We have identified some tweaks to the process can both decrease the time required to perform the disaster recovery and minimize the duration where replication is disabled on the production system. The latter requires a strict condition that operations on files that have been subsequently deleted on the production system are not attempted on the DR-test appliance.

If during a DR-test a file on the production system is deleted, if replication is allowed to update the cloud bucket then the DR-test system would be broken for this file and attempts to read it will result in an error. For this reason, we recommend that replication be paused on the primary appliance for the entire duration for which the DR-test appliance will be used. If care is taken to not read such deleted files from the DR-test appliance, replication can be resumed after the first stage, downloading of metadata, completes.

The first stage of disaster recovery is the downloading of metadata objects. Each file written to the AVA results in two files in the cloud bucket, one containing the file's pointer map and the other its metadata (attributes, permissions, etc). The replication on the production system needs to be disabled for this stage so that the file namespace stored in the cloud remains static.

The number of the metadata files can be determined by running the following CLI command on the production system:

```
> enable
# show replication bucket
```

Example output:

```
listing entries from bucket ava400-ks
2927 entries in bucket ava400-ks
883.7MiB (926580933 bytes) in bucket ava400-ks:
   * 883.3MiB in 261 slabs
   * 189.2KiB in 1331 maps   <- pointer maps
   * 223.2KiB in 1331 rbtmds <- metadata files
```

The progress of stage one can be determined by monitoring a counter that tracks the number of GET requests issued to the cloud. The value of this counter won't necessarily be zero at the beginning of the DR-test so run the following hidden command before beginning the test to obtain the initial value. The command can then be run again at any time to calculate the number of metadata objects that have been downloaded.

```
> enable
# rfsctl exec backend.ops.get
```

To improve performance the AltaVault replication process uses multiple parallel connections and adjusts the number of threads up and down based on performance. The system that adjusts the threads does not work properly for the DR stage because the thread tuning module works based on observed throughput, and since metadata download is ops bound and not throughput bound, thread tuning doesn’t engage and the thread count will remain at the stating/minimum 16 threads. There is a CLI command to increase the number of threads used and testing showed 250% improved metadata download performance when using 128 threads, the maximum number of threads and with a network round trip time of 4ms.

Before starting the service, increase the number of replication threads with the following CLI command:

```
> enable
# config t
(config) # replication num-threads 128
```

Because the service is not initialized the command will result in an error "failed to connect, no process available". This error is benign and can be ignored. The number of threads will be changed and can be verified with the following hidden command:

```
> enable
# rfsctl exec replicator.num_threads
```

At the completion of the first stage the following message is logged:

> Sep 13 16:03:38 localhost rfsd[13852]: [namespace_restore.INFO] (13852) Cloud restore phase complete. Starting namespace generation phase

The second stage of disaster recovery is the population of the namespace using the downloaded pointer maps and metadata files. Here there is a difference between versions as 4.3 began using a database to hold the pointer maps and recreation of this database has been observed to take 12 hours or more. With the condition that deleted files on the production system will not be accessed on the DR-test appliance, replication on the production system does not have to be disabled for this stage.

To reiterate, a potential race condition can arise if a file that has subsequently been deleted on the production system is tried to be read on the DR-test system. If any of the data referenced by this file is deleted from the cloud then the DR-test system will raise a cloud bucket inconsistent alarm. This issue can be avoided by performing front-end file operations on the DR-test system only after first confirming the file(s) exist on the production system.

At the completion of the second stage the following messages are logged:

> Sep 13 16:03:38 localhost rfsd[13852]: [namespace_restore.INFO] (13852) Namespace generation phase complete. Successfully restored user namespace
> Sep 13 16:03:38 localhost rfsd[13852]: [megamount.INFO] (13852) restore namespace successful!