random thoughts on Ganeti operation

plain KVM/DRBD/bridged

Sascha Lucas

GISA GmbH Halle (Saale)

18th of June 2019 Ganeticon / Umeå

node OS: Ubuntu 16.04

- plain package 2.15.2-3 with some tweaks
  - livemigration progress broken

```
--- /old/usr/share/ganeti/2.15/ganeti/hypervisor/hv_kvm/__init__.py
+++ /new/usr/share/ganeti/2.15/ganeti/hypervisor/hv_kvm/__init__.py
@@ -379,7 +379,7 @@
   _MIGRATION_STATUS_RE = re.compile(r"Migration\s+status:\s+(\w+)",
                                     re.M | re.I)
   _MIGRATION_PROGRESS_RE = \
-    re.compile(r"\s*transferred\s+ram:\s+(?P<transferred>\d+)\s+kbytes\s*\n"
+    re.compile(r"\s*transferred\s+ram:\s+(?P<transferred>\d+)\s+kbytes\s*\n.*"
                r"\s*remaining\s+ram:\s+(?P<remaining>\d+)\s+kbytes\s*\n"
                r"\s*total\s+ram:\s+(?P<total>\d+)\s+kbytes\s*\n", re.I)
```

  - bash completion doesn't work out of the box

```
ln -s /usr/share/bash-completion/completions/ganeti /etc/bash_completion.d/
```

  - ganeti-instance-debootstrap broken → use 0.16-3ubuntu1 from 18.04/bionic
- good coverage of canonical 5y supported packages: command `ubuntu-support-status`

Ganeti „distributed switch“

- setup node with `/etc/network/interfaces`

```
source /etc/network/interfaces.group1/*
```

- put VLAN and bridge setup into one file: /e/n/i.group1/net100

```
auto bond0.100
iface bond0.100 inet manual
        vlan-raw-device bond0
        mtu 9000

auto br100
iface br100 inet manual
        bridge_ports bond0.100
        bridge_fd 0
        bridge_stp off
        bridge_maxwait 1
        mtu 9000
```

- distribute config to the cluster:

```
gnt-cluster copyfile /etc/network/interfaces.group1/net100
```

- bring up the interface: `gnt-cluster command ifup bond0.100`[1]
- but ... VLAN aware bridge would be simpler/better

[1] ignore errors from nodes not belonging to node group 1

memory management

part 1

what is default in Ubuntu (I assume in Debian too)

- qemu enables KSM by default: `/etc/default/qemu-kvm`
- KSM merges across NUMA nodes: `/sys/kernel/mm/ksm/merge_across_nodes`
- THP is enabled per default for qemu (madvise): `/sys/kernel/mm/transparent_hugepage/enabled`
- kernel NUMA balancer is enabled per default on multisocket systems

```
$ sysctl kernel.numa_balancing
kernel.numa_balancing = 1
```

All this scans your memory and possibly work against each other. This is probably not what NUMAists want.

memory management

part 2

numad to the rescue

- monitors your system and dynamically adjusts NUMA locality (memory and CPU-masks)
- sets THP_scan_sleep_ms from 10 to 1s
- manually disable qemu KSM
- manually disable kernel NUMA balancer

numad drawbacks

- sometimes to dynamic
- depends on the amount of CPU oversubscribe and instance CPU utilization
- NUMA-Node ping-pong → can trigger instance kernel watchdog on blocking CPU

memory management

numad and THP example

```
$ numastat -c qemu
Per-node process memory usage (in MBs)
PID               Node 0 Node 1  Total
---------------   ------ ------ ------
1003 (qemu-syste       8  32812  32819
1817 (qemu-syste       8  19339  19347
2556 (qemu-syste    4146      0   4146
3326 (qemu-syste    2094      0   2094
4502 (qemu-syste    2088      0   2088
5290 (qemu-syste       7   2097   2104
6050 (qemu-syste    2103      0   2103
6772 (qemu-syste    4121      0   4121
7492 (qemu-syste    6139      0   6139
8242 (qemu-syste    4108      0   4108
9003 (qemu-syste    4130      0   4130
9746 (qemu-syste    6166      0   6166
10511 (qemu-syst    4146      0   4146
12092 (qemu-syst    3420      0   3421
23702 (qemu-syst    7191      0   7191
28563 (qemu-syst       8   2449   2457
29938 (qemu-syst   16435      0  16436
30626 (qemu-syst       8  24537  24545
32157 (qemu-syst    8912      0   8912
---------------   ------ ------ ------
Total              75236  81235 156472
```

```
$ grep -i huge /proc/meminfo
AnonHugePages:  146348032 kB
```

```
$ lscpu | grep NUMA
NUMA node(s):           2
NUMA node0 CPU(s):      0-7,16-23
NUMA node1 CPU(s):      8-15,24-31
```

```
$ taskset -cp 1003
pid 1003's current affinity list: 8-15,24-31
```

post-copy migration

- memory write intensive VMs are hard to migrate
  - even with high migration_bandwidt=1000
  - high migration_downtime (>30 ms) may not be tolerable by instances
  - ... endless copying memory
- solution: post-copy migration
  - memory is copied after switching execution state from source to target node
  - steps to use:

    ```
    $ gnt-cluster modify -H kvm:migration_caps=postcopy-ram # (or x-postcopy-ram on qemu-2.5)
    ```

    on instance source node (ideally after >= 1 cycle/100% of memory transfer)

    ```
    $ echo "migrate_start_postcopy" | socat \
      STDIO UNIX-CONNECT:/var/run/ganeti/kvm-hypervisor/ctrl/some.vm.monitor
    ```

  - migrate_start_postcopy command must timed right to not confuse Ganetis migration status pull (info migrate)
  - in some development branch this feature was added to Ganeti. But it seems never released???

DRBD: simple, stupid

- can be called: software defined / distributed storage (hyper converged)
- needs little to no knowledge to be used by Ganeti
- however Ganeti has non optimal defaults
  - users struggle with static resync vs. dynamic resync controller (which one is active/to tune)
  - higher resync speeds ($>$ 100 MB/s) needs larger buffers
    i.e. 150MB/s → net-custom='--max-buffers 8000 --max-epoch-size 8000'
  - Ganetis setting c-plan-ahead=0 leads to skip DRBDs c-plan-ahead, which is 20 per default
    DRBDs c-plan-ahead != 0 enables the dynamic resync controller
    forcefully disable with disk-custom='--c-plan-ahead 0'
  - are even Debian DSA's struggling? https://dsa.debian.org/howto/install-ganeti/ (section DRBD optimization)
- split brain during live migration
  - DRBD is in dual primary during migration
  - split brain (standalone/standalone) will happen if DRBD gets disconnected during migration
  - happens when migration bandwidth saturates the link
  - qemu will finish migration, because it does not know anything about DRBD

hooks and tweaks

- growing a disk online: see https://github.com/saschalucas/ganeti-hook-grow-disk
- network anti spoofing
  - prevent MAC, ARP and IP spoofing
  - combination of up script ($CONF_DIR/kvm-vif-bridge) and in absence of down script instance-stop.pre-d hook
- I/O-limit your instances with cgroup-v1/blkio-controller
  - needs DIRECTIO (cache=none)
  - a single PV (major/minor) of the ganeti VG
  - example: limit 1000 write IOPS and 300MB/s read

```
mkdir /sys/fs/cgroup/blkio/some.vm
# here 8:16=/dev/sdb substitute with your major/minor number
echo "8:16 1000" > /sys/fs/cgroup/blkio/some.vm/blkio.throttle.write_iops_device
echo "8:16 314572800" > /sys/fs/cgroup/blkio/some.vm/blkio.throttle.read_bps_device
echo PID_OF_QEMU_INST > /sys/fs/cgroup/blkio/some.vm/cgroup.procs
```

- CPU fairness: i.e. 8 Core node, two instances: 4 vCPUs and 8 vCPUs
  - without cgroups: 4c=1/3 and 8c=2/3
  - with cgroups: 4c=1/2 and 8c=1/2

security

not enabled per default

- Want your customers share their data? No? Wiping disk is obligatory: --prealloc-wipe-disks=yes
- KVM can use chroot: --hypervisor-parameters=kvm:use_chroot=true
- UID separation between VMs on the same node:
  - use https://github.com/grnet/nss-uidpool to get 100 UIDs without creating 100 users on each node
  - and --uid-pool=10002-10100 --hypervisor-parameters=kvm:security_model=pool

security

Spectre/MDS mitigation not enabled in qemu by default

- not enabled by your hardware: disable SMT
  check /sys/devices/system/cpu/vulnerabilities/mds
- showing available CPU models: qemu-system-x86_64 -cpu ?
- chose a IBRS variant and enable this flags:
  - +pcid: mitigate the cost of the Meltdown
  - +ssbd: required to enable the CVE-2018-3639 fix
  - +md-clear: signal that host can mitigate MDS
  - enforce: don't start if the host can't fulfill the desired CPU type/flags
  - see also: https://www.berrange.com/tags/ssbd/
- testing the cluster for a common CPU the model:

```
$ gnt-cluster command 'qemu-system-x86_64 -cpu XXXXXXX-IBRS,+pcid,+ssbd,+md-clear,enforce \
  -machine accel=kvm -nographic -nodefaults -boot c,reboot-timeout=1 -no-reboot'
```

- setting a cpu type

```
$ gnt-cluster modify -H kvm:cpu_type='XXXXXXX-IBRS\,+pcid\,+ssbd\,+md-clear\,enforce'
```

instance creation with ganeti-instance-debootstrap

preparation

- features
  - uses kernel and initrd from node (best when node and instance OS are same)
  - no kernel package inside instance (modules from initrd must be sufficient)
  - no boot loader (grub)
  - caches debootstrap in a tar file
- /etc/default/ganeti-instance-debootstrap

```
MIRROR="https://put.your-mirror.here/ubuntu"
ARCH="amd64"
SUITE="xenial"
EXTRA_PKGS="acpid,ssh"
COMPONENTS="main,universe"
GENERATE_CACHE="yes"
CLEAN_CACHE="14"
```

- /etc/ganeti/instance-debootstrap/hooks/000interfaces
  - enhanced with instance network configuration WRT gnt-network
  - see https://github.com/ganeti/instance-debootstrap/pull/1

instance creation with ganeti-instance-debootstrap

in action

```
gnt-instance add -t drbd --disk 0:size=1G --net 0:network=gruen97,ip=172.28.97.234\
  -H kvm:kernel_path=/vmlinuz,initrd_path=/initrd.img,root_path=/dev/vda1,\
     kernel_args='ro elevator=noop net.ifnames=0'\
  -B vcpus=1,memory=1G -o debootstrap+default --no-name-check --no-ip-check test.vm
```

make sure:

- specify an IP if you don't have DHCP
- ganeti-instance-debootstrap assumes eth0 as NIC name, so don't forget net.ifnames=0

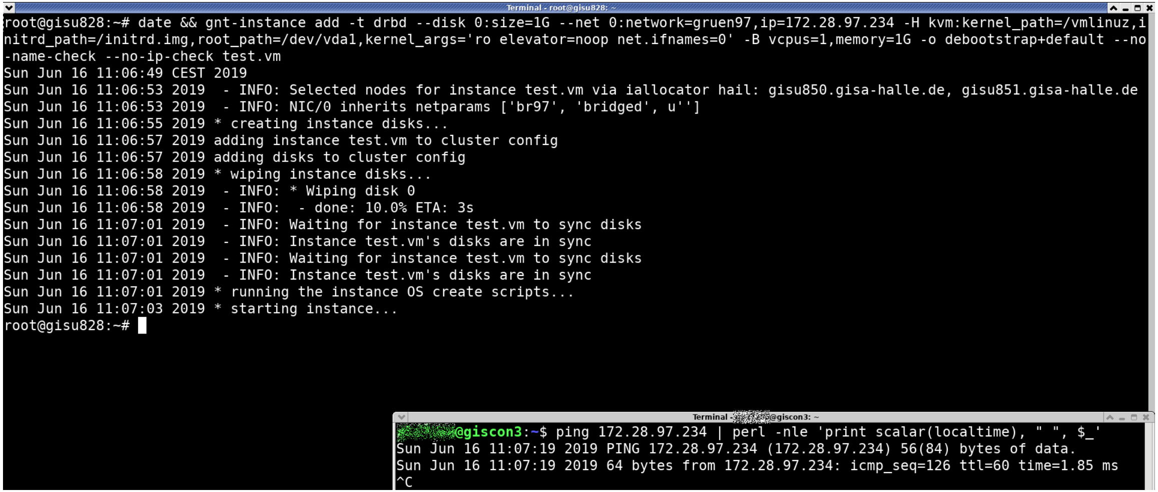
very fast:

- 14s from command submission to instance startup
- 30s from command submission to instance first ping

ganeti-instance-debootstrap can be enhanced i.e. by

- using libguestfs (i.e. install kernel/grub)
- overcome debootstrap limitation $\rightarrow$ install a fully upgraded system (multistrap?)

instance creation with ganeti-instance-debootstrap – who can do faster?



```
root@gisu828:~# date && gnt-instance add -t drbd --disk 0:size=1G --net 0:network=gruen97,ip=172.28.97.234 -H kvm:kernel_path=/vmlinuz,initrd_path=/initrd.img,root_path=/dev/vda1,kernel_args='ro elevator=noop net.ifnames=0' -B vcpus=1,memory=1G -o debootstrap+default --no-name-check --no-ip-check test.vm
Sun Jun 16 11:06:49 CEST 2019
Sun Jun 16 11:06:53 2019  - INFO: Selected nodes for instance test.vm via iallocator hail: gisu850.gisa-halle.de, gisu851.gisa-halle.de
Sun Jun 16 11:06:53 2019  - INFO: NIC/0 inherits netparams ['br97', 'bridged', u'']
Sun Jun 16 11:06:55 2019 * creating instance disks...
Sun Jun 16 11:06:57 2019 adding instance test.vm to cluster config
Sun Jun 16 11:06:57 2019 adding disks to cluster config
Sun Jun 16 11:06:58 2019 * wiping instance disks...
Sun Jun 16 11:06:58 2019  - INFO: * Wiping disk 0
Sun Jun 16 11:06:58 2019  - INFO:  - done: 10.0% ETA: 3s
Sun Jun 16 11:07:01 2019  - INFO: Waiting for instance test.vm to sync disks
Sun Jun 16 11:07:01 2019  - INFO: Instance test.vm's disks are in sync
Sun Jun 16 11:07:01 2019  - INFO: Waiting for instance test.vm to sync disks
Sun Jun 16 11:07:01 2019  - INFO: Instance test.vm's disks are in sync
Sun Jun 16 11:07:01 2019 * running the instance OS create scripts...
Sun Jun 16 11:07:03 2019 * starting instance...
root@gisu828:~#
```

```
[redacted]@giscon3:~$ ping 172.28.97.234 | perl -nle 'print scalar(localtime), " ", $_'
Sun Jun 16 11:07:19 2019 PING 172.28.97.234 (172.28.97.234) 56(84) bytes of data.
Sun Jun 16 11:07:19 2019 64 bytes from 172.28.97.234: icmp_seq=126 ttl=60 time=1.85 ms
^C
```

GISA That's IT.

THANKS

Questions?