LINUX NUMA OPTIMIZATION TOI

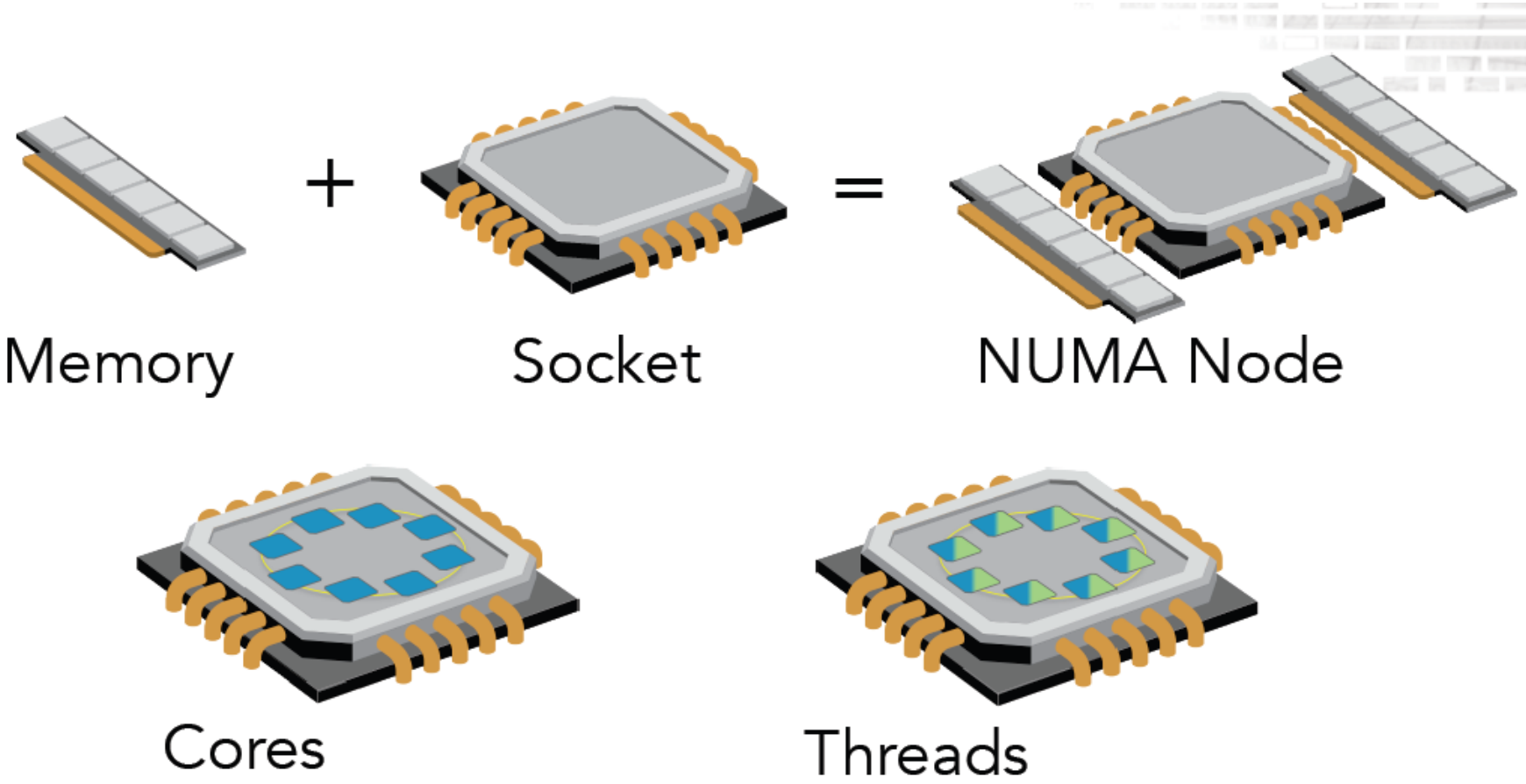
Oliver Yang

Feb, 2016

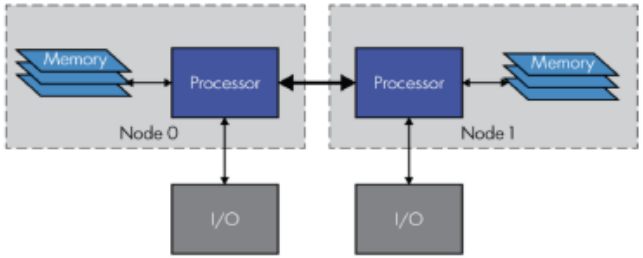
Agenda

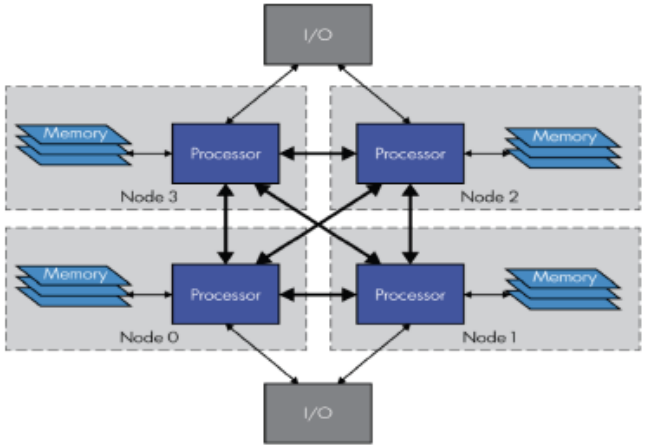
- Overview
- Kernel Facilities
- Known Gaps
- Tools

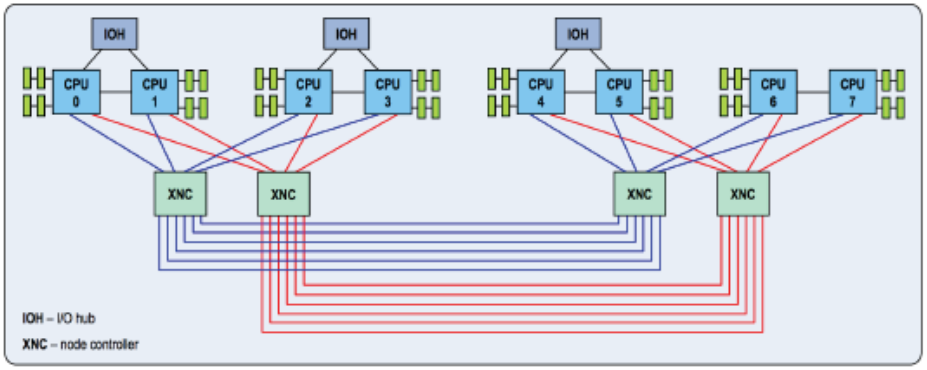
Nodes, Sockets, Cores, Threads



2/4/8 Sockets Servers







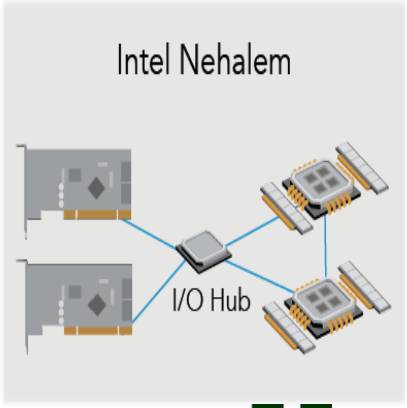

```
# numactl --hardware
node   0   1   2   3   4   5   6   7
  0:  10  12  17  17  19  19  19  19
  1:  12  10  17  17  19  19  19  19
  2:  17  17  10  12  19  19  19  19
  3:  17  17  12  10  19  19  19  19
  4:  19  19  19  19  10  12  17  17
  5:  19  19  19  19  12  10  17  17
  6:  19  19  19  19  17  17  10  12
  7:  19  19  19  19  17  17  12  10
```

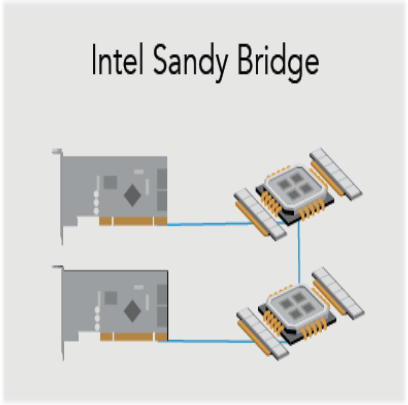
CPU NUMA Affinity

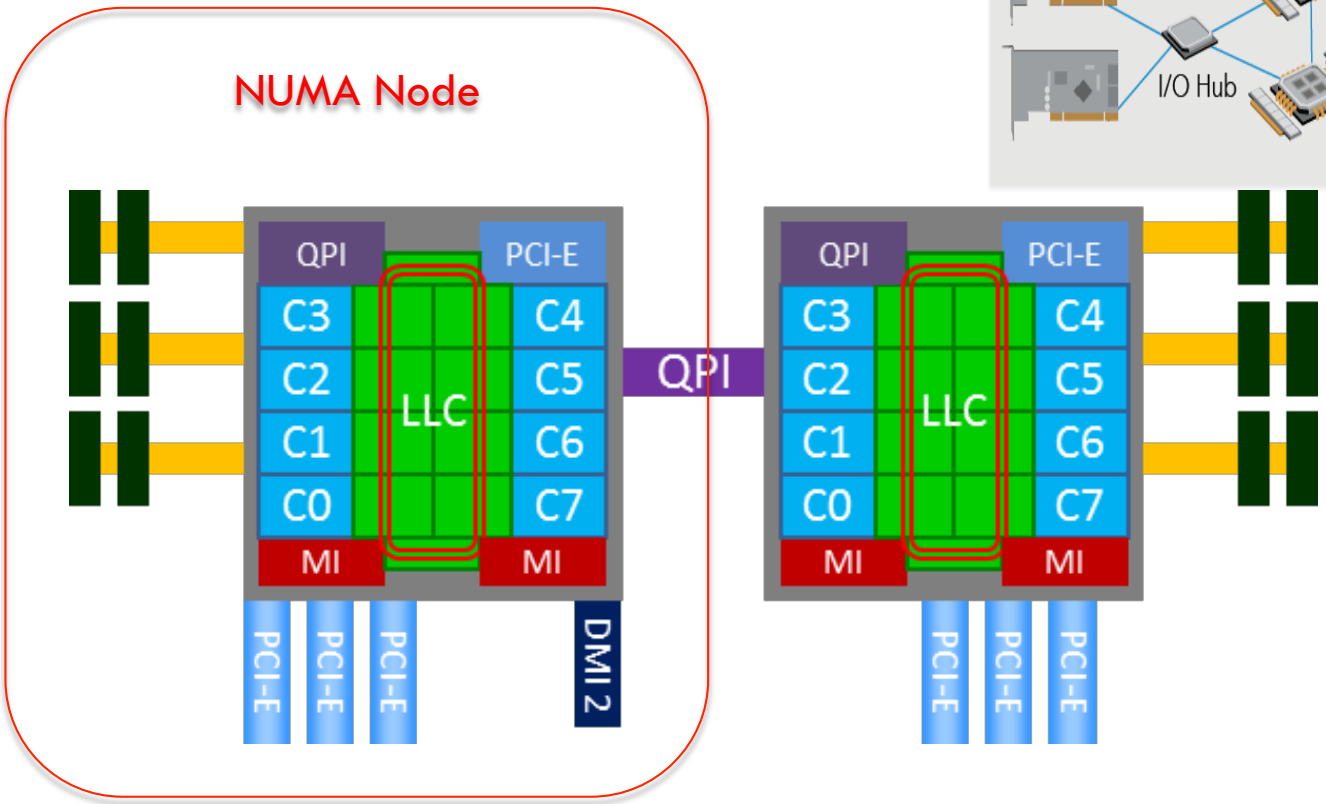
- x86 changes since Nehalem
  - Memory controller is integrated in die
  - Has local and remote memory access
- Worst case NUMA-node relative latency
  - Intel 4 socket / 4 node: 1.5x
  - AMD 4 socket / 8 node: 2.7x
  - 8 socket / 8 node: 2.8x
  - 32 node blade system: 5.5x

Device NUMA Affinity

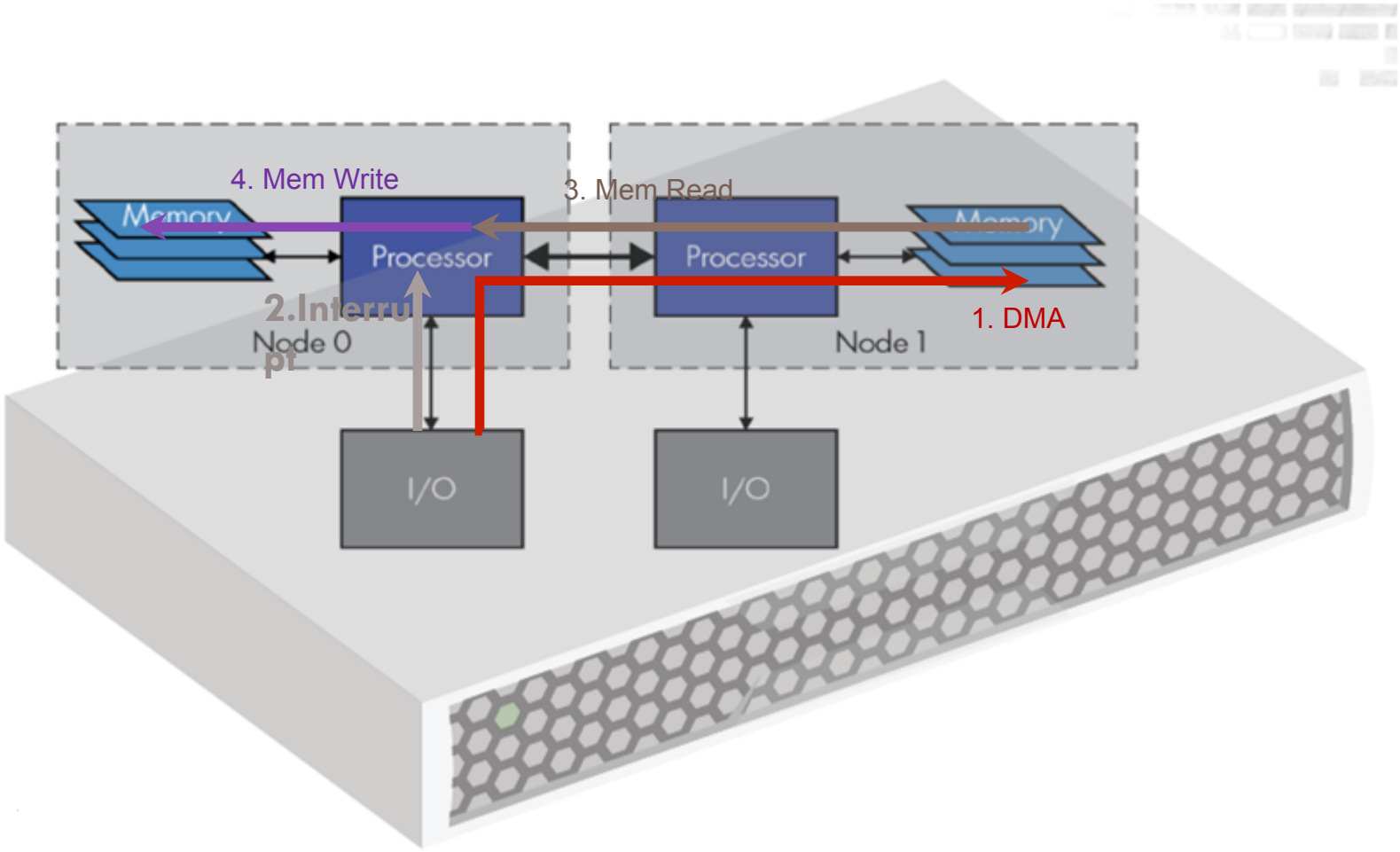
- Changes after the Sandy Bridge
  - No I/O hubs
  - PCIe Root-Complex is in the die
  - Devices connected to CPU directly







Why NUMA Affinities?



Agenda

- Overview
- Kernel Facilities
- Known Gaps
- Tools

Key Requirements

- Device affinity
  - DMA Memory locality
    - Are DMA buffers close to the devices?
  - Interrupt locality
    - Are interrupts close to DMA buffers?
- CPU affinity
  - Host Memory locality
    - Are memory buffers close to the running threads on the processors?
  - Scheduling locality
    - Are processes/threads close to its memory buffers?
  - Protocols stacks (Network/IO etc..) locality
    - Are data locality could be kept in the same NUMA nodes in the stack?

DMA Memory Locality

- Kernel buffer allocation
  - dev_to_node VS. numa_node_id
    - dev_to_node: get device numa node from device tree
    - numa_node_id: get memory numa node per current CPU id
  - Use dma_alloc_coherent directly
  - Specify device numa node id in kernel APIs
    - Use__netdev_alloc_skb instead of alloc_skb (not work for 3.2)

```
int node = dev->dev.parent ? dev_to_node(dev->dev.parent) : -1;
skb = __alloc_skb(length + NET_SKB_PAD, gfp_mask, 0, node);
```

    - Use kmalloc_node instead of kmalloc

```
int node = dev->dev.parent ? dev_to_node(dev->dev.parent) : -1;
data = kmalloc_node(rx_buf_sz, GFP_KERNEL, node);
```

    - Use alloc_pages_node instead of alloc_pages
- User space buffer allocation
  - No standard way, but could be done with kernel customizations.

Interrupt Locality

- Hardware IRQ
  - Capabilities of different type of interrupts

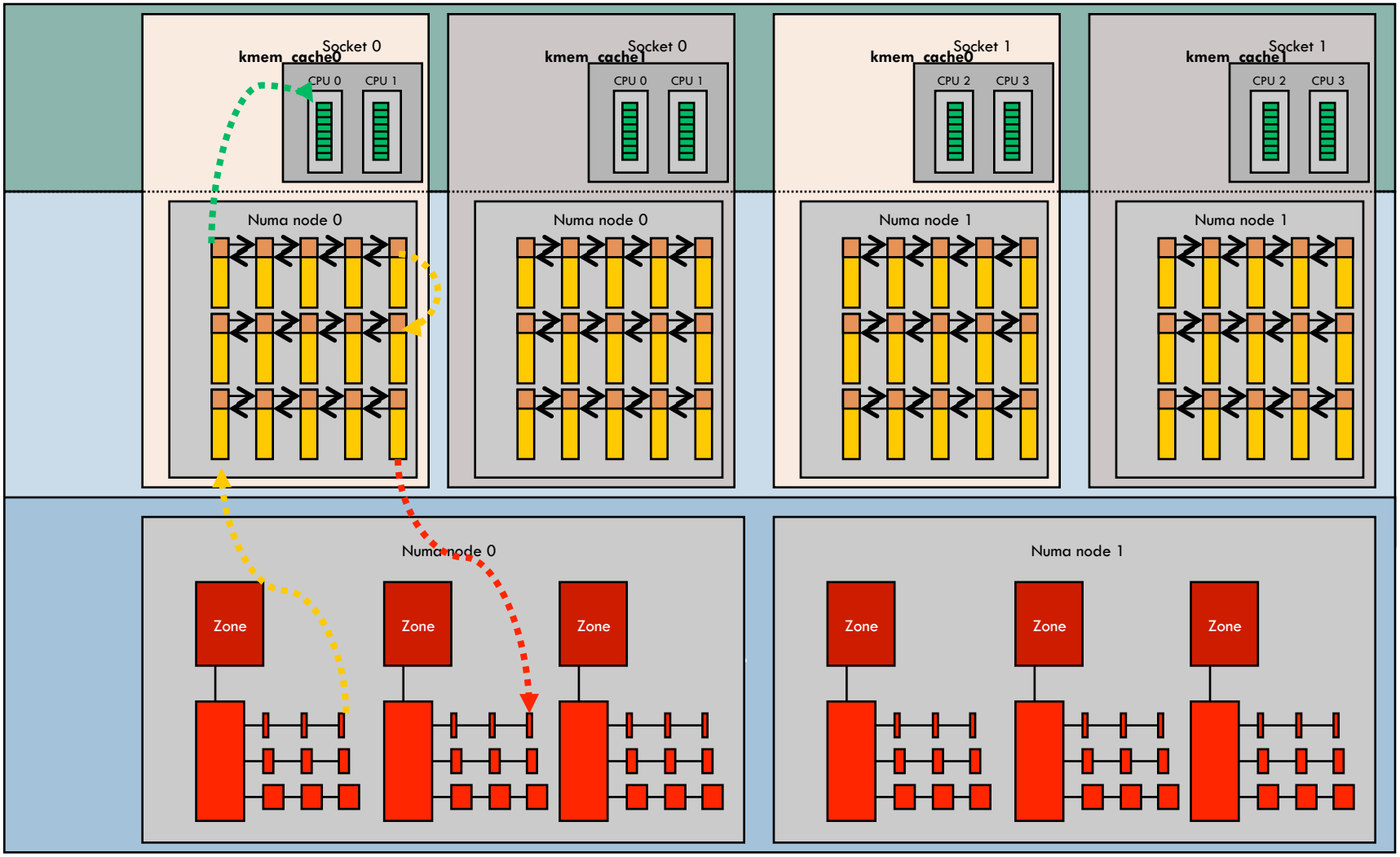
Type	Max Vectors	Bound to multi-CPUs	Shared
INTx	1	No	Yes
MSI	32	No	No
MSI-X	2048	Yes	NO

  - Local APIC interrupt delivery mode in Linux kernel
    - Logical mode when CPU <= 8
      - Local APIC chipset will do irq balance among 8 CPUs
      - No NUMA affinity issues under this mode on latest x86 platform
    - Physical flat mode when CPU > 8
      - One interrupt vectors just can be bound to one CPU at a time.
      - IRQ balance can break NUMA affinity
- Soft IRQ
  - After kernel code triggers it, a soft IRQ will be fired on the same core
    - IRQ balance could impact the affinity of soft IRQ as well

Host Memory Locality

- Kernel memory allocation
  - Slab allocator always returns memory from local NUMA node
    - kmalloc and kmem_cache_alloc are NUMA aware
    - kmalloc VS. kmalloc_node
      - kmalloc allocates memory from local NUMA node per caller's CPU.
      - kmalloc_node allocates memory per node arguments, which could be any of NUMA nodes.
  - Page allocator
    - alloc_pages returns page from local NUMA node by default
    - alloc_pages VS. alloc_pages_node
- User space memory allocation
  - Application hugetlb pages
    - OS kernel can do NUMA aware hugetlb reservation
  - Standard malloc in glibc also considers NUMA affinity
    - Kernel page fault will try to return the page from local NUMA node first
      - Local fault - memory on same node as CPU
      - Remote fault - memory on different node than CPU

NUMA Aware Kernel Memory Allocator



Scheduling Locality

- Scheduler is NUMA aware
  - Trade-off between affinity/locality and CPU availabilities.
    - Scheduling domain are based on ACPI table
      - Support SMT->SMP->NUMA topology and hierarchy
    - Threads migration policy reflects to different scheduling domain
      - Hyper thread CPUs level: No NUMA affinity issues
      - Physical CPUs level: lose cache locality but has the NUMA affinity
      - NUMA node level: Lose both cache locality and NUMA affinity
  - Short running tasks could have better localities
  - Threads migration might break the affinities for long running processes
    - Threads wakeup could trigger threads migration (Major reasons in Linux OS)
    - Runq load balance could be driven periodically by kernel.
- Avoid tasks migration to get better localities
  - Bind kernel thread to a specific CPU
    - Use kthread_create_cpu instead of kthread_create
  - Bind user processes to a specific CPU
    - Use mbind() and set_mempolicy() system call

Summary

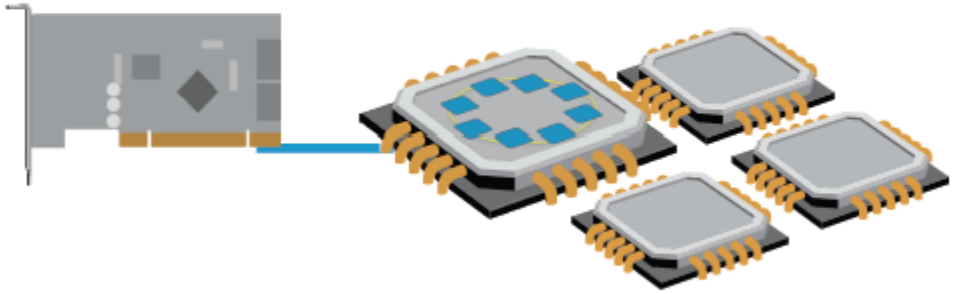
- Major NUMA features are available
  - Kernel APIs are NUMA aware
  - Core kernel services are NUMA aware
- There are some gaps…
  - Performance profiling tools
  - Micro-benchmarks
    - Throughput VS. Latency
  - Had a NUMA mechanism in kernel, but lack of end-to-end affinity from kernel to user space
    - DMA buffer affinity
    - Interrupt affinity
    - Kernel threads affinity (CPU/Cache/Memory)
    - Application threads affinity (CPU/Cache/Memory)

Agenda

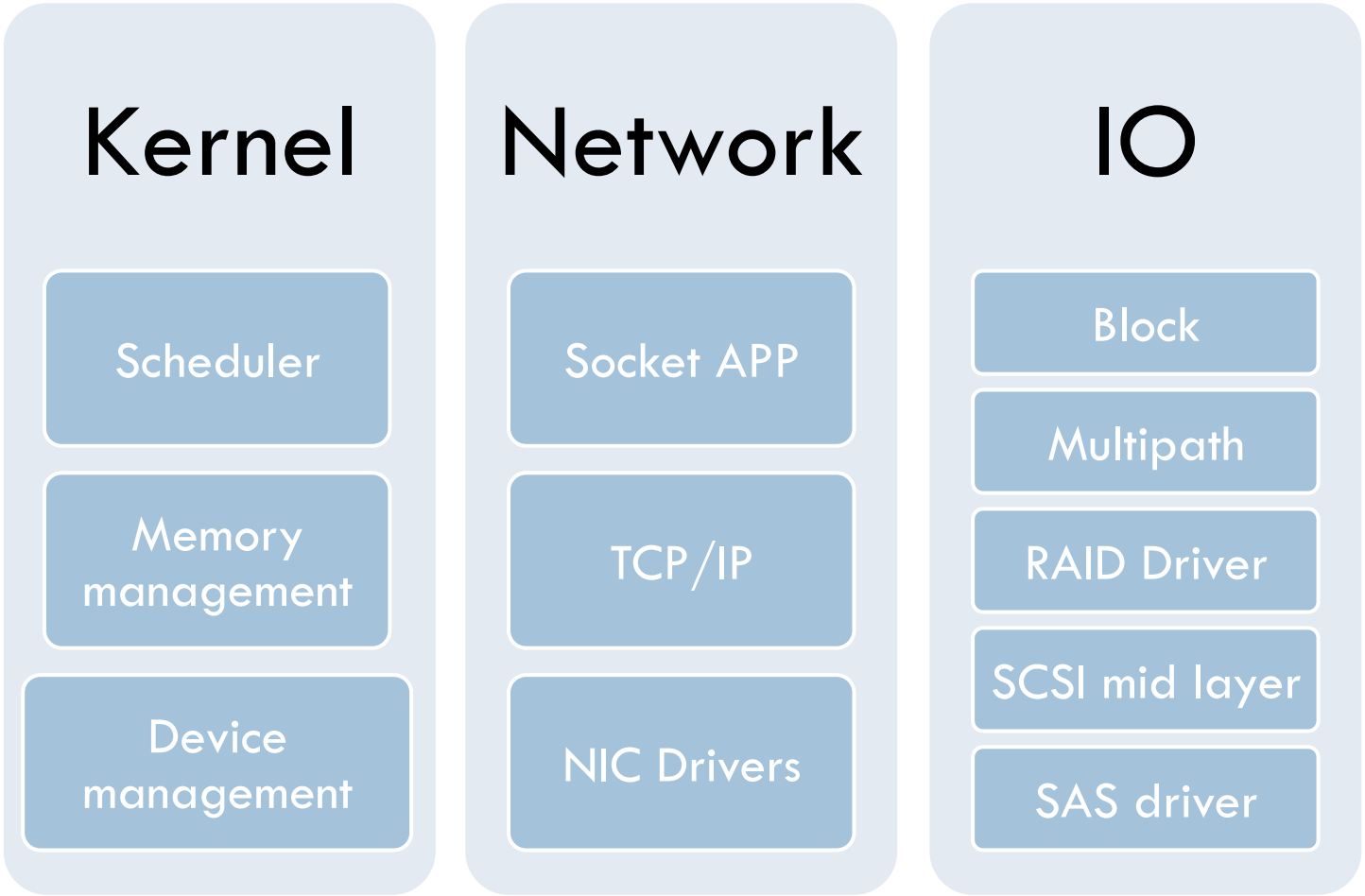
- Overview
- Kernel Facilities
- Known Gaps
- Tools

How To Identify Gaps

- Full stack efforts, need check OS whole data path
  - Hardware
    - SLIC placements vs. QPI/PCIe bandwidth
  - Firmware
    - Key ACPI tables
      - ACPI SRAT for CPU affinity information
      - ACPI SLIT for NUMA domain information
      - ACPI DSDT table for device affinity information
  - Kernel & Drivers
    - DMA affinity
    - Interrupt affinity
    - Kernel threads affinity
      - Thread modeling
      - Threads migrations
    - Memory allocation
  - Application
    - User process affinity
      - Thread modeling
      - Threads migrations
    - Memory allocation



Full Stack View



Kernel

- Device affinity
  - Interrupt affinity
    - Current irqbalance are not good enough.
      - How can it work with TCP flow and GUTs
  - DMA affinity
    - Had the bug in DMA buffer affinity code path.
    - Need evaluate DMA buffer affinity perf differences on Apollo
    - DMA affinity can be determined by user space
- CPU affinity
  - Kernel memory management
    - Need investigate recent NUMA scheduling features in mainline
  - Scheduling
    - Need investigate thread migration features in mainline
- Profiling tools
  - Need port some tools: NUMAtop/Some perf event features/IRQstat etc.
  - Need investigate how to leverage tracepoints

Network

- Device affinity
  - There is a trade-off between CPU availabilities and DMA buffer affinity
    - Not all device drivers are optimized for DMA buffer affinity.
      - For example, ixgbe doesn't support, but bnx2x support
    - How to set interrupts affinity when enable multiple device queues
      - Stop using old irqbalance.
- CPU affinity
  - Very big gap for data copy between kernel and user space
    - Kernel buffer and user buffer might be on different NUMA nodes
    - Following kernel features need to be investigated,
      - Flow Steering support
        - RFS: Receive Flow Steering
        - Flow director support in NICs (with multiple queues/interrupts)
    - Scheduler tuning might be required.

Idea Of Receive Flow Steering

- CPU on which transmits for a flow occur is "remembered"
- On receive for that flow, remembered cpu is looked up and packet steered to that CPU

Network – Flow Director On Intel NIC

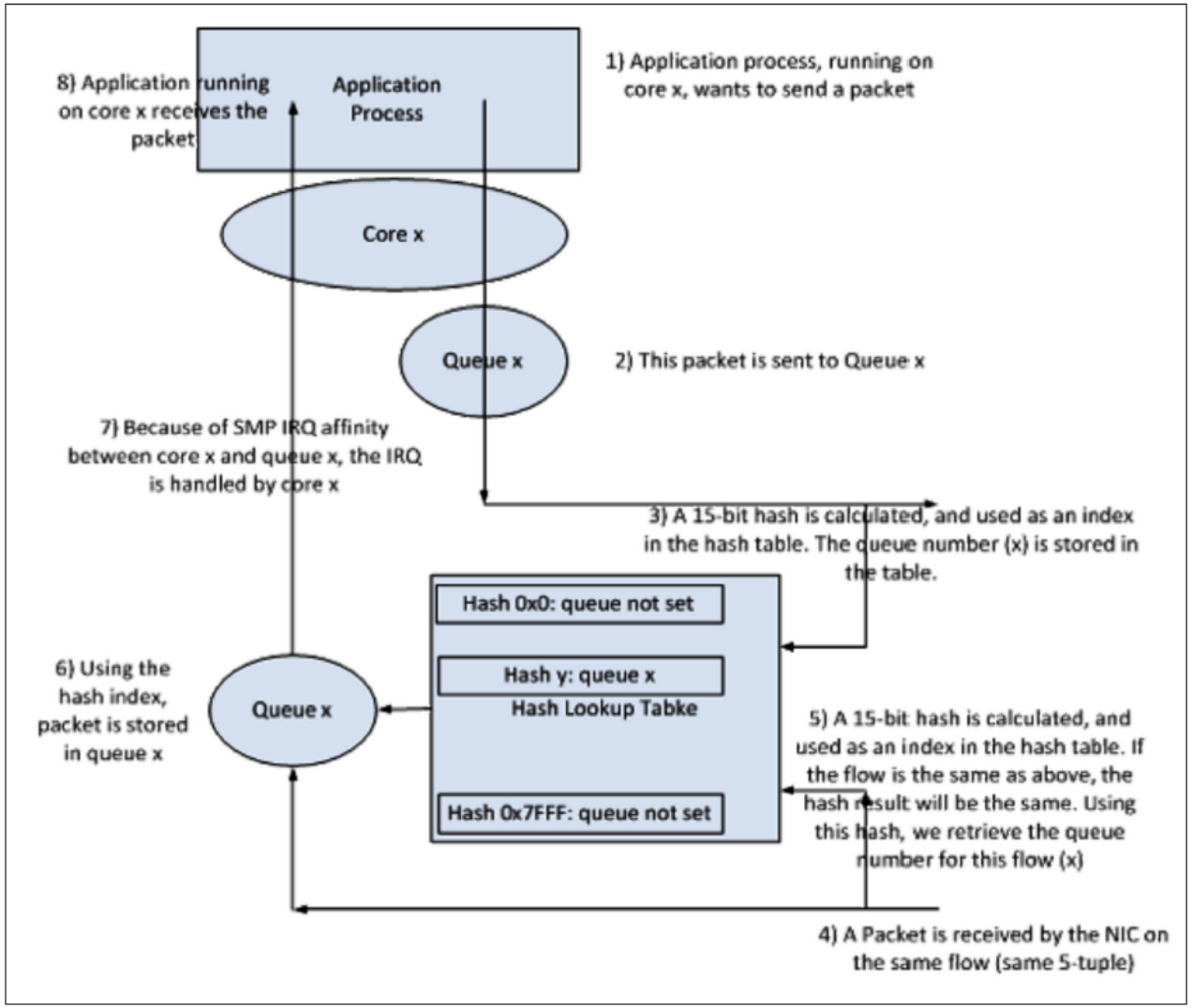


Figure 7: Flow Director Queue Logic

Agenda

- Overview
- Kernel Facilities
- Known Gaps
- Tools

Kernel Investigations – NUMA tools

- NUMA profiling tools
  - numactl
  - numastat
  - numad
  - Sysfs
  - Lstopo
  - NUMAtop: https://01.org/numatop
  - Top(NUMA version): https://gitorious.org/procps/procps
  - Irqstat: https://github.com/lanceshelton/irqstat
  - perf mem-loads event – available in 3.10 kernel
  - c2c data-sharing tool(c2c == “cache to cache”)

NUMA Top

Remote vs. local memory access (RMA/LMA samples)*

(Workload : Multiple 1 socket-wide instances of SPECjbb2005)

4-socket IVY-EX server

Baseline

PID	PROC	RMA(K)	LMA(K)	RMA/LMA	CPI	*CPU%
33142	java	375281.0	131458.3	2.9	0.96	25.9
33140	java	376322.6	124433.0	3.0	1.00	25.9
33141	java	377242.2	121182.1	3.1	1.02	25.8
33143	java	344362.5	157420.1	2.2	0.98	25.7

Pinned

PID	PROC	RMA(K)	LMA(K)	RMA/LMA	CPI	*CPU%
31769	java	104.5	548278.1	0.0	0.78	26.0
31768	java	118.4	565379.1	0.0	0.77	25.9
31771	java	64.1	543975.8	0.0	0.77	25.9
31770	java	138.1	545564.5	0.0	0.78	25.9

Automatic NUMA balancing

PID	PROC	RMA(K)	LMA(K)	RMA/LMA	CPI	*CPU%
32464	java	535.5	556543.3	0.0	0.78	26.3
32463	java	572.5	545804.6	0.0	0.79	26.2
32465	java	33014.9	518331.7	0.1	0.80	26.2
32466	java	32145.0	507227.0	0.1	0.80	26.1

8-socket IVY-EX prototype server

Baseline

PID	PROC	RMA(K)	LMA(K)	RMA/LMA	CPI	*CPU%
30678	java	208041.3	25753.2	8.1	3.09	13.0
30674	java	235889.3	34685.6	6.8	2.27	13.0
30680	java	224735.9	21833.6	10.3	2.81	13.0
30679	java	216502.5	29174.8	7.4	2.91	13.0
30676	java	238901.2	32688.9	7.3	2.56	12.9
30677	java	210094.1	25224.9	8.3	3.06	12.9
30675	java	197423.0	24037.8	8.2	3.44	12.8
30673	java	231181.8	45008.4	5.1	2.40	12.8

Pinned

PID	PROC	RMA(K)	LMA(K)	RMA/LMA	CPI	*CPU%
25494	java	296.1	620725.2	0.0	0.88	14.5
25495	java	277.8	607745.1	0.0	0.88	14.5
25490	java	249.9	586102.3	0.0	0.88	14.4
25491	java	244.1	601810.6	0.0	0.89	14.4
25493	java	233.4	605067.1	0.0	0.88	14.4
25488	java	268.7	617447.6	0.0	0.88	14.4
25495	java	271.0	596257.3	0.0	0.89	14.3
25492	java	89.5	607969.2	0.0	0.88	14.2

Automatic NUMA balancing

PID	PROC	RMA(K)	LMA(K)	RMA/LMA	CPI	*CPU%
26586	java	120637.5	342139.0	0.4	1.13	13.4
26590	java	18820.7	514351.1	0.0	0.91	13.4
26587	java	158040.0	317038.3	0.5	1.13	13.2
26589	java	85090.3	395132.1	0.2	1.03	13.0
26584	java	100579.4	377078.7	0.3	1.02	12.9
26585	java	75059.3	399099.2	0.2	1.01	12.9
26583	java	46739.2	439377.6	0.1	0.98	12.8
26588	java	46514.7	443239.0	0.1	0.96	12.8

Higher RMA/LMA

** Courtesy numatop v1.0.2*