Achieving the Ultimate Performance with KVM

Boyan Krosnov
SREcon20 Americas
About me

● Chief Product Officer & co-founder of StorPool
● 20+ years in ISPs, SDN, SDS
● IT infrastructure with a focus on inventions, performance and efficiency

Boyan Krosnov
linkedin.com/in/krosnov/
@bkrosnov
bk@storpool.com
About StorPool

- NVMe software-defined storage for VMs and containers
- Scale-out, HA, API-controlled
- Since 2011, in commercial production use since 2013
- Based in Sofia, Bulgaria; Profitable and growing
- Mostly virtual disks for KVM and bare metal Linux hosts
- Also used with VMware, Hyper-V, XenServer
- Integrations into OpenStack/Cinder, Kubernetes Persistent Volumes, CloudStack, OpenNebula, OnApp
- Fully managed service; 24/7; monitoring and analytics; 1000s of servers in 100s of zones
Why performance

● Better application performance
  ○ e.g. time to load a page, time to rebuild, time to execute specific query
● Happier customers in cloud and multi-tenant environments
● ROI, TCO - Lower cost per delivered resource (per VM) through higher density

● For public cloud - win customers over from your competitors
● For private cloud - do more with less; win applications / workloads / teams over from public cloud
1. Hardware
2. Compute, Virtualization
3. Networking
4. Storage
5. Conclusion
Compute node hardware

**Usual optimization goal**
- lowest cost per delivered resource
- fixed performance target
- calculate all costs - power, cooling, space, server, network, support/maintenance

Example: cost per VM with 4x dedicated 3 GHz cores and 16 GB RAM

**Unusual**
- Best single-thread performance I can get at any cost
- 4+ GHz cores, etc.
## Compute node hardware

<table>
<thead>
<tr>
<th>Brand</th>
<th>Model</th>
<th>release date</th>
<th>ark.intel status</th>
<th>release price ($)</th>
<th>Cores</th>
<th>TDP (W)</th>
<th>All-Core Turbo Clock (GHz)</th>
<th>Selected 1S or 2S or 4S ?</th>
<th>Total $ per core</th>
<th>Total $/GHz</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gold</td>
<td>5220R</td>
<td>January 2020</td>
<td>Launched</td>
<td>$1,555</td>
<td>24</td>
<td>150</td>
<td>2.9</td>
<td>2S</td>
<td>$244</td>
<td>$84</td>
</tr>
<tr>
<td>Gold</td>
<td>6230R</td>
<td>January 2020</td>
<td>Launched</td>
<td>$1,894</td>
<td>26</td>
<td>150</td>
<td>3.0</td>
<td>2S</td>
<td>$244</td>
<td>$81</td>
</tr>
<tr>
<td>Gold</td>
<td>5218R</td>
<td>January 2020</td>
<td>Launched</td>
<td>$1,273</td>
<td>20</td>
<td>125</td>
<td>2.9</td>
<td>2S</td>
<td>$257</td>
<td>$89</td>
</tr>
<tr>
<td>Gold</td>
<td>6238R</td>
<td>January 2020</td>
<td>Launched</td>
<td>$2,612</td>
<td>28</td>
<td>165</td>
<td>3.0</td>
<td>2S</td>
<td>$262</td>
<td>$87</td>
</tr>
<tr>
<td>Gold</td>
<td>6222V</td>
<td>May 2019</td>
<td>Launched</td>
<td>$1,600</td>
<td>20</td>
<td>115</td>
<td>2.4</td>
<td>2S</td>
<td>$271</td>
<td>$113</td>
</tr>
<tr>
<td>Gold</td>
<td>6240R</td>
<td>January 2020</td>
<td>Launched</td>
<td>$2,200</td>
<td>24</td>
<td>105</td>
<td>3.2</td>
<td>2S</td>
<td>$276</td>
<td>$86</td>
</tr>
<tr>
<td>Silver</td>
<td>4215</td>
<td>April 2019</td>
<td>Launched</td>
<td>$1,002</td>
<td>16</td>
<td>100</td>
<td>2.7</td>
<td>2S</td>
<td>$277</td>
<td>$103</td>
</tr>
<tr>
<td>Gold U</td>
<td>6212U</td>
<td>April 2019</td>
<td>Launched</td>
<td>$1,450</td>
<td>24</td>
<td>165</td>
<td>3.1</td>
<td>1S</td>
<td>$205</td>
<td>$92</td>
</tr>
<tr>
<td>Gold</td>
<td>6230</td>
<td>April 2019</td>
<td>Launched</td>
<td>$1,894</td>
<td>20</td>
<td>125</td>
<td>2.8</td>
<td>2S</td>
<td>$290</td>
<td>$103</td>
</tr>
<tr>
<td>Gold</td>
<td>6230T</td>
<td>May 2019</td>
<td>Launched</td>
<td>$1,988</td>
<td>20</td>
<td>125</td>
<td>2.8</td>
<td>2S</td>
<td>$294</td>
<td>$105</td>
</tr>
<tr>
<td>Gold</td>
<td>5220</td>
<td>April 2019</td>
<td>Launched</td>
<td>$1,555</td>
<td>18</td>
<td>125</td>
<td>2.7</td>
<td>2S</td>
<td>$295</td>
<td>$109</td>
</tr>
<tr>
<td>Gold</td>
<td>6230N</td>
<td>May 2019</td>
<td>Launched</td>
<td>$2,040</td>
<td>20</td>
<td>125</td>
<td>2.9</td>
<td>2S</td>
<td>$298</td>
<td>$103</td>
</tr>
<tr>
<td>Gold</td>
<td>5220T</td>
<td>May 2019</td>
<td>Launched</td>
<td>$1,727</td>
<td>18</td>
<td>105</td>
<td>2.7</td>
<td>2S</td>
<td>$298</td>
<td>$110</td>
</tr>
<tr>
<td>Gold</td>
<td>6262V</td>
<td>May 2019</td>
<td>Launched</td>
<td>$2,900</td>
<td>24</td>
<td>135</td>
<td>2.5</td>
<td>2S</td>
<td>$299</td>
<td>$120</td>
</tr>
<tr>
<td>Gold</td>
<td>5218T</td>
<td>May 2019</td>
<td>Launched</td>
<td>$1,349</td>
<td>16</td>
<td>105</td>
<td>2.7</td>
<td>2S</td>
<td>$303</td>
<td>$112</td>
</tr>
</tbody>
</table>
Compute node hardware

Intel

lowest cost per core:
- 2.9 GHz: Xeon Gold 5220R - 24 cores ($244/core)
- 3.2 GHz: Xeon Gold 6240R - 24 cores ($276/core)
- 3.6 GHz: Xeon Gold 6248R - 24 cores ($308/core)

lowest cost per GHz:
- Xeon Gold 6230R - 26 cores @ 3.0 GHz ($81/GHz)
Compute node hardware

AMD

per core @2.5 GHz: AMD EPYC 7702P 1-socket 64 cores ($210/core)
per core @3.0 GHz: AMD EPYC 7502 2-socket 32 cores ($251/core)
per core @3.5 GHz: AMD EPYC 7F72 2-socket 24 cores ($320/core)

per GHz: 7502, 7702P, 7742 tied for first place
Compute node hardware

Form factor

from                         to
Compute node hardware

- firmware versions and BIOS settings
- Understand power management -- esp. C-states, P-states, HWP and “bias”
  - Different on AMD EPYC: "power-deterministic", "performance-deterministic"
- Think of rack level optimization - how do we get the lowest total cost per delivered resource?
Agenda

1. Hardware
2. Compute, Virtualization
3. Networking
4. Storage
5. Conclusion
Tuning KVM

RHEL7 Virtualization_Tuning_and_Optimization_Guide link
https://pve.proxmox.com/wiki/Performance_Tweaks
http://www.slideshare.net/janghoonsim/kvm-performance-optimization-for-ubuntu

... but don’t trust everything you read. Perform your own benchmarking!
CPU and Memory

Recent Linux kernel, KVM and QEMU
... but beware of the bleeding edge
E.g. qemu-kvm-ev from RHEV (repackaged by CentOS)

tuned-adm virtual-host
tuned-adm virtual-guest
CPU

Typical
- (heavy) oversubscription, because VMs are mostly idling
- HT
- NUMA
- route IRQs of network and storage adapters to a core on the NUMA node they are on

Unusual
- CPU Pinning
Understanding oversubscription and congestion

Linux scheduler statistics: /proc/schedstat
(linux-stable/Documentation/scheduler/sched-stats.txt)

Next three are statistics describing scheduling latency:
7) sum of all time spent running by tasks on this processor (in ms)
8) sum of all time spent waiting to run by tasks on this processor (in ms)
9) # of tasks (not necessarily unique) given to the processor

* In nanoseconds, not ms.

20% CPU load with large wait time (bursty congestion) is possible
100% CPU load with no wait time, also possible

Measure CPU congestion!
Understanding oversubscription and congestion
Memory

Typical
● Dedicated RAM
● huge pages, THP
● NUMA
● use local-node memory if you can

Unusual
● Oversubscribed RAM
● balloon
● KSM (RAM dedup)
1. Hardware
2. Compute, Virtualization
3. Networking
4. Storage
5. Conclusion
Networking

Virtualized networking
- hardware emulation (rtl8139, e1000)
- paravirtualized drivers - virtio-net

regular virtio vs vhost-net vs vhost-user

Linux Bridge vs OVS in-kernel vs OVS-DPDK

Pass-through networking
SR-IOV (PCIe pass-through)
**virtio-net QEMU**

- Multiple context switches:
  1. virtio-net driver → KVM
  2. KVM → qemu/virtio-net device
  3. qemu → TAP device
  4. qemu → KVM (notification)
  5. KVM → virtio-net driver (interrupt)

- Much more efficient than emulated hardware
- shared memory with qemu process
- qemu thread process packets
virtio vhost-net

- Two context switches (optional):
  1. virtio-net driver → KVM
  2. KVM → virtio-net driver (interrupt)

- shared memory with the host kernel (vhost protocol)
- Allows Linux Bridge Zero Copy
- qemu / virtio-net device is on the control path only
- kernel thread [vhost] process packets
virtio vhost-usr / OVS-DPDK

- No context switches
- Shared memory between the guest and the Open vSwitch (requires huge pages)
- Zero copy
- qemu / virtio-net device is on the control path only
- KVM not in the path
- ovs-vswitchd process packets.
- Poll-mode-driver (PMD) takes 1 CPU core, 100%
PCI Passthrough

- No paravirtualized devices
- Direct access from the guest kernel to the PCI device
- Host, KVM and qemu are not on the data nor the control path.
- NIC driver in the guest

- No virtual networking
- No live migrations
- No filtering
- No control
- Shared devices via SR-IOV
Virtual Network Performance

All measurements are between two VMs on the same host

`# ping -f -c 100000 vm2`
# top -H -p 14826

```
top - 01:01:31 up 15 days, 21:48, 2 users, load average: 7.28, 6.03, 5.98
Threads: 6 total, 2 running, 4 sleeping, 0 stopped, 0 zombie
%Cpu(s): 16.9 us, 9.5 sy, 0.0 ni, 73.6 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 13175620+total, 1550248 free, 12736860+used, 2837352 buff/cache
KiB Swap:  0 total,  0 free,  0 used. 2975956 avail Mem

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>14852</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>2994152</td>
<td>64440</td>
<td>19700</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>28 vnc_worker</td>
</tr>
<tr>
<td>14849</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>2994152</td>
<td>64440</td>
<td>19700</td>
<td>R</td>
<td>40.0</td>
<td>0.0</td>
<td>0:35.98</td>
<td>CPU 1/KVM</td>
</tr>
<tr>
<td>14847</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>2994152</td>
<td>64440</td>
<td>19700</td>
<td>S</td>
<td>6.0</td>
<td>0.0</td>
<td>0:09.50</td>
<td>CPU 0/KVM</td>
</tr>
<tr>
<td>14834</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>2994152</td>
<td>64440</td>
<td>19700</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00.11</td>
<td>30 I0 iotthread1</td>
</tr>
<tr>
<td>14833</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>2994152</td>
<td>64440</td>
<td>19700</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00.01</td>
<td>9 qemu-kvm</td>
</tr>
<tr>
<td>14826</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>2994152</td>
<td>64440</td>
<td>19700</td>
<td>R</td>
<td>99.9</td>
<td>0.0</td>
<td>1:09.07</td>
<td>12 qemu-kvm</td>
</tr>
</tbody>
</table>
```
virtio vhost-net

```
# top -H -p 18225 -p 18241
```

```
top - 01:09:50 up 15 days, 21:57, 2 users, load average: 7.84, 7.14, 6.62
Threads: 7 total, 2 running, 5 sleeping, 0 stopped, 0 zombie
%Cpu(s): 17.0 us, 10.0 sy, 0.0 ni, 73.0 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 13175620+total, 1545220 free, 12737219+used, 2838800 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 2972400 avail Mem

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>P</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>18225</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64260</td>
<td>19524</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00.16</td>
<td>22</td>
<td>qemu-kvm</td>
</tr>
<tr>
<td>18232</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64260</td>
<td>19524</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>12</td>
<td>qemu-kvm</td>
</tr>
<tr>
<td>18234</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64260</td>
<td>19524</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00.13</td>
<td>22</td>
<td>i0 iothread1</td>
</tr>
<tr>
<td>18241</td>
<td>root</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>R</td>
<td>93.7</td>
<td>0.0</td>
<td>1:09.34</td>
<td>30</td>
<td>vhost-18225</td>
</tr>
<tr>
<td>18248</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64260</td>
<td>19524</td>
<td>S</td>
<td>0.7</td>
<td>0.0</td>
<td>0:10.92</td>
<td>23</td>
<td>CPU 0/KVM</td>
</tr>
<tr>
<td>18250</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64260</td>
<td>19524</td>
<td>R</td>
<td>65.4</td>
<td>0.0</td>
<td>0:53.86</td>
<td>13</td>
<td>CPU 1/KVM</td>
</tr>
<tr>
<td>18253</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64260</td>
<td>19524</td>
<td>S</td>
<td>0.0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>21</td>
<td>vnc_worker</td>
</tr>
</tbody>
</table>
```
virtio vhost-usr / OVS-DPDK

```
# top -H -p 18225 -p 27142
```

top - 01:19:59 up 15 days, 22:07, 2 users, load average: 4.18, 5.42, 6.22
Threads: 17 total, 2 running, 15 sleeping, 0 stopped, 0 zombie
%Cpu(s): 8.8 us, 3.8 sy, 0.0 ni, 87.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem: 13175620+total, 1551636 free, 12736377+used, 2840804 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 2980800 avail Mem

<table>
<thead>
<tr>
<th>PID</th>
<th>USER</th>
<th>PR</th>
<th>NI</th>
<th>VIRT</th>
<th>RES</th>
<th>SHR</th>
<th>S</th>
<th>%CPU</th>
<th>%MEM</th>
<th>TIME+</th>
<th>P</th>
<th>COMMAND</th>
</tr>
</thead>
<tbody>
<tr>
<td>27157</td>
<td>root</td>
<td>10</td>
<td>-10</td>
<td>18.982</td>
<td>151148</td>
<td>15868 S 0.0</td>
<td>0.1</td>
<td>0:01.61</td>
<td>5</td>
<td>0svs-vswitchd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>27156</td>
<td>root</td>
<td>10</td>
<td>-10</td>
<td>18.982</td>
<td>151148</td>
<td>15868 S 0.0</td>
<td>0.1</td>
<td>0:02.79</td>
<td>5</td>
<td>0svs-vswitchd</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23318</td>
<td>root</td>
<td>10</td>
<td>-10</td>
<td>18.982</td>
<td>151148</td>
<td>15668 R 99.9</td>
<td>0.1</td>
<td>2845:43</td>
<td>6</td>
<td>pmd21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18253</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64376</td>
<td>19524 S 0.0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>21</td>
<td>vnc_worker</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18250</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64376</td>
<td>19524 S 1.0</td>
<td>0.0</td>
<td>3:53.02</td>
<td>18</td>
<td>CPU 1/KVM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18248</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64376</td>
<td>19524 R 99.9</td>
<td>0.0</td>
<td>6:40.92</td>
<td>31</td>
<td>CPU 0/KVM</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18234</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64376</td>
<td>19524 S 0.0</td>
<td>0.0</td>
<td>0:00.13</td>
<td>7</td>
<td>iotthread1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18232</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64376</td>
<td>19524 S 0.0</td>
<td>0.0</td>
<td>0:00.00</td>
<td>12</td>
<td>qemu-kvn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18225</td>
<td>oneadmin</td>
<td>20</td>
<td>0</td>
<td>3000308</td>
<td>64376</td>
<td>19524 S 0.0</td>
<td>0.0</td>
<td>0:00.25</td>
<td>14</td>
<td>qemu-kvn</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Additional reading

- Deep dive into Virtio-networking and vhost-net

- Open vSwitch DPDK support
  https://docs.openvswitch.org/en/latest/topics/dpdk/
1. Hardware
2. Compute, Virtualization
3. Networking
4. Storage
5. Conclusion
Storage - virtualization

Virtualized

live migration
thin provisioning, snapshots, etc.

vs. Full bypass

only speed
Virtualized

- cache=none -- direct IO, bypass host buffer cache
- io=native -- use Linux Native AIO, not POSIX AIO (threads)
- virtio-blk vs virtio-scsi
- virtio-scsi multiqueue
- iothread

vs. Full bypass

- SR-IOV for NVMe devices
Storage - virtualization

Virtualized with io_uring

  guest kernel -> qemu ----(Linux Native AIO)--- > host kernel
  guest kernel -> qemu ----(io_uring)--- > host kernel

Virtualized with io_uring passthrough

  guest kernel ----(io_uring)--- > host kernel
Storage - vhost-user

**Virtualized with qemu bypass**

before:

guest kernel -> host kernel -> qemu -> host kernel -> storage client

with vhost-user:

guest kernel -> storage client
- Highly scalable and efficient architecture
- Scales up in each storage node & out with multiple nodes
Storage benchmarks

Beware: lots of snake oil out there!

- performance numbers from hardware configurations totally unlike what you’d use in production
- synthetic tests with high iodepth - 10 nodes, 10 workloads * iodepth 256 each. (because why not)
- testing with ramdisk backend

- synthetic workloads don't approximate real world (example)
Latency

.ops per second

best service

Latency
Latency (ops per second) vs. Latency

- **Best service**
- **Lowest cost per delivered resource**
- **Only pain**
- **Benchmarks**
Real load
2.5-3x lower latency at fixed load
Agenda

1. Hardware
2. Compute, Virtualization
3. Networking
4. Storage
5. Conclusion
Conclusion

KVM has the right tools to get very good performance, but not by default.

These are complex systems so don't guess, measure! Measure what matters to your team/company.

Work with partners who understand performance, because you can gain a lot!
Follow StorPool Online

@storpool

StorPool Storage

StorPool Storage

StorPool Storage

StorPool Storage

StorPool Storage

StorPool Storage

StorPool Storage

StorPool Storage
Thank you!

Boyan Krosnov
bk@storpool.com

www.storpool.com
@storpool