OpenDedup OST Connector 2.1.0

The OpenDedup OST connector provides integration between NetBackup and OpenDedup. It supports the following OST features:

- Write/Backup to OpenDedup volumes
- Read/Restore from OpenDedup volumes
- Accelerator support from backups to OpenDedup Volumes

OpenDedup provides an open source filesystem, SDFS, that includes inline deduplication to local or cloud storage targets. Backup specific features of OpenDedup are as follows:

- **In-Line deduplication to a Cloud Storage Backend** - SDFS can send all of its data to AWS, AZURE, Google, or any S3 compliant backend.
- **Performance** - Compressed Multi-Threaded upload and download of data to the cloud.
- **Local Cache** - SDFS will cache the most recently accessed data locally. This is configurable but set to 10 GB by default
- **Security** - All data can be encrypted using AES-CBC 256 when sent to the cloud
- **Throttling** - Upload and download speeds can be throttled.
- **Cloud Recovery/Replication** - All local metadata is replicated to the cloud and can be recovered back to local volume.
- **Glacier Support** - Supports S3 Lifecycle policies and retrieving data from Glacier
- **AWS Region Support** - Supports all AWS Regions

Requirements :

CentOS/RHEL 6.7+ - This is been tested on CentOS 7.0 and CentOS 6.7..
NetBackup 7.6.1+ - This has been tested and developed for NBU 7.7 and has been tested back to 7.6.1.
CPU : 2+ CPU System
Memory : 1GB of RAM + 256MB of RAM per TB of unique data stored
Disk :

- 2GB of additional local storage per TB of unique data.
- 2GB of additional storage per TB of logical storage that is used for filesystem metadata
- 10GB for local cache of cloud data (Configurable)

Quick Start Instructions

Setting up the OST Connector on a Netbackup Media/Master Server:

On a NetBackup media or media/master server perform the following steps:

1. Download and install the ost package and sdfs:
   On A Standard Media Server Run:
   wget http://www.opendedup.org/downloads/ost-2.1.tar.gz
   tar -xzvf ost-2.0.tar.gz
   cd dist
   ./media-install.sh
   /etc/init.d/netbackup stop
   /etc/init.d/netbackup start

   For the Appliance Run:
   NOTE: The opendedupe dedupe engine cannot run on the appliance only the OST connector. To use with an appliance run opendedupe on a separate server

   Step 1 - Install opendedupe on a Separate Server:
   1. Follow the Installation (not initialization) instructions at http://opendedup.org/odd/linux-quickstart-guide/

   Step 2 - Install the OST Connector on the appliance:
   1. Download http://www.opendedup.org/downloads/ost_appliance2.0_OST_redhat_64.tar.gz
   2. Install the OpenDedupe OST Appliance package by following the OST plugin installation instructions in the Appliance documentation:
   3. https://www.veritas.com/content/support/en_US/58991/appliance-docs/appliance-docs-30.html
   4. Create an Appliance user, if one does not exist.
   5. Login to the Appliance using the user credentials created in the previous step.
   6. Modify the /usr/openv/ostconf/OpenDedupe/ostconfig.xml file

2. Create an sdfs volume

   Local Storage
   sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=256GB --backup-volume --sdfscli-disable-ssl
   For an appliance setup run this on a separate opendedupe server
   sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=256GB --backup-volume --sdfscli-listen-addr 0.0.0.0 --sdfscli-listen-port 6442 --sdfscli-require-auth --sdfscli-disable-ssl

AWS Storage

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --aws-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --backup-volume --sdfscli-disable-ssl

For an appliance setup run this on a separate opendedupe server

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --aws-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --backup-volume --sdfscli-listen-addr 0.0.0.0 --sdfscli-listen-port 6442 --sdfscli-require-auth --sdfscli-disable-ssl

ECS Storage

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --aws-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --cloud-url <url> --backup-volume --sdfscli-disable-ssl

For an appliance setup run this on a separate opendedupe server

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --aws-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --simple-s3 --cloud-url <url> --backup-volume --sdfscli-listen-addr 0.0.0.0 --sdfscli-listen-port 6442 --sdfscli-require-auth --sdfscli-disable-ssl

Azure Storage

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --azure-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --sdfscli-disable-ssl

For an appliance setup run this on a separate opendedupe server

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --azure-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --backup-volume --sdfscli-listen-addr 0.0.0.0 --sdfscli-listen-port 6442 --sdfscli-require-auth --sdfscli-disable-ssl

Google Storage

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --google-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --backup-volume --sdfscli-disable-ssl

For an appliance setup run this on a separate opendedupe server

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --google-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --backup-volume --sdfscli-listen-addr 0.0.0.0 --sdfscli-listen-port 6442 --sdfscli-require-auth --sdfscli-disable-ssl

3. Mount the SDFS Volume under /opendedupe/volumes/

   mkdir /opendedupe/volumes/pool0
   mount -t sdfs pool0 /opendedupe/volumes/pool0

   optional add volume to fstab by using the following line is /etc/fstab

pool0 /opendedupe/volumes/pool0 sdfs defaults 0 0

4. Add the sdfs volume to /etc/fstab so it can be mounted at startup

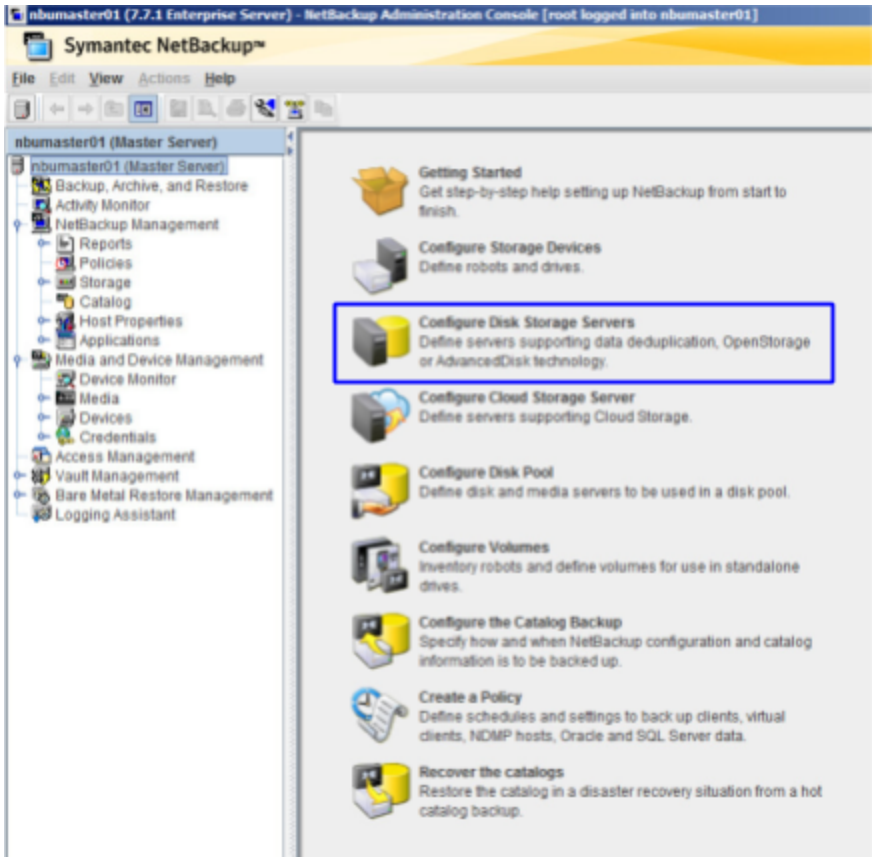
   Add the following line to the fstab as referenced by the pool0 example

   pool0 /opendedup/volumes/pool0 sdfs defaults 0 0

5. Edit /etc/sdfs/ostconfig.xml if needed.

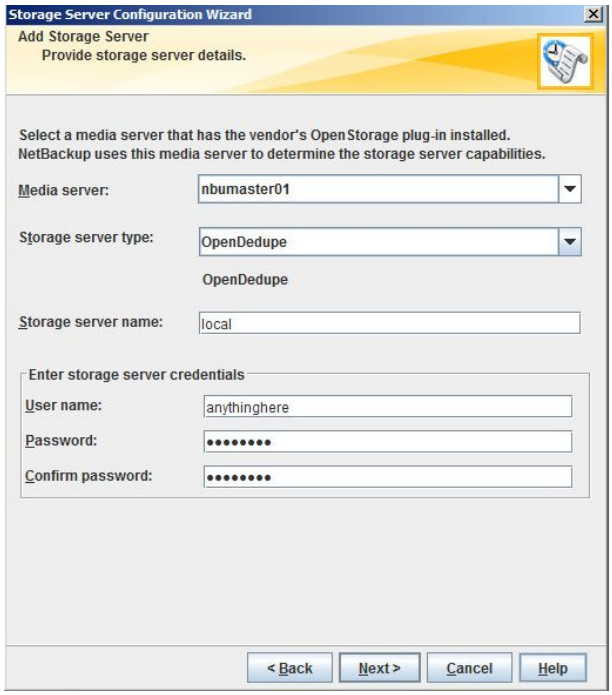
Create a OST Disk Pool and STU In the NetBackup Console.

1. Login to the Netbackup Master from the Java Console
2. Select “Configure Disk Storage Servers” from the Wizard Page.

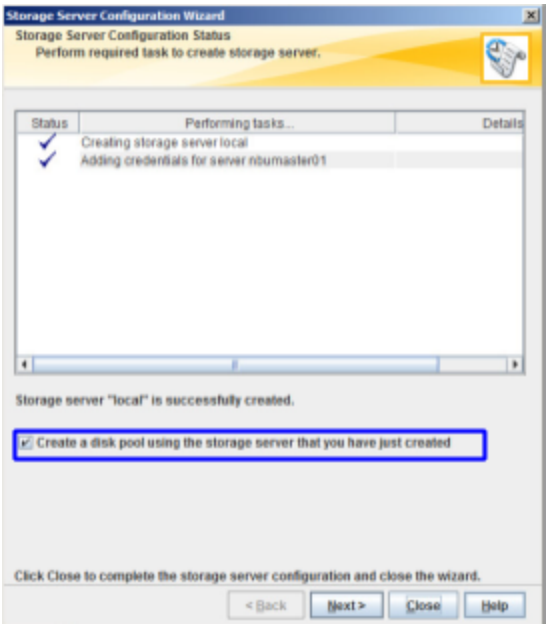


2. Select the “OpenStorage” option from the “Select the type of disk storage that you want to configure.”

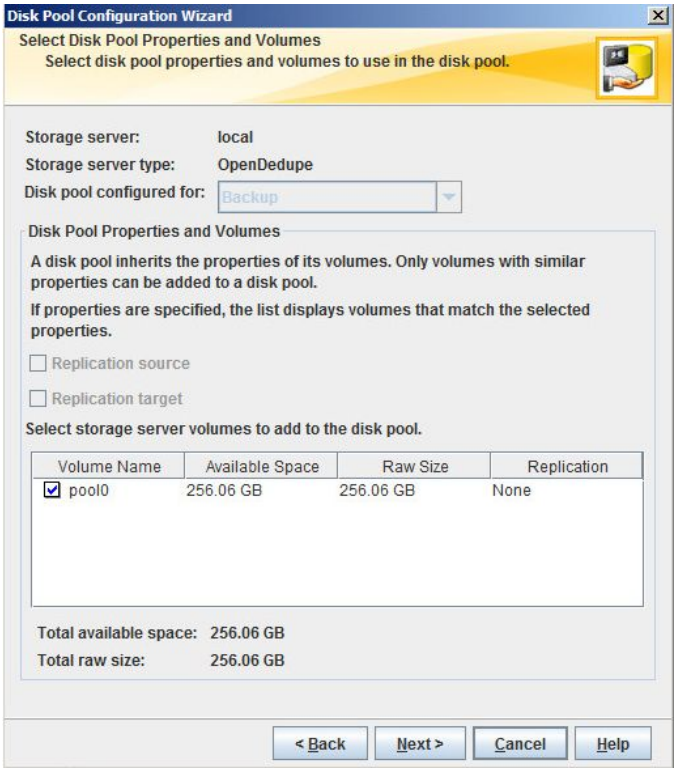
3. Add the following options to the Storage Server Details
    a. Storage server type : OpenDedupe
    b. Storage Server name : the name in the <NAME></NAME> tag in /etc/sdfs/ostconfig.xml. This is “local” by default.
    c. Username : anything can go in here. It is not used
    d. Password/Confirm Password : Anything can go in here as well



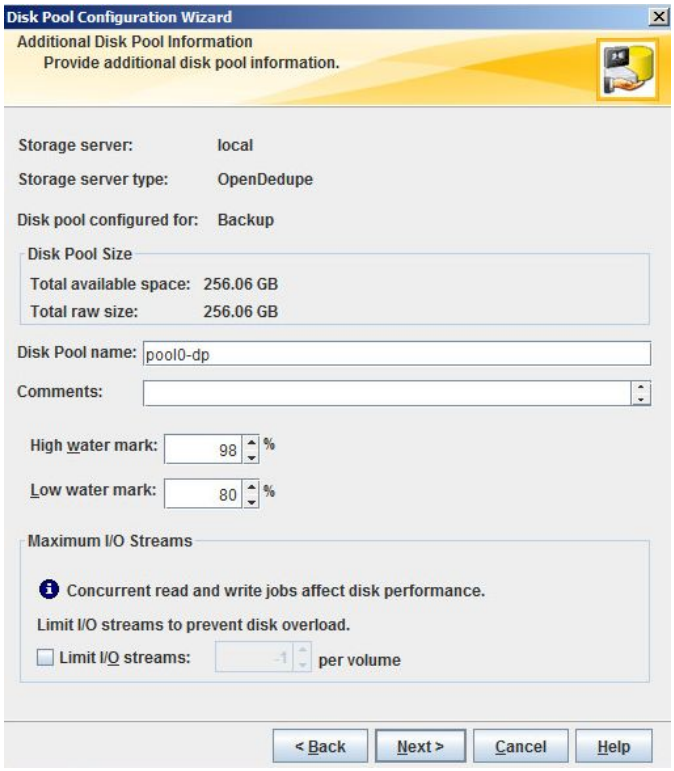
4. Finish the storage configuration wizard and make sure **“Create a disk pool using the storage server that you just created”** is selected



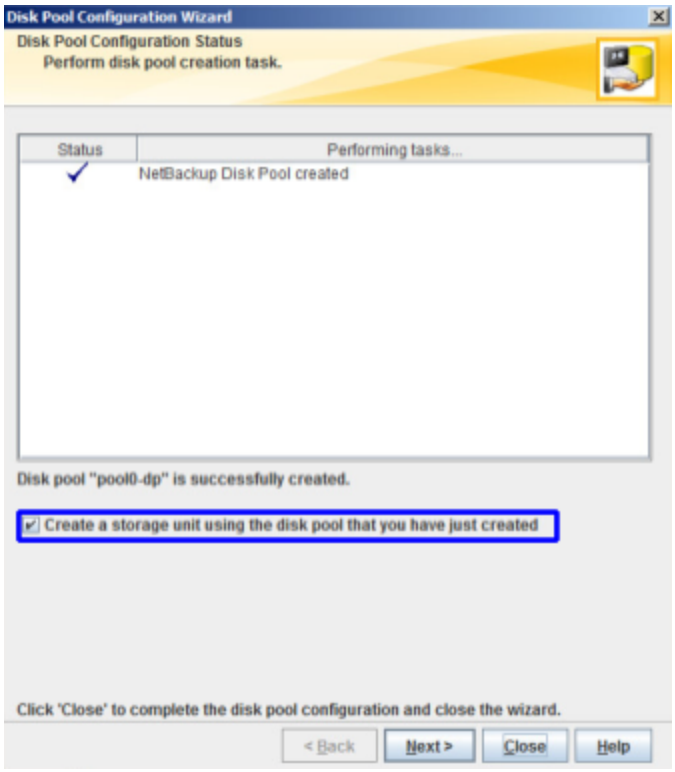
5. Select the storage pool that was just created.



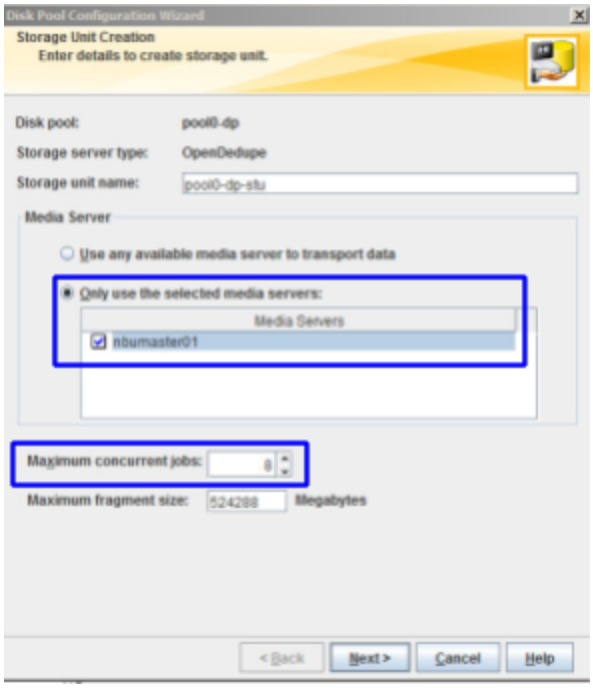
6. Add a disk pool name



7. Finish the Wizard and Select “Create a storage unit using the disk pool that you just created”.



8. In the Storage Unit Creation page select “**Only use the selected media servers**” and select the media server that the storage was created on. For **maximum concurrent jobs** select “8”.



Setting up multiple media servers in the same domain

To setup the OST connector on multiple media servers in the same domain additional steps must be taken on each media server before adding the storage pools in NetBackup.

Step 1: Follow “Setting up the OST Connector” instructions outlined in the document on each media server that will use the OST Connector.

Step 2: Edit /etc/sdfs/ostconfig.xml and change the <name> tag to something unique in the NetBackup domain such as the host name with an incremented number **e.g.** ***<NAME>hostname-0</NAME>***

Step 3: Follow “Create a OST Disk Pool and STU In the NetBackup Console“ instructions and use the name in the <NAME> tag as the Storage Server name designated in Step 3 of the “Setting up the OST Connector” section.

Setting up multiple SDFS volumes on a media servers

The OST connector supports multiple SDFS volumes on the same media server but additional steps are required to support this configuration.

Step 1: Follow “Setting up the OST Connector” instructions outlined in the document on each media server that will use the OST Connector.

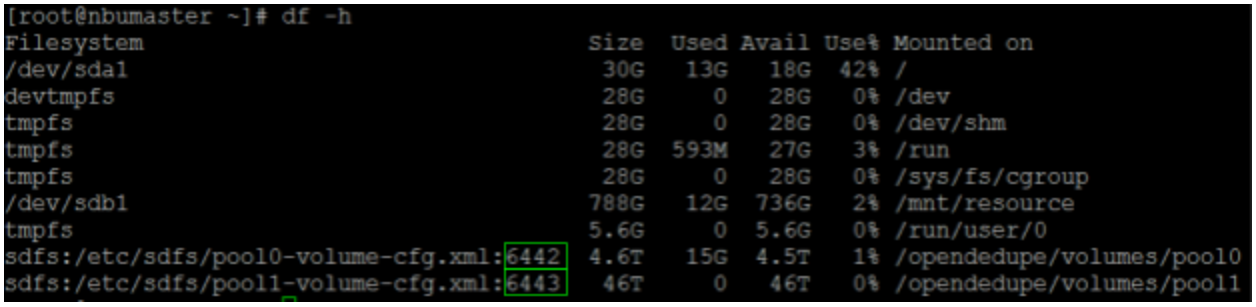
Step 2: Run the mkfs.sdfs command for each additional SDFS volume. E.g.

```
sudo mkfs.sdfs --volume-name=pool1 --volume-capacity=1TB --aws-enabled true --cloud-access-key
<access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name>
```

Step 3: Create a mount point for each additional volume under /opendedupe/volumes/ . e.g.

```
mkdir /opendedupe/volumes/pool1
mount -t sdfs pool1 /opendedupe/volumes/pool1
```

Step 4: Mount the new volume and get the control port number of the additional volume. The port number will be appended to the filesystem column when running df -h. In the example below pool0 has a tcp control port of 6442 and pool1 has a control port of 6443.



```
[root@nbumaster ~]# df -h
Filesystem                                 Size  Used Avail Use% Mounted on
/dev/sda1                                   30G   13G   18G  42% /
devtmpfs                                    28G     0   28G   0% /dev
tmpfs                                       28G     0   28G   0% /dev/shm
tmpfs                                       28G  593M   27G   3% /run
tmpfs                                       28G     0   28G   0% /sys/fs/cgroup
/dev/sdb1                                  788G   12G  736G   2% /mnt/resource
tmpfs                                      5.6G     0  5.6G   0% /run/user/0
sdfs:/etc/sdfs/pool0-volume-cfg.xml:6442   4.6T   15G  4.5T   1% /opendedupe/volumes/pool0
sdfs:/etc/sdfs/pool1-volume-cfg.xml:6443    46T     0   46T   0% /opendedupe/volumes/pool1
```

Step 5: Edit the /etc/sdfs/ostconfig.xml and add a new <CONNECTION> tag inside of the <CONNECTIONS> tag for the new volume. In the new <CONNECTION> tag add the port identified in Step 4 to the <URL> tag (https://localhost:6443/) add a name that is unique the <NAME> tag and specify the new volume name in the <LSU_NAME> tag (pool1). Below is a complete example of a ostconfig.xml with two volumes.

```
<!-- This is the config file for the OST connector for opendedup and Netbackup -->
<CONNECTIONS>
<CONNECTION>
<!--NAME is the local server name that you will reference within Netbackup -->
<NAME>
local
</NAME>
<LSU_NAME>
pool0
</LSU_NAME>
<URL>
https://localhost:6442/
</URL>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>passwd</PASSWD>
<!--
```

```
<SERVER_SHARE_PATH>
A_SUBDIRECTORY_UNDER_THE_MOUNT_PATH
</SERVER_SHARE_PATH>
-->
</CONNECTION>
<!-- Below is the new volume-->
<CONNECTION>
<!--NAME is the local server name that you will reference within Netbackup -->
<NAME>
hostname0
</NAME>
<LSU_NAME>
pool1
</LSU_NAME>
<URL>
https://localhost:6443/
</URL>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>passwd</PASSWD>
<!--
<SERVER_SHARE_PATH>
A_SUBDIRECTORY_UNDER_THE_MOUNT_PATH
</SERVER_SHARE_PATH>
-->
</CONNECTION>
</CONNECTIONS>
```

Replicaiton (AIR) Setup

SDFS 2.0 can perform auto image replication. With OpenDedupe there are 2 types of auto image replication:

- Zero Data Movement - With Zero Data Movement, no data acutally gets replicated but rather the target downloads the metadata associated with the source image from the source bucket and creates a virtual copy of that metadata. For this process to work both the source and target opendedupe volumes must share the same backend object storage bucket. The opendedupe ostconfig.xml refers to this as ***direct*** replication.
- Traditional Optimized Replication - This is optimized replication as performed by any other deduplication technology. It does replication of unique data between two independent sdfs volumes connected to different buckets by sending only new unique to the target sdfs domain. The opendedupe ostconfig.xml refers to this as ***indirect*** replication.

Regardless of the auto image replication type requires a few changes to sdfs configuration to allow communication over the control port on the source side and improved performance.

Things to check before you get started.

- Make sure you can ping the source by host name from the target
- Make sure you can ping the target by host name from the source
- Make sure that all firewalls allow communication from the source to the destination on port 6442
- Make sure that all firewalls allow communication from the destination to the source on port 6442

Creating a sdfs volume on the source

Step 1 - Create the volume :

Below is an example of creating a volume on the source that can be used with an AWS Bucket. Please note the section in bold :

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --aws-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name <unique bucket name> --backup-volume **--sdfscli-listen-addr 0.0.0.0 --sdfscli-listen-port 6442 --sdfscli-require-auth --sdfscli-disable-ssl**

Step 2 - Create the mountpoint

mkdir -p /opendedupe/volumes/pool0

Step 3 - Mount the volume on the source

sudo mount -t sdfs pool0 /opendedupe/volumes/pool0

Step 4 - Check the volume is mounted

sdfscli --volume-info --password password --nossl

Creating a sdfs volume on the destination for Zero Data Movement Air

Step 1 - Create the volume :

Below is an example of creating a volume on the source that can be used with an AWS Bucket. Please note the section in bold. On the destination the bucket ne :

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --aws-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name **<same bucket name as source>** --backup-volume **--sdfscli-listen-addr 0.0.0.0 --sdfscli-listen-port 6442 --sdfscli-require-auth --sdfscli-disable-ssl**

Step 2 - Create the mountpoint

mkdir -p /opendedupe/volumes/pool0

Step 3 - Mount the volume on the source
sudo mount -t sdfs pool0 /opendedupe/volumes/pool0

Step 4 - Check the volume is mounted
sdfscli --volume-info --password password --nossl

Creating a sdfs volume on the destination for Traditional Air

Step 1 - Create the volume :
Below is an example of creating a volume on the source that can be used with an AWS Bucket. Please note the section in bold. On the destination the bucket ne :

sudo mkfs.sdfs --volume-name=pool0 --volume-capacity=1TB --aws-enabled true --cloud-access-key <access-key> --cloud-secret-key <secret-key> --cloud-bucket-name **<same bucket name as source>** --backup-volume
--sdfscli-listen-addr 0.0.0.0 --sdfscli-listen-port 6442 --sdfscli-require-auth --sdfscli-disable-ssl

Step 2 - Create the mountpoint
mkdir -p /opendedupe/volumes/pool0

Step 3 - Mount the volume on the source
sudo mount -t sdfs pool0 /opendedupe/volumes/pool0

Step 4 - Check the volume is mounted
sdfscli --volume-info --password password --nossl

Zero Data Movement - Edit the ostconfix.xml on the Source and destination to match as the example below shows

```
<!-- This is the config file for the OST connector for opendedup and Netbackup -->
<CONNECTIONS>
<CONNECTION>
<!--NAME is the local server name that you will reference within Netbackup -->
<NAME>
sourceserver
</NAME>
<!--LSU_NAME is the name of the volume to be mounted and maps to the path under
/opendedup/volumes/ where the volume will be mounted. As an example if the volume
name is pool0 the LSU_NAME would be pool0 and the volume would be mounted at
/opendedup/volumes/pool0
<LSU_NAME>
pool0
```

</LSU_NAME>
<!--URL - this is the url that the plugin will use to communicate with sdfs volume.
If the volume is local and the only one url should be set to https://localhost:6442.
Otherwise do a df -h on the host containing the volume the mount point for the volume
will contain the port used to connect to the volume. Use this port plus the host name
as the url. https://<server-name>:<volume-tcp-port>/-->
<URL>
http://sourceserver-hostname:6442/
</URL>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>admin</PASSWD>
<!--SERVER_SHARE_PATH - This is the subdirectory under the mount path that corresponds
to the LSU_NAME. This is required if you are mounting the volume remotely via nfs and
exporting a subdirectory or are using a subdirectory under a local sdfs mount as the volume path.
An NFS example is if the sdfs volume is mounted on the remote server to /media/pool0 and the folder
"nfs" is being exported under this mount (/media/pool0/nfs). The export would be mounted locally
to /opendedupe/volumes/pool0 and the SERVER_SHARE_PATH would be set to "nfs".
A local example would be that you mount an sdfs volume to /opendedupe/volumes/ and create a
subdirectory, where the backups will be stored, under the mount called "pool0". In this example,
the SERVER_SHARE_PATH would be set to the subdirectory name of "pool0".
-->
<!--
<SERVER_SHARE_PATH>
A_SUBDIRECTORY_UNDER_THE_MOUNT_PATH
</SERVER_SHARE_PATH>
-->
</CONNECTION>
<CONNECTION>
<!--NAME is the local server name that you will reference within Netbackup -->
<NAME>
targetserver
</NAME>
<!--LSU_NAME is the name of the volume to be mounted and maps to the path under
/opendedup/volumes/ where the volume will be mounted. As an example if the volume
name is pool0 the LSU_NAME would be pool0 and the volume would be mounted at
/opendedup/volumes/pool0
<LSU_NAME>
pool0
</LSU_NAME>
<!--URL - this is the url that the plugin will use to communicate with sdfs volume.
If the volume is local and the only one url should be set to https://localhost:6442.
Otherwise do a df -h on the host containing the volume the mount point for the volume
will contain the port used to connect to the volume. Use this port plus the host name
as the url. https://<server-name>:<volume-tcp-port>/-->
<URL>
http://targetserver-hostname:6442/
</URL>
<!-- for zero data movement the DIRECT xml tag should direct for tradition air comment out the section DIRECT section
below -->
<DIRECT>direct</DIRECT>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>admin</PASSWD>
<!--SERVER_SHARE_PATH - This is the subdirectory under the mount path that corresponds
to the LSU_NAME. This is required if you are mounting the volume remotely via nfs and
exporting a subdirectory or are using a subdirectory under a local sdfs mount as the volume path.
An NFS example is if the sdfs volume is mounted on the remote server to /media/pool0 and the folder
"nfs" is being exported under this mount (/media/pool0/nfs). The export would be mounted locally

to /opendedupe/volumes/pool0 and the SERVER_SHARE_PATH would be set to "nfs".
A local example would be that you mount an sdfs volume to /opendedupe/volumes/ and create a
subdirectory, where the backups will be stored, under the mount called "pool0". In this example,
the SERVER_SHARE_PATH would be set to the subdirectory name of "pool0".
-->
<!--
<SERVER_SHARE_PATH>
A_SUBDIRECTORY_UNDER_THE_MOUNT_PATH
</SERVER_SHARE_PATH>
-->
</CONNECTION>
</CONNECTIONS>
<!-- for zero data movement sele
<REPLICATIONS>
        <REPLICATION target-server="targetserver" target-lsu="pool0" source-server="sourceserver"
source-lsu="pool0"/>
</REPLICATIONS>

Traditional AIR - Edit the ostconfix.xml on the Source and destination to match as the example below shows

<!-- This is the config file for the OST connector for opendedup and Netbackup -->
<CONNECTIONS>
<CONNECTION>
<!--NAME is the local server name that you will reference within Netbackup -->
<NAME>
sourceserver
</NAME>
<!--LSU_NAME is the name of the volume to be mounted and maps to the path under
/opendedup/volumes/ where the volume will be mounted. As an example if the volume
name is pool0 the LSU_NAME would be pool0 and the volume would be mounted at
/opendedup/volumes/pool0
<LSU_NAME>
pool0
</LSU_NAME>
<!--URL - this is the url that the plugin will use to communicate with sdfs volume.
If the volume is local and the only one url should be set to https://localhost:6442.
Otherwise do a df -h on the host containing the volume the mount point for the volume
will contain the port used to connect to the volume. Use this port plus the host name
as the url. https://<server-name>:<volume-tcp-port>/-->
<URL>
http://sourceserver-hostname:6442/
</URL>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>admin</PASSWD>
<!--SERVER_SHARE_PATH - This is the subdirectory under the mount path that corresponds
to the LSU_NAME. This is required if you are mounting the volume remotely via nfs and
exporting a subdirectory or are using a subdirectory under a local sdfs mount as the volume path.
An NFS example is if the sdfs volume is mounted on the remote server to /media/pool0 and the folder
"nfs" is being exported under this mount (/media/pool0/nfs). The export would be mounted locally
to /opendedupe/volumes/pool0 and the SERVER_SHARE_PATH would be set to "nfs".
A local example would be that you mount an sdfs volume to /opendedupe/volumes/ and create a

```
subdirectory, where the backups will be stored, under the mount called "pool0". In this example,
the SERVER_SHARE_PATH would be set to the subdirectory name of "pool0".
-->
<!--
<SERVER_SHARE_PATH>
A_SUBDIRECTORY_UNDER_THE_MOUNT_PATH
</SERVER_SHARE_PATH>
-->
</CONNECTION>
<CONNECTION>
<!--NAME is the local server name that you will reference within Netbackup -->
<NAME>
targetserver
</NAME>
<!--LSU_NAME is the name of the volume to be mounted and maps to the path under
/opendedup/volumes/ where the volume will be mounted. As an example if the volume
name is pool0 the LSU_NAME would be pool0 and the volume would be mounted at
/opendedup/volumes/pool0
<LSU_NAME>
pool0
</LSU_NAME>
<!--URL - this is the url that the plugin will use to communicate with sdfs volume.
If the volume is local and the only one url should be set to https://localhost:6442.
Otherwise do a df -h on the host containing the volume the mount point for the volume
will contain the port used to connect to the volume. Use this port plus the host name
as the url. https://<server-name>:<volume-tcp-port>/-->
<URL>
http://targetserver-hostname:6442/
</URL>
<!-- for zero data movement the DIRECT xml tag should direct for tradition air comment out the section DIRECT section
below -->
<DIRECT>indirect</DIRECT>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>admin</PASSWD>
<!--SERVER_SHARE_PATH - This is the subdirectory under the mount path that corresponds
to the LSU_NAME. This is required if you are mounting the volume remotely via nfs and
exporting a subdirectory or are using a subdirectory under a local sdfs mount as the volume path.
An NFS example is if the sdfs volume is mounted on the remote server to /media/pool0 and the folder
"nfs" is being exported under this mount (/media/pool0/nfs). The export would be mounted locally
to /opendedupe/volumes/pool0 and the SERVER_SHARE_PATH would be set to "nfs".
A local example would be that you mount an sdfs volume to /opendedupe/volumes/ and create a
subdirectory, where the backups will be stored, under the mount called "pool0". In this example,
the SERVER_SHARE_PATH would be set to the subdirectory name of "pool0".
-->
<!--
<SERVER_SHARE_PATH>
A_SUBDIRECTORY_UNDER_THE_MOUNT_PATH
</SERVER_SHARE_PATH>
-->
</CONNECTION>
</CONNECTIONS>
<!-- for zero data movement sele
<REPLICATIONS>
        <REPLICATION target-server="targetserver" target-lsu="pool0" source-server="sourceserver"
source-lsu="pool0"/>
</REPLICATIONS>
```

Detailed OST XML Configuration setup

The OST driver is configured from the XML file located at /etc/sdfs/ostconfig.xml. It contains all of the configuration parameters required to allow SDFS to communicate with netbackup over OST.

Below is a typical ostconfig.xml :

```
<!-- This is the config file for the OST connector for opendedup and Netbackup -->
<CONNECTIONS>
<CONNECTION>
<!--NAME is the local server name that you will reference within Netbackup -->
<NAME>
local
</NAME>
<!--LSU_NAME is the name of the volume to be mounted and maps to the path under
/opendedup/volumes/ where the volume will be mounted. As an example if the volume
name is pool0 the LSU_NAME would be pool0 and the volume would be mounted at
/opendedup/volumes/pool0
<LSU_NAME>
pool0
</LSU_NAME>
<!--URL - this is the url that the plugin will use to communicate with sdfs volume.
If the volume is local and the only one url should be set to https://localhost:6442.
Otherwise do a df -h on the host containing the volume the mount point for the volume
will contain the port used to connect to the volume. Use this port plus the host name
as the url. https://<server-name>:<volume-tcp-port>/-->
<URL>
https://localhost:6442/
</URL>
<!--PASSWD - The password of the volume if one is required for this sdfs volume -->
<PASSWD>passwd</PASSWD>
<!--SERVER_SHARE_PATH - This is the subdirectory under the mount path that corresponds
to the LSU_NAME. This is required if you are mounting the volume remotely via nfs and
exporting a subdirectory or are using a subdirectory under a local sdfs mount as the volume path.
An NFS example is if the sdfs volume is mounted on the remote server to /media/pool0 and the folder
"nfs" is being exported under this mount (/media/pool0/nfs). The export would be mounted locally
to /opendedupe/volumes/pool0 and the SERVER_SHARE_PATH would be set to "nfs".
A local example would be that you mount an sdfs volume to /opendedupe/volumes/ and create a
subdirectory, where the backups will be stored, under the mount called "pool0". In this example,
the SERVER_SHARE_PATH would be set to the subdirectory name of "pool0".
-->
<!--
<SERVER_SHARE_PATH>
A_SUBDIRECTORY_UNDER_THE_MOUNT_PATH
```

```
</SERVER_SHARE_PATH>
-->
</CONNECTION>
</CONNECTIONS>
```

Scaling and Sizing

When running OpenDedupe at scale disk, cpu, and memory requirements need to be considered to size appropriately.

Data Stored on Disk

Cloud Volumes - SDFS Stores file metadata, a local hashtable, and a cache of unique blocks on local disk.

Local Volumes - SDFS Stores file metadata, a local hashtable, and all unique blocks on local disk.

Data Types:

File MetaData - Information about files and folders stored on opendedupe volumes. This data is also stored in the cloud for DR purposes when using cloud storage. File MetaData represents .21% of the non deduplicated size of the file stored.

HashTable - The hashtable is the lookup table that is used to identify whether incoming data is unique. The hashtable is stored on local disk and in the cloud for object storage backed instances. For local instances the hashtable is stored on local disk only. The hashtable is .4% of the unique storage size.

Local Cache - For Object storage backed volumes, active data is cached locally. The local cache stores compressed, deduplicate blocks only. This local cache size is set to 10GB by default but can be set to any capacity required with a minimum of 1GB. The local cache helps with restore performance and accelerated backup performance.

Local Unique Data Store - OpenDedupe stores all unique blocks locally for volumes not backed by object storage. For Object storage backed volumes this is not used. Local storage size will depend on the data being backed up and retention but typically represents 100% of the front end data for a 60 Day retention. OpenDedupe uses a

similar variable block deduplication method to a DataDomain so it will be inline with its sizing requirements.

Storage Performance:

Minimum local disk storage performance:

- 2000 random read IOPS
- 2400 random write IOPS
- 180 MB/s of streaming reads
- 120 MB/s of streaming writes

Supported Filesystems:

- VXFS
- XFS
- EXT4

Storage Requirements:

The following percentages should be used to calculate local storage requirements for Object Backed dedupe Volumes:

- MetaData: **.**21% of Non-Deduped Data Stored
- Local Cache: 10GB by default
- HashTable: .2% of Deduped Data

An example for 100TB of deduped data with an 8:1 dedupe rate would be as follows:

- Logical Data Stored on Disk = 8x100TB = 800TB
- Local Cache = 10GB
- Unique Data Stored in the Object Store 100TB
- MetaData
    - .21%Logical Data Stored on Disk=MetaData Size
    - .0021x800TB=1.68TB
- HashTable
    - .2% * Unique Storage
    - .002* 100TB = 400GB
- Total Volume Storage Requirements
    - Local Cache + MetaData + Hashtable
    - 10GB + 1.68TB + 400GB = 2.09TB

The following percentages should be used to calculate local storage requirements for local dedupe Volumes:

- MetaData: 2.1% of Non-Deduped Data Stored
- Local Cache: 10GB by default
- HashTable: .2% of Deduped Data
- Unique Data

An example for 100TB of deduped data with an 8:1 dedupe rate would be as follows:

- Logical Data Stored on Disk = 8x100TB = 800TB
- Unique Data Stored on disk 100TB
- MetaData
    - 21%Logical Data Stored on Disk=MetaData Size
    - .0021x800TB=1.68TB
- HashTable
    - .2% * Unique Storage
    - .002* 100TB = 400GB
- Total Volume Storage Requirements
    - Unique + MetaData + Hashtable +
    - 100TB + 1.68TB + 400GB = 102.08TB

Memory Sizing :

Memory for OpenDedupe is primarily used for internal simplified lookup tables (bloom filter) that indicate, with some likelihood that a hash is already stored or not. These data structures take about 256MB per TB of data stored.1GB of additional base memory is required for other uses.

In addition to memory used by opendedupe you will want to have memory available for filesystem cache to cache the most active parts of the lookup hashtable into ram. For a volume less than 1TB you will need an additional 1GB of ram. For a volume less than 100GB you will need an addition 8GB of RAM. For a volume over 100TB you will need an additional 16GB of ram.

An example for 100TB of deduped data:

- Hash Table Memory
    - 200MB per 1TB of Storage

    - 200MB x 100TB = 25.6 GB
- 1GB of base memory
- 8GB of Free RAM for Disk Cache
- Total = 25.6+1+8=34.6GB of RAM

CPU Sizing:

As long as the disk meets minimum IO and IOPs requirements the primary limiter for OpenDedupe performance will be CPU at higher dedupe rates. At lower dedupe rates volumes will be limited by the speed of the underlying disk.

For a single 16 Core CPU, SDFS will perform at :

- 2GB/s for 2% Unique Data
- Speed of local disk for 100% unique data. Using minimum requirements this would equal 120MB/s.

Troubleshooting (Logs)

For help in troubleshooting issues associated with SDFS of the OST plugin you can email the sdfs forum. The forum can be found at :

http://groups.google.com/group/dedupfilesystem-sdfs-user-discuss?pli=1

SDFS Logs:

SDFS creates logs under **/var/logs/sdfs/<volume-name>-volume-cfg.xml.log**. Errors can be identified in this log file

OST Plugin Logs:
The OpenDedupe OST plugin log can be found in **/tmp/logs/opendedup.log** .

NetBackup Logs:
Pertinent OST related errors and logging are trapped in the BPTM log. Netback logging for bptm can be enable by creating the bptm logging directory.

mkdir /usr/openv/netbackup/logs/bptm