Disk Types Proposal v3

*** * * SHARED EXTERNALLY * * ***

nlavine@google.com

Introduction

This document proposes the following changes in how PKB deals with disk types:

1. Specify disk types using provider-specific strings, rather than generic names
2. Record metadata about disks with common keys and values for easy comparison.
3. Rename “scratch_disk_*” flags to “data_disk_*”
4. Add “boot_disk_*” flags to specify the boot disk. This will at least include boot_disk_size; it may or may not include more

Examples

Here are some examples of PKB commands under our proposal and the metadata that would be recorded with the resulting samples.

Command: pkb.py --data_disk_type=pd-ssd --data_disk_size=500
Metadata:

```
|data_disk_0_type:pd-ssd|data_disk_0_size:500|data_disk_0_replication:zone|data_disk_0_media:ssd|data_is_boot:False|boot_disk_type:pd-standard|boot_disk_size:10|boot_disk_replication:zone|boot_disk_media:hdd|
```

Command: pkb.py --cloud=DigitalOcean --data_disk_size=30
Metadata:

```
|data_disk_0_type:digitalocean_disk|data_disk_0_size:30|data_disk_0_replication:none|data_disk_0_media:ssd|data_is_boot:True|boot_disk_type:digitalocean_disk|boot_disk_size:30|boot_disk_replication:none|boot_disk_media:ssd|
```

Command: pkb.py --data_disk_type=local-ssd --gcp_num_local_ssds=1
Metadata:

```
|data_disk_0_type:local-ssd|data_disk_0_size:375|data_disk_0_replication:none|data_disk_0_media:ssd|data_is_boot:False|boot_disk_type:pd-s
```

```
tandard|boot_disk_size:10|boot_disk_replication:zone|boot_disk_media:hdd|
```

Command: pkb.py --cloud=AWS --data_disk_type=instance-store --machine_type=d2.2xlarge

Metadata:

```
|data_disk_0_type:local|data_disk_0_size:12000|data_disk_0_replication:none|data_disk_0_media:hdd|data_is_boot:False|boot_disk_type:ami|
```

Command: pkb.py --cloud=Kubernetes --data_disk_type=rbd --data_disk_size=100

Metadata:

```
|data_disk_0_type:rbd|data_disk_0_size:100|data_disk_0_replication:zone|data_is_boot:False|boot_disk_type:pod|
```

Command: pkb.py --benchmark_config_file=bench.yaml (static VMs in config file)

Metadata:

```
|data_disk_0_type:static|boot_disk_type:static| … (static VM file can specify more metadata if it wants to)
```

Recording Disk Metadata

Even though disks from different providers will have different names, we still want to record metadata about them so we can compare benchmark results more easily. Each provider's _disk.py file will contain a mapping from that provider's disk types to dictionaries of metadata keys and values, and the samples from a benchmark will contain the metadata keys and values corresponding to the disk type used.

To make results comparable across providers, keys will come from the following list:

- data_disk_0_size: size, in GB. If a RAID array, the total size.
- data_disk_0_type: the provider-dependent name for the disk type.
- data_disk_0_media: primary storage medium. Possible values "hdd," "ssd."
- data_disk_0_replication: replication policy. Possible values "none," "zone," "region" (note: GCP, AWS, Azure, and Rackspace all use the term "region," so I think we can call it standard.)
- data_disk_0_num_stripes: the number of physical devices that are RAIDed together to form this logical disk.
- data_disk_1_{size,type,media,replication}: same as above, for a second logical data disk (data_disk_1 has a different mount point than data_disk_0, so RAID arrays do not qualify.)
- num_data_disks: the number of data disks. I.e. will be 2 for copy benchmark, 1 for most (all?) other benchmarks.

- data_is_boot: whether the data disk is also the boot disk. Possible values “True,” “False.” We assume that data disk 0 is the boot disk if there is more than one.
- boot_disk_{size,type,media,replication,num_stripes}: same as above, but for the boot disk.

Providers can omit keys that don’t make sense for them. For instance, Kubernetes may not be able to record the primary storage medium of a Rados Block Device. Providers may also add keys that only make sense with that provider, but they should be prefixed with the provider’s abbreviation. For instance, “aws_provisioned_iops.”

Changing Disk Type Names

We will call all disk types by the names that providers give them, instead of our own names. This means that on AWS (for instance), the disk types will be gp2, io1, standard, and local. On GCP, they will be standard, pd-ssd, and local-ssd. Etc. for other providers. This fixes the problem that different providers’ disk types are hard to map onto a set of standard PD names.

There will be a transition period, during which the new and old behaviors will both work. If a user tries to use the old disk type names, PKB will print a warning and then automatically translate their command to use the new names. This will happen for both flags and configs. For instance, pkb.py --cloud=GCP --scratch_disk_type=remote_ssd --scratch_disk_size=50 will be equivalent to pkb.py --cloud=GCP --data_disk_type=pd-ssd --data_disk_size=50 during the transition.

When the user passes a disk type flag, we will stop with an error if the disk type is not in the disk type metadata table for that provider. However, we will not necessarily check that the disk type flag is compatible with the disk size flag, or any other provider-specific constraints. This fits PKB’s general philosophy of passing info through to provider-specific tools with minimal changes and letting them accept or reject it.

Changing Disk Flags

We will rename PKB’s current scratch_disk_type and scratch_disk_size flags to data_disk_type and data_disk_size, because we have received feedback that “scratch_disk” is not always a good name for the benchmarking disk. In addition, we will add a flag boot_disk_size to let the user specify the size of the boot disk on providers that support it (at least GCP, and I believe Rackspace when booting from a network disk).

During the transition period, users will be able to use the old and the new flag names. If they use the old names, PKB will print a warning but continue. This applies only to flags - configs never used the name scratch_disk, and so are not affected.