TRIM and Scrub commands for maintaining a ZFS storage in EVE

Motivation

When using physical SSDs in a ZFS configuration, write performance degrades over time when the storage system is used for a long time. This problem occurs due to the nature of the SSD. Depending on the type of SSD and how long that drive is loaded, the negative impact on write performance can be significant.
To solve this problem, ZFS supports the TRIM feature, which helps prevent performance degradation of SSDs.

There is also a need to periodically check the integrity of the data in EVE in the ZFS configuration using the Scrub function. This verification is necessary in order to detect and prevent errors before they cause hardware failure or data corruption in EVE, which is not acceptable to us.

The requirement to be able to maintain the storage system and periodically check it for errors comes from EVE customers and also is in line with the requirements for improving the storage system in EVE.
And this proposal is to add the ability to instantly launch or schedule these commands on the controller side. Since commands can negatively affect performance in the course of their work, it is proposed to add the ability for users to set the time for their launch themselves through the controller.

TRIM

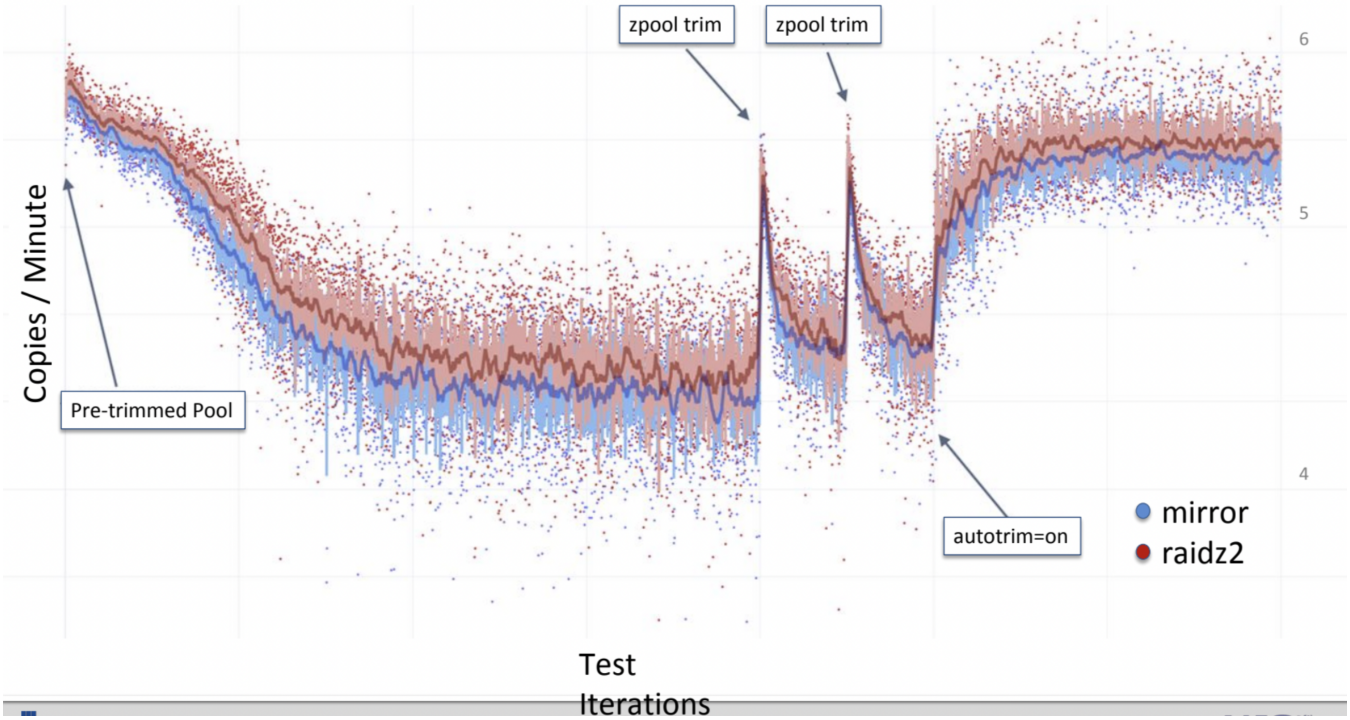
TRIM is a command which allows the filesystem to notify the storage device which blocks are no longer in use. an read more about what TRIM is on the wiki.

This functionality is already implemented in OpenZFS and all we need is to add the ability to control this through the controller.

The benefit for EVE that we get after implementing the ability to run this command from the controller is the following:

- Reduced write amplification (fewer writes)
- Higher write throughput (less read-erase-modify)
- Increased device longevity (finite erase-write cycles)

An example of the positive impact of the TRIM team was presented at the OpenZfs conference in 2019 in the presentation of this functionality:



SCRUB

ZFS checksums every block of data that is written to disk, and compares this checksum when the data is read back into memory. If the checksums don't match we know the data was changed by something other than ZFS (assuming a ZFS bug isn't the culprit), and assuming we are using ZFS to RAID protect the storage the issue will be automatically fixed for us.

But what if you have a lot of data on a disk that isn't read often? ZFS provides a scrub option to read back all of the data in the file system and validate that the data still matches the computed checksum. This feature in EVE can be accessed by running the zpool utility with the "scrub" option and the name of the pool to scrub.

In this proposal, it is proposed to add functionality that allows you to run this function from the controller, including according to the schedule.

Weak sides

The weak side is the fact that performance decreases when executing these commands since for their execution they, like everyone else, require resources.

A study was also conducted to determine the negative impact on performance when reading and writing at the time of executing these commands. And it turned out that both commands in the process of their execution show a decrease in performance on the device by at least half when reading from disk or zpool. And only the Scrub command has a negative impact on system performance when writing. The scrub operation consumes all I/O resources on the system.

There are levers in ZFS to limit these commands in system resources, but another big new study is needed to test them. After which it will be possible to think about improving this functionality in EVE.

EVE API Additions

The ***StorageCmdConfig*** message is expected to be received from the controller:

```
// StorageCmdType is the type of command for servicing the storage system
enum StorageCmdType {
        STORAGE_CMD_TYPE_UNSPECIFIED = 0;
        STORAGE_CMD_TYPE_TRIM = 1; // Command for Trim ops
        STORAGE_CMD_TYPE_SCRUB = 2; // Command for Scrub ops
}

// StorageCmdRunType - type for determining the frequency of commands running
// when servicing the storage system
enum StorageCmdRunType {
        STORAGE_CMD_RUN_TYPE_UNSPECIFIED = 0;
        STORAGE_CMD_RUN_TYPE_NOT_RUN = 1;
        STORAGE_CMD_RUN_TYPE_NOW = 2;
        STORAGE_CMD_RUN_TYPE_EVERY_HOUR = 3;
        STORAGE_CMD_RUN_TYPE_EVERY_3_HOURS = 4;
        STORAGE_CMD_RUN_TYPE_EVERY_6_HOURS = 5;
        STORAGE_CMD_RUN_TYPE_EVERY_12_HOURS = 6;
        STORAGE_CMD_RUN_TYPE_EVERY_DAY = 7;
        STORAGE_CMD_RUN_TYPE_EVERY_2_DAYS = 8;
        STORAGE_CMD_RUN_TYPE_EVERY_3_DAYS = 9;
        STORAGE_CMD_RUN_TYPE_EVERY_WEEK = 10;
        STORAGE_CMD_RUN_TYPE_EVERY_2_WEEKS = 11;
        STORAGE_CMD_RUN_TYPE_EVERY_MONTH = 12;
        STORAGE_CMD_RUN_TYPE_EVERY_3_MONTHS = 13;
        STORAGE_CMD_RUN_TYPE_EVERY_6_MONTHS = 14;
        STORAGE_CMD_RUN_TYPE_EVERY_YEAR = 15;
}

// StorageCmdConfig describes the desired command configuration for serving storage.
message StorageCmdConfig {
        string pool_name = 1;
        StorageCmdType cmd_type = 2;
        StorageCmdRunType run_type = 3;
}
```

From the EVE side, a message ***StorageInfo*** with information about the current state of the commands will be added to the message ***StorageServiceCmd*** with information about the state of the storage system:

```
// StorageServiceCmd - information about servicing storage systems (Trim/Scrub)
message StorageServiceCmd {
        org.lfedge.eve.config.StorageCmdType cmd_type = 1; // Scrub or Trim
        org.lfedge.eve.config.StorageCmdRunType run_type = 2; // The run instruction (time period)
        uint64 last_update_time = 3; // Time of the last scheduled start check
        uint64 next_run_time = 4; // The time when the command will be run next
        uint64 last_change_cmd_run_type_time = 5; // The time the command was last modified
        repeated uint64 launch_times_history_list = 6; // Stores history of previous runs
}
```

Reducing performance impact when running TRIM and Scrub

At the moment, there are not many options for these commands to regulate the resources used.
Only those options with default value that affect performance will be described here.

For Scrub:

`zfs_vdev_nia_delay``=5 (int)`

For non-interactive I/O (scrub, resilver, removal, initialize and rebuild), the number of concurrently-active I/O operations is limited to zfs_*_min_active, unless the vdev is "idle". When there are no interactive I/O operations active (synchronous or otherwise), and zfs_vdev_nia_delay operations have been completed since the last interactive operation, then the vdev is considered to be "idle", and the number of concurrently-active non-interactive operations are increased to zfs_*_max_active. See ZFS I/O SCHEDULER.

`zfs_vdev_nia_credit``=5 (int)`

Some HDDs tend to prioritize sequential I/O so strongly, that concurrent random I/O latency reaches several seconds. On some HDDs, this happens even if sequential I/O operations are submitted one at a time, and so setting zfs_*_max_active=1 does not help. To prevent non-interactive I/O, like scrub, from monopolizing the device, no more than zfs_vdev_nia_credit operations can be sent while there are outstanding incomplete interactive operations. This enforced wait ensures the HDD services the interactive I/O within a reasonable amount of time. See ZFS I/O SCHEDULER.

`zfs_vdev_scrub_max_active``=3 (int)`

Maximum scrub I/O operations active to each device.

`zfs_vdev_scrub_min_active``=1 (int)`

Minimum scrub I/O operations active to each device.

For TRIM:

`zfs_trim_queue_limit``=10 (uint)`

The maximum number of queued TRIMs outstanding per leaf vdev.

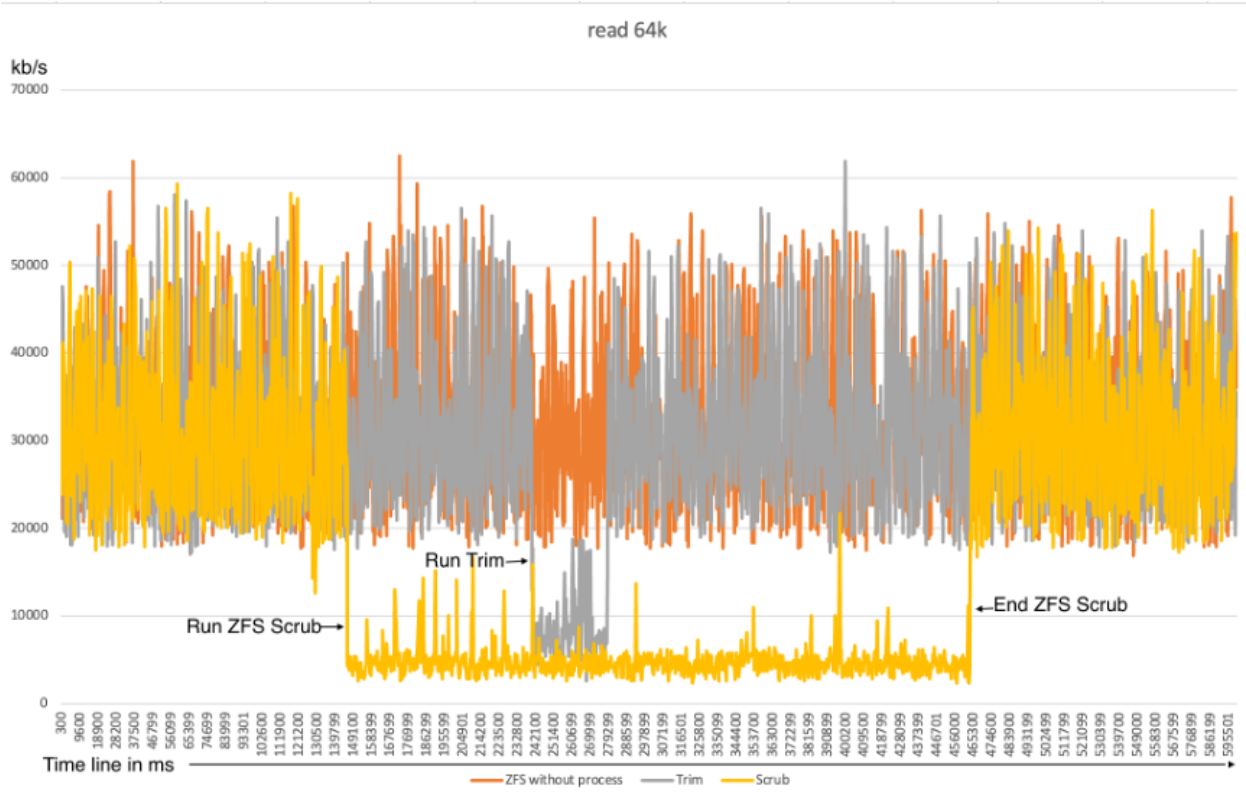
`zfs_vdev_trim_max_active``=2 (int)`

Maximum trim/discard I/O operations active to each device.

`zfs_vdev_trim_min_active``=1 (int)`

Minimum trim/discard I/O operations active to each device.

Investigating performance when changing options



Explanations for the chart:
This graph depicts 3 read performance tests under the same conditions and shows the impact of TRIM and Scrub commands on performance during these tests.

- The orange color indicates a test in which no commands were executed, i.e. clean test.
- The gray color shows the test in which the TRIM command was run, so we can observe short a performance drop when running this command.
- The yellow color shows a test in which the Scrub command was run, which resulted in performance degradation on most of the test.

Thus, this graph gives a visual understanding of the impact of the TRIM and Scrub commands on the performance of read operations.

TRIM command

The problem with this command is that a significant performance drop (more than 50%) occurs only for the read performance when it is executed. For this command, after testing and checking, it was determined that changing only one zfs_trim_queue_limit option to a value of 5 reduces the drawdown to 30-35%. Changing the rest of the trim_min_active and trim_max_active commands does not produce a positive effect, since they are already at their minimum values. The Trim command has almost no effect on write performance (Permissible performance degradation within 5 - 7%).

Recommended value for set in EVE:
`zfs_trim_queue_limit`=5

Scrub command

The scrub command has a strong impact on both read and write operations. Moreover, if the zpool is large, it can take a long time, which is a very critical factor for us (performance drop for a long time).

There are 2 ways to solve this:

1. Increase the limits, with the expectation that the sooner we finish, the sooner we will return to normal mode. This option can hang all I/O from applications, so this is not suitable for us.
2. Or, lower the available limits for this command as much as possible, and in this case, the process will be significantly delayed, but the performance will be eaten up by 60 - 70% less than on the default settings.

The average here is difficult to achieve since there are not many options that can affect this.

Recommended value for set in EVE:

`zfs_vdev_scrub_max_active`=1
`zfs_vdev_nia_credit=1`

Discussion

- Do we want to be able to pause these commands through the controller?
- The EVE API Additions topic suggested options for time periods to automatically run these commands on a schedule.
  If you have any objections or suggestions, let's discuss them in the comments.