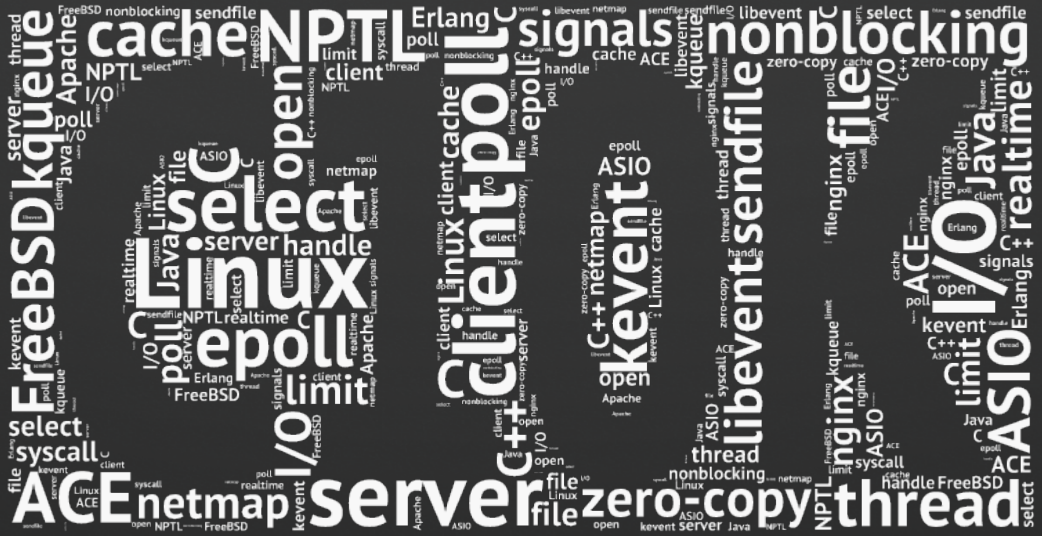
从C10K到C10M

高性能网络的探索与实践


cache
NPTL
Erlang
signals
nonblocking
kqueue
FreeBSD
select
Linux
epoll
limit
open
client
poll
kevent
ASIO
sendfile
libevent
file
I/O
realtime
Java
ACE
netmap
server
zero-copy
thread
syscall
nginx
handle
C++
Apache

C10K Killer

Model

- blocking/nonblocking
- synchronous/asynchronous
- multiplexing

Implement

- epoll/kqueue/iocp

Then

Framework

libevent/libev/boost::asio

Software

nginx/haproxy/squid

More Performance

CPU Affinity & Data Locality

RPS/RFS/RSS/
XPS

IRQ Tuning

Kernel Tuning

CPU Affinity & Data Locality

sched_set_affinity()
or
taskset/numactl

RPS/RFS/RSS/ XPS

RPS – Receive Packet Steering
RFS – Receive Flow Steering
RSS – Receive Side Scaling (hardware)
XPS – Transmit Packet Steering

IRQ Tuning

Interrupt Coalescing

NAPI/Newer newer NAPI

SMP IRQ Affinity

echo $bitmask > /proc/irq/{#}/smp_affinity

Kernel Tuning

Offload

TSO/GSO LRO/GRO

Kernel Parameter

net.ipv4.tcp_*/net.core.*

Next, we talk about ..

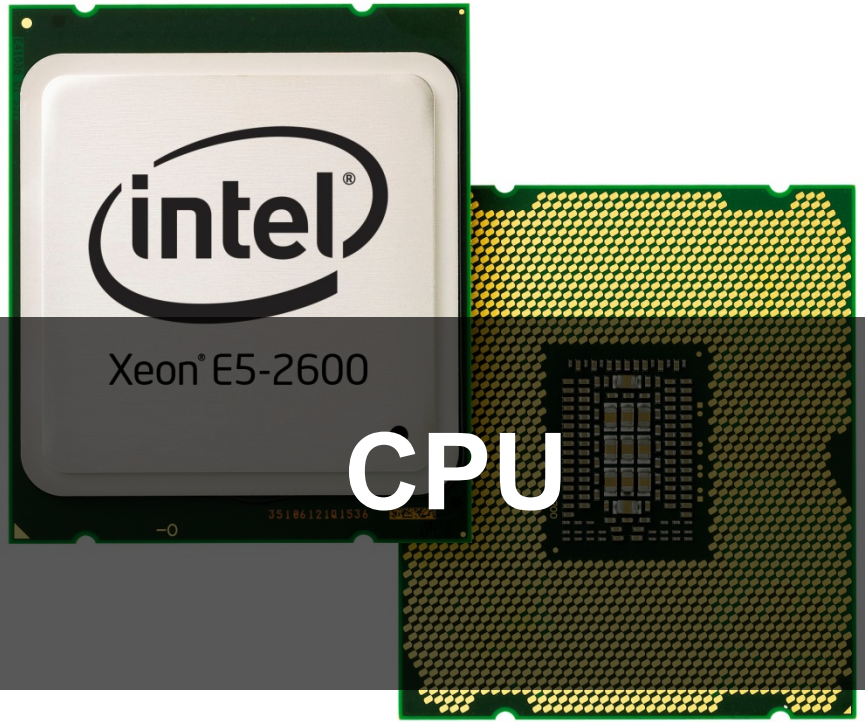
{

Hardware

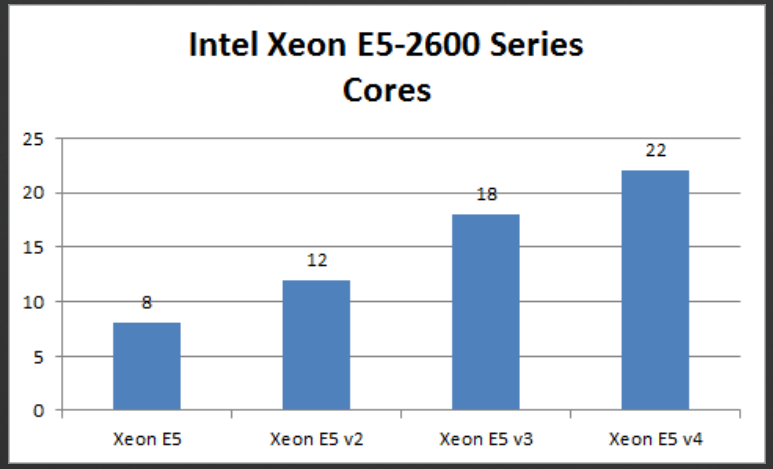
OS

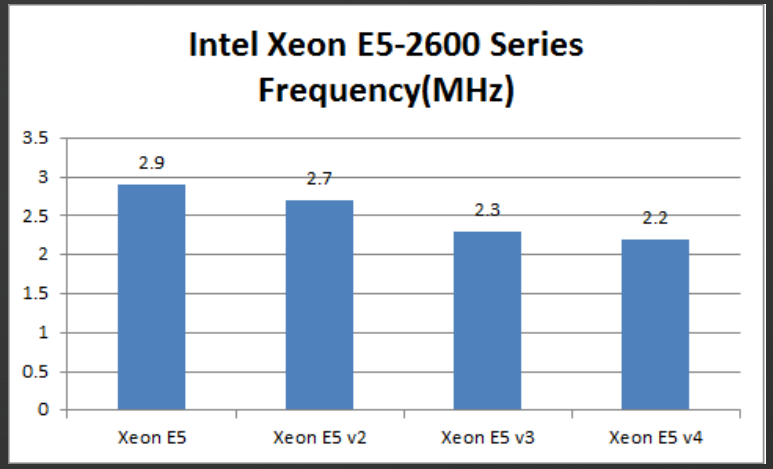
Application

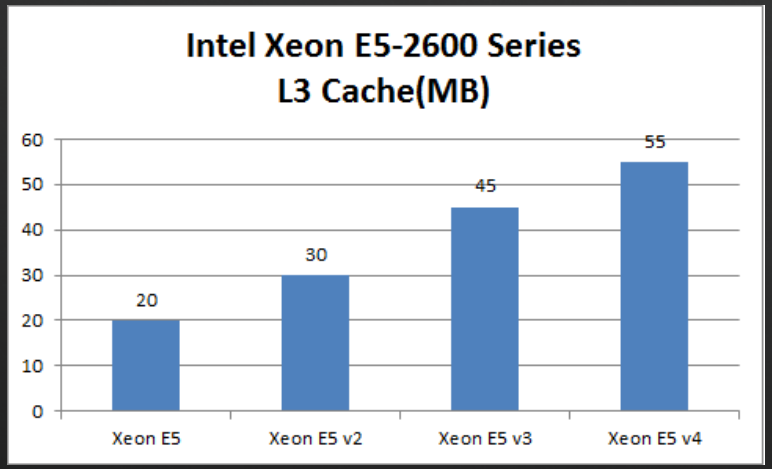
Modern Hardware

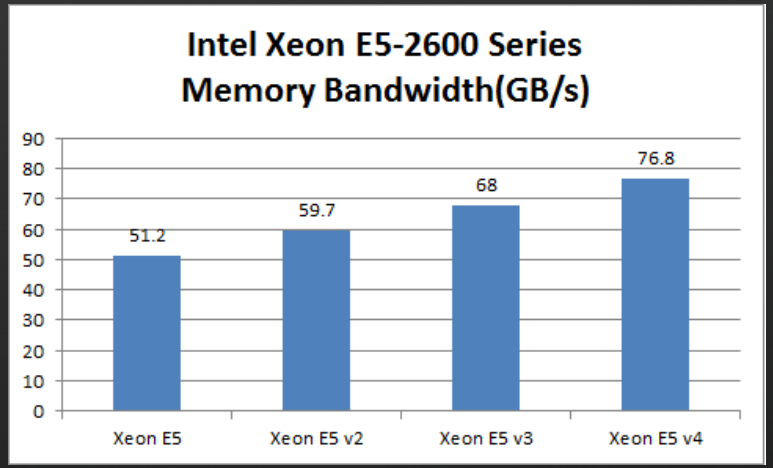


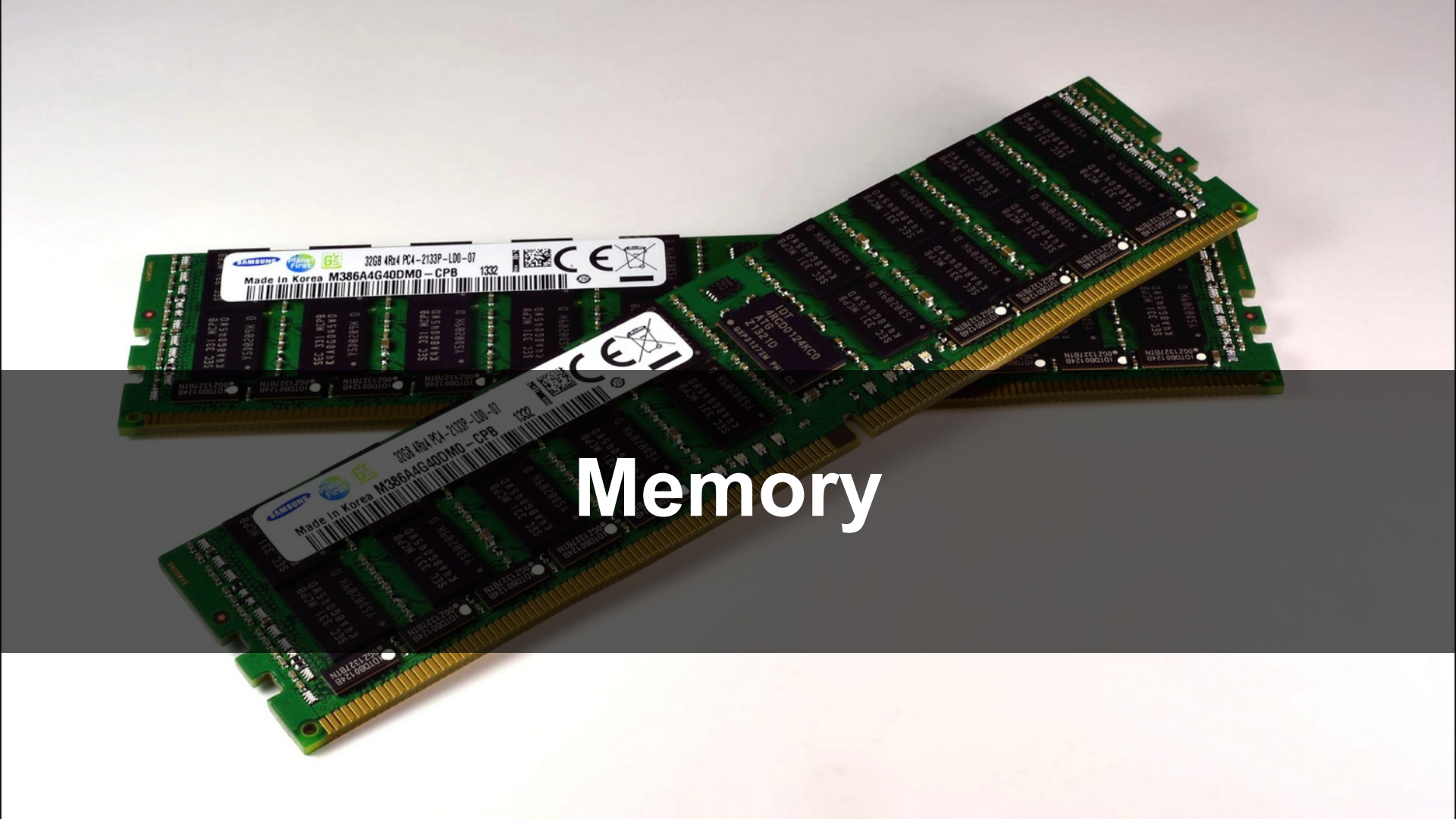
CPU


Intel Xeon E5-2600 Series
Cores
25
20
15
10
5
0
8
12
18
22
Xeon E5
Xeon E5 v2
Xeon E5 v3
Xeon E5 v4


Intel Xeon E5-2600 Series
Frequency(MHz)
3.5
3
2.5
2
1.5
1
0.5
0
2.9
2.7
2.3
2.2
Xeon E5
Xeon E5 v2
Xeon E5 v3
Xeon E5 v4


Intel Xeon E5-2600 Series
L3 Cache(MB)
60
50
40
30
20
10
0
20
30
45
55
Xeon E5
Xeon E5 v2
Xeon E5 v3
Xeon E5 v4


Intel Xeon E5-2600 Series
Memory Bandwidth(GB/s)
90
80
70
60
50
40
30
20
10
0
51.2
59.7
68
76.8
Xeon E5
Xeon E5 v2
Xeon E5 v3
Xeon E5 v4


Made in Korea M386A4G40DM0 – CPB
Made in Korea M386A4G40DM0 – CPB

Memory

DDR4 Transition by Segment

Servers: full support in 2014

- Small DDR4 volumes in 1H 2014 – validation phase

  Volumes picked up in 2H 2014

  >50% adoption by end of 2015

Technology Forecast

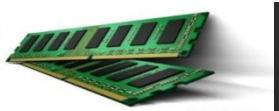
- Mobile DRAM share was down slightly in Q1-15
  - 32% of bits shipped in Q1-15 were mobile DRAM (from 34% in Q4-14)
  - Expected to settle at 45%-50% on annual basis
  - Peak quarterly share will exceed 50%

Upcoming Intel® Platforms Continue to Support DDR4

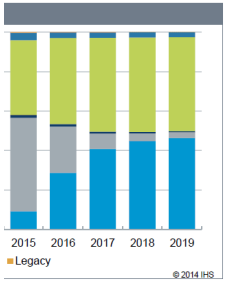
Intel® Xeon® Processor E5-2600 v4 product family support DDR4 up to 2400.

Mainstream PCs now supporting DDR4, 6th Generation Intel® Core™ processor 6000 series now supporting DDR4 up to 2133.

Memory Ecosystem ready for DDR4 on future Intel Xeon and Intel Core Processor Extreme Editions. Inquire about DDR4 availability and pricing through your preferred memory vendor.



2015

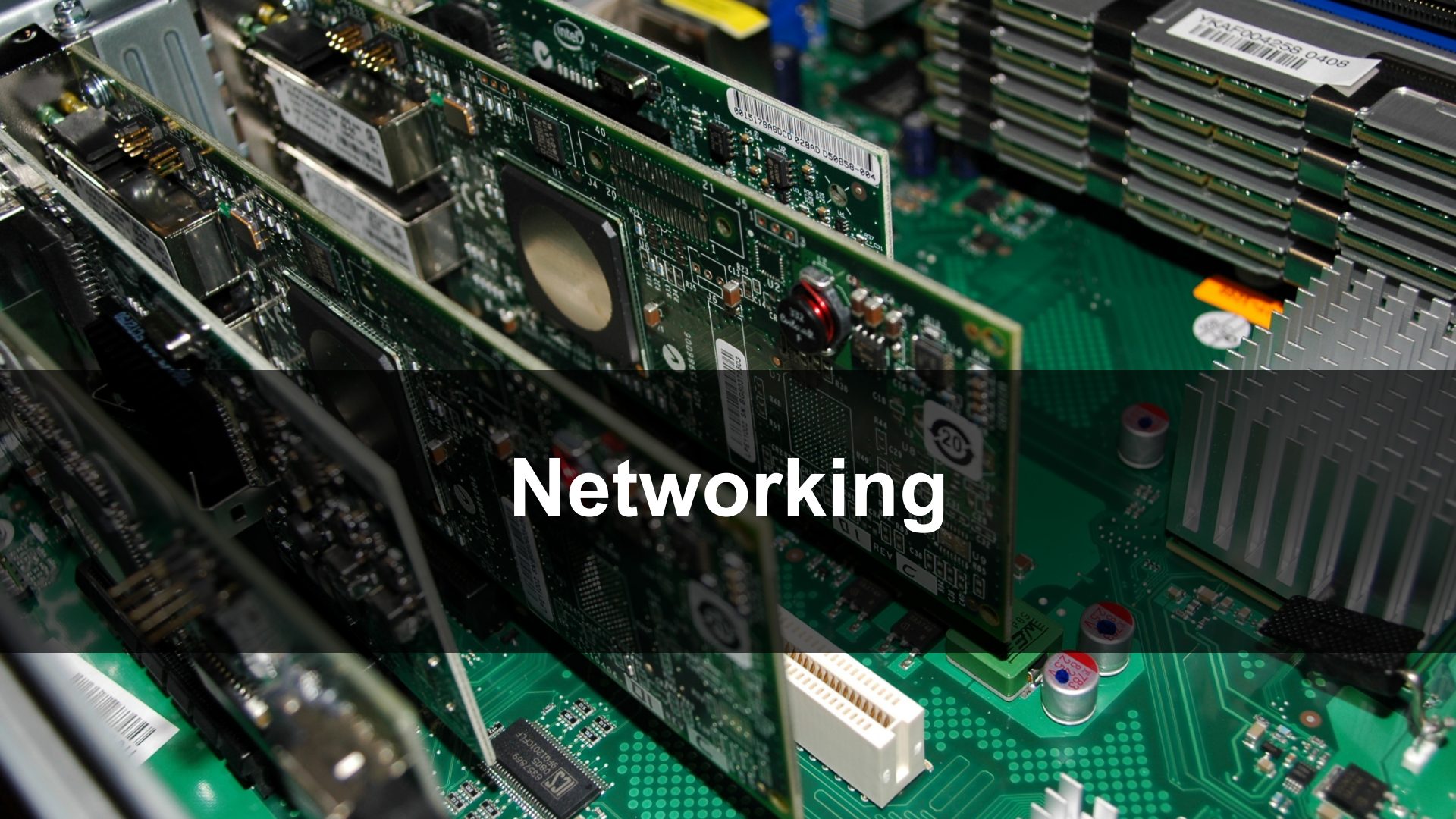


Desktops: minor support 2014, widespread in 2015

Small DDR4 penetration in 2H 2014

Adoption ramps quickly in 2016

IDF15 INTEL DEVELOPER FORUM



Networking

Intel® Ethernet Controllers and PHYs

Product	Codename	Device	Package Size/ Physical Package	Host Interface/ Bus Type	Performance Features
10/40 GbE Silicon					
Intel® XL710-AM2 	Fortville	Dual Port 40GbE MAC/PHY, KR4/CR4/XLAUI/XLPPI, KR/SFI/ XAUI, KX4/KX/SGMII	25x25 mm 576-pin Flip-Chip	PCI Express* v3.0 (8.0 GT/s) x8/x4/x1	1536 queue pairs in the device and 16 per virtual function, 128 VFs for the device, 384 VSI in the device. VXLAN, NVGRE and Geneve stateless offloads, Intel(r) Ethernet Flow Director, Optimized for Intel(r) Data Plane Developer Kit for small packet performance, FCoE CRC offload, FCoE direct data placement
Intel® XL710-AM2 	Fortville	Single Port 40GbE MAC/PHY, KR4/CR4/XLAUI/XLPPI, KR/ SFI/XAUI, KX4/KX/SGMII	25x25 mm 576-pin Flip-Chip	PCI Express* v3.0 (8.0 GT/s) x8/x4/x1	1536 queue pairs in the device and 16 per virtual function, 128 VFs for the device, 384 VSI in the device. VXLAN, NVGRE and Geneve stateless offloads, Intel(r) Ethernet Flow Director, Optimized for Intel(r) Data Plane Developer Kit for small packet performance, FCoE CRC offload, FCoE direct data placement
Intel® X710-AM2 	Fortville	Single Port 40GbE MAC/PHY, KR4/CR4/XLAUI/XLPPI, KR/ SFI/XAUI, KX4/KX/SGMII	25x25 mm 576-pin Flip-Chip	PCI Express* v3.0 (8.0 GT/s) x8/x4/x1	1536 queue pairs in the device and 16 per virtual function, 128 VFs for the device, 384 VSI in the device. VXLAN, NVGRE and Geneve stateless offloads, Intel(r) Ethernet Flow Director, Optimized for Intel(r) Data Plane Developer Kit for small packet performance, FCoE CRC offload, FCoE direct data placement
Intel® X540-AT2 Intel® X540-BT2 	Twinville	Dual Port 10GbE MAC/PHY, 10GBase-T, 1000Base-T, 100Base-T	25x25 mm 576-pin Flip-Chip	PCI Express* v2.1 (5.0 GT/s) x8/x4/x2/x1	128 Tx and 128 Rx queues, Receive Side Coalescing (RSC), low latency interrupts, Intel® Ethernet Flow Director, Intel® Virtualization Technology for Connectivity (Intel® VT-c) (VMDq, SR-IOV), FCoE CRC offload, FCoE direct data placement
Intel® 82599ES 	Niantic	Dual Port 10GbE MAC/PHY, KR, SFI, XAUI, KX/KX4, CX4, BX, SGMII	25x25 mm 576-pin Flip-Chip	PCI Express* v2.0 (5.0 GT/s) x8/x4/x2/x1	128 Tx and 128 Rx queues, Receive Side Coalescing (RSC), low latency interrupts, Intel® Ethernet Flow Director, Intel® Virtualization Technology for Connectivity (Intel® VT-c) (VMDq, SR-IOV), FCoE CRC offload, FCoE direct data placement
Intel® 82599EB 	Niantic	Dual Port 10GbE MAC/PHY, XAUI, KX/KX4, CX4, BX, SGMII	25x25 mm 576-pin Flip-Chip	PCI Express* v2.0 (5.0 GT/s) x8/x4/x2/x1	128 Tx and 128 Rx queues, Receive Side Coalescing (RSC), low latency interrupts, Intel® Ethernet Flow Director, Intel® Virtualization Technology for Connectivity (Intel® VT-c) (VMDq, SR-IOV), FCoE CRC offload, FCoE direct data placement
Intel® 82599EN 	Niantic	Single Port 10GbE MAC/ PHY, SFI	25x25 mm 576-pin Flip-Chip	PCI Express* v2.0 (5.0 GT/s) x8/x4/x2/x1	64 Tx and 64 Rx queues, Receive Side Coalescing (RSC), low latency interrupts, Intel® Ethernet Flow Director, Intel® Virtualization Technology for Connectivity (Intel® VT-c) (VMDq, SR-IOV), FCoE CRC offload, FCoE direct data placement


We are ready!


We are ready!
But…

Packet Processing

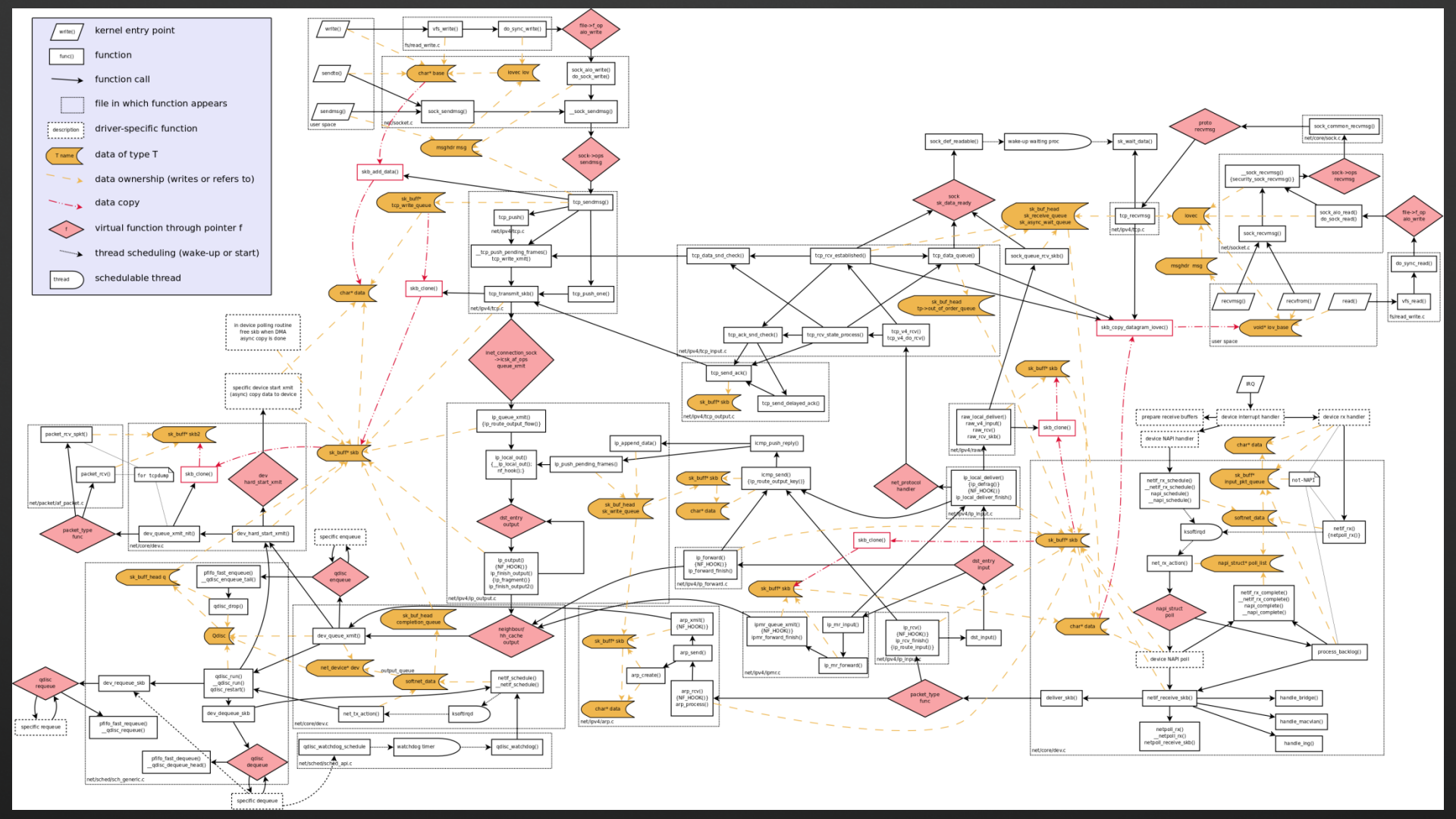
Task Scheduling

Data Access

The kernel isn’t the solution.

The kernel is the **problem**.

- Shmoocon 2013


kernel entry point
function
function call
file in which function appears
driver-specific function
data of type T
data ownership (writes or refers to)
data copy
virtual function through pointer f
thread scheduling (wake-up or start)
schedulable thread
write()
vfs_write()
do_sync_write()
file->f_op aio_write
fs/read_write.c
sendto()
sendmsg()
user space
char* base
iovec iov
sock_aio_write() do_sock_write()
sock_sendmsg()
__sock_sendmsg()
net/socket.c
msghdr msg
sock->ops sendmsg
skb_add_data()
sk_buff* tcp_write_queue
tcp_sendmsg()
tcp_push()
net/ipv4/tcp.c
__tcp_push_pending_frames() tcp_write_xmit()
char* data
skb_clone()
tcp_transmit_skb()
tcp_push_one()
inet_connection_sock ->icsk_af_ops queue_xmit
in device polling routine free skb when DMA async copy is done
specific device start xmit (async) copy data to device
tcp_data_snd_check()
tcp_rcv_established()
tcp_data_queue()
sock_queue_rcv_skb()
tcp_ack_snd_check()
tcp_rcv_state_process()
tcp_v4_rcv() tcp_v4_do_rcv()
net/ipv4/tcp_input.c
tcp_send_ack()
sk_buff* skb
tcp_send_delayed_ack()
net/ipv4/tcp_output.c
sock_def_readable()
wake-up waiting proc
sk_wait_data()
sock sk_data_ready
sk_buff_head sk_receive_queue sk_async_wait_queue
tcp_recvmsg
net/ipv4/tcp.c
sk_buff_head tp->out_of_order_queue
skb_copy_datagram_iovec()
proto recvmsg
sock_common_recvmsg()
net/core/sock.c
__sock_recvmsg() (security_sock_recvmsg())
sock->ops recvmsg
sock_aio_read() do_sock_read()
file->f_op aio_write
sock_recvmsg()
net/socket.c
iovec
msghdr msg
recvmsg()
recvfrom()
read()
void* iov_base
user space
do_sync_read()
vfs_read()
fs/read_write.c
packet_rcv_spkt()
packet_rcv()
net/packet/af_packet.c
sk_buff* skb2
for tcpdump
skb_clone()
dev hard_start_xmit
packet_type func
dev_queue_xmit_nit()
dev_hard_start_xmit()
net/core/dev.c
ip_queue_xmit() (ip_route_output_flow())
ip_local_out() (__ip_local_out() nf_hook():)
ip_append_data()
ip_push_pending_frames()
icmp_push_reply()
dst_entry output
sk_buff_head sk_write_queue
ip_output() (NF_HOOK()) ip_finish_output() (ip_fragment()) ip_finish_output2()
net/ipv4/ip_output.c
sk_buff* skb
char* data
icmp_send() (ip_route_output_key())
net protocol handler
ip_local_deliver() (ip_defrag() (NF_HOOK()) ip_local_deliver_finish())
net/ipv4/ip_input.c
raw_local_deliver() raw_v4_input() raw_rcv() raw_rcv_skb()
net/ipv4/raw.c
skb_clone()
ip_forward() (NF_HOOK()) ip_forward_finish()
net/ipv4/ip_forward.c
dst_entry input
IRQ
prepare receive buffers
device interrupt handler
device rx handler
device NAPI handler
char* data
netif_rx_schedule() __netif_rx_schedule() napi_schedule() __napi_schedule()
sk_buff* input_pkt_queue
not-NAPI
softnet_data
netif_rx() (netpoll_rx())
ksoftirqd
net_tx_action()
napi_struct* poll_list
napi_struct poll
netif_rx_complete() __netif_rx_complete() napi_complete() __napi_complete()
device NAPI poll
process_backlog()
netif_receive_skb()
deliver_skb()
handle_bridge()
handle_macvlan()
handle_ing()
netpoll_rx() __netpoll_rx() netpoll_receive_skb()
net/core/dev.c
specific enqueue
qdisc enqueue
sk_buff_head q
pfifo_fast_enqueue() __qdisc_enqueue_tail()
qdisc_drop()
Qdisc
sk_buff_head completion_queue
neighbour/ hh_cache output
net_device* dev
output_queue
softnet_data
netif_schedule() __netif_schedule()
net_tx_action()
ksoftirqd
dev_queue_xmit()
qdisc_run() __qdisc_run() qdisc_restart()
dev_dequeue_skb
net/core/dev.c
arp_xmit() (NF_HOOK())
arp_send()
arp_create()
arp_rcv() (NF_HOOK()) arp_process()
char* data
net/ipv4/arp.c
ipmr_queue_xmit() (NF_HOOK()) ipmr_forward_finish()
ip_mr_input()
ip_mr_forward()
net/ipv4/ipmr.c
ip_rcv() (NF_HOOK()) ip_rcv_finish() (ip_route_input())
dst_input()
net/ipv4/ip_input.c
packet_type func
qdisc requeue
dev_requeue_skb
pfifo_fast_requeue() __qdisc_requeue()
specific requeue
qdisc dequeue
pfifo_fast_dequeue() __qdisc_dequeue_head()
specific dequeue
net/sched/sch_generic.c
qdisc_watchdog_schedule
watchdog timer
qdisc_watchdog()
net/sched/sch_api.c

Global accept queue ☹

Global established table ☹

Optimize networking stack ☺

Move networking stack to userspace ☺

Hook networking stack ☺

Userspace I/O ☺

Raw Packets?

Maybe I need a protocol stack.


CPU3
CPU0
CPU1
CPU2

Multiple process/thread

!=

Multiple core

Connection locality ☺

Lockless data structures ☺

Keep data structures per core ☺

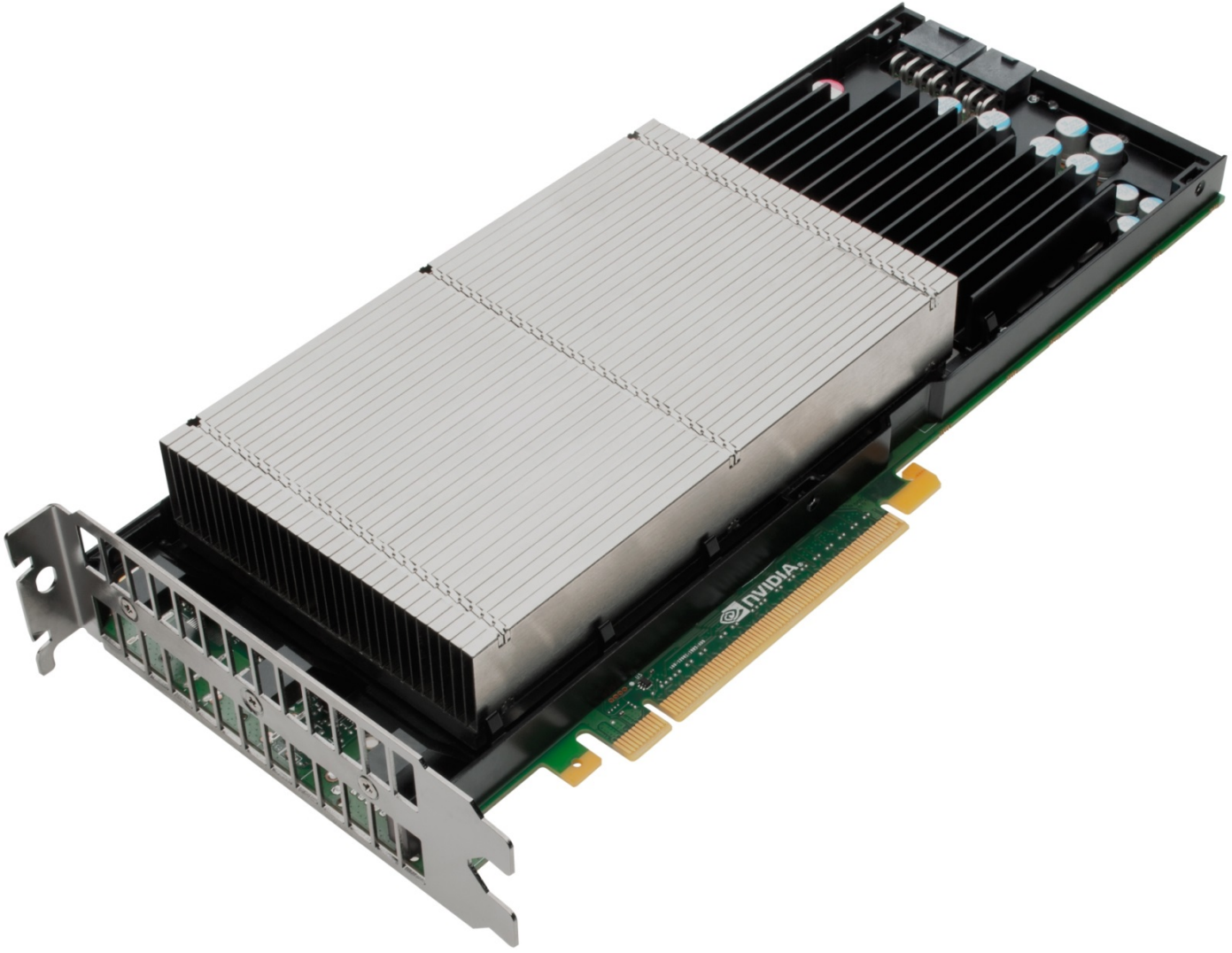
Latency Comparison Numbers

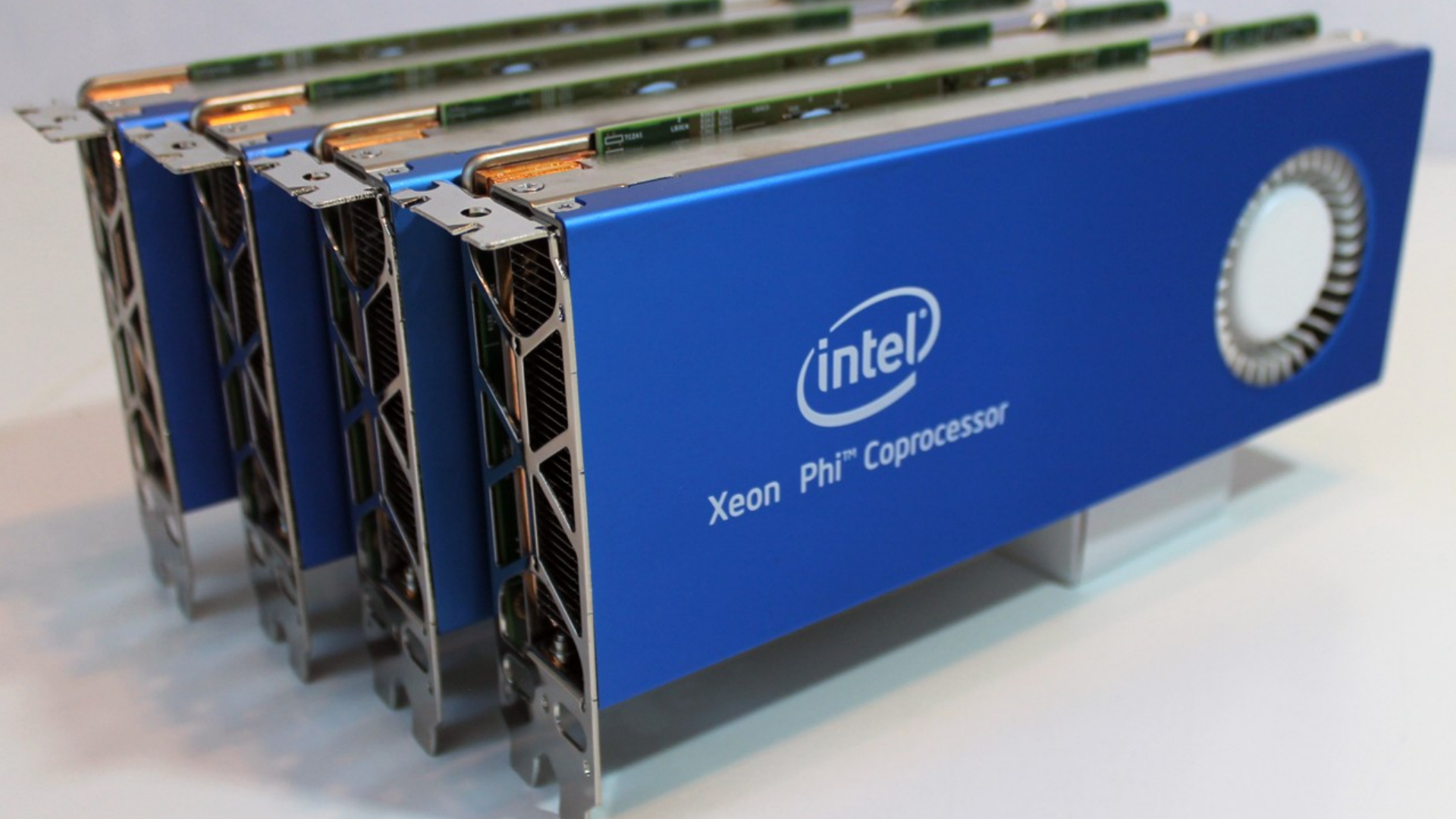
Latency Comparison Numbers		
Register/Buffer reference	0.4 ns	(1 cycle)
L1 cache reference	1.6 ns	(4 cycles)
L2 cache reference	4.8 ns	(12 cycles)
L3 cache reference	10.4 ns	(26 cycles)
Linux SYS_gettid syscall	50.0 ns	
Main Memory reference	61.0 ns	
Process context switches with CPU affinity	1000.0 ns	
Process context switches without CPU affinity	1300.0 ns	
Send 1K bytes over 1 Gbps network	10000.0 ns	
Read 4K randomly from SSD	150000.0 ns	
Read 1 MB sequentially from memory	250000.0 ns	
Round trip within same datacenter	500000.0 ns	
Read 1 MB sequentially from SSD	1000000.0 ns	
CPU: Intel(R) Core(TM) i7-5500U CPU @ 2.40GHz		
RAM: 8 GB (Double PC3L-12800 11-11-11-28)		

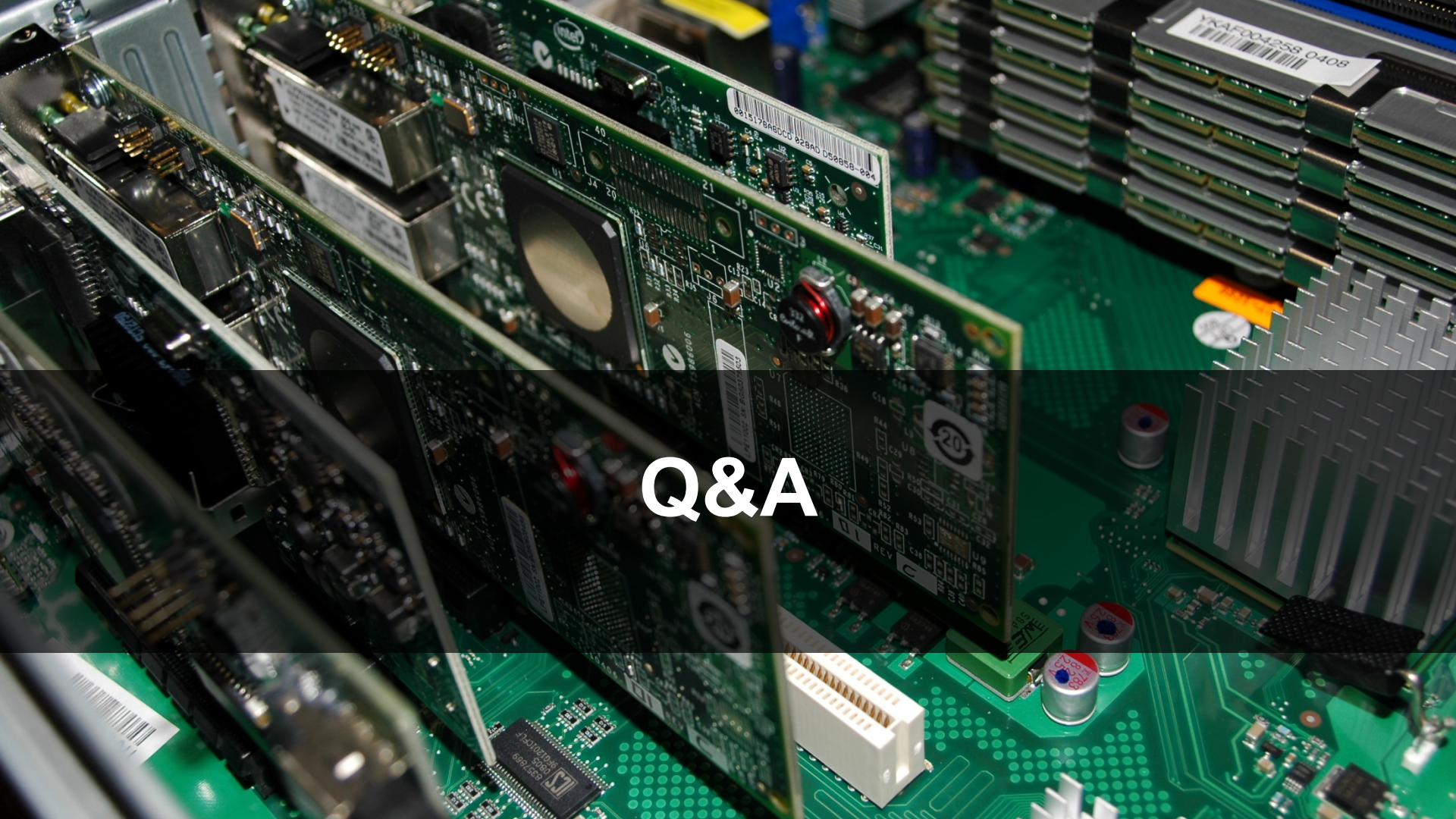
No nested pointer ☺

Hugepages ☺

Preallocate memory ☺


NVIDIA


intel
Xeon Phi™ Coprocessor


Q&A