Resuming ZFS send - oshogbo//vx

Resuming ZFS send

July 19, 2019, 2:42 p.m.

One of the amazing functionalities of ZFS is the possibility of sending a whole dataset from one place to another. This mechanism is amazing to create backups of your ZFS based machines. Although, there were some issues with this functionality for a long time when a user sent a big chunk of data. What if you would do that over the network and your connection has disappeared? What if your machine was rebooted as you are sending a snapshot?

For a very long time, you didn't have any options - you had to send a snapshot from the beginning. Now, this limitation was already bad enough. However, another downside of this approach was that all the data which you already send was thrown away. Therefore, ZFS had to go over all this data and remove them from the dataset. Imagine the terabytes of data which you sent via the network was thrown away because as you were sending the last few bytes, the network went off.

In this short post, I don't want to go over the whole ZFS snapshot infrastructure (if you think that such a post would be useful, please leave a comment). Now, to get back to the point, this infrastructure is used to clone the datasets. Some time ago a new feature called “Resuming ZFS send” was introduced. That means that if there was some problem with transmitting the dataset from one point to another you could resume it or throw them away. But the point is, that yes, you finally have a choice. Let's examine an example. Let's assume we have a dataset named '*ztank/test'* which have some data.

```
$ zfs list ztank/test
NAME          USED  AVAIL   REFER   MOUNTPOINT
ztank/test    7.21G   930G   7.21G   /tank/test
```

As you probably know we can use '*zfs send'* and '*zfs recv'* to send and recv dataset. We can send the whole dataset or snapshot. Sending snapshot it's much handier then dataset. First, when we send snapshot the dataset doesn't have to be unmounted. In the case when we send dataset it have to. Another advantage is that we can send data incrementally from one snapshot to another one, but it's not something that we want to discuss here. Let's send a dataset '*ztank/test'* and let's receive it in dataset '*ztank/newtest'*.

```
# zfs send ztank/test | zfs recv ztank/newtest
```

In the mine time, we can see that a new dataset was created. And the size of it is growing.

```
# zfs list ztank/newtest
NAME          USED  AVAIL  REFER  MOUNTPOINT
ztank/newtest   658M   929G   658M  /tank/newtest
```

The sending didn't finish. Let's, cancel it when we’re in the middle of sending it. What will happen?

```
$ zfs list ztank/newtest
cannot open 'ztank/newtest': dataset does not exist
```

The whole data sent (almost 1GB) was thrown away by ZFS, because we have to say ZFS to not throw away this data. This can be accomplished by passing the option 's' to the receiving side. So the default behavior of ZFS didn’t change. Let’s try one more time:

```
# zfs send ztank/test | zfs recv -s ztank/newtest
```

This time we also canceled it in the middle, but this time the file system was not destroyed.

```
$ zfs list ztank/newtest
NAME          USED  AVAIL  REFER  MOUNTPOINT
ztank/newtest  1.10G   929G  1.10G  /tank/newtest
```

When we will try to mount it, it will fail because the whole dataset was not sent (although the error communicate could be a little bit more helpful).

```
# zfs mount tank/newtest
cannot open 'tank/newtest': dataset does not exist
```

So, to resume sending the data we have to fetch a special token from the receiving site. The token is kept in the ZFS properties under '*recive_resume_token'* name.

```
$ zfs get -H -o value receive_resume_token ztank/newtest
1-d4f61d893-c0-
789c636064000310a500c4ec50360710e72765a5269740d80cd8e4d3d28a534b4032666e30793624f9a4ca92d462209dc19ee2864d7f497e7a
```

Now we can pass this token to the send command to inform it from which point he should resume the send. To do that we have to use the option 't'.zfs send -t '1-d4f61d893-c0-
789c636064000310a500c4ec50360710e72765a5269740d80cd8e4d3d28a534b4032666e30793624f9a4ca92d462209dc19ee2864d7f497e7a
| zfs recv -s ztank/newtest

Now if there will be some other interruption we just need to fetch a new token and resume sending again and again. We can do exactly the same over the network. If we would like to abort the send we can simply use command '*zfs recive'* with the '*-A'* options. This command will clean up evrything for us:

```
$ zfs get -H -o name,value receive_resume_token ztank/newtest
ztank/newtest1-bac48b79b-c0-
```

789c636064000310a500c4ec50360710e72765a5269740d80cd8e4d3d28a534b4032d3ca61f26c48f2499525a9c540fa85da93726cfa4bf2d34b
# zfs recv -A ztank/newtest
$ zfs get -H -o name,value receive_resume_token ztank/newtest
cannot open 'ztank/newtest': dataset does not exist

If the sending process was interrupted and we ask ZFS to not clean up the data, we can't send other data to the same dataset. Instead, we have to continue in the place interrupt occurred, or we have to abort it. Another option is to force send using '*-F*' options which rollback all the changes, which basically means we will rollback all the send.

The resumable options work only with the dataset, because if we would redirect blob from the zfs send the command to the file there is no way to follow and get a token.

The ZFS resumable send is very interesting, and straightforward functionality, which can help us save a lot of time and bandwidth.

⇐ Confusion with used/free disk space in ZFS Work Computer Google CTF Beginners Write Up ⇒