



Performance y escalabilidad del kernel Linux aplicado a redes de alta velocidad

Matías Zabaljáuregui

Agenda

- Introducción
- Intrusión del Sistema Operativo
- Performance en el procesamiento de red
- La escalabilidad de TCP/IP en Linux
- Un prototipo básico y resultados preliminares
- Conclusiones y trabajo a futuro

Introducción

- Contexto
- Algunos Problemas
- Objetivos de la Tesis

The Tera Era

“An age when people need teraflops of computing power, terabits of communication bandwidth, and terabytes of data storage to handle the information all around them”

Contexto – Factores Fundamentales

- Computación Centrada en las Redes
- Arquitecturas SMP / Multicore

Contexto – Computación Centrada en las Redes

- Tecnología de Comunicaciones
 - Infiniband, Fibre Channel, Myrinet...
 - Ethernet
 - Proyecto 100 Gbps (Julio de 2007)
- Distribución de datos, procesadores e instrumentos
 - WWW y Web Services
 - HPC y e-science
 - Almacenamiento (SAN, NAS)
 - Multimedia, sensores, RFID, etc

Contexto – Arquitecturas SMP / Multicore

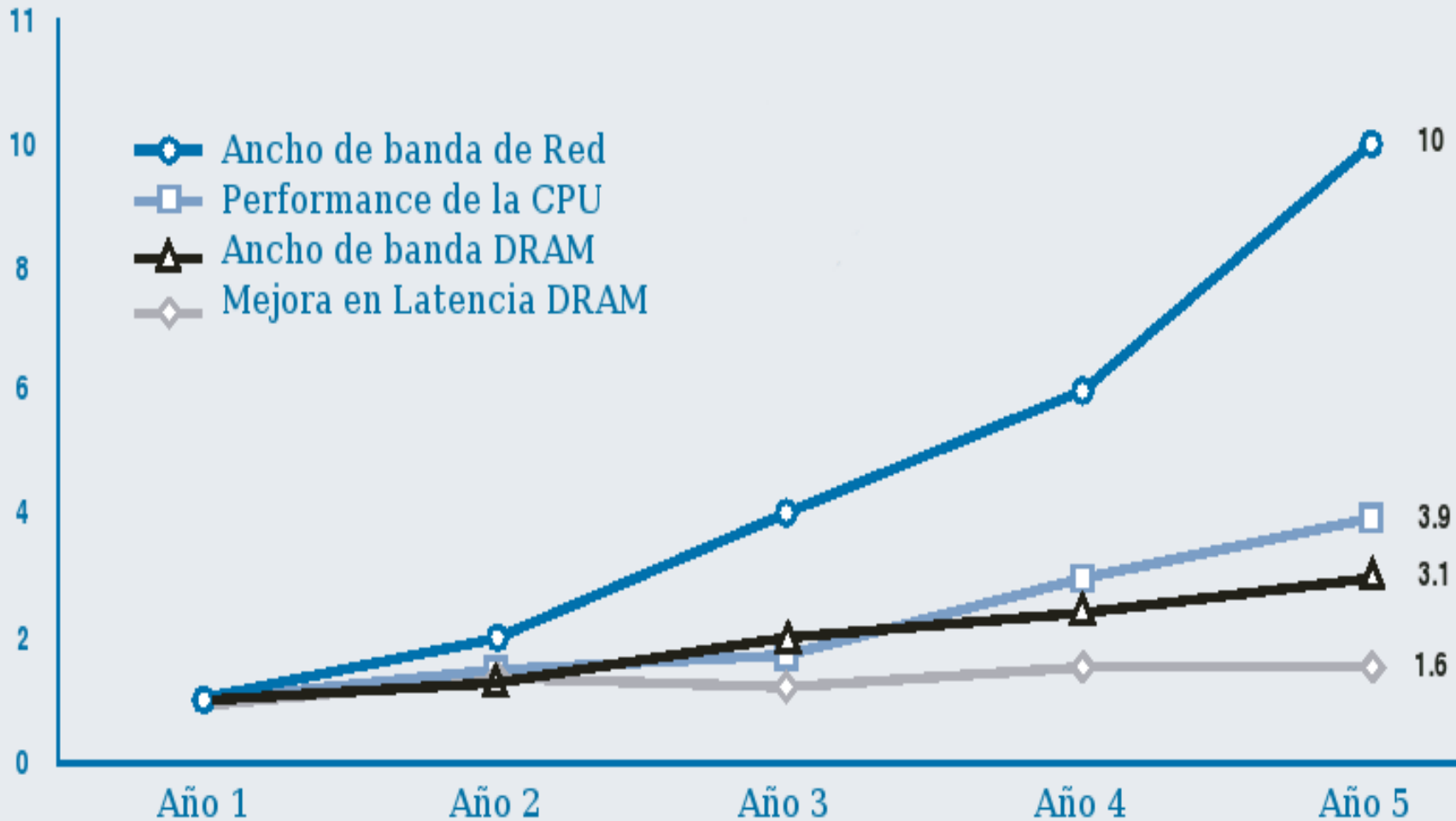
- Servidores
 - Xeon® processor family
 - Itanium® 2 processor
- Escritorios
 - Core™2 Extreme processor
- Dispositivos de Red
 - IXP2855 Network Processor
- Consolas de Juegos
 - Cell (IBM, Sony y Toshiba)
- Dispositivos embebidos
 - Core™2 Duo E6400 y T7400
- Experimentales
 - Intel Teraflops Research Chip (80 cores)

Algunos Problemas

- Limitaciones de las plataformas actuales
- El software debe ser re-diseñado

Proyección de mejoras en RED, CPU y RAM

Proyección de la mejora relativa en tecnología de RED, CPU y DRAM



Algunos Problemas – Limitaciones de Plataforma

- La barrera de la memoria principal
 - John von Neumann. 1946. "The Principles of Large-Scale Computing Machines"
- Las placas de red como dispositivos *periféricos*
 - *La última milla* en las LANs
- Cuellos de botella de SMP / Multicore
 - Arquitecturas donde “todo se comparte”

Algunos Problemas – El software

- Software de usuario
 - Paralelizar las aplicaciones clientes
 - Escalabilidad de los procesos servidores
- Sistemas Operativos
 - Evaluar la paralelización de las funciones del kernel
 - Incrementar la eficiencia en la E/S

Algunos Problemas – El kernel

- Interrupciones
- Sincronización
- Copias de datos en memoria
- Scheduling

Intrusión del
Sistema Operativo

Objetivo de la Tesis

Adaptar la idea de un kernel activo para mejorar la performance y escalabilidad del subsistema de red del kernel Linux

Objetivo de la Tesis

Adaptar la idea de un kernel activo para mejorar la performance y escalabilidad del subsistema de red del kernel Linux

- Estudiar el codiseño hardware/software del subsistema de red en un servidor SMP con Ethernet Gigabit y el kernel Linux 2.6
- Identificar y clasificar los costos que reducen la performance y/o escalabilidad del kernel Linux en el procesamiento de red
- Analizar las soluciones propuestas hasta ahora en el ámbito académico e industrial
- Implementar un prototipo para medir el efecto de las ideas planteadas en este trabajo

Intrusión del Sistema Operativo

- El rol del Sistema Operativo
- La Intrusión del Sistema Operativo
- La evolución de los Sistemas Operativos

El rol del Sistema Operativo – Virtualización

- Protección
- Multiplexación
- Traducción

Fueron combinadas con las funciones de alto nivel

La Intrusión del Sistema Operativo – Clasificación

- Intrusión por recursos
- Intrusión por tiempo
 - Intrusión por política
 - Intrusión por mecanismo

La Intrusión del Sistema Operativo – FI

- Factor de Intrusión

tiempo real de ejecución

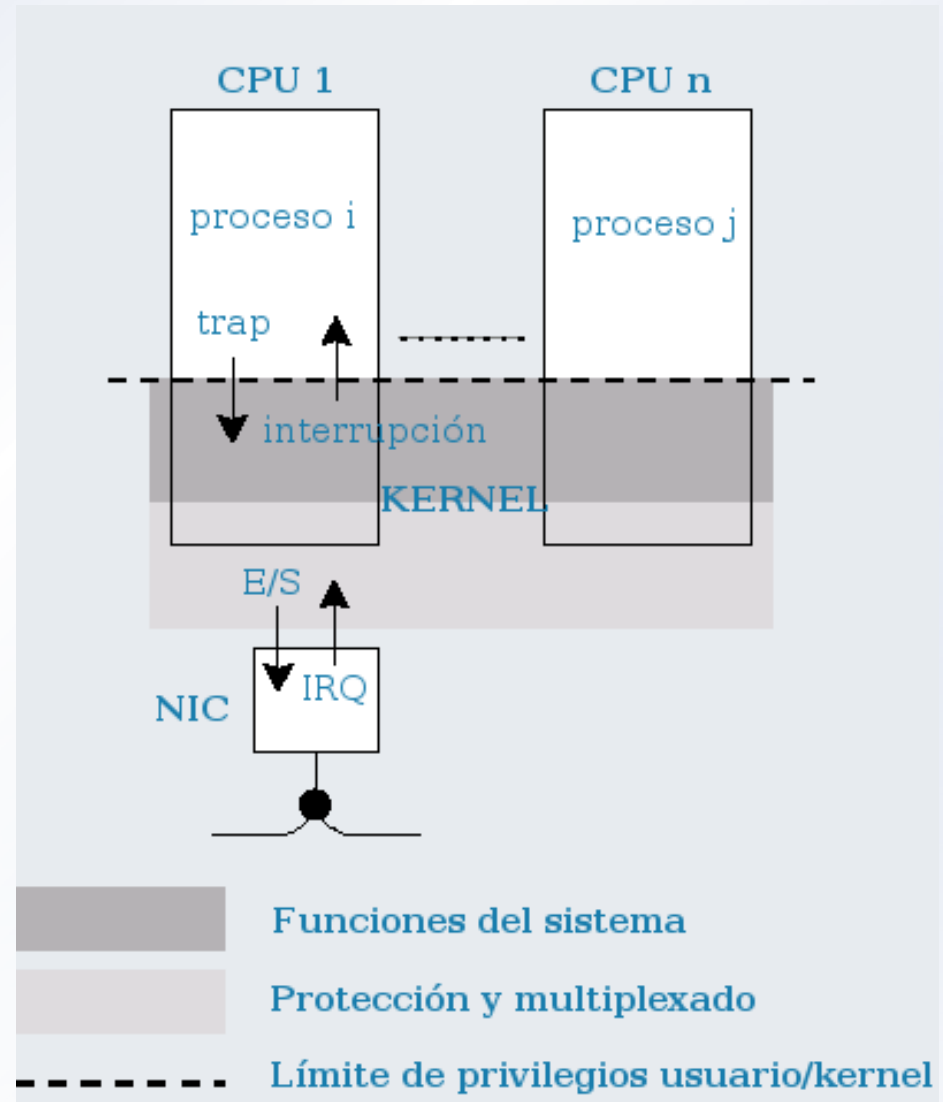
tiempo ideal de ejecución

- *Livelock*

$$FI == \infty$$

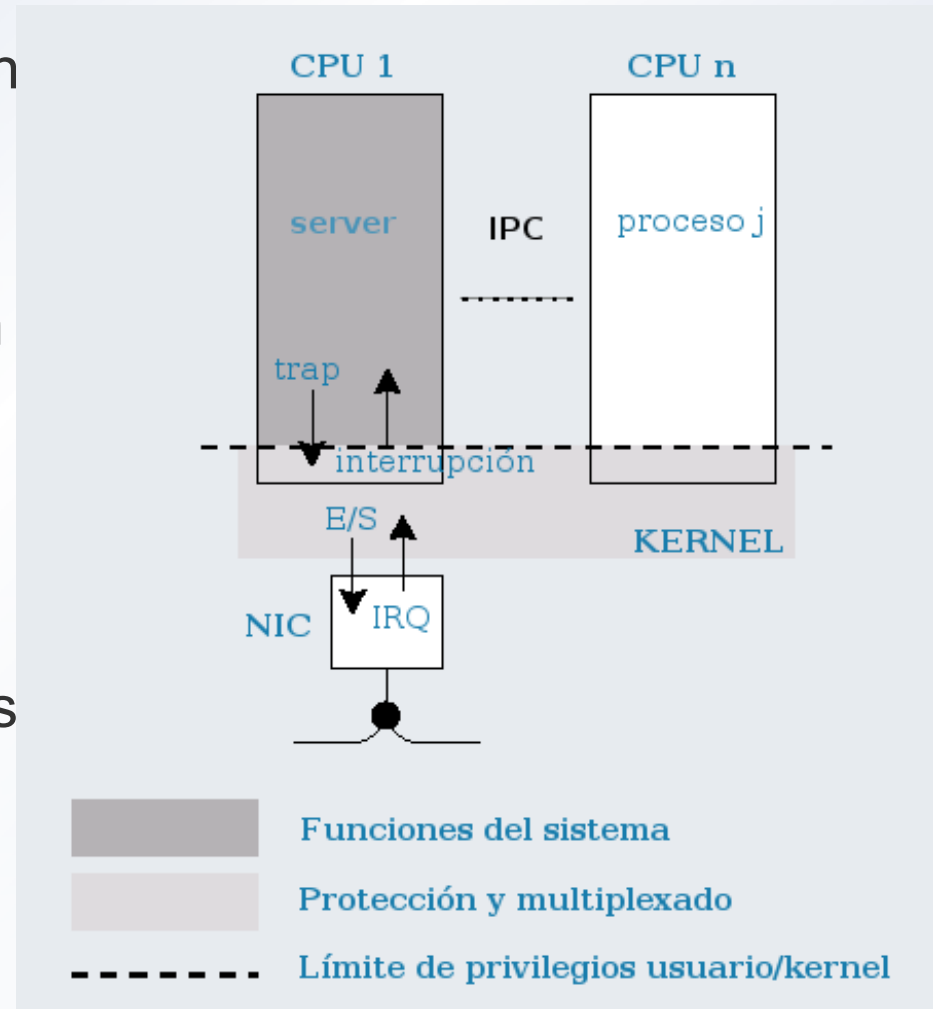
La evolución – Kernel Monolítico

- La virtualización es implementada a través de la abstracción
- Un único espacio de direccionamiento
- TCP/IP incluido en el kernel por herencia de Multics



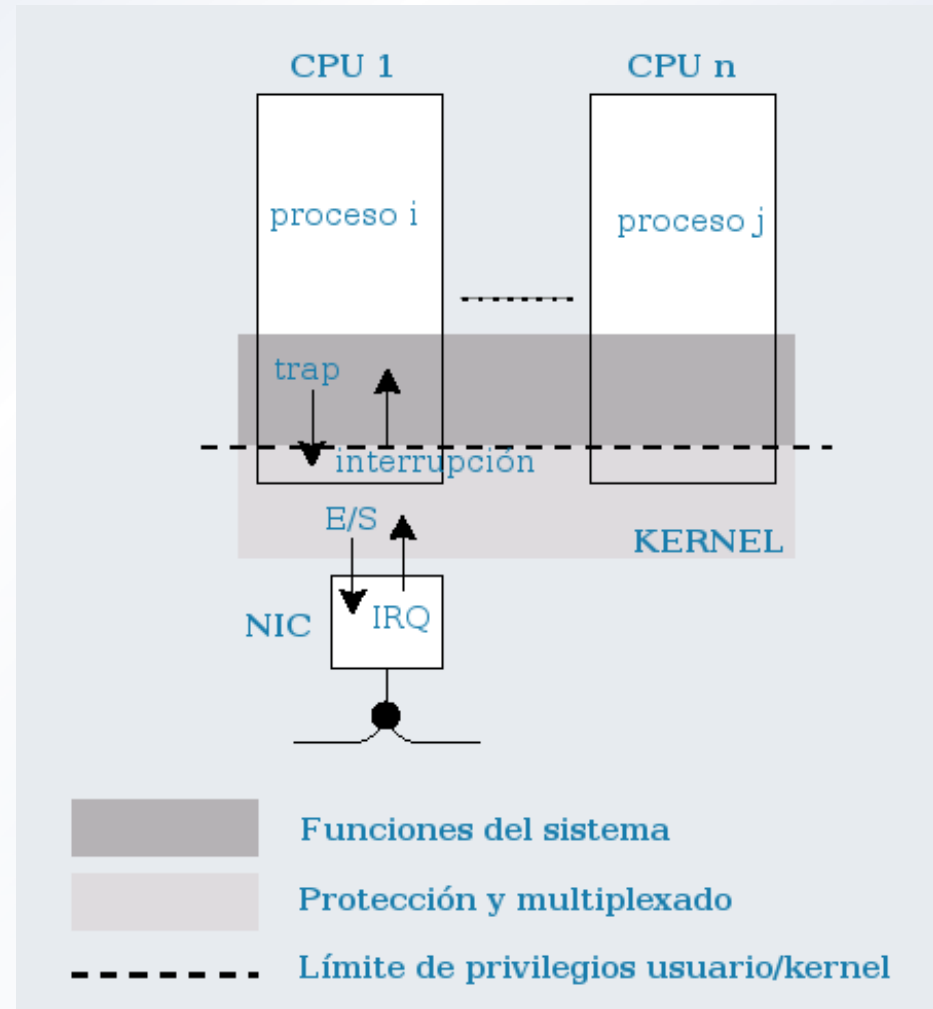
La evolución – Microkernel

- Servicios del sistema operativo en procesos *servers*
- El kernel implementa sólo un mínimo de servicios de virtualización
- El sistema operativo puede ofrecer diferentes estrategias
- Los requerimientos de E/S y los datos asociados deben cruzar límites de protección adicionales



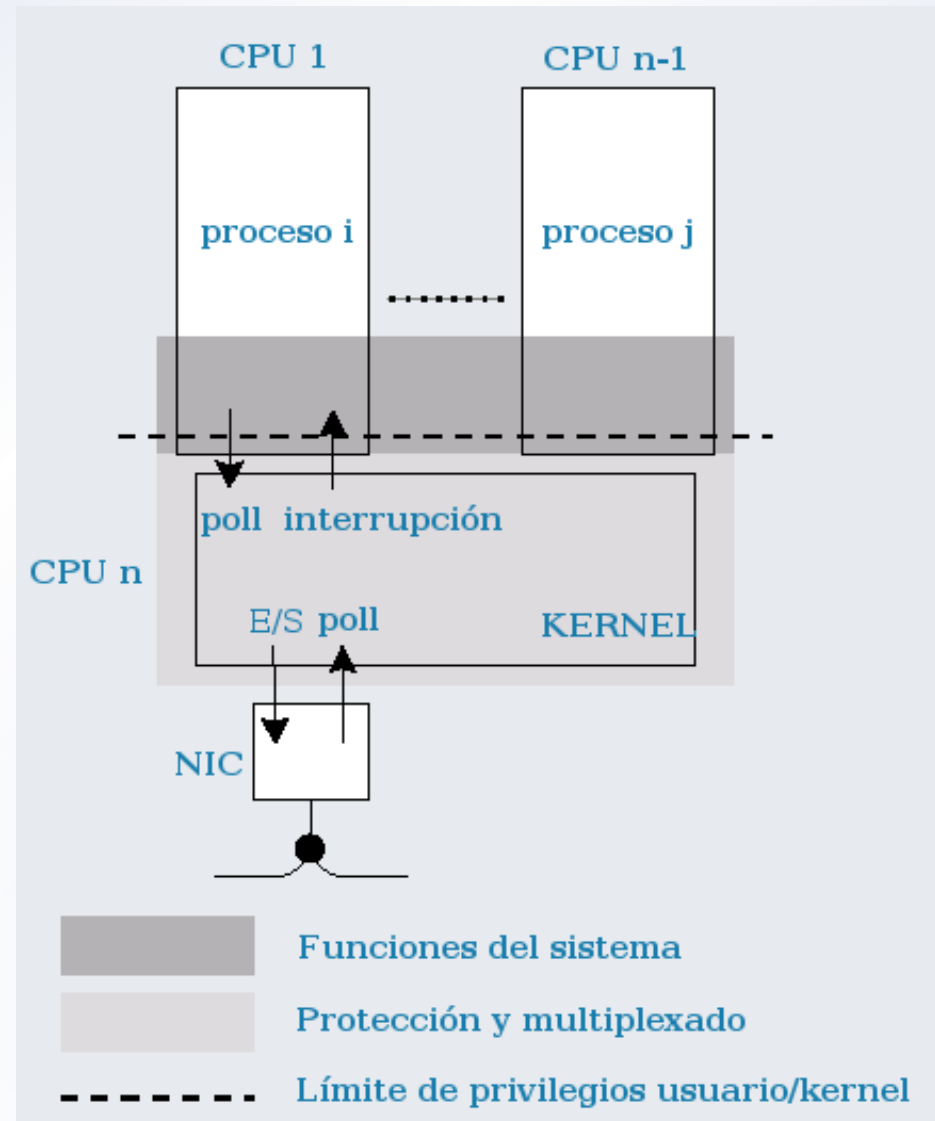
La evolución – Kernel Vertical

- El kernel provee primitivas de virtualización de bajo nivel a las aplicaciones
- Sin procesos Servers. Se evita el IPC
- Las librerías compartidas son una parte integral de este tipo de sistema



La evolución – Kernel Activo

- El kernel se ejecuta continuamente en un procesador dedicado
- Eliminación de las interrupciones asincrónicas (se reduce el overhead y la complejidad del código)
- Objetos en memoria compartida como el mecanismo principal de comunicación entre las aplicaciones y el kernel



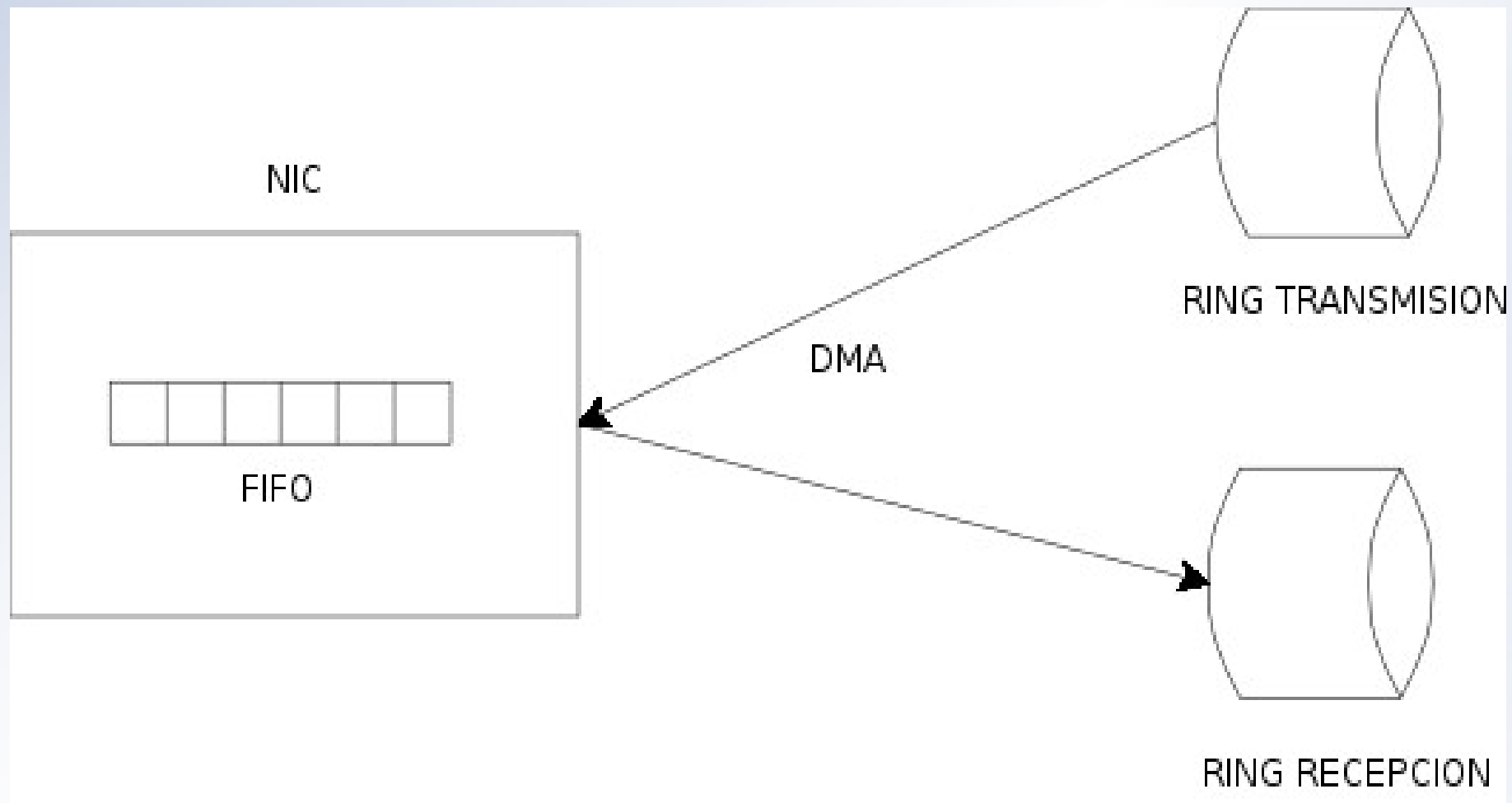
Performance en el procesamiento de red

- Temas de hardware
- Subsistema de red de Linux 2.6
- Los costos del procesamiento de red

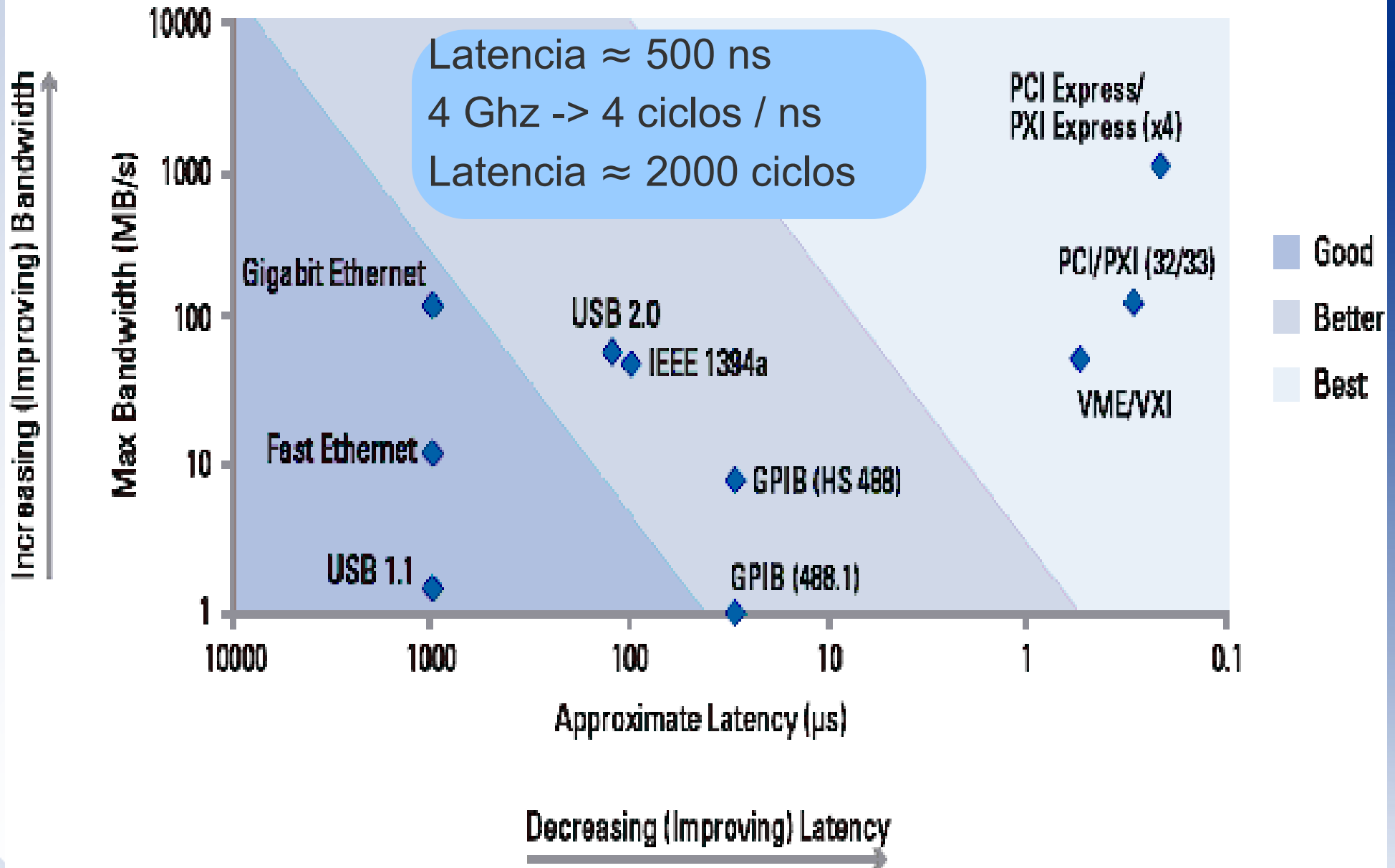
Temas de hardware

- Network Interface Controllers (NICs)
- Tecnología de buses de E/S
- La latencia de la memoria principal y la inefectividad de la memoria cache

Network Interface Controllers (NICs)



Tecnología de buses de E/S



La latencia de la memoria principal y la inefectividad de la memoria cache

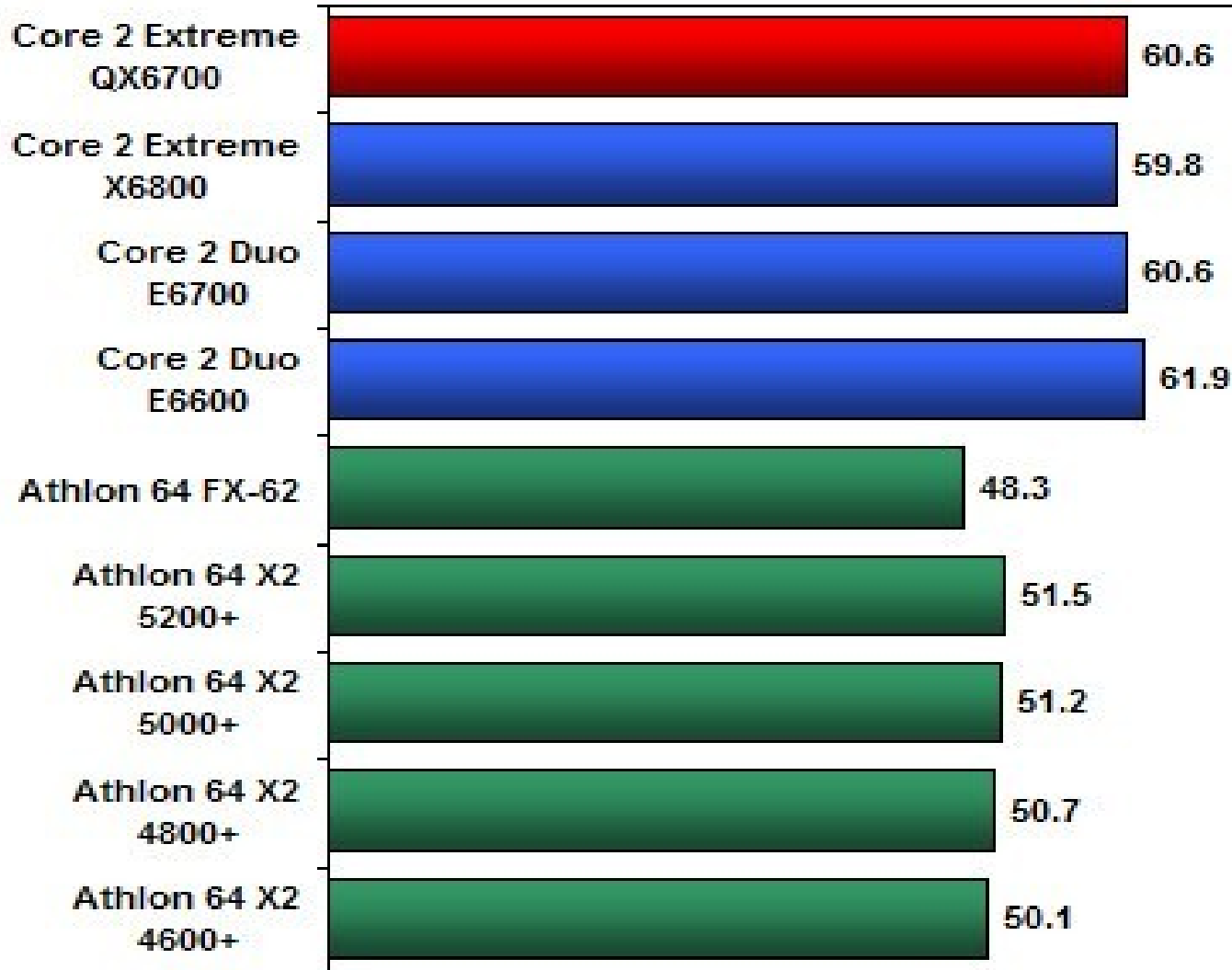
- Latencia de memoria principal
- Accesos de TCP/IP a memoria principal
- Inefectividad de la memoria cache
- Regla “1 GHz / 1 Gbps”

Latencia de memoria principal

- La performance de DRAM se caracteriza por su latencia de acceso (ha mejorado sólo a un 7% anual)
- CPUs de 4 GHz vs bus de memoria de 400 MHz (DDR2-800)
- DDR2 es conocido por sus altas latencias de lectura, que llegan a ser de hasta 9 ciclos del bus de memoria
- Se requiere tiempo adicional para atravesar el bus de memoria, el northbridge y FSB hasta el procesador

Latencia de memoria principal

**Everest Memory Latency (ns)
(Lower is Better)**



latencia

involucra
inco

VS

- La de

- CP (DE

- El c al me **acce** tama

- Ine

Accesos de TCP/IP a memoria principal

- El ancho de banda de memoria principal en las computadoras modernas de propósito general está en el mismo orden que el ancho de banda de las redes de alta velocidad emergentes
- El camino de datos de toda transferencia por red involucra al menos tres accesos a memoria principal. Son cinco accesos en el peor caso (tamaño de los paquetes vs tamaño de cache)

Inefectividad de la memoria cache

- TCP/IP es una *Flow Application*
- Polución de cache
- Tamaño de la Cache y Correspondencia
- Política de escritura y mecanismos de coherencia

Regla “1 GHz / 1 Gbps”

- Esta relación empeora para transferencias con unidades más pequeñas.
- Tomando medidas a dos frecuencias de CPU (800 MHz y 2.4 GHz) se ha mostrado que la performance de red sólo escala en un 60 %
- La tecnología SMT ayuda a disminuir los ciclos de CPU ociosos en los accesos a memoria de las aplicaciones con más de un thread de ejecución

Subsistema de red de Linux 2.6

- Mecanismos de Linux involucrados en el procesamiento de red
- Componentes del procesamiento TCP/IP

Mecanismos de Linux involucrados en el procesamiento de red

- Interrupciones y funciones diferibles
- Los threads del kernel y *ksoftirqd*
- La estructura *softnet_data*
- Los socket buffers

Interrupciones y funciones diferibles

- *El manejador de interrupciones se divide en top half y bottom half*
- *NET_TX_SOFTIRQ y NET_RX_SOFTIRQ*
- *Softirqs*
 - *Se planifican y ejecutan en la misma CPU que recibió la interrupción del dispositivo y, aunque se serializan por CPU, deben ser funciones reentrantes*
 - *Los manejadores de softirqs se ejecutarán cuando el kernel verifique la existencia de softirqs pendientes. Cuando el tráfico de red es alto, siempre hay softirqs pendientes*
 - *Se suele discutir la escalabilidad de este mecanismo, ya que sólo hay dos contextos de procesamiento de red por CPU*

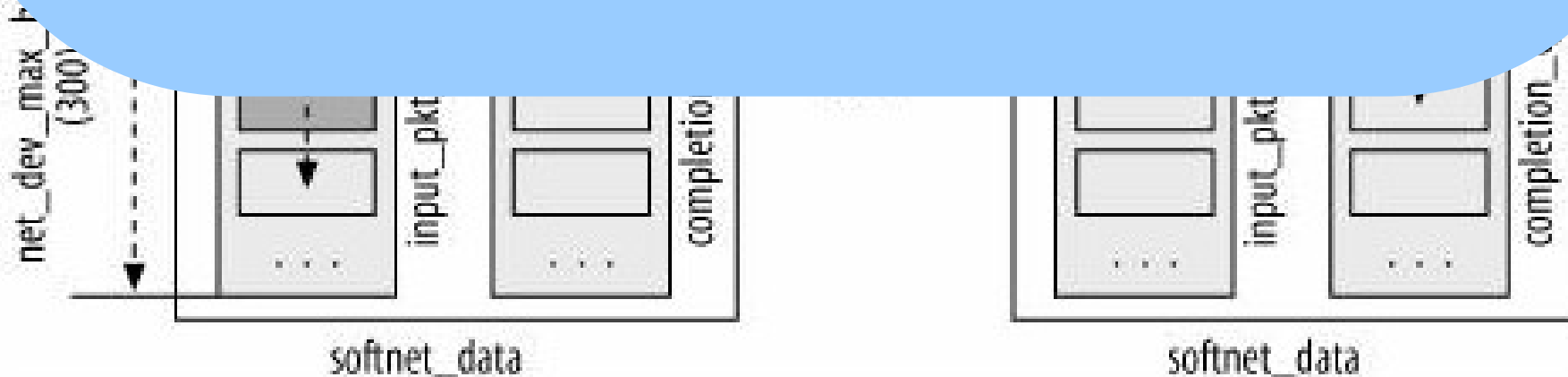
Los threads del kernel y *ksoftirqd*

- Threads del kernel
 - no necesitan manejar un contexto en modo usuario
 - son planificados por el scheduler general, compitiendo con los procesos del usuario
- *ksoftirqd*
 - uno asociado a cada procesador de la máquina
 - verificar si existen softirqs que hayan quedado sin ejecutarse
- Bajo cargas intensas de red, el scheduler intercalará procesamiento de red con el resto de los procesos

La estructura *softnet_data*



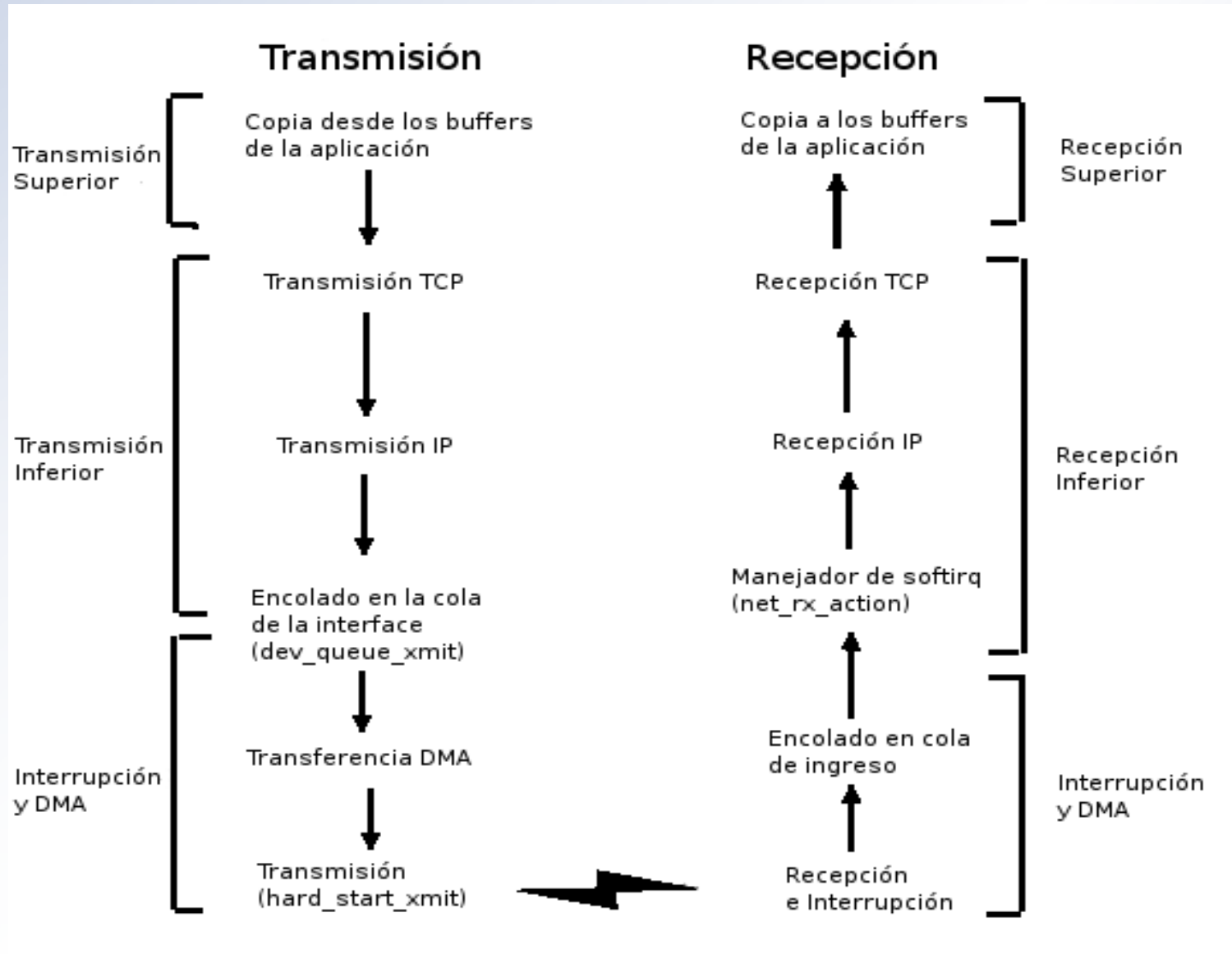
Desde la versión 2.5 del kernel hay dos formas de proceder en la recepción de una trama: el método tradicional y el método **NAPI**



Los socket buffers

- Los socket buffers (`sk_buff`) de Linux son estructuras utilizadas por todas las capas del subsistema de red para referenciar paquetes
- Consiste en los datos de control y una referencia a la memoria donde se ubican los datos del paquete.

Componentes del procesamiento TCP/IP



Los costos del procesamiento de red

- Operaciones DTO
- Operaciones NDTO

DTO = Data Touching
Overhead

NDTO = Non Data
Touching Overhead

- Operaciones DTO

- Transferencia de datos entre dominios de protección

- Manipulación de datos

Operaciones NDTO

- Creación y manipulación de buffers de red
- Operación del sistema operativo
- Operaciones específicas de los protocolos

DTO vs NDTO

Cuadro 8.2: Profiling del kernel durante la prueba A

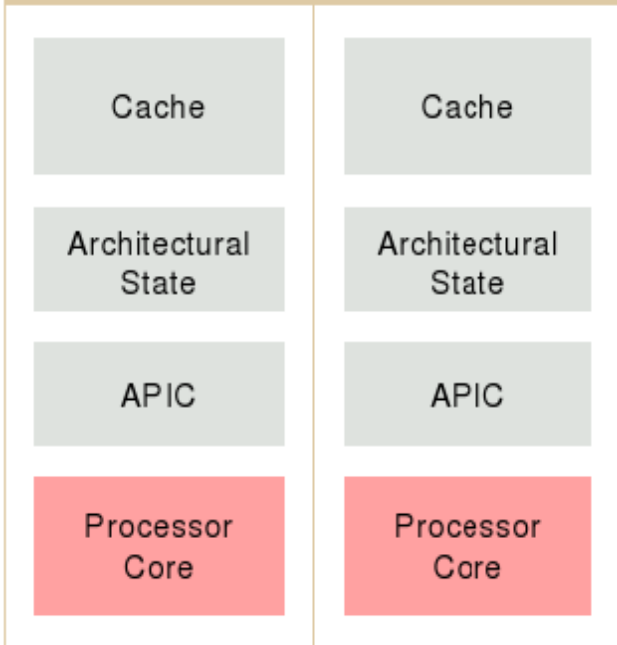
muestras	%	muestras	%	símbolo
69213	18.2416	58248	55.5786	<code>_copy_to_user_ll</code>
53019	13.9735	1942	1.8530	<code>handle_IRQ_event</code>
21261	5.6035	2779	2.6516	<code>tcp_v4_rcv</code>
12545	3.3063	1364	1.3015	<code>_do_IRQ</code>
11823	3.1160	460	0.4389	<code>qdisc_restart</code>
11375	2.9980	267	0.2548	<code>_spin_lock</code>
9692	2.5544	1514	1.4446	<code>irq_entries_start</code>
9555	2.5183	1709	1.6307	<code>schedule</code>
8774	2.3125	1168	1.1145	<code>_spin_unlock</code>
8430	2.2218	3066	2.9255	<code>kfree</code>
7924	2.0884	3322	3.1698	<code>tcp_rcv_established</code>

La escalabilidad de TCP/IP en Linux

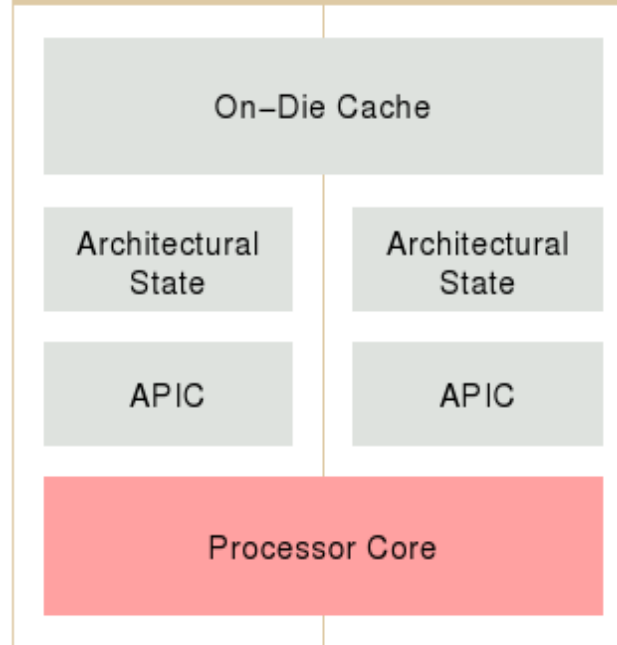
- Multiprocesamiento
- Paralelización de protocolos
- Distribución de interrupciones en SMP
- Problemas de Escalabilidad
- Afinidad de procesos e interrupciones

Multiprocesamiento

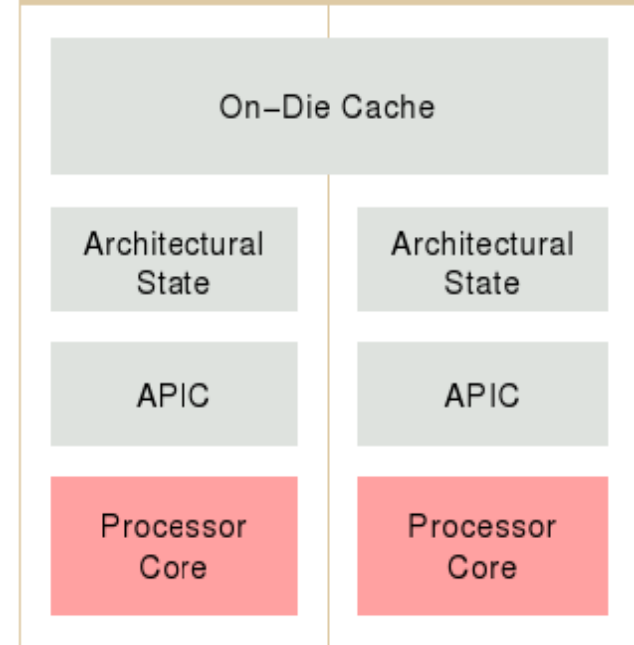
Dual Processor



Hyper-Threading



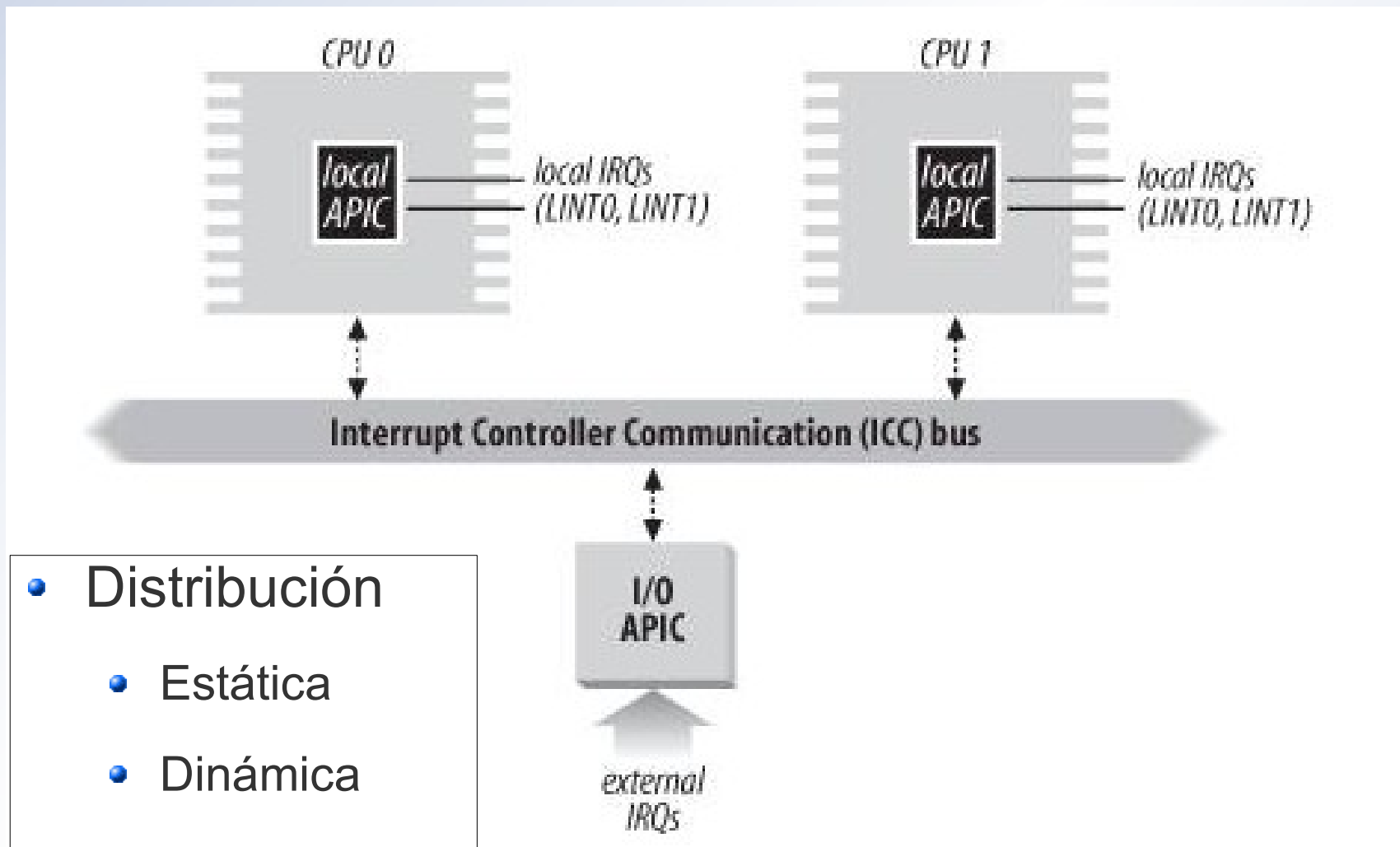
Dual Core



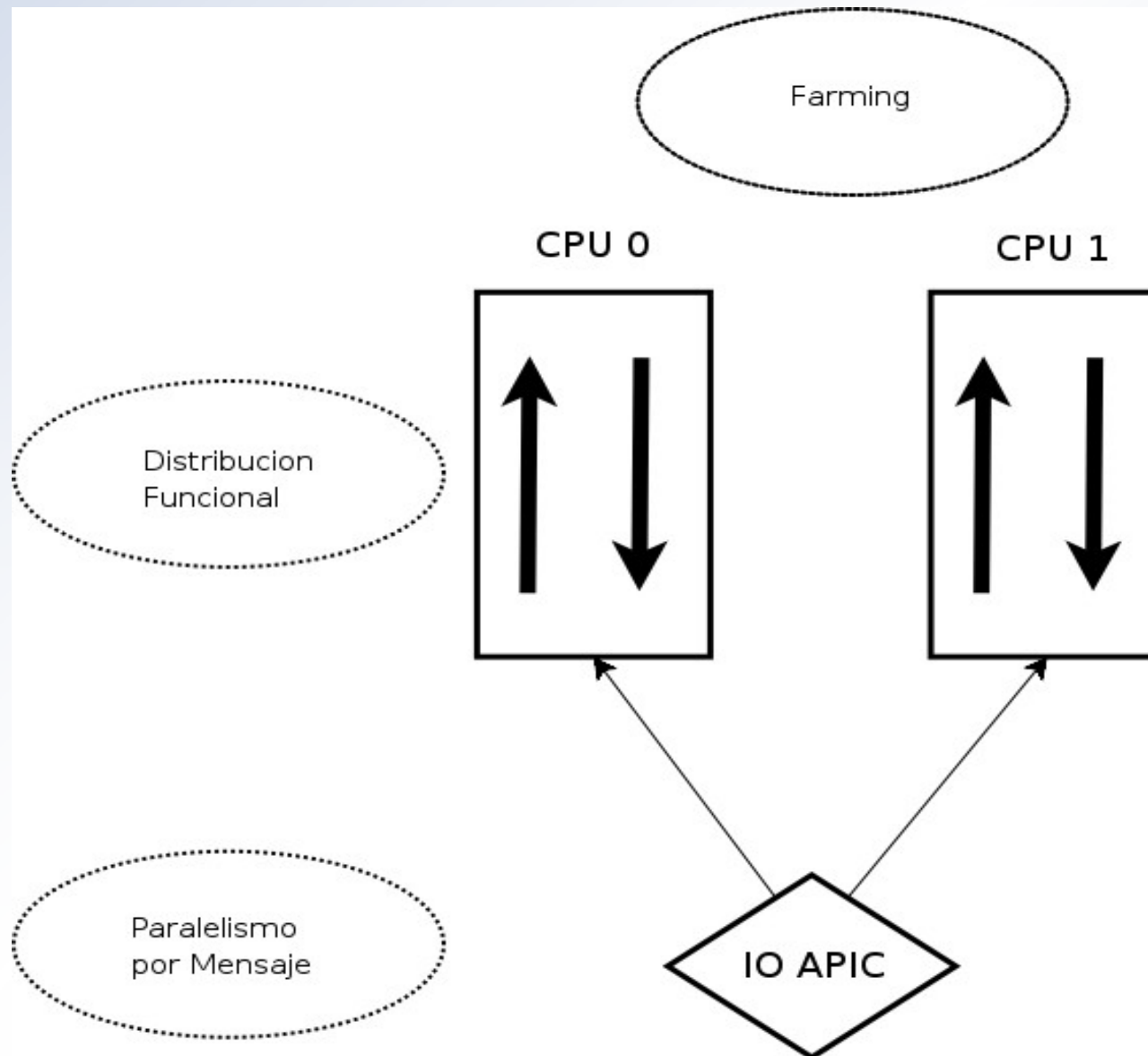
Paralelización de protocolos

- Formas de Paralelismo
 - Farming
 - Paralelismo Temporal
 - Distribución Funcional (Multiple data parallelism)
 - Single data parallelism
- Niveles de Paralelismo
 - Basado en la tarea
 - Basado en mensajes

Distribución de interrupciones en SMP



Paralelismo del procesamiento de red en Linux



Problemas de Escalabilidad

Cuadro 8.2: Profiling del kernel durante la prueba A

muestras	%	muestras	%	símbolo
69213	18.2416	58248	55.5786	_copy_to_user_ll
53019	13.9735	1942	1.8530	handle_IRQ_event
21261	5.6035	2779	2.6516	tcp_v4_rcv
12545	3.3063	1364	1.3015	_do_IRQ
11823	3.1160	460	0.4389	qdisc_restart
11375	2.9980	267	0.2548	_spin_lock
9692	2.5544	15		
9555	2.5183	17		
8774	2.3125	11		
8430	2.2218	30		
7924	2.0884	33		

en los
to general

E	
90	1418

Cuadro 8.3: Profiling del kernel durante la prueba B

muestras	%	muestras	%	símbolo
113573	16.5949	1631	1.1642	_spin_lock_irqsave
77729	11.3575	61287	43.7467	_copy_to_user_ll
49851	7.2840	1816	1.2963	handle_IRQ_event
40771	5.9573	4382	3.1279	tcp_v4_rcv
25860	3.7786	5935	4.2364	tcp_rcv_established
20138	2.9425	4636	3.3092	kfree
18753	2.7401	3475	2.4805	schedule
17351	2.5353	709	0.5061	_spin_lock
15908	2.3244	619	0.4418	qdisc_restart

- Se debe reducir como sea posible

Afinidad de procesos e interrupciones

Cuadro 8.3: Profiling del kernel durante la prueba B

muestras	%	muestras	%	símbolo
113573	16.5949	1631	1.1642	_spin_lock_irqsave
77729	11.3575	61287	43.7467	_copy_to_user_ll
49851	7.2840	1816	1.2963	handle_IRQ_event
40771	5.9573	4382	3.1279	tcp_v4_rcv
25860	3.7786			
20138	2.9425			
18753	2.7401			
17351	2.5353			
15908	2.3244			

Cuadro 8.4: Profiling del kernel durante la prueba C

muestras	%	muestras	%	símbolo
71262	12.2796	38204	44.7831	_copy_to_user_ll
63721	10.9802	2039	2.3901	handle_IRQ_event
28590	4.9265	1795	2.1041	schedule
23986	4.1332	1883	2.2073	_spin_unlock
20121	3.4672	227	0.2661	_spin_lock
19769	3.4065	2567	3.0091	tcp_v4_rcv
19551	3.3690	1624	1.9037	default_idle
17150	2.9552	1631	1.9119	_switch_to
14749	2.5415	473	0.5545	qdisc_restart
11698	2.0158	1902	2.2295	try_to_wake_up

Proyectos Relacionados

- Offloading
- Onloading: Intel IO/AT
- Asynchronous I/O (AIO)

Offloading

- Stateless
 - Checksum Offloading
 - Large Segment Offload
 - Split Headers
 - Receive-side scaling
 - Large Receive Offload
- Stateful
 - TCP Offload Engines

Onloading: Intel IO/AT

- Procesamiento de la pila de protocolos
- Threads Livianos
- Acceso directo a la cache
- Copias Asíncronas en Memoria

AIO y memoria compartida

- AIO divide cada operación de E/S en dos fases distintas: generar un requerimiento de operación y recuperar la información del evento de terminación
- Ejemplos: Windows asynchronous I/O de Microsoft, POSIX AIO y Linux AIO. Estas APIs están pensadas para una pila de E/S (sockets y archivos) basadas en el kernel
- La memoria compartida evita las copias en memoria, implementando el modelo compartido virtual

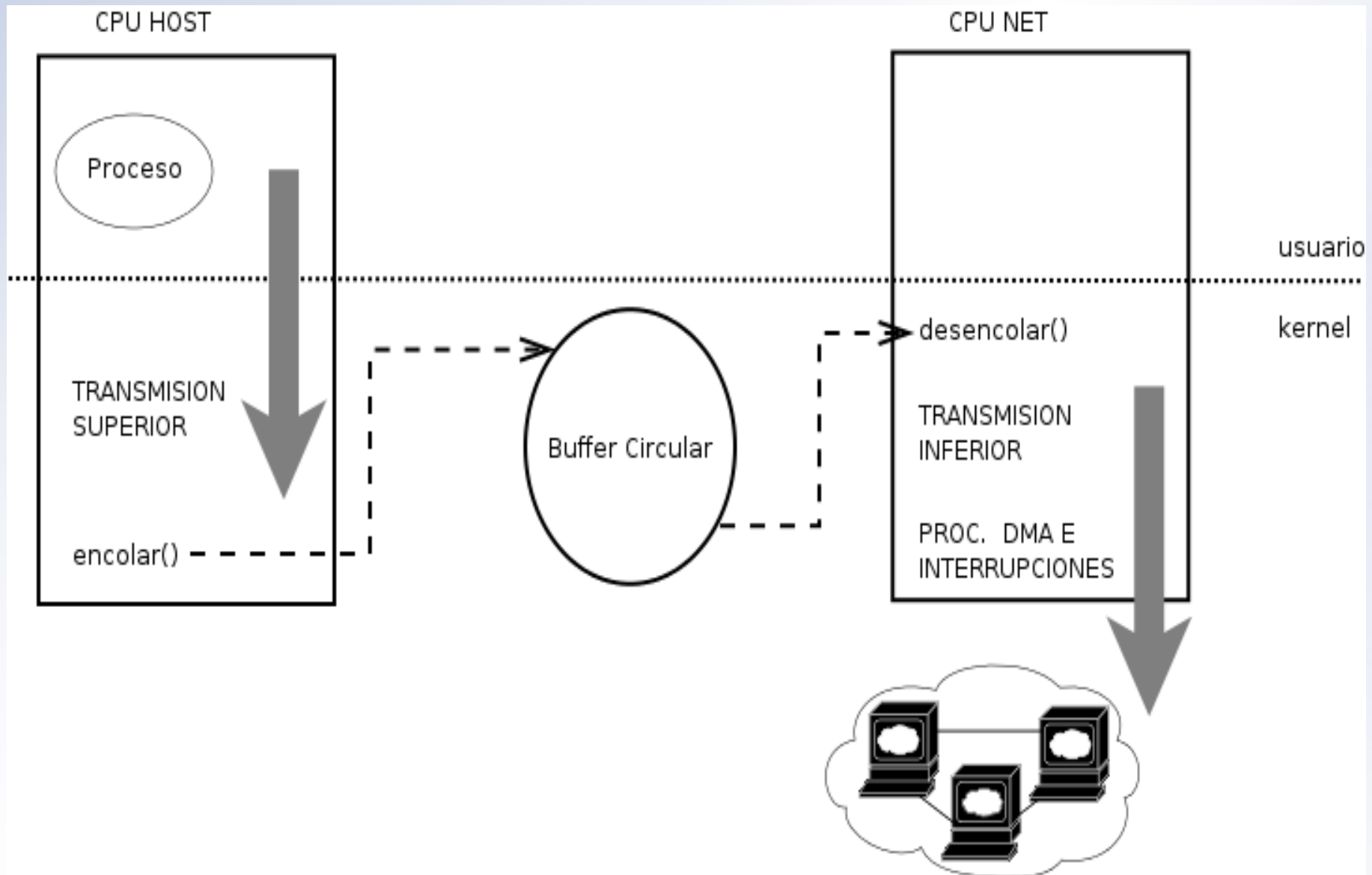
Un prototipo básico y resultados preliminares

- Diseño e implementación del prototipo
- Mediciones y resultados preliminares

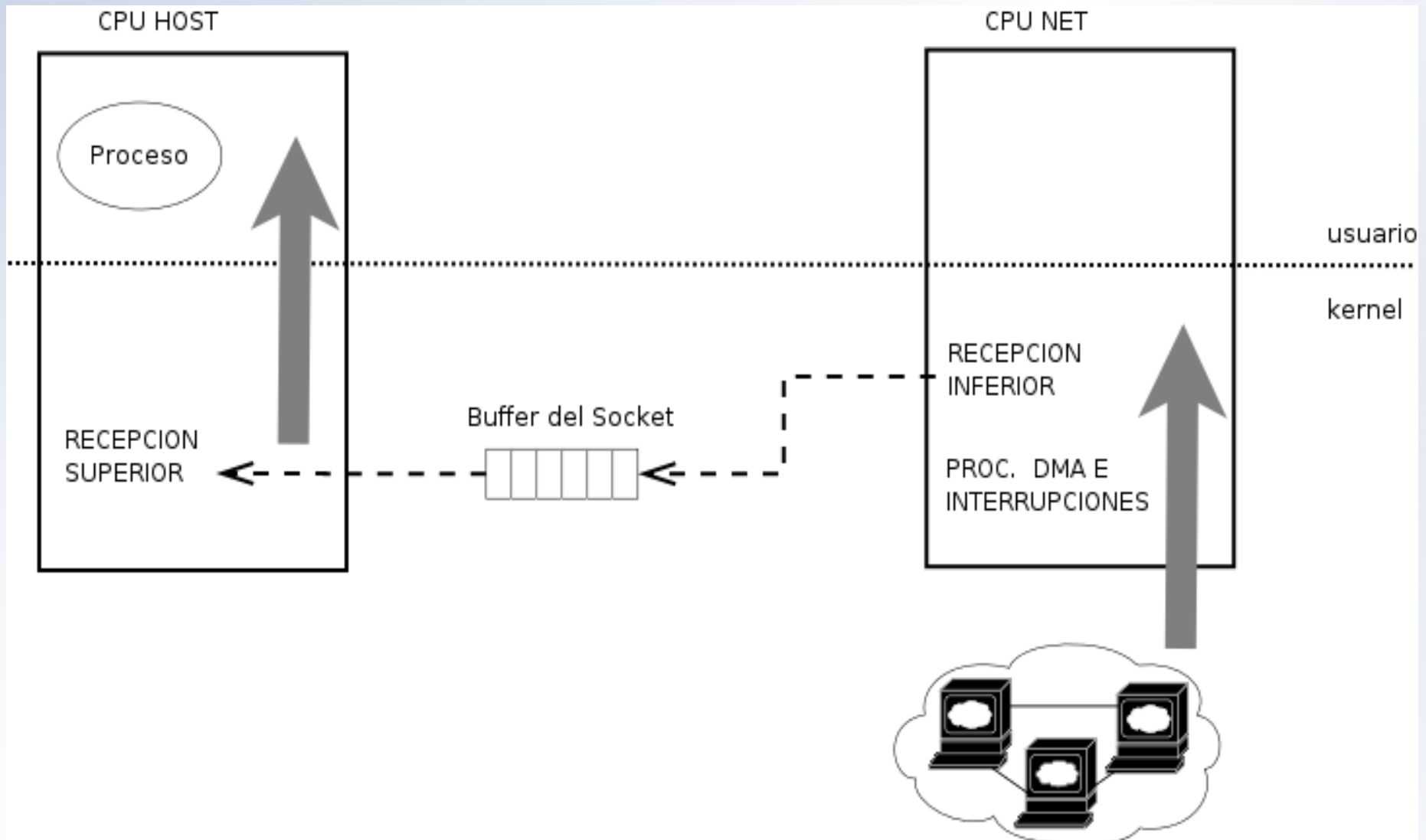
Diseño e implementación del prototipo

- El procesamiento de red es aislado del resto del kernel y los procesos de usuario
- CPU-HOST y CPU-NET
- Una CPU-NET no necesita ser multiplexada, entonces se puede optimizar para su función
- Se evitan interrupciones y se reducen costos de sincronización y cambios de contexto

Diseño e implementación del prototipo – Transmisión



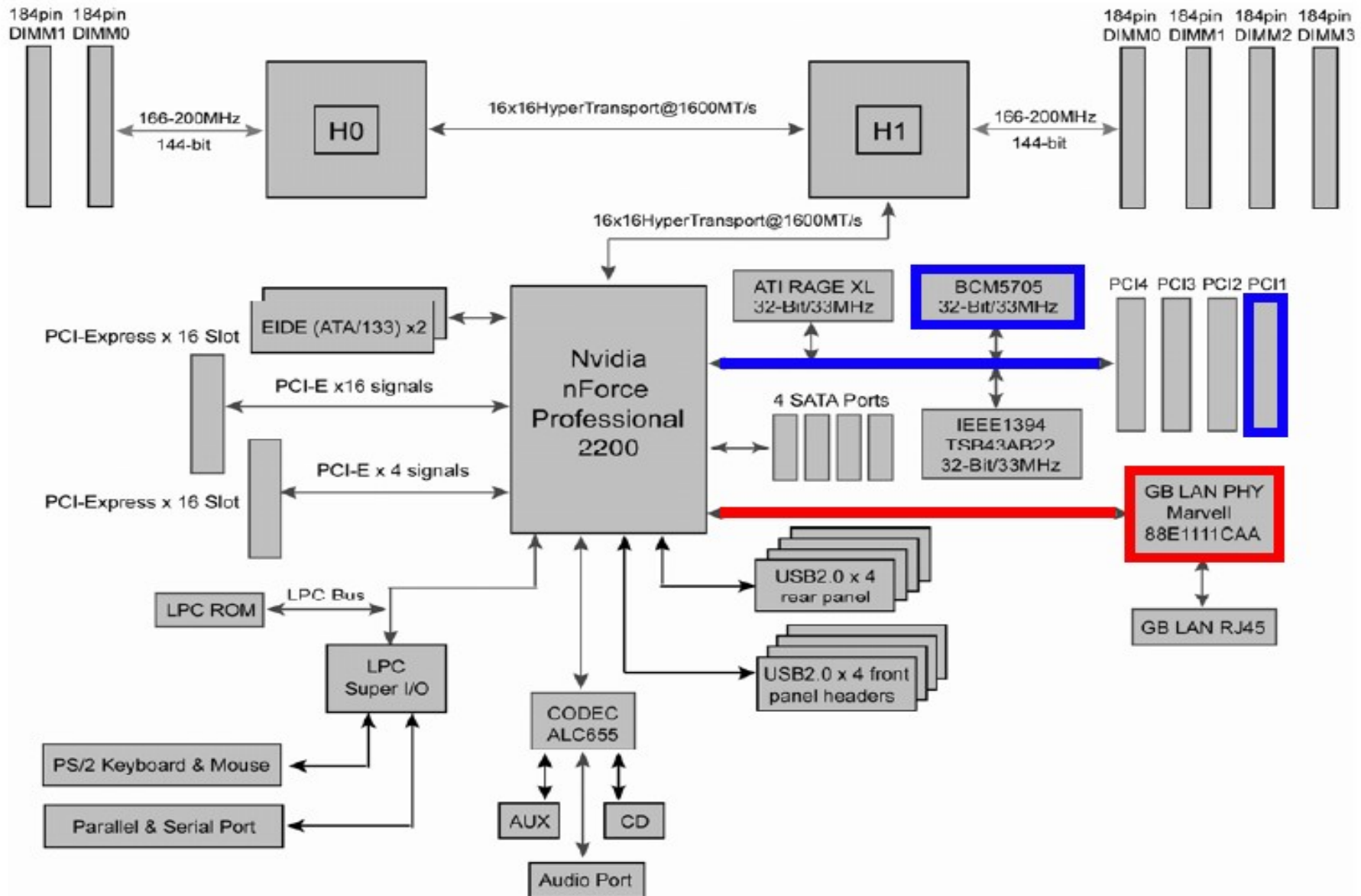
Diseño e implementación del prototipo – Recepción



Diseño e implementación del prototipo – Código

- Se modificó el kernel Linux 2.6.17.8 para implementar la arquitectura separada de CPU-HOST y CPU-NET
 - net/core/dev.c
 - include/linux/interrupt.h
 - net/ipv4/tcp_output.c
 - drivers/net/tg3.c
 - drivers/net/forcedeth.c
 - drivers/net/r8169.c
- Se creó un módulo del kernel que dispara el thread que toma el control de la CPU1, la cual cumplirá el rol de CPU-NET

Diseño e implementación del prototipo – Hardware



Mediciones y resultados preliminares – Herramientas

- **Netperf:** permite medir el rendimiento de red promedio de un enlace
- **Sar:** gran cantidad de indicadores de performance en una máquina SMP identificándolos por CPU
- **Oprofile:** profiler estadístico que usa los contadores de performance que ofrecen los procesadores modernos y que permiten obtener información detallada de la ejecución de los procesos, librerías y el propio kernel

Mediciones y resultados preliminares – Pruebas A,B,C

- **Prueba A)** Se activa sólo una CPU de la máquina SMP para todo el procesamiento.
- **Prueba B)** En este caso se usan ambas CPUs para procesar la actividad del kernel y procesos de usuario. Las interrupciones se distribuyen entre los procesadores, y los procesos de usuario se migran cada vez que el scheduler decida balancear la carga.
- **Prueba C)** En esta prueba se asocia el procesamiento de cada NIC con una CPU en particular. Es decir, en una misma CPU se ejecutan: el manejador de interrupciones de una de las NICs, la softirq correspondiente y el proceso de Netperf que atiende el servidor asociado a la dirección IP de esa NIC.

Mediciones y resultados preliminares – Pruebas D,E

- **Prueba D)** se utiliza la CPU-NET para realizar el procesamiento de los siguientes componentes de TCP/IP:
 - Procesamiento de interrupciones
 - Recepción Inferior
- **Prueba E)** se utiliza la CPU-NET para realizar el procesamiento de los siguientes componentes de TCP/IP:
 - Procesamiento de interrupciones
 - Recepción Inferior
 - Transmisión Inferior

Mediciones y resultados preliminares – Pruebas X,Z

- **Prueba X)** En esta prueba se asocia el procesamiento de cada NIC con una CPU en particular. Es decir, en una misma CPU se ejecutan: el manejador de interrupciones de una de las NICs, la softirq correspondiente y el proceso de Netperf que atiende el servidor asociado a la dirección IP de esa NIC
- **Prueba Z)** se utiliza la CPU-NET para realizar el procesamiento de los siguientes componentes de TCP/IP:
 - Procesamiento de interrupciones
 - Recepción Inferior
 - Transmisión Inferior

Mediciones y resultados preliminares

Cuadro 8.7: Comparación de pruebas X y Z

medidas\pruebas	X	Z
Rendimiento(mbps)	1492	1477
%Idle general	34,44	24,77
cswch/s	157186,17	99275,49
intr/s	131363	0

Cuadro 8.1: Comparación de pruebas A, B, C, D y E

medidas\pruebas	A	B	C	D	E
Rendimiento(Mbps)	1379	1504	1482	1490	1418
%Idle	50,03	15,94	42,14	22,61	25,19
Cswch/s	44617,77	50454,35	222893,68	131499,63	101608,68
Intr/s	121823	105444	133873	0	0

Mediciones y resultados preliminares

Cuadro 8.9: Profiling del kernel durante la prueba Z

muestras	%	muestras	%	símbolo
148053	10.3428	116811	38.1698	_copy_to_user_ll
120776	8.4372	1291	0.4219	net_rx_action
95700	6.6855	26140	8.5416	desencolar
68739	4.8020	1621	0.5297	qdisc_restart
65176	4.5531	6827	2.2308	tcp_v4_rcv
59659	4.1677	8393	2.7495	sk_data_ready
50669	3.5397	8619	2.7911	net_rx_action
48140	3.3630	13402	4.3221	net_rx_action
43827	3.0617	9406	2.9871	net_rx_action
36767	2.5685	2824	0.8911	net_rx_action
30361	2.1210	2063	0.6531	net_rx_action
28993	2.0254	9818	3.0441	net_rx_action

Cuadro 8.8: Profiling del kernel durante la prueba X

muestras	%	muestras	%	símbolo
154270	13.1928	93867	49.7272	_copy_to_user_ll
153381	13.1167	1070	0.5668	_spin_lock
94636	8.0930	2845	1.5072	handle_IRQ_event
58272	4.9833	3033	1.6068	_do_IRQ
45311	3.8749	3621	1.9183	schedule
39026	3.3374	4737	2.5095	tcp_v4_rcv
30196	2.5823	1067	0.5653	qdisc_restart
26420	2.2594	3570	1.8913	_switch_to
24781	2.1192	2303	1.2200	default_idle

Conclusiones y trabajo a futuro

- Conclusiones
- Trabajo a futuro

Conclusiones

- El procesamiento de red es demasiado costoso
- Se comprobó que una parte significativa del procesamiento de red corresponde a la intrusión del sistema operativo
 - copias en memoria
 - interrupciones
 - sincronización
- Se demostró que es posible reducir la intrusión del sistema operativo generada por el procesamiento de E/S de red si se construye un kernel pensado para esta tarea

Trabajo a futuro

- Investigar otros usos para kernels activos: virtualización, tiempo real
- Estudiar mejoras en la performance de la interfaz entre el kernel y los procesos de usuario, a través del uso de memoria compartida y AIO
- Implementar la posibilidad de reconfiguración dinámica del subconjunto de CPUs dedicadas al rol de CPU-NET, dependiendo de la carga de procesamiento de red en un momento dado
- Estudiar las posibles aplicaciones de este trabajo en un entorno de Grid Computing, en particular para un *Storage Element*

Gracias por su atención – Preguntas

