CACHE POOL

PURPOSE

Use a pool of fast storage devices (probably SSDs) and use it as a cache for an existing slower and larger pool.

Use a replicated pool as a front-end to service most I/O, and destage cold data to a separate erasure coded pool that does not currently (and cannot efficiently) handle the workload.

We should be able to create and add a cache pool to an existing pool of data, and later remove it, without disrupting service or migrating data around.

USE CASES

READ-WRITE POOL, WRITEBACK

We have an existing data pool and put a fast cache pool “in front” of it. Writes will go to the cache pool and immediately ack. We flush them back to the data pool based on the defined policy.

READ-ONLY POOL, WEAK CONSISTENCY

We have an existing data pool and add one or more read-only cache pools. We copy data to the cache pool(s) on read. Writes are forwarded to the original data pool. Stale data is expired from the cache pools based on the defined policy.

This is likely only useful for specific applications with specific data access patterns. It may be a match for rgw, for example.

INTERFACE

Set up a read/write cache pool foo-hot for pool foo:

```
ceph osd tier add foo foo-hot
ceph osd tier cache-mode foo-hot writeback
```

Direct all traffic for foo to foo-hot:

```
ceph osd tier set-overlay foo foo-hot
```

Set the target size and enable the tiering agent for foo-hot:

```
ceph osd pool set foo-hot hit_set_type bloom
ceph osd pool set foo-hot hit_set_count 1
ceph osd pool set foo-hot hit_set_period 3600   # 1 hour
ceph osd pool set foo-hot target_max_bytes 1000000000000  # 1 TB
ceph osd pool set foo-hot min_read_recency_for_promote 1
ceph osd pool set foo-hot min_write_recency_for_promote 1
```

Drain the cache in preparation for turning it off:

```
ceph osd tier cache-mode foo-hot forward
rados -p foo-hot cache-flush-evict-all
```

When cache pool is finally empty, disable it:

```
ceph osd tier remove-overlay foo
ceph osd tier remove foo foo-hot
```

Read-only pools with lazy consistency:

```
ceph osd tier add foo foo-east
ceph osd tier cache-mode foo-east readonly
ceph osd tier add foo foo-west
ceph osd tier cache-mode foo-west readonly
```

TIERING AGENT

The tiering policy is defined as properties on the cache pool itself.

HITSET METADATA

First, the agent requires HitSet information to be tracked on the cache pool in order to determine which objects in the pool are being accessed. This is enabled with:

```
ceph osd pool set foo-hot hit_set_type bloom
ceph osd pool set foo-hot hit_set_count 1
ceph osd pool set foo-hot hit_set_period 3600   # 1 hour
```

The supported HitSet types include 'bloom' (a bloom filter, the default), 'explicit_hash', and 'explicit_object'. The latter two explicitly enumerate accessed objects and are less memory efficient. They are there primarily for debugging and to demonstrate pluggability for the infrastructure. For the bloom filter type, you can additionally define the false positive probability for the bloom filter (default is 0.05):

```
ceph osd pool set foo-hot hit_set_fpp 0.15
```

The hit_set_count and hit_set_period define how much time each HitSet should cover, and how many such HitSets to store. Binning accesses over time allows Ceph to independently determine whether an object was accessed at least once and whether it was accessed more than once over some time period ("age" vs "temperature").

The `min_read_recency_for_promote` defines how many HitSets to check for the existence of an object when handling a read operation. The checking result is used to decide whether to promote the object asynchronously. Its value should be between 0 and `hit_set_count`. If it's set to 0, the object is always promoted. If it's set to 1, the current HitSet is checked. And if this object is in the current HitSet, it's promoted. Otherwise not. For the other values, the exact number of archive HitSets are checked. The object is promoted if the object is found in any of the most recent `min_read_recency_for_promote` HitSets.

A similar parameter can be set for the write operation, which is `min_write_recency_for_promote`.

```
ceph osd pool set {cachepool} min_read_recency_for_promote 1
ceph osd pool set {cachepool} min_write_recency_for_promote 1
```

Note that the longer the `hit_set_period` and the higher the `min_read_recency_for_promote`/`min_write_recency_for_promote` the more RAM will be consumed by the ceph-osd process. In particular, when the agent is active to flush or evict cache objects, all hit_set_count HitSets are loaded into RAM.

CACHE MODE

The most important policy is the cache mode:

> ceph osd pool set foo-hot cache-mode writeback

The supported modes are 'none', 'writeback', 'forward', and 'readonly'. Most installations want 'writeback', which will write into the cache tier and only later flush updates back to the base tier. Similarly, any object that is read will be promoted into the cache tier.

The 'forward' mode is intended for when the cache is being disabled and needs to be drained. No new objects will be promoted or written to the cache pool unless they are already present. A background operation can then do something like:

```
rados -p foo-hot cache-try-flush-evict-all
rados -p foo-hot cache-flush-evict-all
```

to force all data to be flushed back to the base tier.

The ‘readonly’ mode is intended for read-only workloads that do not require consistency to be enforced by the storage system. Writes will be forwarded to the base tier, but objects that are read will get promoted to the cache. No attempt is made by Ceph to ensure that the contents of the cache tier(s) are consistent in the presence of object updates.

CACHE SIZING

The agent performs two basic functions: flushing (writing ‘dirty’ cache objects back to the base tier) and evicting (removing cold and clean objects from the cache).

The thresholds at which Ceph will flush or evict objects is specified relative to a ‘target size’ of the pool. For example:

```
ceph osd pool set foo-hot cache_target_dirty_ratio .4
ceph osd pool set foo-hot cache_target_dirty_high_ratio .6
ceph osd pool set foo-hot cache_target_full_ratio .8
```

will begin flushing dirty objects when 40% of the pool is dirty and begin evicting clean objects when we reach 80% of the target size.

The target size can be specified either in terms of objects or bytes:

```
ceph osd pool set foo-hot target_max_bytes 1000000000000  # 1 TB
ceph osd pool set foo-hot target_max_objects 1000000       # 1 million objects
```

Note that if both limits are specified, Ceph will begin flushing or evicting when either threshold is triggered.

OTHER TUNABLES

You can specify a minimum object age before a recently updated object is flushed to the base tier:

```
ceph osd pool set foo-hot cache_min_flush_age 600   # 10 minutes
```

You can specify the minimum age of an object before it will be evicted from the cache tier:

```
ceph osd pool set foo-hot cache_min_evict_age 1800   # 30 minutes
```