CANONICAL

KVM Performance Optimization

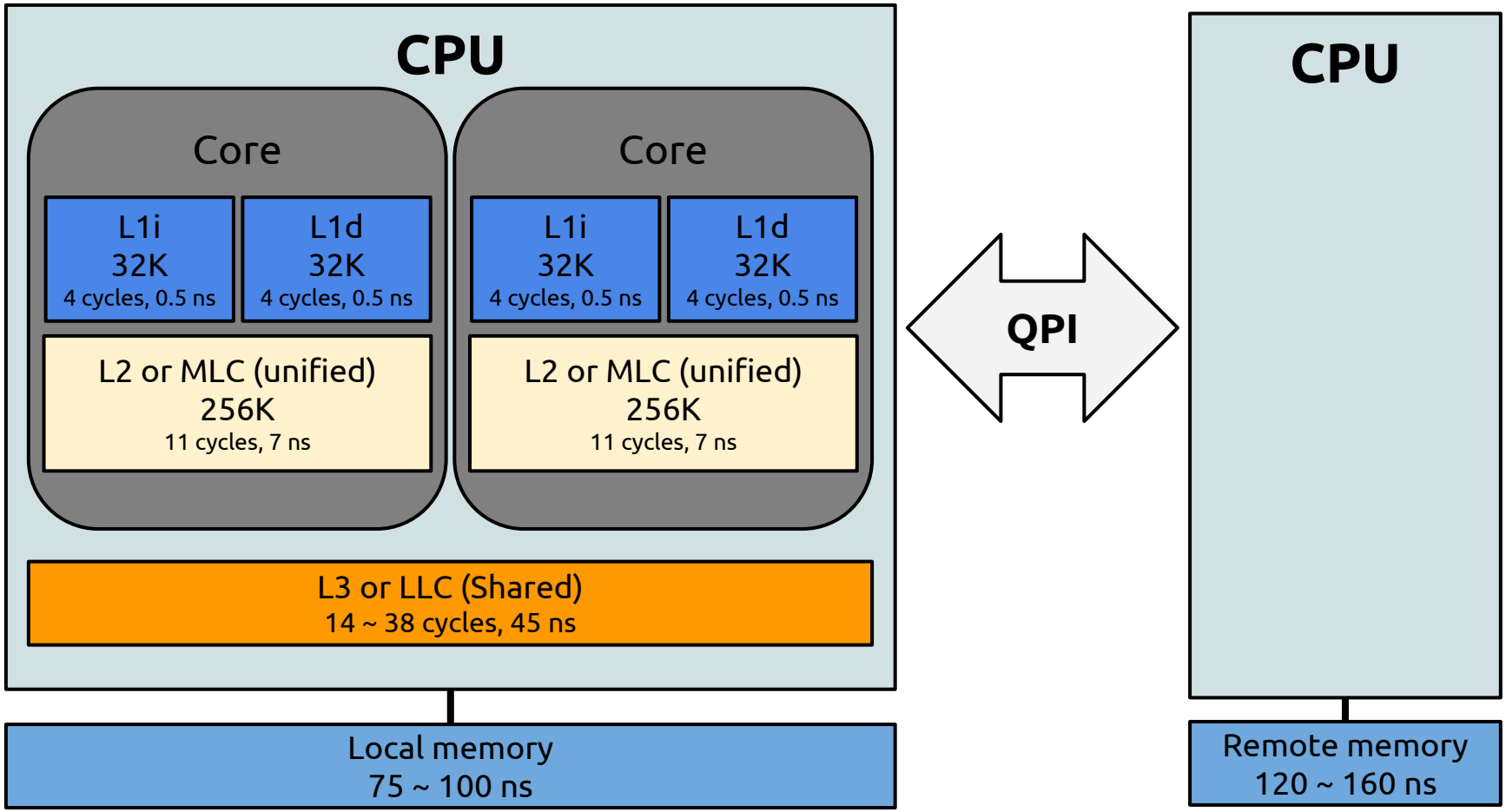
Paul Sim
Cloud Consultant
paul.sim@canonical.com

Index

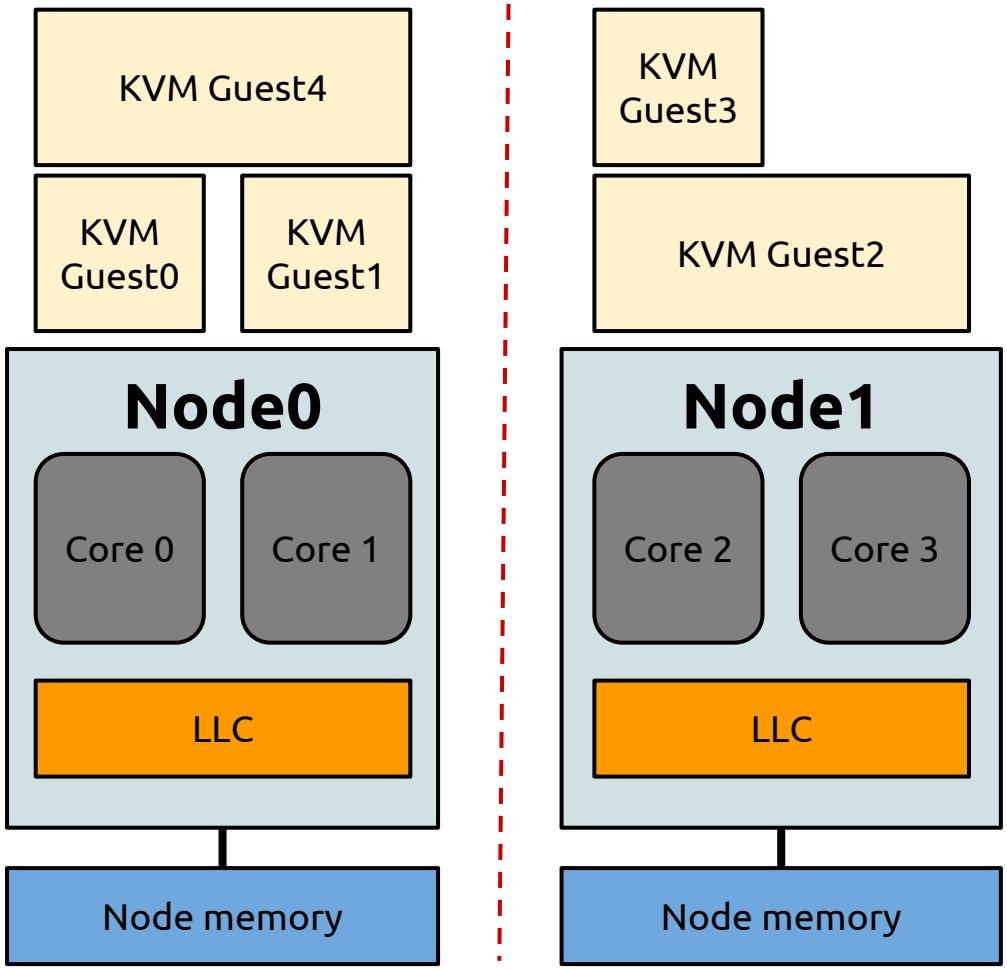
- **CPU & Memory**
  - vCPU pinning
  - NUMA affinity
  - THP (Transparent Huge Page)
  - KSM (Kernel SamePage Merging) & virtio_balloon
- **Networking**
  - vhost_net
  - Interrupt handling
  - Large Segment offload
- **Block Device**
  - I/O Scheduler
  - VM Cache mode
  - Asynchronous I/O

• CPU & Memory

Modern CPU cache architecture and latency (Intel Sandy Bridge)



CPU & Memory - vCPU pinning



*** Pin specific vCPU on specific physical CPU**
*** vCPU pinning increases CPU cache hit ratio**

1. Discover cpu topology.

- virsh capabilities

2. pin vcpu on specific core

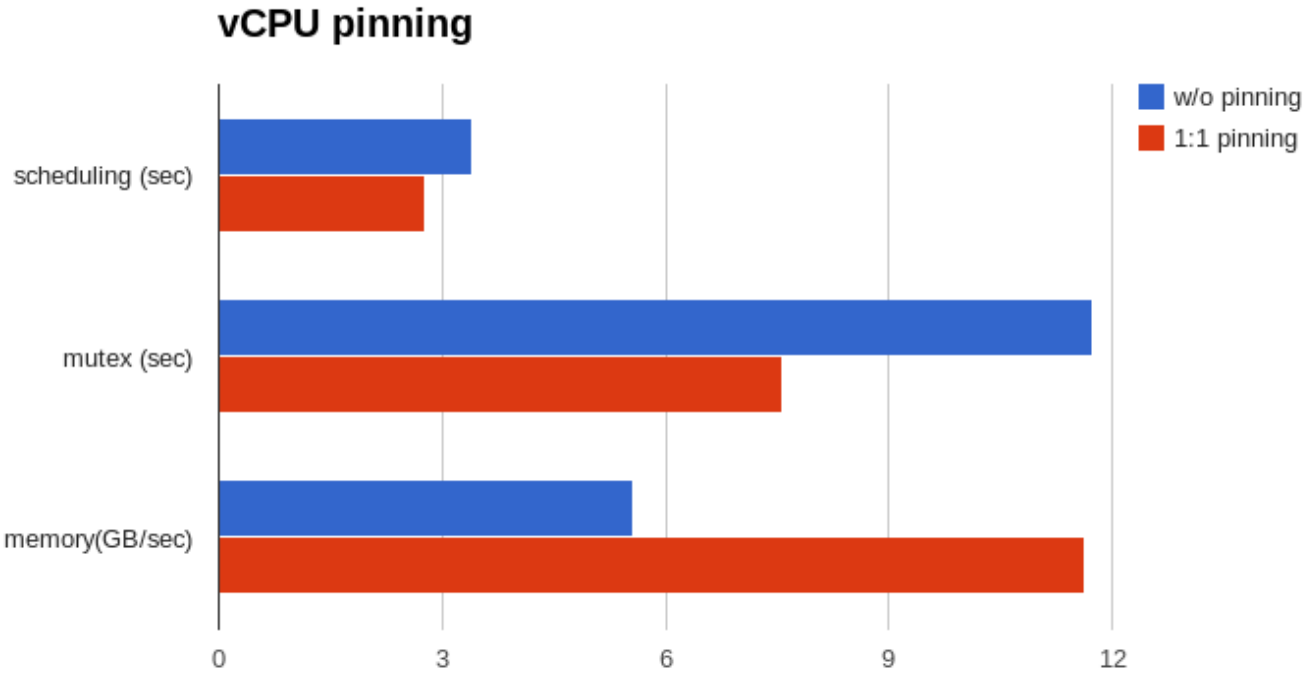
- vcpupin <domain> vcpu-num cpu-num

3. print vcpu information

- vcpuinfo <domain>

*** Multi-node virtual machine?**

• CPU & Memory - vCPU pinning

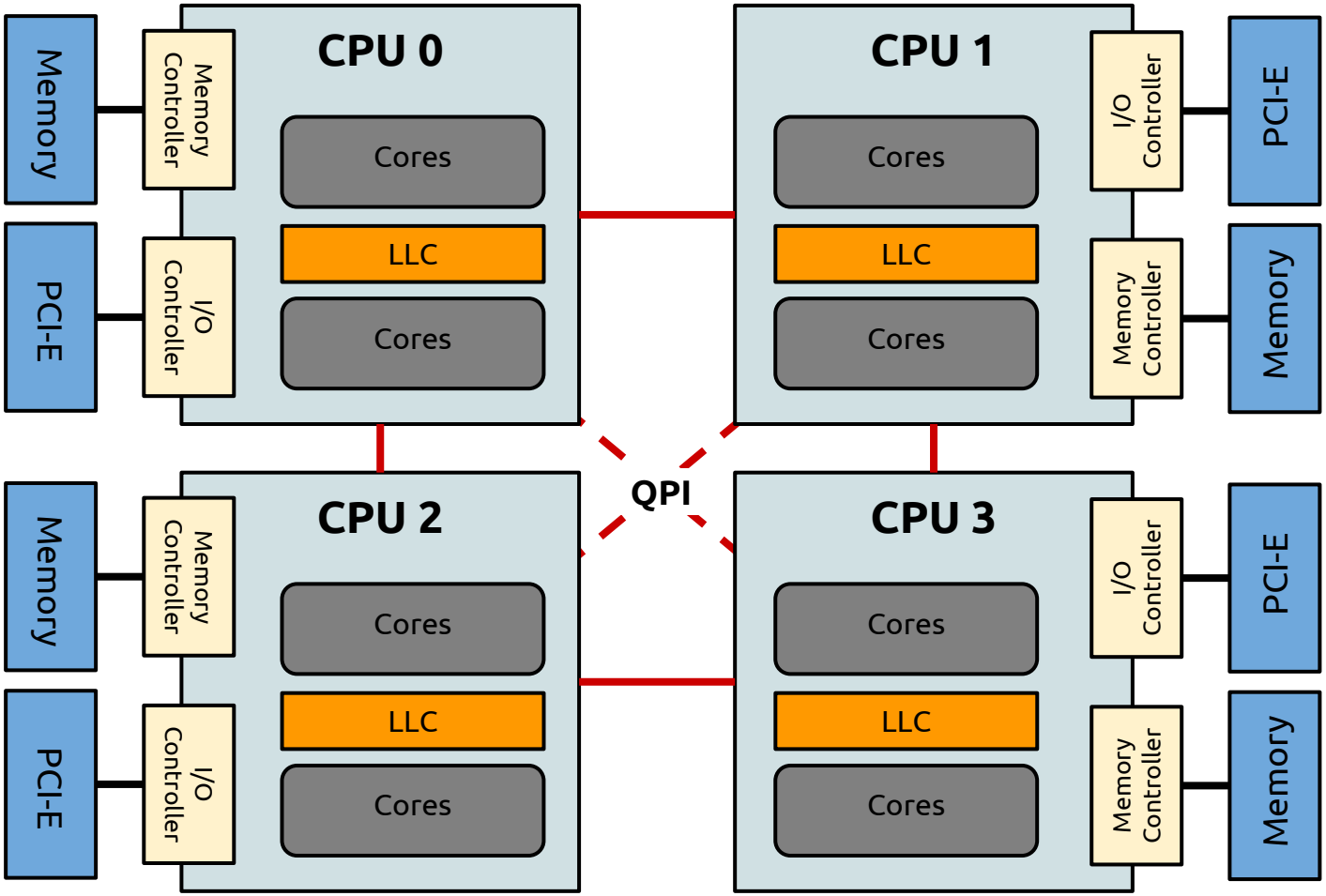


*** Two times faster memory access than without pinning**

- Shorter is better on scheduling and mutex test.
- Longer is better on memory test.

• CPU & Memory - NUMA affinity

CPU architecture (Intel Sandy Bridge)



• CPU & Memory - NUMA affinity

```
janghoon@machine-04:~$ sudo numactl --hardware
available: 4 nodes (0-3)
node 0 cpus: 0 1 2 3 4 5 6 7 8 9 40 41 42 43 44 45 46 47 48 49
node 0 size: 24567 MB
node 0 free: 6010 MB
node 1 cpus: cpus: 10 11 12 13 14 15 16 17 18 19 50 51 52 53 54 55 56 57 58 59
node 1 size: 24576 MB
node 1 free: 8557 MB
node 2 cpus: 20 21 22 23 24 25 26 27 28 29 60 61 62 63 64 65 66 67 68 69
node 2 size: 24567 MB
node 2 free: 6320 MB
node 3 cpus: 30 31 32 33 34 35 36 37 38 39 70 71 72 73 74 75 76 77 78 79
node 3 size: 24567 MB
node 3 free: 8204 MB
node distances:
node  0  1  2  3
  0: 10 16 16 22
  1: 16 10 22 16
  2: 16 22 10 16
  3: 22 16 16 10
```

*** Remote memory is expensive!**

• CPU & Memory - NUMA affinity

1. Determine where the pages of a VM are allocated.

- cat /proc/<*PID*>/numa_maps

- cat /sys/fs/cgroup/memory/sysdefault/libvirt/qemu/<*KVM name*>/memory.numa_stat

```
total=244973 N0=118375 N1=126598
file=81 N0=24 N1=57
anon=244892 N0=118351 N1=126541
unevictable=0 N0=0 N1=0
```

2. Change memory policy mode

- cgset -r cpuset.mems=<*Node*> sysdefault/libvirt/qemu/<*KVM name*>/emulator/

3. Migrate pages into a specific node.

- migratepages *<PID> from-node to-node*

- cat /proc/<*PID*>/status

*** Memory policy modes**

1) **interleave** : Memory will be allocated using round robin on nodes. When memory cannot be allocated on the current interleave target fall back to other nodes.

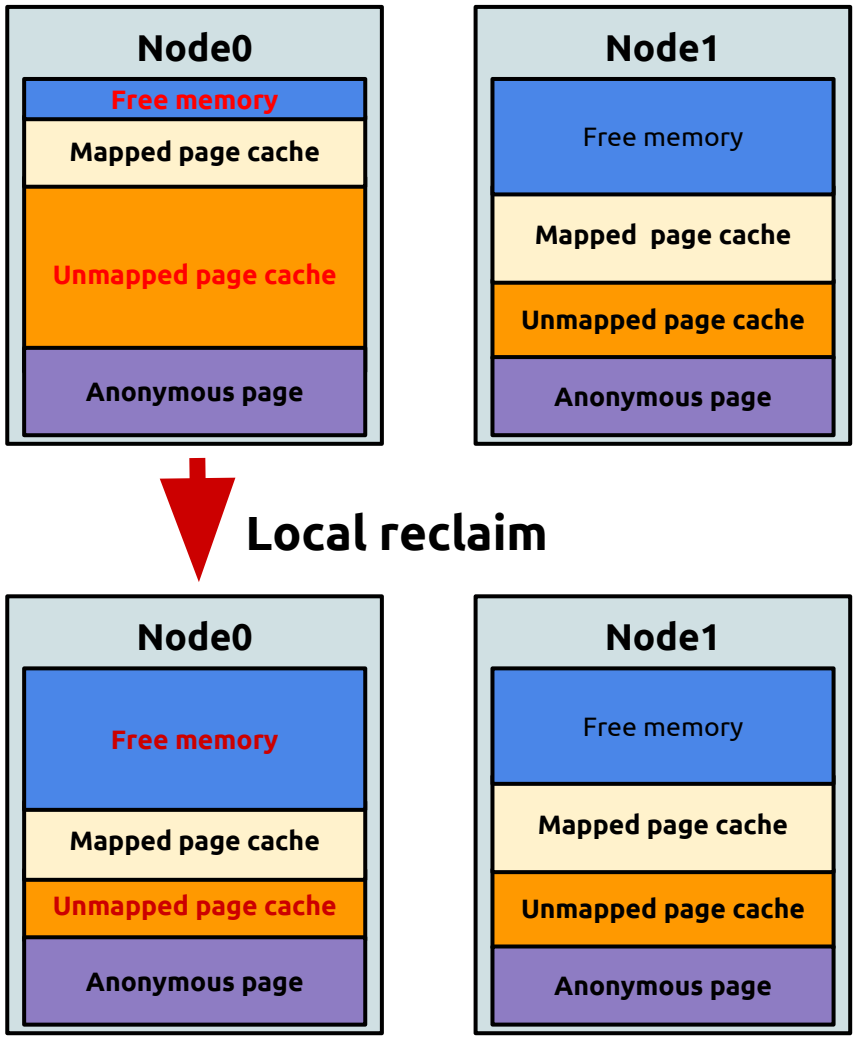
2) **bind** : Only allocate memory from nodes. Allocation will fail when there is not enough memory available on these nodes.

3) **preferred** : Preferably allocate memory on node, but if memory cannot be allocated there fall back to other nodes.

*** "preferred" memory policy mode is not currently supported on cgroup**

• CPU & Memory - NUMA affinity

- NUMA reclaim



1. Check if zone reclaim is enabled.

- cat /proc/sys/vm/zone_reclaim_mode

0(default): Linux kernel allocates the memory to a remote NUMA node where free memory is available.

1 : Linux kernel reclaims unmapped page caches for the local NUMA node rather than immediately allocating the memory to a remote NUMA node.

It is known that a virtual machine causes zone reclaim to occur in the situations when KSM(Kernel Same-page Merging) is enabled or Hugepage are enabled on a virtual machine side.

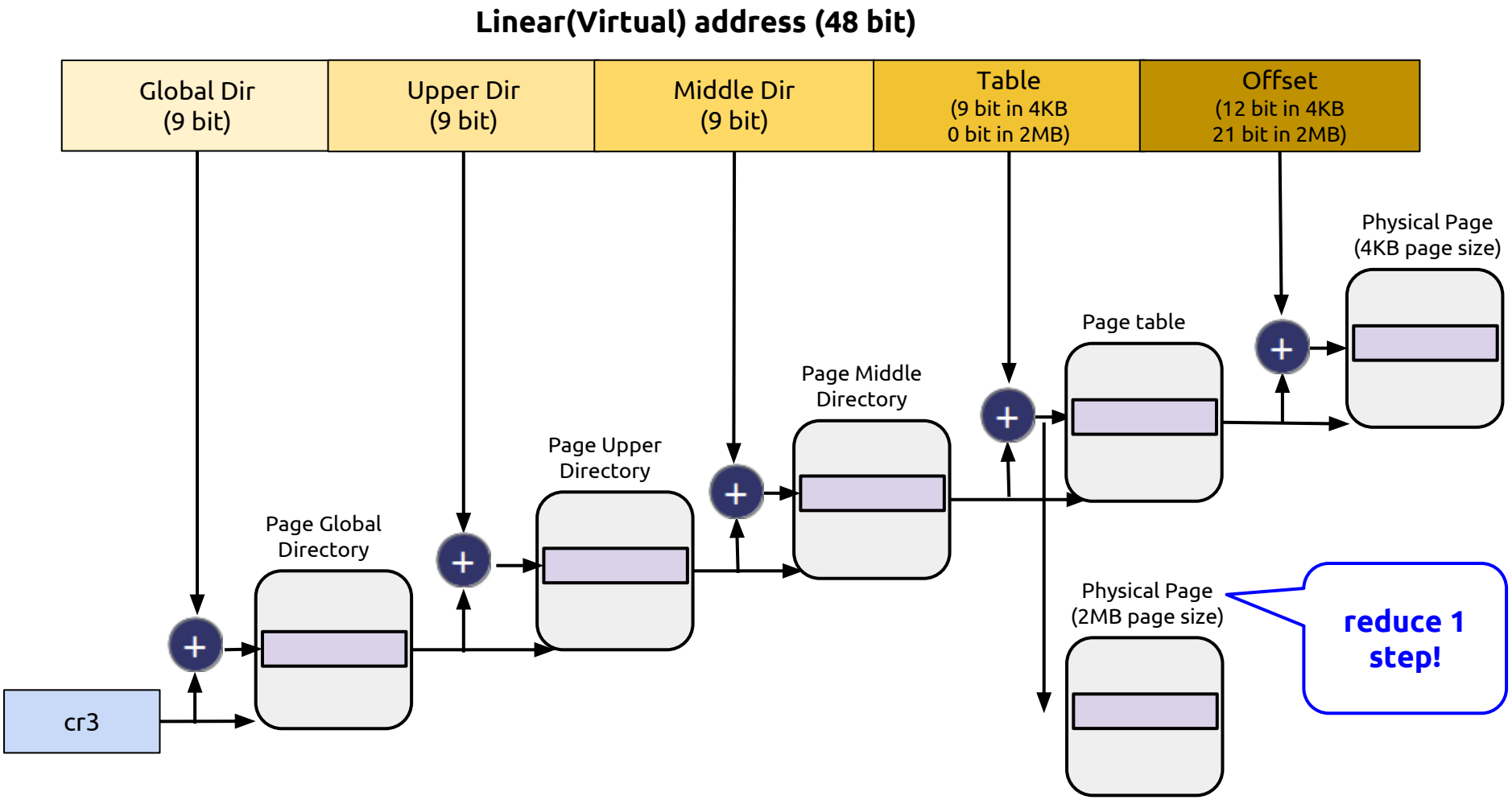
• CPU & Memory - THP (Transparent Huge Page)

- Paging level in some 64 bit architectures

Platform	Page size	Address bit used	Paging levels	splitting
Alpha	8KB	43	3	10 + 10 + 10 + 13
IA64	4KB	39	3	9 + 9 + 9 + 12
PPC64	4KB	41	3	10 + 10 + 9 + 12
x86_64	4KB(default)	48	4	9 + 9 + 9 + 9 + 12
x86_64	**2MB(THP)**	**48**	**3**	**9 + 9 + 9 + 21**

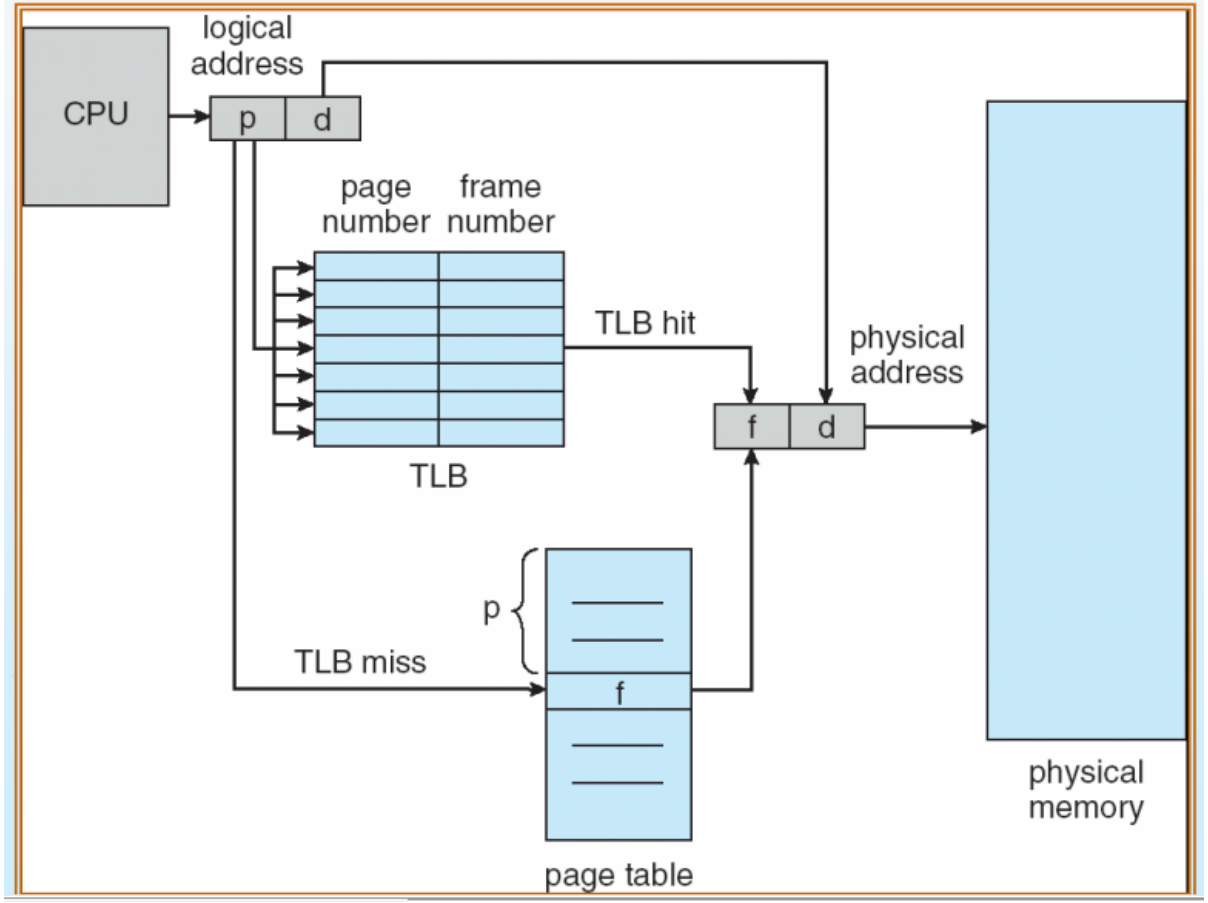
• CPU & Memory - THP (Transparent Huge Page)

- Memory address translation in 64 bit Linux



• CPU & Memory - THP (Transparent Huge Page)

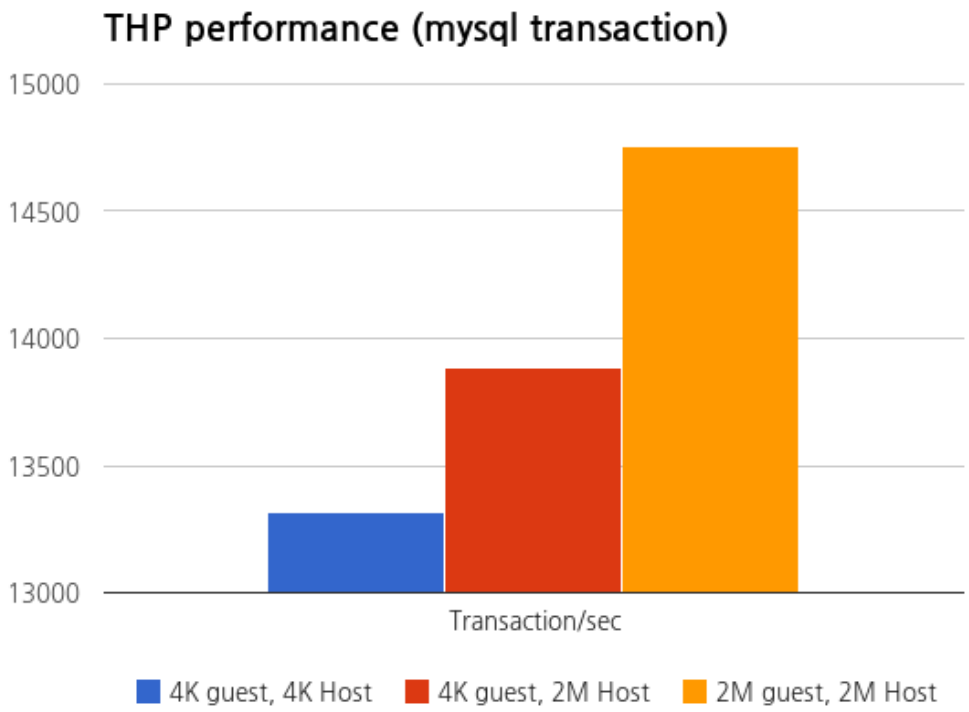
Paging hardware with TLB (Translation lookaside buffer)



* Translation lookaside buffer (TLB) : a cache that memory management hardware uses to improve virtual address translation speed.
* TLB is also a kind of cache memory in CPU.
* Then, how can we increase TLB hit ratio?
 - TLB can hold only 8 ~ 1024 entries.
- Decrease the number of pages.
 i.e. On 32GB memory system,
 8,388,608 pages with 4KB page block
 16,384 pages with 2MB page block

• CPU & Memory - THP (Transparent Huge Page)

THP performance benchmark - MySQL 5.5 OLTP testing



*** A guest machine also has to use HugePage for the best effect**

• CPU & Memory - THP (Transparent Huge Page)

1. Check current THP configuration
- cat /sys/kernel/mm/transparent_hugepage/enabled
2. Configure THP mode
- echo *mode* > /sys/kernel/mm/transparent_hugepage/enabled
3. monitor Huge page usage
- cat /proc/meminfo | grep Huge

```
janghoon@machine-04:~$ grep Huge /proc/meminfo
AnonHugePages:   462848 kB
HugePages_Total:       0
HugePages_Free:        0
HugePages_Rsvd:        0
HugePages_Surp:        0
Hugepagesize:       2048 kB
```

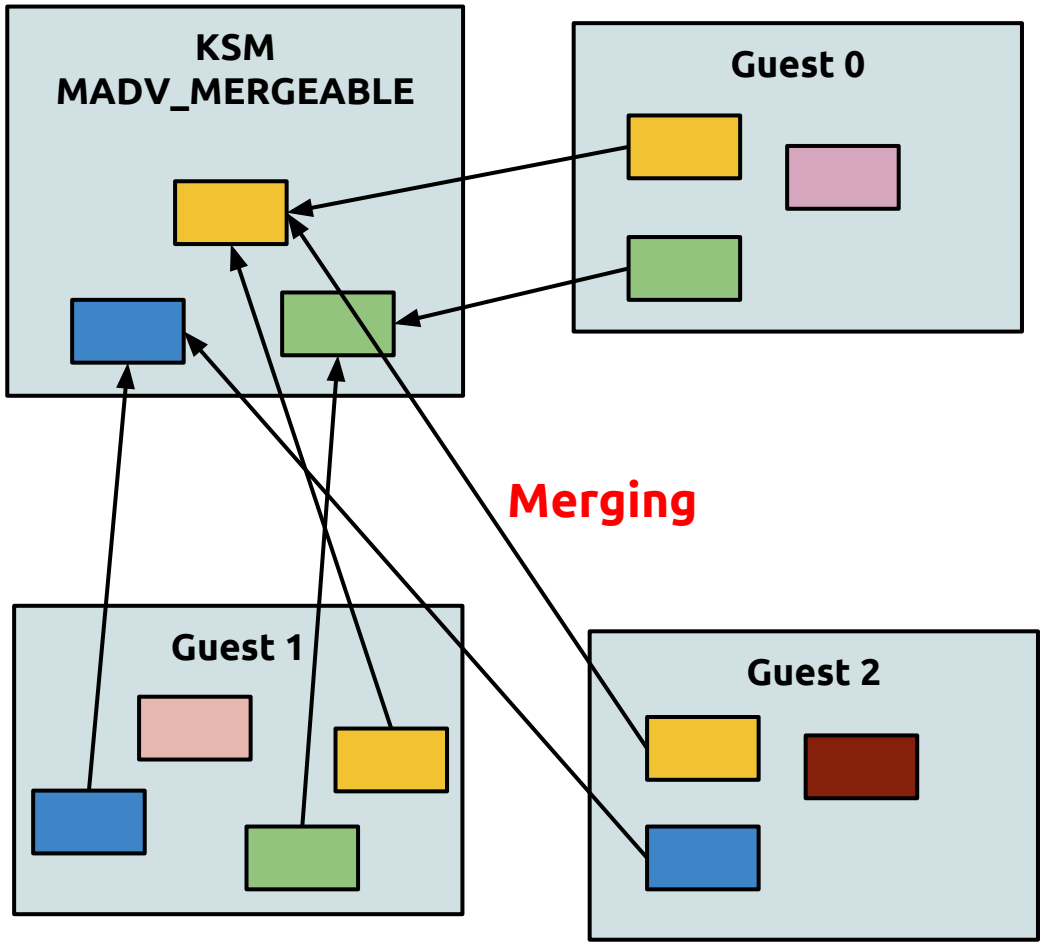
- grep Huge /proc/*<PID>*/smaps
4. Adjust parameters under /sys/kernel/mm/transparent_hugepage/khugepaged
- grep thp /proc/vmstat

*** THP modes**
1) **always :** use HugePages always
2) **madvise** : use HugePages only in specific regions, madvise(MADV_HUGEPAGE). Default on Ubuntu precise
3) **never** : not use HugePages

** Currently it **only works for anonymous memory mappings** but in the future it can expand over the pagecache layer starting with tmpfs. - Linux Kernel Documentation/vm/transhuge.txt*

• CPU & Memory - KSM & virtio_balloon

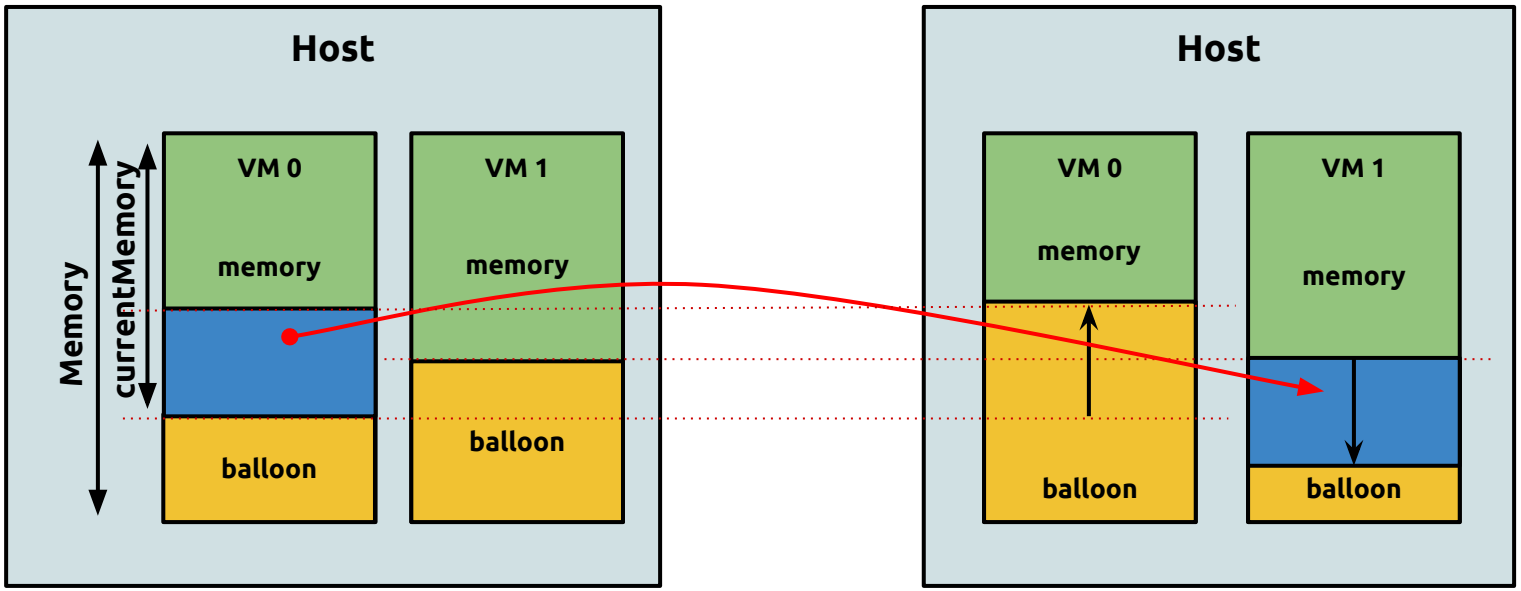
- KSM (Kernel SamePage Merging)



1. A kernel feature in KVM that shares memory pages between various processes, over-committing
the memory.
2. Only merges anonymous (private) pages.
3. Enable KSM
- echo “1” > /sys/kernel/mm/ksm/run
4. Monitor KSM
- Files under /sys/kernel/mm/ksm/
5. For NUMA (in Linux 3.9)
- /sys/kernel/mm/ksm/merge_across_nodes
0 : merge only pages in the memory area of the same NUMA node
1 : merge pages across nodes

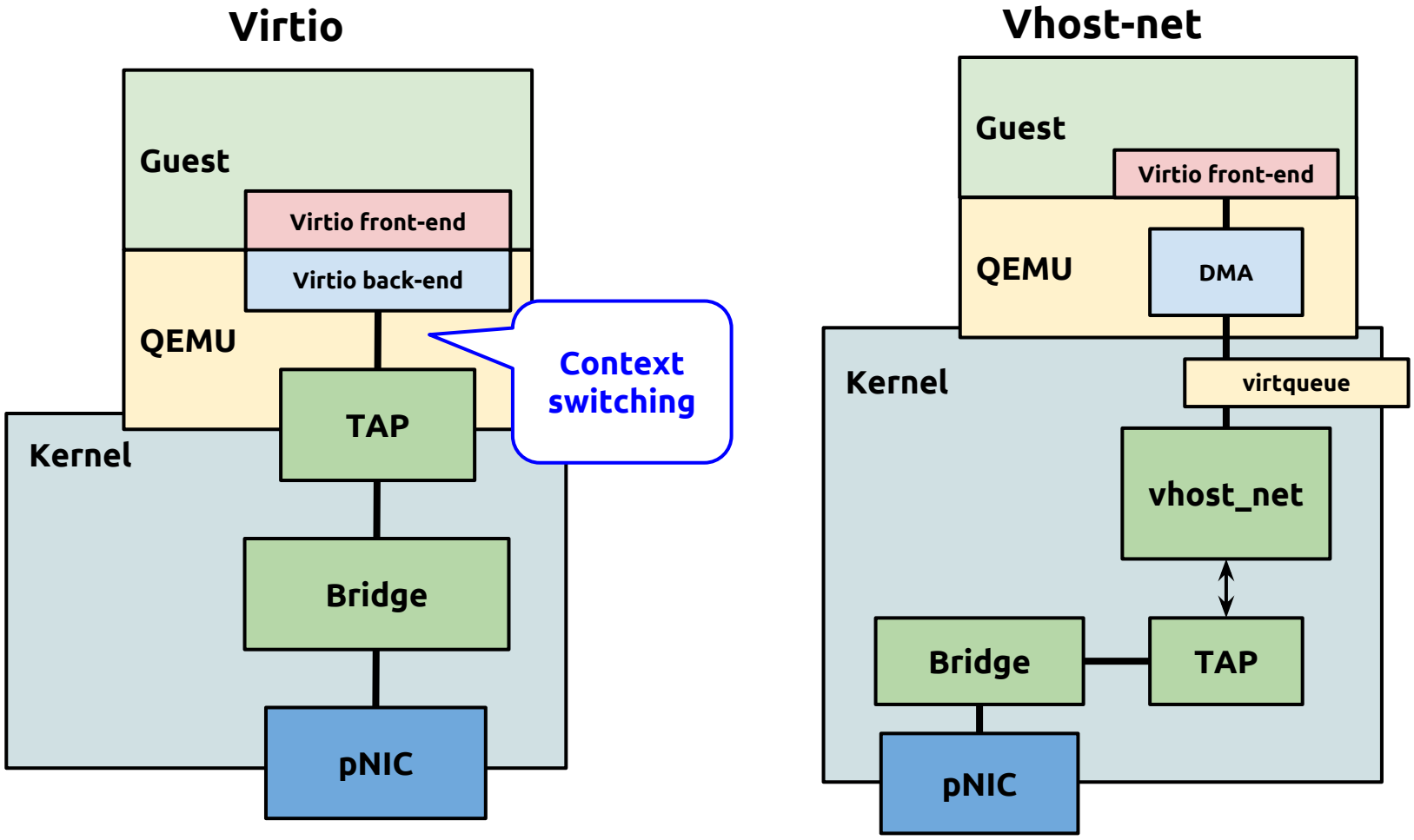
• CPU & Memory - KSM & virtio_balloon

- Virtio_balloon

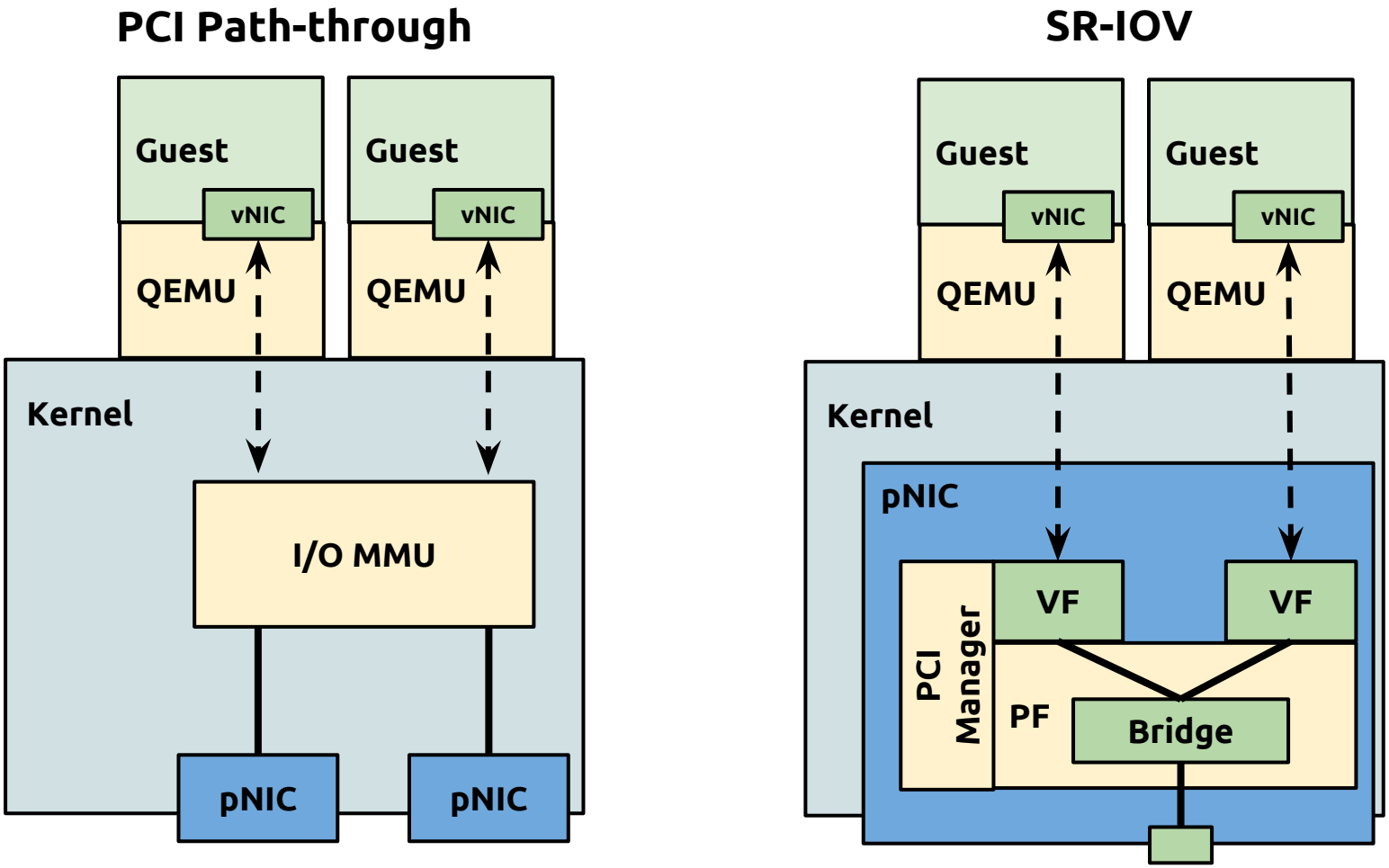


1. The hypervisor sends a request to the guest operating system to return some amount of memory back to the hypervisor.
2. The virtio_balloon driver in the guest operating system receives the request from the hypervisor.
3. The virtio_balloon driver inflates a balloon of memory inside the guest operating system
4. The guest operating system returns the balloon of memory back to the hypervisor.
5. The hypervisor allocates the memory from the balloon elsewhere as needed.
6. If the memory from the balloon later becomes available, the hypervisor can return the memory to the guest operating system

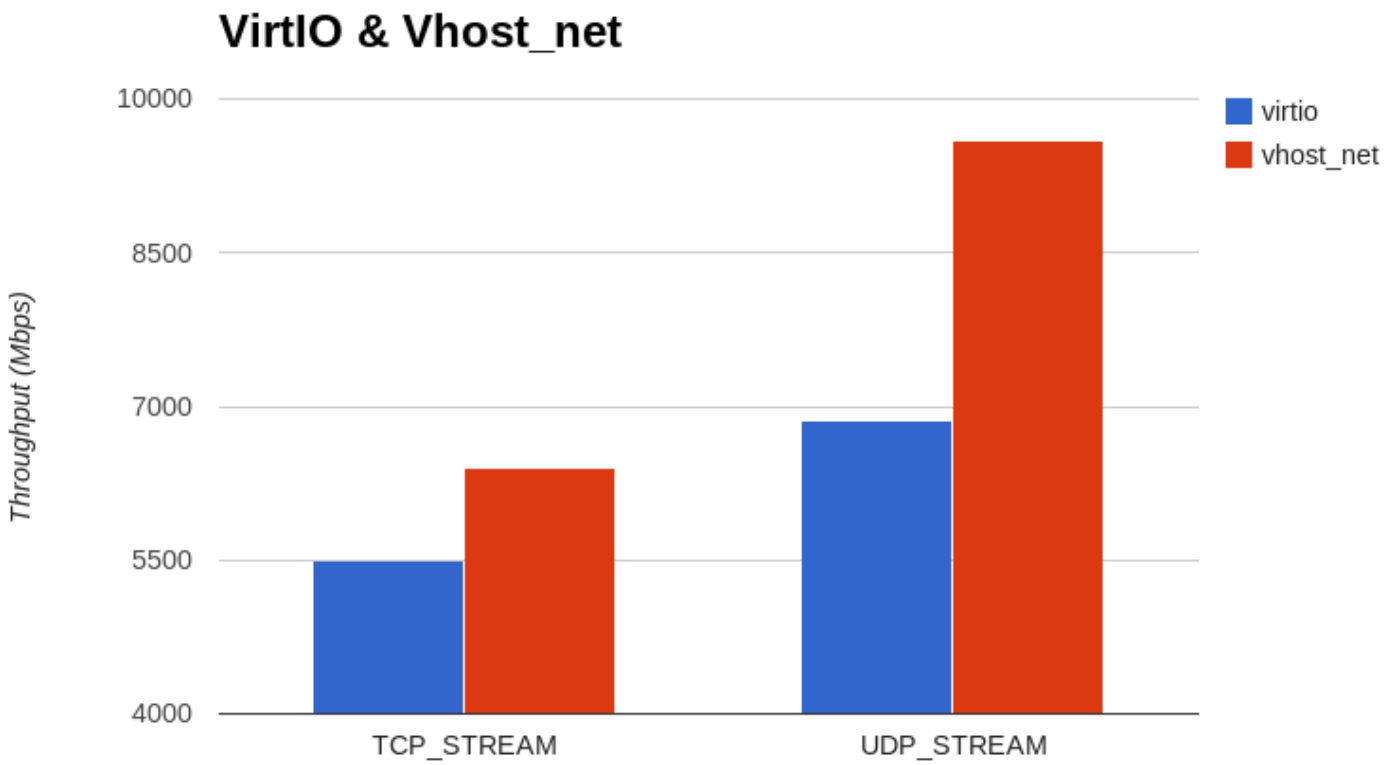
• Networking - vhost-net



• Networking - vhost-net



• Networking - vhost-net



• Networking - Interrupt handling

1. Multi-queue NIC

```
root@machine-04:~# cat /proc/interrupts | grep eth0
 71:       0       0       0       0       0       0       0       0  IR-PCI-MSI-edge     eth0
 72:       0       0     213    679058       0       0       0       0  IR-PCI-MSI-edge     eth0-TxRx-0
 73:       0       0       0       0       0       0       0       0  IR-PCI-MSI-edge     eth0-TxRx-1
 74:  562872       0       0       0       0       0       0       0  IR-PCI-MSI-edge     eth0-TxRx-2
 75:  561303       0       0       0       0       0       0       0  IR-PCI-MSI-edge     eth0-TxRx-3
 76:  583422       0       0       0       0       0       0       0  IR-PCI-MSI-edge     eth0-TxRx-4
 77:      21       5  563963       0       0       0       0       0  IR-PCI-MSI-edge     eth0-TxRx-5
 78:      23       0  548548       0       0       0       0       0  IR-PCI-MSI-edge     eth0-TxRx-6
 79:  567242       0       0       0       0       0       0       0  IR-PCI-MSI-edge     eth0-TxRx-7
```

- RPS(Receive Packet Steering) / XPS(Transmit Packet Steering)

: *A mechanism for intelligently selecting which receive/transmit queue to use when receiving/transmitting a packet on a multi-queue device. it can configure CPU affinity per-device-queue, disabled by default.*

: configure cpu mask

- echo 1 > /sys/class/net/*<device>*/queues/rx-*<queue num>*/rps_cpus

- echo 16 > /sys/class/net/*<device>*/queues/tx-*<queue num>*/xps_cpus

around 20 ~ 100 % improvement

• Networking - Interrupt handling

2. MSI(Message Signaled Interrupt), MSI-X

- Make sure your MSI-X enabled

```
janghoon@machine-04:~$ sudo lspci -vs 01:00.1 | grep MSI
        Capabilities: [50] MSI: Enable- Count=1/1 Maskable+ 64bit+
        Capabilities: [70] MSI-X: Enable+ Count=10 Masked-
        Capabilities: [a0] Express Endpoint, MSI 00
```

3. NAPI(New API)

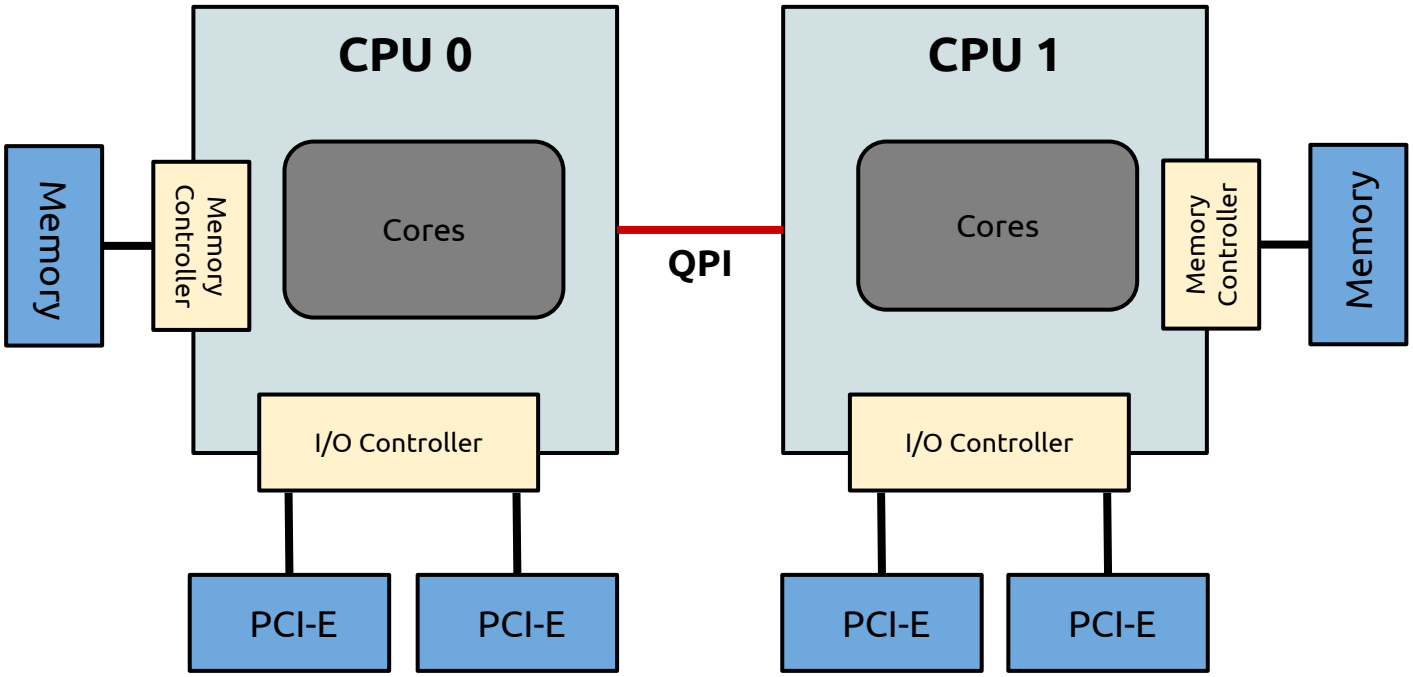
- *a feature that in the linux kernel aiming to improve the performance of high-speed networking, and avoid interrupt storms.*
- Ask your NIC vendor whether it's enabled by default.

Most modern NICs support Multi-queue, MSI-X and NAPI.
However, You may need to make sure these features are configured correctly and working properly.

http://en.wikipedia.org/wiki/Message_Signaled_Interrupts
http://en.wikipedia.org/wiki/New_API

Networking - Interrupt handling

4. IRQ Affinity (Pin Interrupts to the local node)



Each node has PCI-E devices connected directly (On Intel Sandy Bridge).

Where is my 10G NIC connected to?

• Networking - Interrupt handling

4. IRQ Affinity (Pin Interrupts to the local node)

    1) stop irqbalance service

    2) determine node that a NIC is connected to

        - lspci -tv

        - lspci -vs *<PCI device bus address>*

        - dmidecode -t slot

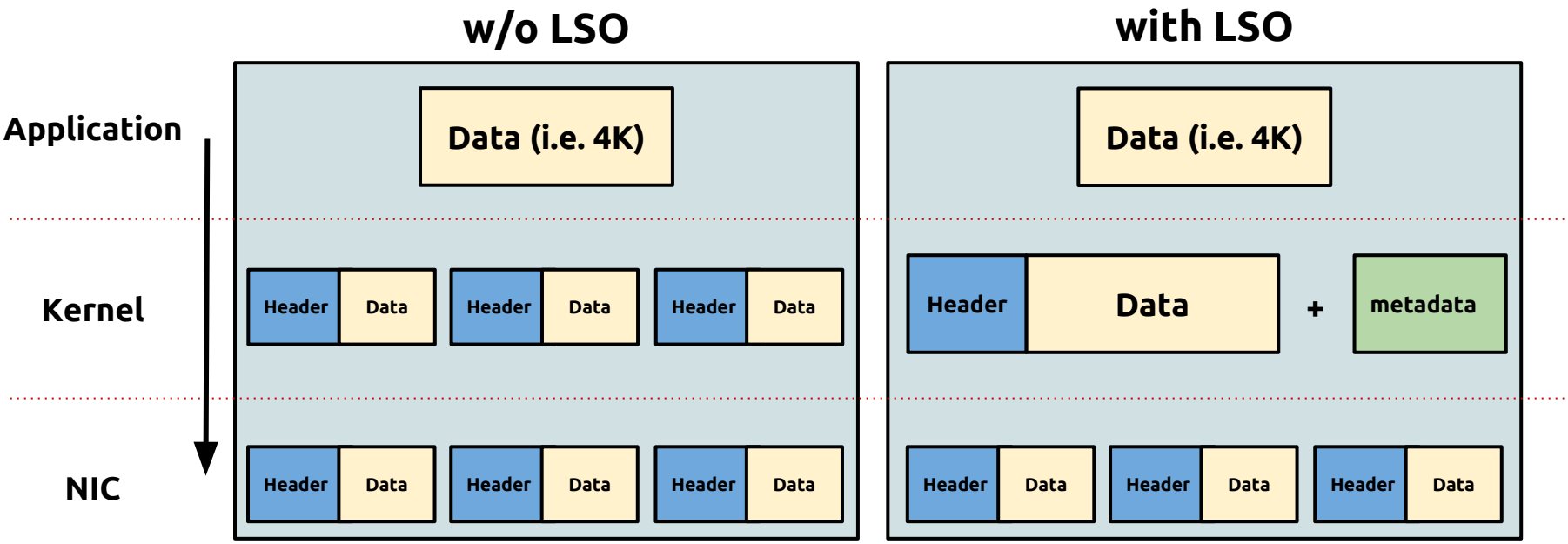
        - cat /sys/devices/pci*/*<bus address>*/numa_node (-1 : not detected)

        - cat /proc/irq/*<Interrupt number>*/node

    3) Pin interrupts from the NIC to specific node

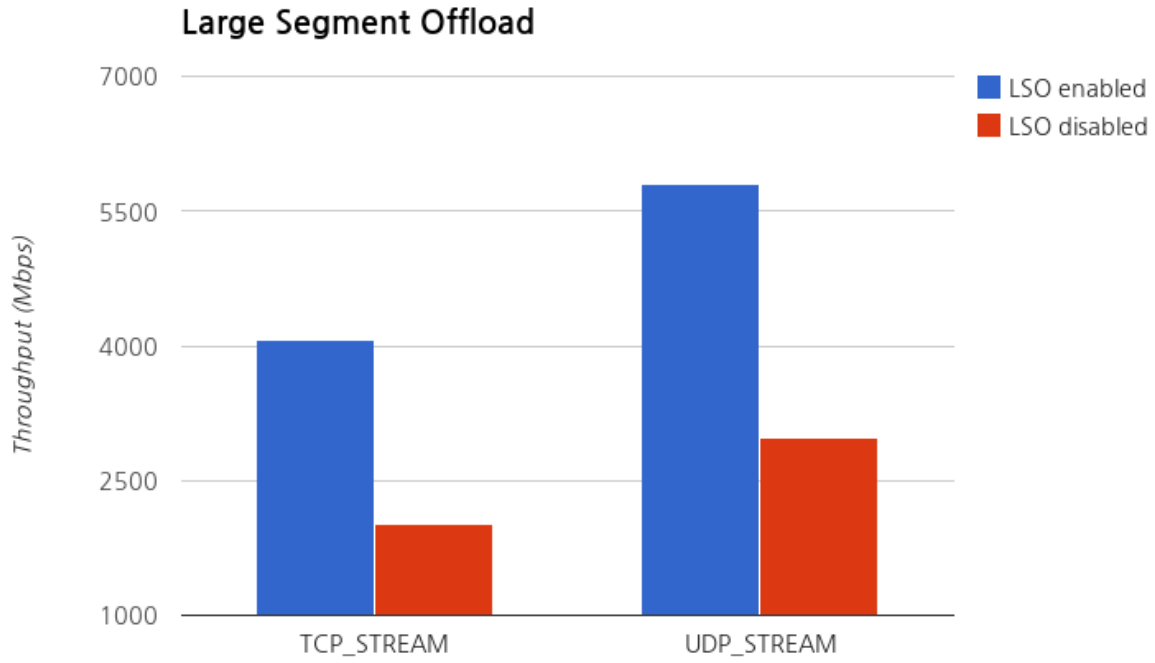
        - echo f > /proc/irq/*<Interrupt number>*/smp_affinity

• Networking - Large Segment Offload




```
janghoon@ubuntu-precise:~$ sudo ethtool -k eth0
Offload parameters for eth0:
tcp-segmentation-offload: on
udp-fragmentation-offload: on
generic-segmentation-offload: on
generic-receive-offload: on
large-receive-offload: off
```

• Networking - Large Segment Offload

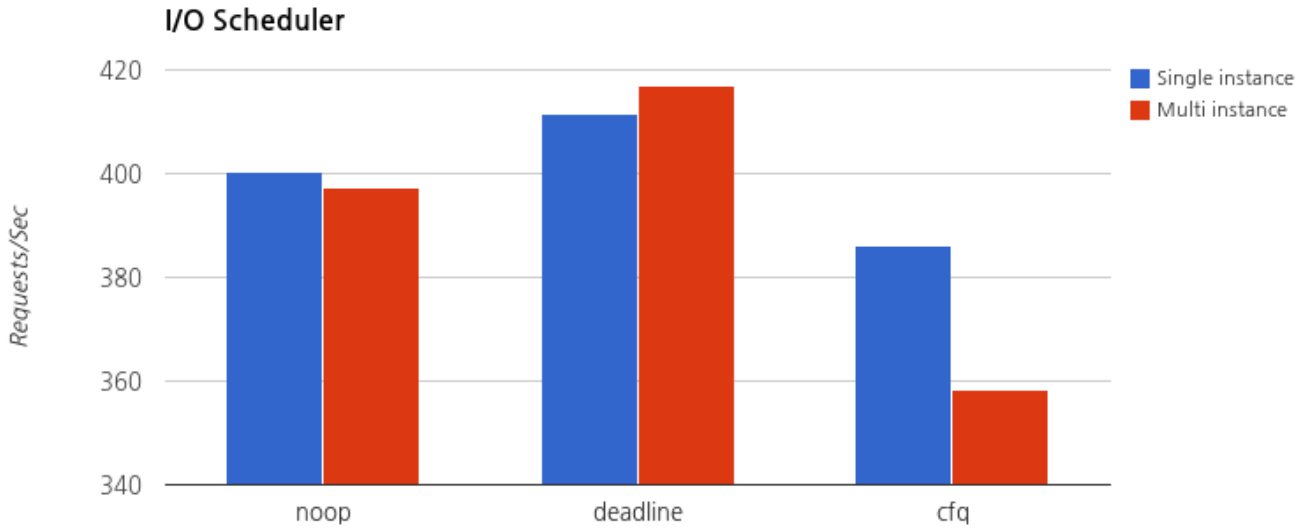


1. 100% throughput performance improvement
2. GSO/GRO, TSO and UFO are enabled by default on Ubuntu precise 12.04 LTS
3. Jumbo-frame is not much effective

• Block device - I/O Scheduler

1. Noop : basic FIFO queue.
2. Deadline : I/O requests place in a priority queue and are guaranteed to be run within a certain time. low latency.
3. CFQ (Completely Fair Queueing) : I/O requests are distributed to a number of per-process queues. default I/O scheduler on Ubuntu.

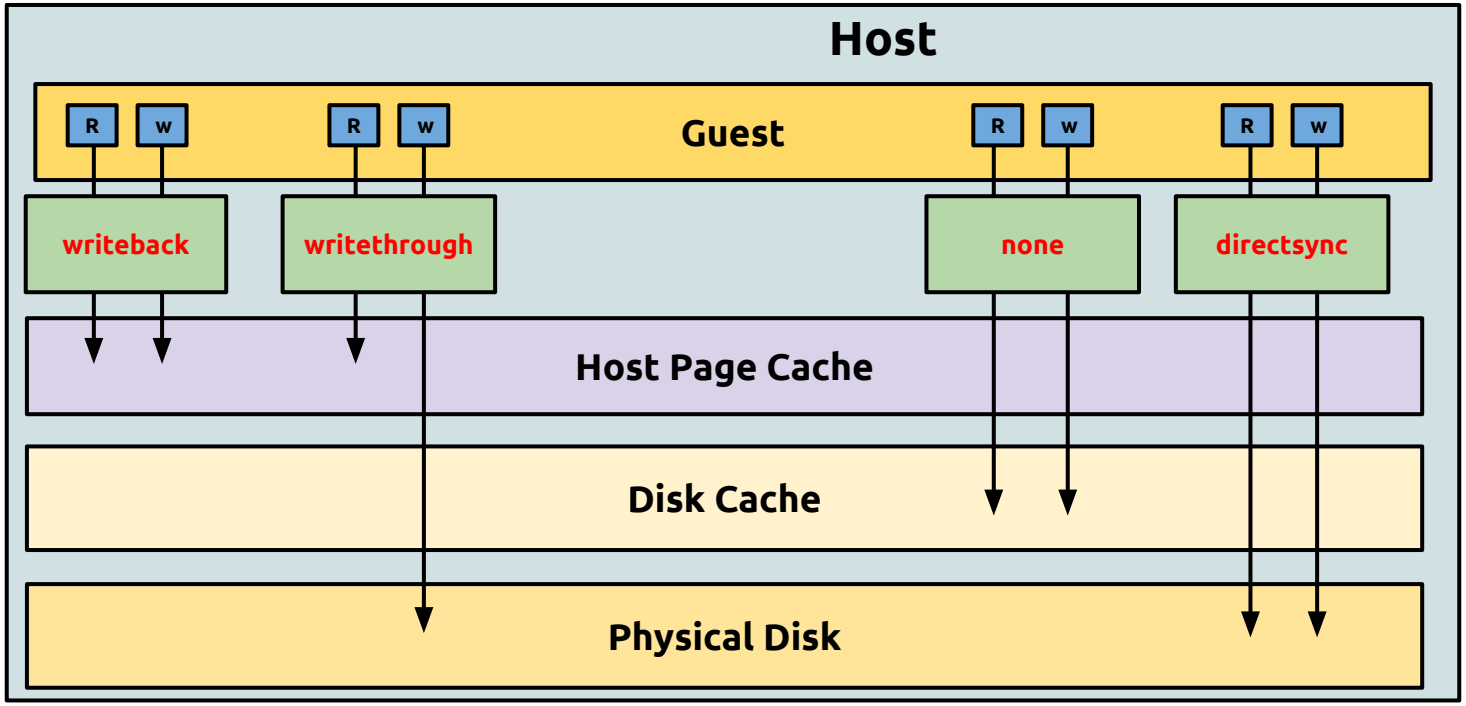
```
janghoon@ubuntu-precise:~$ sudo cat /sys/block/sdb/queue/scheduler
noop deadline [cfq]
```



• Block device - VM Cache mode

Disable host page cache

```
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2' cache='none'/>
  <source file='/mnt/VM_IMAGES/VM-test.img'/>
  <target dev='vda' bus='virtio'/>
  <alias name='virtio-disk0'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
</disk>
```



• Block device - Asynchronous I/O

```
<disk type='file' device='disk'>
  <driver name='qemu' type='qcow2' cache='none' io='native'/>
  <source file='/dev/sda7'/>
      <target dev='vda' bus='virtio'/>
  <address type='pci' domain='0x0000' bus='0x00' slot='0x04' function='0x0'/>
</disk>
```