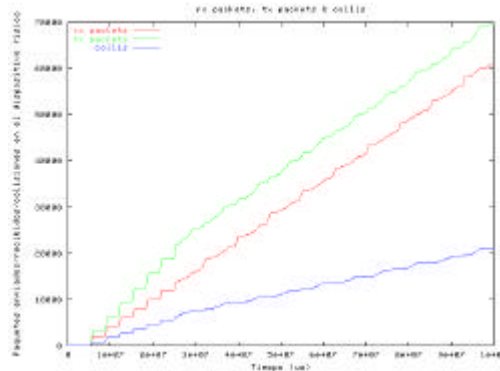


Fig.2: Paquetes enviados/recibidos y colisiones

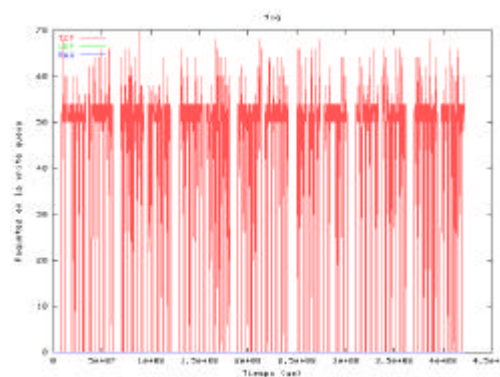


Fig.3: Ocupación de la write_queue para cada protocolo

Se ha concluido la investigación con un análisis comparativo del rendimiento de los kernel 2.2 y 2.4 para ver cuál de los dos se comporta mejor ante aplicaciones distribuidas, concluyendo una semejanza de comportamiento.

5. Conclusiones

El objetivo principal de este TFC ha sido estudiar el comportamiento y detectar los posibles cuellos de botella en las comunicaciones que implementa el sistema operativo Linux en distintas versiones, la 2.0.36, la 2.2.15 y la 2.4.16.

Del análisis de este comportamiento podemos concluir soluciones que nos permitan aumentar el rendimiento global, tanto de los elementos del subsistema de comunicaciones del kernel, como de aplicaciones que lo usan, como puede ser el PVM.

Referencias

- [1] Homepage del netfilter/iptables project. <http://netfilter.samba.org/>
- [2] David A. Rusling. "The Linux Kernel" *Linux Documentation Project*.
- [3] Josep Lluís Lérída Monsó. "Anàlisi del sistema de comunicacions de PVM en un sistema operatiu LINUX." *EUP-UdL*, octubre de 1999