Backup System

Document Author: arho.virkki@vtt.fi

Automated Backups

Wiki, version control system, raw data, PostgreSQL data base and several other localtions are automatically backed up to NAS at */nas/backup*.The cron scheduler at *ktp@gradient* is responsible for running the backup scrtips. For details, see `crontab -l`.

Backup scripts

The backup script are stored in *Common.git* repository and located at *Common/backup/*. Currently, the scripts include several shell and R scripts.

```
.
 data
ăă  backup_git.R
ăă  backup_raw_data.R
 KVM
ăă  backup_all_domains.sh
ăă  backup_quickboot_domain.sh
ăă  backup_reboot_domain.sh
ăă  virsh_shutdown_domain.sh
 postgres
     backup_pg.sh
```

Backing up PostgreSQL

There are several possibilites of making a full backup of a running PostgreSQL cluster. We use now Option 1 below since it was fastest.

Option 1: Run pg_dumpall at gradient.vsshp.net

First, install a matching version of the PostgreSQL *pg_dumpall* tool. (In this case postgresql94 - PostgreSQL client programs and libraries: http://yum.postgresql.org/9.4/redhat/rhel-7-x86_64/repoview/postgresql94.html) The development libraries are not strictly necessary (but are needed for R support, if even needed).

```
wget http://yum.postgresql.org/9.4/redhat/rhel-7-x86_64/postgresql94-9.4.8-1PGDG.rhel7.x86_64.rpm
wget
    http://yum.postgresql.org/9.4/redhat/rhel-7-x86_64/postgresql94-libs-9.4.8-1PGDG.rhel7.x86_64.rpm
wget
    http://yum.postgresql.org/9.4/redhat/rhel-7-x86_64/postgresql94-devel-9.4.8-1PGDG.rhel7.x86_64.rpm
```

Install the packages with

```
sudo rpm -ivh postgresql94-*
```

Ensure that the password for the PostgreSQL root user is set at the database machine. Then, create a *.pgpass* file under the *ktp* user home directory. The generic format for the file is

```
hostname:port:database:username:password
```

In this case, we allow *postgres* user to access all databases (*) with

```
echo "ktppg.vsshp.net:5432:*:postgres:<passwd here>" > .pgpass
chmod og-rwx .pgpass
chmod 0600 .pgpass
```

where the password is set to *ktp* (normal short) password. Try that the arrangement works with

```
psql -U postgres -h ktppg.vsshp.net -d postgres
```

Then, make a copy of the database with

```
today=$(date --iso-8601)
mkdir -p  "/var/local/backup/ktppg/$today"

time /usr/pgsql-9.4/bin/pg_dumpall \
-w -h ktppg.vsshp.net -U postgres -l postgres | lz4 | \
split -a 2 -b 1G - "/var/local/backup/ktppg/$today/pgdump.lz4_"
```

The execution took about 2 hours and 36 minutes as of this writing (2016-08-03)

```
real    156m44.363s
user    48m14.420s
sys     26m17.773s
```

Option 2: Run at gradient but call pg_dumpall at ktppg

Copy the public key of ktp@gradient to ktp@ktppg

```
scp .ssh/id_rsa.pub ktp@ktppg.vsshp.net
```

Then, add the key to the postgres user

```
sudo sh -c "cat id_rsa.pub >> ~postgres/.ssh/authorized_keys"
```

Now we can backup the whole database from ktp@gradient with the following script:

```
#!/bin/bash

# PostgreSQL cluster backup script

# Author(s) : Arho Virkki
# Copyright : VTT Technical Reseach Centre of Finland
# Date      : 2016-07-29

today=$(date --iso-8601)
mkdir -p  "/var/local/backup/ktppg/$today"

time ssh postgres@ktppg.vsshp.net "pg_dumpall | lz4 " | \
split -b 1G - "/var/local/backup/ktppg/$today/pgdump.lz4_"
```

The execution took about three hours (2016-08-03)

```
[ktp@gradient postgres]$ ./backup_pg.sh

real    185m44.510s
user    5m32.734s
sys     4m3.767s
```

Option 3: Run the backup at ktppg.vsshp.net

ktp@ktppg:~$ sudo apt-get install liblz4-tool

Allow

```
sudo su - postgres
ssh-keygen
ssh-copy-id -i .ssh/id_rsa.pub ktp@gradient.vsshp.net
```

The execution took about 3 and half hours (2016-08-03)

```
time sudo -u postgres sh -c \
"pg_dumpall | lz4 | ssh ktp@gradient.vsshp.net \
\"split -a 2 -b 1G - /nas/backup/ktppg/$(date --iso-8601).lz4_\""

real    211m47.591s
user    28m51.380s
sys     15m27.717s
```

Backing up KVM virtual machines

Shut down for maintenance

While it is possible to back up live machines with snapshots, it is safest to power off the machine to ensure a consistent state of the virtual disk. Otherwise, we need to make sure that e.g. no database transactions are running while the snapshot was taken for the backup.

Run the backup script

The backup scripts reside on the *Common* repository under `backup/KVM`, and an instance of *Common* should be found at `ktp@gradient.vsshp.net:/home/ktp/Common`. There are also symbolic links at `/usr/bin` for *sudo* access.

Examples:

```
sudo backup_reboot_domain.sh ktpgit
time sudo backup_reboot_domain.sh ktpgit pbzip2
```

Typical output:

```
[ktp@gradient images]$ time sudo backup_reboot_domain.sh ktptest pbzip2
Waiting for ktptest to shut off..
Backing up ktptest into
/nas/backup/images/2016-01-15_ktptest.xml
/nas/backup/images/2016-01-15_ktptest.qcow2.tar.bz2
Backup done
Domain ktptest started

real    8m6.144s
user    56m55.455s
sys 5m38.021s
```

Ensure that the machine responds to ACPI poweroff

On Ubuntu hosts, save the original script which responds to power button

```
cd /etc/acpi/
sudo mv powerbtn.sh powerbtn_orig.sh
```

and edit the *powerbtn.sh* to only contain the following line

```
#!/bin/shs
/sbin/poweroff
```

to disable any user interactivity required for poweroff (such as the "Would you like to…" prompts).

Why the backup is slow?

A "recent" discussion at *comp.unix.internals* (1990) explains that "…you cannot tell the difference between a hole and an equivalent number of nulls without reading raw blocks…". Hence tar needs to read the whole file, since it is a file-system independent tool (xfs, ext2/3/4, ntfs and nfs all work). For details, see: http://www.delorie.com/gnu/docs/tar/tar_118.html

Appendix A: Tar Performance with Different Compression Levels

For details, see e.g.

- http://www.gnu.org/software/tar/manual/tar.pdf
- http://serverfault.com/questions/66338/how-do-you-synchronise-huge-sparse-files-vm-disk-images-be

Tar with no compression

```
time tar -cSf /nas/backup/images/`date --iso-8601`_ktptest.qcow2.tar \
-C /var/lib/libvirt/images/ ktptest.qcow2

real    9m48.137s
user    2m5.844s
sys     4m35.557s

time tar -xvSf /nas/backup/images/2016-01-14_ktptest.qcow2.tar
ktptest.qcow2

real    2m42.673s
user    0m1.598s
sys     0m43.269s
```

Tar with gzip (I/O bound)

```
time tar -cSzf /nas/backup/images/`date --iso-8601`_ktptest.qcow2.tar.gz \
-C /var/lib/libvirt/images/ ktptest.qcow2

real    17m35.627s
user    13m28.720s
sys     4m21.880s

time tar -xvzf /nas/backup/images/2016-01-14_ktptest.qcow2.tar.gz

real    2m48.644s
user    2m9.246s
sys     1m3.905s
```

Tar with lz4

```
time tar -I lz4 \
-cSf /nas/backup/images/`date --iso-8601`_ktptest.qcow2.tar.lz4 \
-C /var/lib/libvirt/images/ ktptest.qcow2

real    8m8.091s
user    3m3.050s
sys     4m1.563s

time tar -I lz4 -xvf /nas/backup/images/2016-01-14_ktptest.qcow2.tar.lz4

real    2m44.864s
user    0m18.914s
sys     1m12.399s
```

Tar with pbzip2

```
time tar -I pbzip2 \
-cSf /nas/backup/images/`date --iso-8601`_ktptest.qcow2.tar.bz2 \
-C /var/lib/libvirt/images/ ktptest.qcow2

real    8m1.982s
user    54m55.448s
sys     5m28.025s

time tar -I pbzip2 -xvf /nas/backup/images/2016-01-14_ktptest.qcow2.tar.bz2

real    1m42.990s
user    15m34.306s
sys     1m52.933s
```

File size comparison

```
[arho@gradient images]$ ls -lha
-rw-r--r-- 1 arho wheel  14G 14.1. 16:04 2016-01-14_ktptest.qcow2.tar
-rw-r--r-- 1 arho wheel 5,7G 14.1. 22:45 2016-01-14_ktptest.qcow2.tar.bz2
-rw-r--r-- 1 arho wheel 6,0G 14.1. 22:24 2016-01-14_ktptest.qcow2.tar.gz
-rw-r--r-- 1 arho wheel 7,9G 14.1. 21:59 2016-01-14_ktptest.qcow2.tar.lz4
```

Appendix B: Typical Execution Times

Initial sized of the images

```
[ktp@gradient images]$ ls -lhs
total 929G
 59G -rw-r--r-- 1 qemu qemu  2.1T Jan 16 11:17 ktpanalytics.qcow2
 17G -rw-r--r-- 1 qemu qemu  513G Jan 16 11:44 ktpdoc.qcow2
 44G -rw-r--r-- 1 qemu qemu  2.1T Jan 16 11:17 ktpgit.qcow2
523G -rw-r--r-- 1 qemu qemu   11T Jan 16 11:45 ktphadoop.qcow2
270G -rw-r--r-- 1 qemu qemu   11T Jan 16 11:17 ktppg.qcow2
 19G -rw-r--r-- 1 root root  257G Jan 15 15:01 ktptest.qcow2
```

Corresponding execution times

```
[ktp@gradient KVM]$ sudo ./backup_all_domains.sh
This will take long, and automatically reboot all virtual
machines along the way!
Are you sure [y/n]: y
Backing up ktpdoc into
/nas/backup/images/2016-01-15_ktpdoc.xml
/nas/backup/images/2016-01-15_ktpdoc.qcow2.tar.lz4
Backup done
Domain ktpdoc started

real    14m28.701s
user    4m51.211s
sys     8m5.817s

Waiting for ktpgit to shut off..
Backing up ktpgit into
/nas/backup/images/2016-01-15_ktpgit.xml
/nas/backup/images/2016-01-15_ktpgit.qcow2.tar.lz4
Backup done
Domain ktpgit started

real    49m24.771s
user    16m55.121s
sys     29m56.354s

Waiting for ktpanalytics to shut off...
Backing up ktpanalytics into
/nas/backup/images/2016-01-15_ktpanalytics.xml
/nas/backup/images/2016-01-15_ktpanalytics.qcow2.tar.lz4
Backup done
Domain ktpanalytics started

real    55m36.373s
user    18m6.019s
sys     31m46.295s

Waiting for ktppg to shut off...
Backing up ktppg into
/nas/backup/images/2016-01-15_ktppg.xml
/nas/backup/images/2016-01-15_ktppg.qcow2.tar.lz4
Backup done
Domain ktppg started

real    262m9.829s
user    89m34.183s
sys     160m33.088s

Waiting for ktphadoop to shut off...
ing up ktphadoop into
/backup/images/2016-01-16_ktphadoop.xml
/backup/images/2016-01-16_ktphadoop.qcow2.tar.lz4
Backup done
Domain ktphadoop started

    302m29.830s
    98m37.440s
    174m44.484s
```