How do I configure BeeGFS on demand?

BeeOND ("**Bee**GFS **OnD**emand", pronounced like the word "beyond") was developed to enable easy creation of one or multiple BeeGFS instances on the fly. Nowadays in most HPC cluster systems compute nodes are equipped with internal hard disks or SSDs, which could deliver an additional performance advantage. The problem with the internal drives in compute nodes is that they provide neither the advantage of a single name space across multiple machines nor the flexibility and performance of a shared parallel file system. BeeOND solves this problem by creating a shared parallel filesystem on a "per-job basis" across all compute nodes that are part of the particular compute job, exactly for the runtime of the job.

To configure BeeOND on a Bright cluster you need to enable the beegfs repo for your distribution in the software images you are using. In the following example we are using the **default-image** in a **RHEL7** distribution.

```
cd /cm/images/default-image/etc/yum.repos.d

wget https://www.beegfs.io/release/latest-stable/dists/beegfs-rhel7.repo
```

Then you need to install the **beeond** package and reboot the affected compute nodes.

```
yum install beeond --installroot=/cm/images/default-image
```

For users with sudo permissions

If an HPC users has permissions to run **sudo**, they can create and delete the filesystem themselves in their job. An example of a **slurm** workload manager job follows:

```
#!/bin/bash
```

```
#SBATCH --nodes=3

module load shared

module load slurm

# create the nodelist by asking slurm what nodes are allocated to

# this job

scontrol show hostnames $SLURM_JOB_NODELIST > nodefile-$SLURM_JOB_ID

# create the filesystem by using the allocated jobs

sudo beeond start -n nodefile-$SLURM_JOB_ID -d /tmp -c /mnt/$SLURM_JOB_ID

# copy the work files into the beeond FS

beeond-cp copy -n nodefile-$SLURM_JOB_ID /home/sbrennan/dataset1 /mnt/$SLURM_JOB_ID

# the actual job lives here

sleep 60

# tear down the filesystem at the end

sudo beeond stop -n nodefile-$SLURM_JOB_ID -L -d
```

This approach has a couple of disadvantages, namely:

- Users need to be able to use **sudo**
- If a job returns before to run **beeond stop**, the temporary filesystem will be not removed

Using workload manager’s hooks

A better approach would be to rely on the workload manager’s hooks to setup and tear down the temporary filesystem for the jobs. This allows users to get this functionality without having to modify their jobs. They also won’t need to use **sudo**.

Workload Management: How do I configure BeeGFS on demand?

Slurm for example provides pre and post job hooks that we can use to automatically setup and tear down beeond. To do this we need to:

- Create a few scripts that allow multiple hooks to run one after the other
- Create the beeond hooks
- Modify the slurm configuration in order to use them

We first need to provide a generic prolog for **slurmd**. This will run on the head node before every job.

```
[root@master ~] cat >/cm/local/apps/slurm/var/prolog-slurmctld <<EOF
#!/bin/bash
PREJOBS_DIR=/cm/local/apps/slurm/var/ctld-prologs
# execute previous prejob
/cm/local/apps/cmd/scripts/prolog-prejob
# execute our own prologs
for script in ${PREJOBS_DIR}/*; do
    if [ -x "${script}" ]; then
    source "${script}"
    fi
done
EOF
[root@master ~] chmod +x /cm/local/apps/slurm/var/prolog-slurmctld
```

We then do the same for a generic epilog:

```
[root@master ~] cat >/cm/local/apps/slurm/var/epilog-slurmctld <<EOF
#!/bin/bash
EPILOGS_DIR=/cm/local/apps/slurm/var/ctld-epilogs
# execute our own epilogs
for script in ${EPILOGS_DIR}/*; do
    if [ -x "${script}" ]; then
        source "${script}"
    fi
done
EOF
[root@master ~] chmod +x /cm/local/apps/slurm/var/epilog-slurmctld
```

Next step is to then create the specific scripts for **beeond**. The prolog:

```
[root@master ~] mkdir -p /cm/local/apps/slurm/var/ctld-prologs
[root@master ~] cat >/cm/local/apps/slurm/var/ctld-prologs/beeond.sh <<EOF
#!/bin/bash
# slurm strips out PATH during hooks execution
export PATH=$PATH:/bin:/cm/shared/apps/slurm/current/bin
NODEFILE=/cm/shared/jobs-nodelists/nodes-${SLURM_JOB_ID}
```

```
CLIENT_MNT=/mnt/job-${SLURM_JOB_ID}

JOBS_DATA_DIR=/tmp

# setup beeond

mkdir -p $(dirname ${NODEFILE})

scontrol show hostnames "${SLURM_JOB_NODELIST}" > "${NODEFILE}"

beeond start -n "${NODEFILE}" -d "${JOBS_DATA_DIR}" -c "${CLIENT_MNT}"

EOF

[root@master ~] chmod +x /cm/local/apps/slurm/var/ctld-prologs/beeond.sh
```

And the epilog:

```
[root@master ~] mkdir -p /cm/local/apps/slurm/var/ctld-epilogs

[root@master ~] cat >/cm/local/apps/slurm/var/ctld-epilogs/beeond.sh <<EOF

#!/bin/bash

# slurm strips out PATH during hooks execution

export PATH=$PATH:/bin:/cm/shared/apps/slurm/current/bin

NODEFILE=/cm/shared/jobs-nodelists/nodes-${SLURM_JOB_ID}

# tear down beeond

beeond stop -n "${NODEFILE}" -L -d

rm -f "${NODEFILE}"

EOF

[root@master ~] chmod +x /cm/local/apps/slurm/var/ctld-epilogs/beeond.sh
```

The next step is to change the slurm configuration, in order to use **root** as user for **slurmd.** To do that set **SlurmdUser=root** in **/etc/slurm/slurm.conf**.

Then you have to change the epilog in slurm.conf:

EpilogSlurmctld=/cm/local/apps/cmd/scripts/epilog-slurmctld

To change the prolog instead, you have to change the parameter through **cmsh:**

```
[cluster1]% device use master

[cluster1->device[cluster1]]% roles

[cluster1->device[cluster1]->roles]% use slurmserver

[cluster1->device[cluster1]->roles[slurmserver]]% set prologslurmctldprogram
/cm/local/apps/cmd/scripts/prolog-slurmctld

[cluster1->device[cluster1]->roles[slurmserver]]% commit
```

Now for every job you would have a temporary scratch space automatically setup. An example of job that takes advantage of that would be:

```
[mike@cluster1 ~]$ cat example.job

#!/bin/bash

#SBATCH --ntasks=2

NODEFILE=/cm/shared/jobs-nodelists/nodes-${SLURM_JOB_ID}

CLIENT_MNT=/mnt/job-${SLURM_JOB_ID}

echo copy some stuff
```

Workload Management: How do I configure BeeGFS on demand?

```
dd if=/dev/zero of=data bs=4k count=1k

beeond-cp copy -n "${NODEFILE}" data "${CLIENT_MNT}"

echo read it back

ls -lah "${CLIENT_MNT}/data"

echo done
```

Unique solution ID: #1429
Author: Michele Bertasi
Last update: 2018-05-24 12:45