Downstream Ironic Software RAID support at CERN

Arne Wiebalck (Arne.Wiebalck@cern.ch)
Daniel Abad (D.Abad@cern.ch)

CERN, September 2018

TL;DR

CERN added downstream software RAID support to Ironic. Just as for hardware RAID, the software RAID setup is done during cleaning: the hardware manager creates/deletes md devices and -- triggered by the conductor -- the IPA sets up grub2 to boot from the md device. The current implementation is basic, and hence limited, but could be used as a starting point to discuss the integration of software RAID support into upstream Ironic.

Introduction

CERN runs an OpenStack deployment with close to 10k hypervisors across two data centres. We have recently introduced Ironic to add bare metal provision to our service offering (in addition to VMs and containers). The main goals include: to consolidate accounting, to simplify our provisioning workflows, and to allow for new use cases (e.g. containers on bare metal). Our current Ironic deployment has ~1'300 active nodes, and the plan is to enrol the majority of the other physical nodes over the course of the next 6-12 months.

All hypervisors in the CERN OpenStack deployment rely on software/md RAID: RAID-1 and RAID-10 for service hypervisors, RAID-0 for batch compute. Many of the other (non-virtualised) services, such as storage, use software RAIDs as well. Therefore, the support of software RAID in Ironic is of great interest to us.

Over the past weeks, we have made a first attempt to add software RAID support to Ironic and we will briefly describe in the following what we did to get to a working prototype. We share this with the intention to get feedback and start a design that could be of wider interest and that would be apt to be included into Ironic's code base.

Initial implementation

The approach we took is:

- md device creation/deletion is done in our hardware manager as cleaning steps (this should eventually move to deployment steps once these become available);

- We always create a small RAID-1 at the beginning of the devices for the system. This avoids additional complexity (grub support for booting from RAID-N) and gives some flexibility (as the RAID-N can be assembled by the hardware manager, by user-data/cloud-init, or by a config' management system such as Puppet):
    - All disks get two partitions: p1/p2;
    - The p1 partitions form a RAID-1;
    - The p2 partitions form the desired RAID-N.

- The RAID-1 is exported via iSCSI to the conductor to deploy the same whole disk image(s) as we use for virtual instances; this avoids to have dedicated "physical instance" images.

Our approach required to patch code in the following areas:

- CERN hardware manager
    - `create_configuration()`: creates partitions p1/p2, RAID-1 and RAID-N;
    - `delete_configuration()`: removes all md devices and partitions.
    - The information which type of RAID to create is taken from the node field '`target_raid_config`'

- Ironic
    - Trigger the installation of grub on the holder drives for whole disk images (which is usually not necessary as whole disk images come with grub); this is done in:

      Patched:
      `ironic/drivers/modules/agent_base_vendor.py: configure_local_boot()`

- IPA
    - Allow for md devices to be considered for deployment: https://review.openstack.org/#/c/592639 (this is currently part of our hardware manager, simply b/c it was initially easier to add it here than patching the IPA);
    - Detect md devices and identify the holder device to install grub there when the conductor requests it.

      New:
      `ironic_python_agent/extensions/image.py: _is_md_device():`
      Check if a device is an md device.

      `ironic_python_agent/extensions/image.py: _md_get_components():`
      Get the components of an md device, needed e.g. for cleanup or md restart.

      `ironic_python_agent/extensions/image.py: _md_get_holders()::`
      Get the holder disks of an md device, needed to identify the correct place for grub.

      `ironic_python_agent/extensions/image.py: _md_restart():`
      Restart an md device, needed for grub to see all devices.

      Patched:

`ironic_python_agent/extensions/image.py: _get_partitions():`
Return the deploy md device. Should find it eventually via root_uuid.

`ironic_python_agent/extensions/image.py: _install_grub2():`
Install grub on all holder disks in case we deploy on an md device.

Limitations / Drawbacks / Shortcuts:

- The size of p1 is currently hardcoded to 16GB;
- We support the creation of only one single RAID-N;
- All available disks are used to form the RAID-N, i.e. no support for disk selection;
- No partitioning is supported.
- The deploy partition is hard coded (first partition that the conductor creates on the exported md device). Need to pass the root_uuid and help _get_partition() to identify the correct partition eventually.
- Changing the RAID config of a node requires the node to go through cleaning (and the target RAID config needs to be changed before).

Code:

CERN Hardware Manager:
https://github.com/cernops/cern-ironic-hardware-manager/commit/7f6d892ec4848a09000ed1f28f3137bf8ba917f0

IPA:
https://github.com/cernops/ironic-python-agent/commit/bddac76c4d100af0103a6bc08b81dd71681a9c02

Ironic:
https://github.com/cernops/ironic/commit/581e65f1d8986ac3e859678cb9aadd5a5b06ba60