TECHNICAL WHITE PAPER

ORACLE RMAN FOR TEST/DEV

PUT YOUR BACKUPS TO USE WITH FLASHBLADE

PURESTORAGE®

TABLE OF CONTENTS

INTRODUCTION

Data is often the most valuable asset in an organization. Enterprises must not only put their data to work – through processing and analytics – they must also protect that data. Organizations often deploy a robust backup strategy in order to meet stringent RPO and RTO regulatory requirements, and many have service-level agreements (SLAs) to comply with internal or external users, but it's an increasingly difficult task with ever growing database sizes.

Legacy backup appliances are typically expensive, and suffer from very slow restore times – which raises costs in the event of a failure, and limits their utility when provisioning for test/dev and analytics. Modern all-flash storage, such as FlashBlade, can dramatically accelerate backup and restore operations while simultaneously opening new opportunities for consolidation.

THE PROBLEM

Data restore has traditionally been quite slow. To keep up with data protection requirements, organizations have turned to storage and backup vendors for purpose-built appliances to speed up the process. Vendors leverage various technologies, like de-duplication to optimize space requirements on rotating media, and accelerated metadata using flash memory storage. While purpose-built appliances may have optimized backup times, they take a big hit during database restoration. The data rehydration process during recovery operations generates large random IO access patterns on spinning disk, resulting in poor performance.

In addition, customers also face challenges keeping up with demand in their test and development environment for fast provisioning of production database clones that DBAs can iterate on. While the FlashRecover snapshot functionality of Pure Storage® FlashArray allows customers to clone databases in seconds, with no additional storage requirement, many customers want to segregate their non-production environments from production and thus end up deploying additional infrastructure, including storage, to support a test and development environment. And yet, it would make perfect sense to take advantage of Oracle® backups to create the database clones to be used by testers and developers.

Ultimately, the speed at which an enterprise can restore data in the event of a failure, or quickly provision test and development environments, defines its business edge. In this paper, we will explore how to put dormant backups to use by accelerating database cloning with best-of-breed solutions from Pure Storage and Oracle.

THE SOLUTION

FlashBlade™ is a ground-breaking scale-out flash storage system from Pure Storage. Engineered with a massively parallel architecture from software to hardware, it has emerged as the industry-leading solution for use cases requiring the highest performance, like artificial intelligence and modern analytics. Many customers have tapped this performance to modernize their backup and recovery infrastructure.

> The current FlashBlade system can support a read rate of 16 GBps and a write rate of 4.5 GBps, in a single 4U chassis. The sustained write rate of 4.5 GBps is equivalent to a backup rate of 15TB/hour, which is well above the normal required backup rate of many customers. The restore rates from FlashBlade can be higher than 45TB/hour with concurrent restore operations.

Oracle customers can now direct their RMAN backups to FlashBlade and use dNFS (Direct NFS) to significantly accelerate backup and restore. The high bandwidth capabilities of FlashBlade, along with the RMAN feature, **DUPLICATE**, can clone databases very quickly from these periodic backups. This not only allows the customer to take advantage of dormant backups, but also helps in validating restore/recovery procedures. Customers can realize these backup/restore rates without needing any exotic and costly software on their hosts for block differencing purposes.

While the performance of FlashBlade is very impressive, even more attractive is its non-disruptive scalability. The capacity and performance of FlashBlade can be scaled in a non-disruptive manner, simplifying the operational procedures of a customer's backup/recovery process. The system can be scaled in seconds without running special commands or extra cables. FlashBlade's in-line compression feature reduces capacity requirements by almost 66% for most typical databases – driving costs down significantly while not sacrificing performance. Allowing the storage system to handle compression also saves dramatically on host CPU cycles.

OVERVIEW

The purpose of this technical white paper is to provide techniques and guidelines to rapidly create database clones from RMAN backups on FlashBlade. The goal of this document is not only to help Oracle database administrators meet their backup/recovery SLAs, but to enable them to rapidly provision non-production databases from their backups for secondary purposes like test and development.

AUDIENCE

The target audience for this document includes, but is not limited to, Oracle database administrators, storage administrators, IT managers, system architects, sales engineers, field consultants, professional services, and partners who are looking to design and deploy Oracle RMAN **DUPLICATE** to clone Oracle databases on FlashBlade. A working knowledge of Oracle, Linux®, server, storage, and networks is assumed but is not a prerequisite to read this document.

RMAN DATABASE DUPLICATION

Oracle Recovery Manager (RMAN) is a utility that enables effective backup and recovery of Oracle databases. RMAN utility is available free of charge and comes with Oracle software installation.

While RMAN has various features, like the ability to detect corrupt blocks in data files, backup databases without placing the database in 'hot backup' mode, perform cross-platform data conversion (useful for migration), parallelize backup and recovery using multiple processes, etc., one of the key features that helps Oracle Database administrators is database duplication.

Database duplication (also referred to as cloning) is the use of the **DUPLICATE** command within RMAN that copies all or a subset of the data from a source database. The duplicate database is useful for various scenarios, such as:

- Validation of backup and recovery procedures
- Creation of a database to perform reports
- Testing of an upgrade of a database to a new release
- Testing of an application's effect on database performance
- Creation of a standby database

One of the advantages of using the **DUPLICATE** command to create a database clone is with the DBID generation of the cloned database. If the cloning is performed outside of RMAN, a new DBID has to be generated for the cloned database using the DBNEWID utility, whereas in RMAN the **DUPLICATE** command will automatically assign a new DBID to the cloned database. This is useful if the cloned database has to be registered in the same recovery catalog as the source database.

RMAN supports two types of duplication:

- Active database duplication
- Backup-based duplication

In Active database duplication, RMAN connects to the source database as **TARGET** and to the destination database as **AUXILIARY**, and copies data to the auxiliary instance over the network without creating any backup of the source database.

In Backup-based duplication, RMAN creates a duplicate database by using existing RMAN backups and copies. This supports three different ways a duplication can be performed:

1. Duplication without a connection to the source database. RMAN obtains metadata about the backups from a recovery catalog.
2. Duplication without a connection to the source database and without a recovery catalog. RMAN obtains metadata about the backups and copies from the **BACKUP LOCATION** clause as part of the **DUPLICATE** command.
3. Duplication with a connection to the source database. RMAN obtains metadata about the backups from the source database control file or from the recovery catalog.

For ease of use and clarity, we opted to go with Option #1 (duplication without a connection to the source database and using recovery catalog) for this paper as that is the most prevalent option, even though all other options are absolutely possible using FlashBlade as the RMAN backup target.

If your requirement is to duplicate the database without a backup, our recommendation is to use Pure FlashArray's native snapshot functionality instead of active database duplication, as the FlashRecover snapshot is very efficient,

instantaneous, and consumes very little storage space at the time of creation. You can certainly perform active database duplication to FlashBlade if it is a one-time request, but if the database has to be refreshed periodically, backup-based duplication would be the best solution, as the active database duplication has to transfer the whole database over the network to FlashBlade every time, which might mean more time and resource consumption as opposed to backup-based duplication.

This paper assumes you are backing up your source database using RMAN to FlashBlade and thus putting the backups to use by creating clones of the source database for secondary purposes.

ORACLE DNFS

Oracle dNFS (direct Network File System) is the NFS client functionality directly integrated in the Oracle RDBMS server. dNFS makes the task of configuring an Oracle database on NAS storage like FlashBlade much simpler compared to Standard NFS (aka Kernel NFS). Direct NFS Client on Oracle 12c supports NFSv3, NFSv4, and NFSv4.1 protocols to access the NFS server.

The key benefits of Direct NFS Client include simplicity, ease of administration, load balancing, high availability, and cost effectiveness. Oracle has optimized the I/O code path by avoiding kernel overhead and, as such, is able to improve I/O performance.

Direct NFS Client is capable of performing concurrent direct I/O by bypassing Operating System level caches. It also performs asynchronous I/O, which allows processing to continue while the I/O request is submitted and processed. These two key performance and scalability features provide unparalleled performance when compared to Kernel NFS clients. Another key feature of Direct NFS Client is high availability. Direct NFS Client delivers optimized performance by automatically load balancing requests across all specified paths (up to 4 parallel network paths). If one network path fails, then Direct NFS Client will reissue I/O commands over any remaining paths, ensuring fault tolerance and high availability.

One of the primary challenges of Kernel NFS administration is inconsistency of configurations across different platforms. Direct NFS Client eliminates this problem by providing a standard NFS client implementation across all platforms supported by the Oracle Database. This also makes NFS a viable option on platforms like Windows, which doesn't natively support NFS.

As NFS is a shared file system, it supports Real Application Cluster (RAC) databases as well as single instance databases. Direct NFS Client recognizes when an instance is part of an RAC and automatically optimizes the mount points for RAC, relieving the administrator of manually configuring NFS parameters.

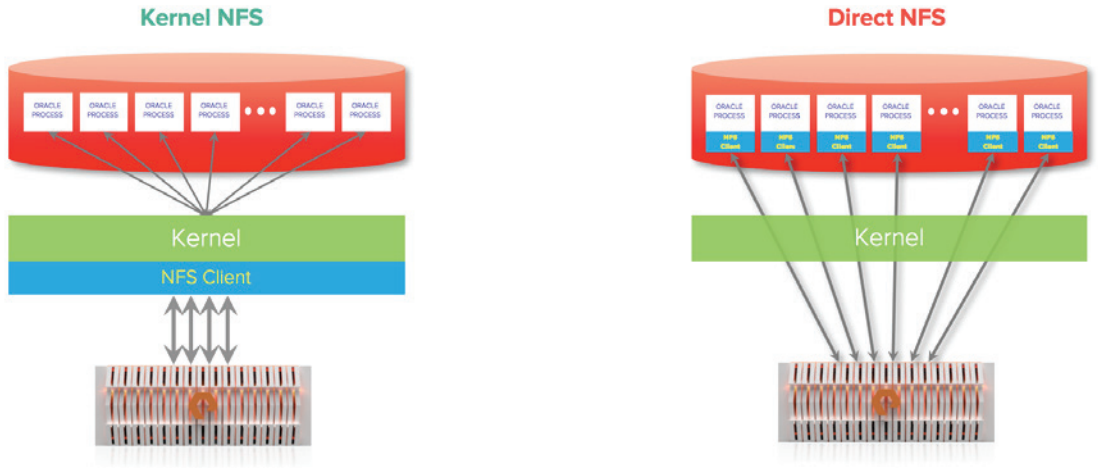

FIGURE 1. Kernel NFS vs Direct NFS communication with FlashBlade

dNFS is highly recommended in order to take advantage of FlashBlade's architecture, which thrives on parallel connections to its blades rather than a single connection. dNFS makes separate connections to the storage system for every server process.

For example, with a single mount point that is mounted using Kernel NFS, there is only one connection from the host to the storage system, and all Oracle server processes will pass through that single connection.

Alternatively, in a dNFS system with a single mount point with four paths (as defined in the **oranfstab** file), there will be 4 connections to the storage system for every Oracle server process. With FlashBlade's blade architecture, each of these connections will land on a different blade, based on host and port combinations, enabling better utilization of all compute/network resources across the FlashBlade system.

Direct NFS Client Usage

To use Direct NFS Client, the NFS file systems must first be mounted and available over regular NFS mounts. The mount options used in mounting the file systems are not relevant, as Direct NFS Client manages the configuration after installation. Direct NFS Client searches for the mount entries in the following order. It will use the first entry found if duplicate entries exist in the configuration files.

1. $ORACLE_HOME/dbs/oranfstab
2. /etc/oranfstab
3. /etc/mtab

SAMPLE ORANFSTAB FILE

```
Server: FlashBlade
local: 192.168.201.160
path: 192.168.201.100
local: 192.168.202.160
path: 192.168.202.100
local: 192.168.203.160
path: 192.168.203.100
local: 192.168.204.160
path: 192.168.204.100
nfs _ version: nfsv3
export: /rman1        mount: /m01
```

> The syntax of the **oranfstab** is critical. Be sure to follow keywords like **server**, **local**, **path**, **export**, **mount**, and **nfs_version** with a colon and the value.

For multiple paths that are on the same subnet, a static path[1] has to be setup on Linux. For all other platforms you can include the attribute **dontroute** inside the **oranfstab** file.

Enabling & Disabling Direct NFS Client

To enable the Direct NFS Client, the standard Oracle Disk Manager (ODM) library that supports Direct NFS Client should be used. It can be enabled as follows:

```
cd $ORACLE _ HOME/rdbms/lib
make -f ins _ rdbms.mk dnfs _ on
```

To disable the Direct NFS Client, perform the following:

```
cd $ORACLE _ HOME/rdbms/lib
make -f ins _ rdbms.mk dnfs _ off
```

1 See Oracle Support Document "How to Setup Direct NFS client multipaths in same subnet Doc ID # 822481.1"

RMAN Duplicate Process Overview

As we opted to use backup-based duplication, the database duplication process on FlashBlade requires the backups to be available on FlashBlade. The high-level steps are:

Perform backup of the source database to filesystem(s) hosted on FlashBlade over NFS protocol, with backup metadata synchronized with the recovery catalog database. This is a regular activity that you may have already configured.

Make the backups that are on FlashBlade available on the server where backup-based duplication processes will be performed. If the duplicate database is to be created on the same server as the source, this is not necessary.

Perform the backup-based duplication process on the server through the RMAN client by connecting to the recovery catalog and to the auxiliary instance (instance to be duplicated).

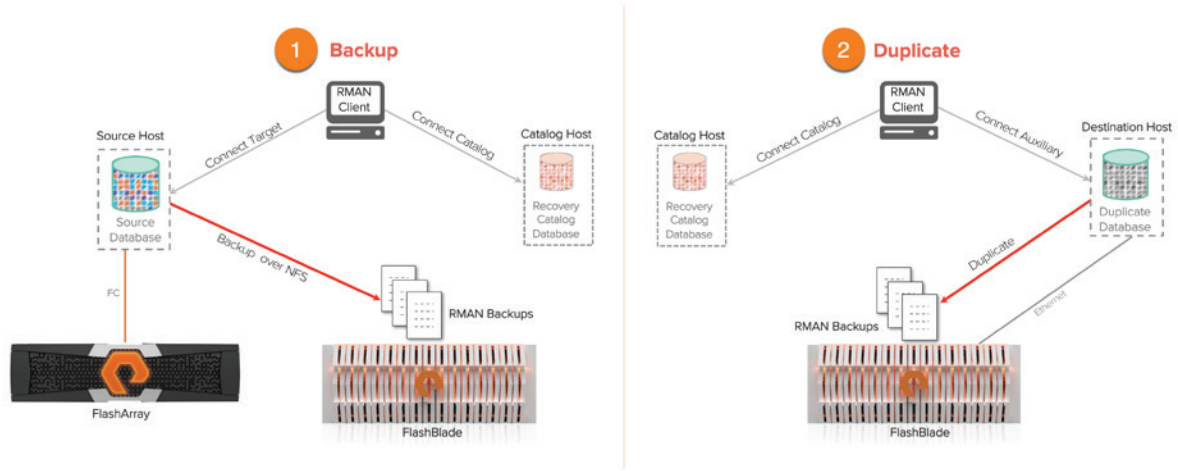

FIGURE 2. Solution design connectivity

This paper will cover two scenarios of RMAN database duplication.

1. Duplicate database onto a remote server
2. Duplicate database on the same source server

For clarity, in both these scenarios we will give the database a new name as well as use the RMAN catalog database to retrieve the metadata that is used for duplicating the database.

Environment Overview

The testing environment includes a physical database server that hosts two source databases (**OLTP** and **DW**). The server was setup with Oracle Linux 7.4 on a Cisco UCS® B200 M4 blade. The source databases were setup with Oracle 12.2 (**OLTP**) and Oracle 12.1 (**DW**) RDBMS enterprise edition on filesystems hosted on a Pure Storage FlashArray//M50 that was connected through FC protocol. The source databases were registered with a recovery catalog database that was setup on another Linux host and, for ease of use, that database was also hosted on the same Pure FlashArray. RMAN backup of the source databases was performed on the FlashBlade system over NFS protocol. One of the cloned databases (**DWDEV**) through RMAN **DUPLICATE** will be hosted on the source server and the second one (**DEVOLTP**) on a remote server (Cisco UCS B200 M4 blade) that runs the same exact OS and database version as that of source. The same FlashBlade system will be hosting the cloned databases over NFS protocol.

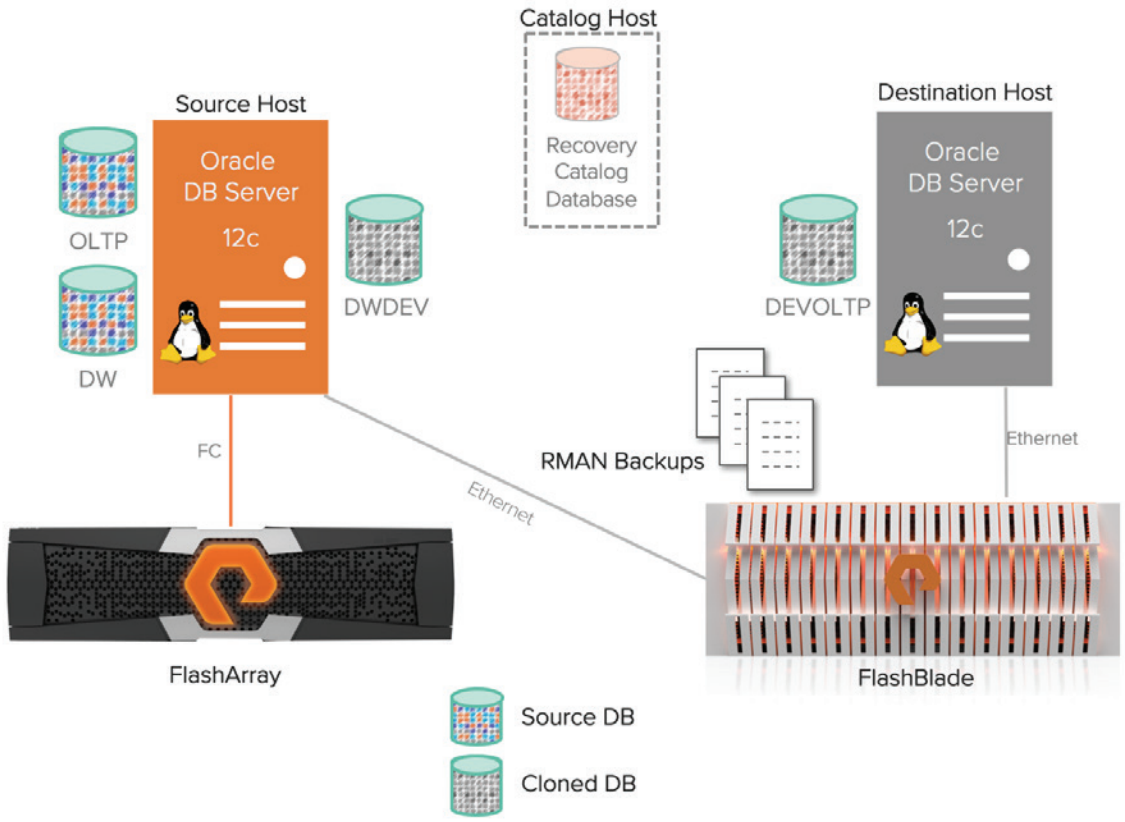

FIGURE 3. RMAN environment

Server Configuration

Two Intel® CPU-based Cisco UCS B-series B200 M4 blade servers were deployed to host the source Oracle database and the target cloned database. Each server has a Cisco UCS VIC 1340 card and was connected by four ports from

each Cisco Fabric extender of the Cisco UCS chassis to the Cisco Fabric Interconnect, which was in turn connected to the Cisco MDS 9148S for upstream connectivity to access the Pure Storage FlashArray//M50 LUNs. The server configuration is described in Table 1.

COMPONENT	DESCRIPTION
PROCESSOR	2 X INTEL XEON E5-2609 V4 1.7 GHZ (2 CPUS WITH 8 CORES EACH)
MEMORY	64 GB @ 2.4GHZ (2 X 32GB)
HBA	4 X 10G PORTS ON CISCO UCS VIC 1340 (UCSB-MLOM-40G-03) 40GBPS
NIC	6 INTERFACES CONFIGURED ON CISCO UCS VIC 1340 (1 FOR PUBLIC, 1 FOR PRIVATE, 4 FOR BACKUP)
UCS FIRMWARE (ACROSS ALL COMPONENTS)	3.1 (2B)

TABLE 1. UCS Blade configuration

The LAN setup under the service profile for the Oracle nodes was setup with six VLANs and each network interface was configured to one of the VLANs. One interface was used for public traffic, one for private traffic, and the remaining four for FlashBlade traffic. At the host level, the four network interfaces for FlashBlade were configured across four subnets on which the NFS filesystems/volumes from FlashBlade for the RMAN backups and RMAN duplicate target were to be mounted. Four private IP addresses were configured for these four interfaces.

```
[root@oradb01 ~]# ip addr |grep inet |grep 20
    inet 192.168.201.160/24 brd 192.168.201.255 scope global enp9s0
    inet 192.168.202.160/24 brd 192.168.202.255 scope global enp16s0
    inet 192.168.203.160/24 brd 192.168.203.255 scope global enp17s0
    inet 192.168.204.160/24 brd 192.168.204.255 scope global enp18s0

[root@oradb01 ~]# route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.21.122.1     0.0.0.0         UG    100    0        0 enp6s0
10.21.122.0     0.0.0.0         255.255.255.0   U     100    0        0 enp6s0
192.168.201.0   0.0.0.0         255.255.255.0   U     100    0        0 enp9s0
192.168.202.0   0.0.0.0         255.255.255.0   U     100    0        0 enp16s0
192.168.203.0   0.0.0.0         255.255.255.0   U     100    0        0 enp17s0
192.168.204.0   0.0.0.0         255.255.255.0   U     100    0        0 enp18s0
```

RMAN Target – FlashBlade Configuration

COMPONENT	DESCRIPTION
FLASHBLADE	14 X 17TB BLADES
CAPACITY	224 TB RAW 151.63 TB USABLE (WITH NO DATA REDUCTION)
CONNECTIVITY	4 X 40 GB/S ETHERNET (DATA) 1 GB/S ETHERNET (MANAGEMENT PORT)
PHYSICAL	4U
SOFTWARE	PURITY//FB 2.1.1

TABLE 2. FlashBlade configuration

FlashBlade network settings were configured with four subnets across four VLANs. The NFS filesystems are to be mounted on these four subnets (**201** to **204**) on the Oracle database servers to perform RMAN backup and duplicate functions.

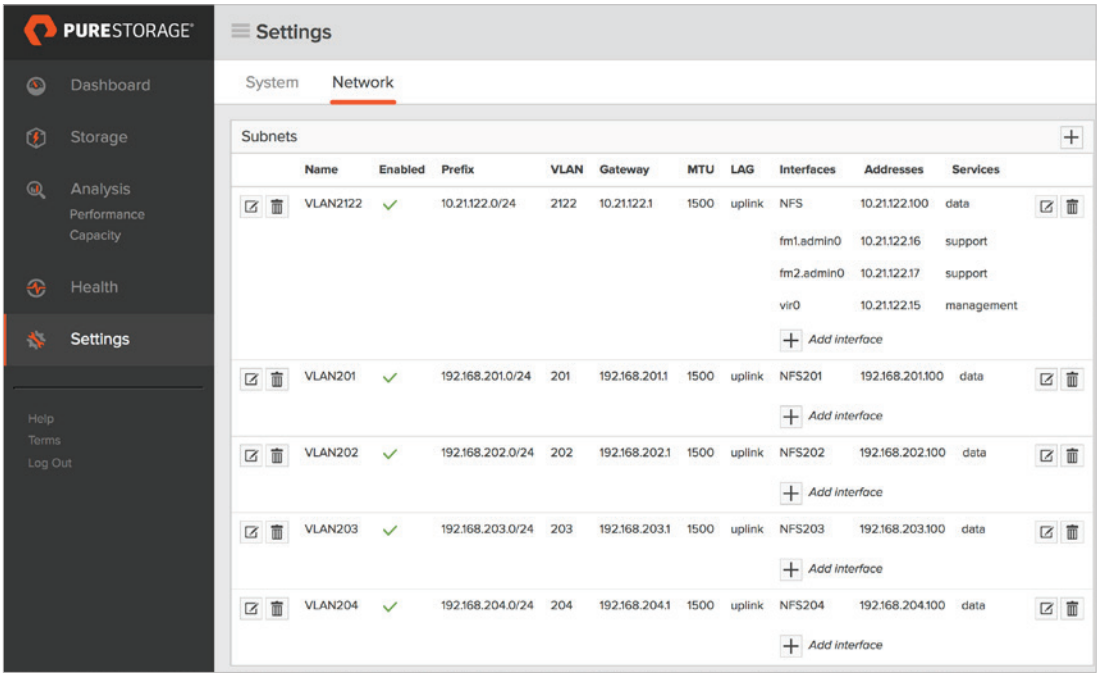

FIGURE 4. FlashBlade network settings

For the RMAN backup, four NFS filesystems were created on FlashBlade named **rman01**, **rman02**, **rman03**, and **rman04**, each with a size of 5TB. For the RMAN clone, 3 NFS filesystems were created named **devdata01**, **devdata02**, and **devarch01**, each with a size of 2TB. Each filesystem will be mounted onto the source/destination hosts on one of the four subnets.

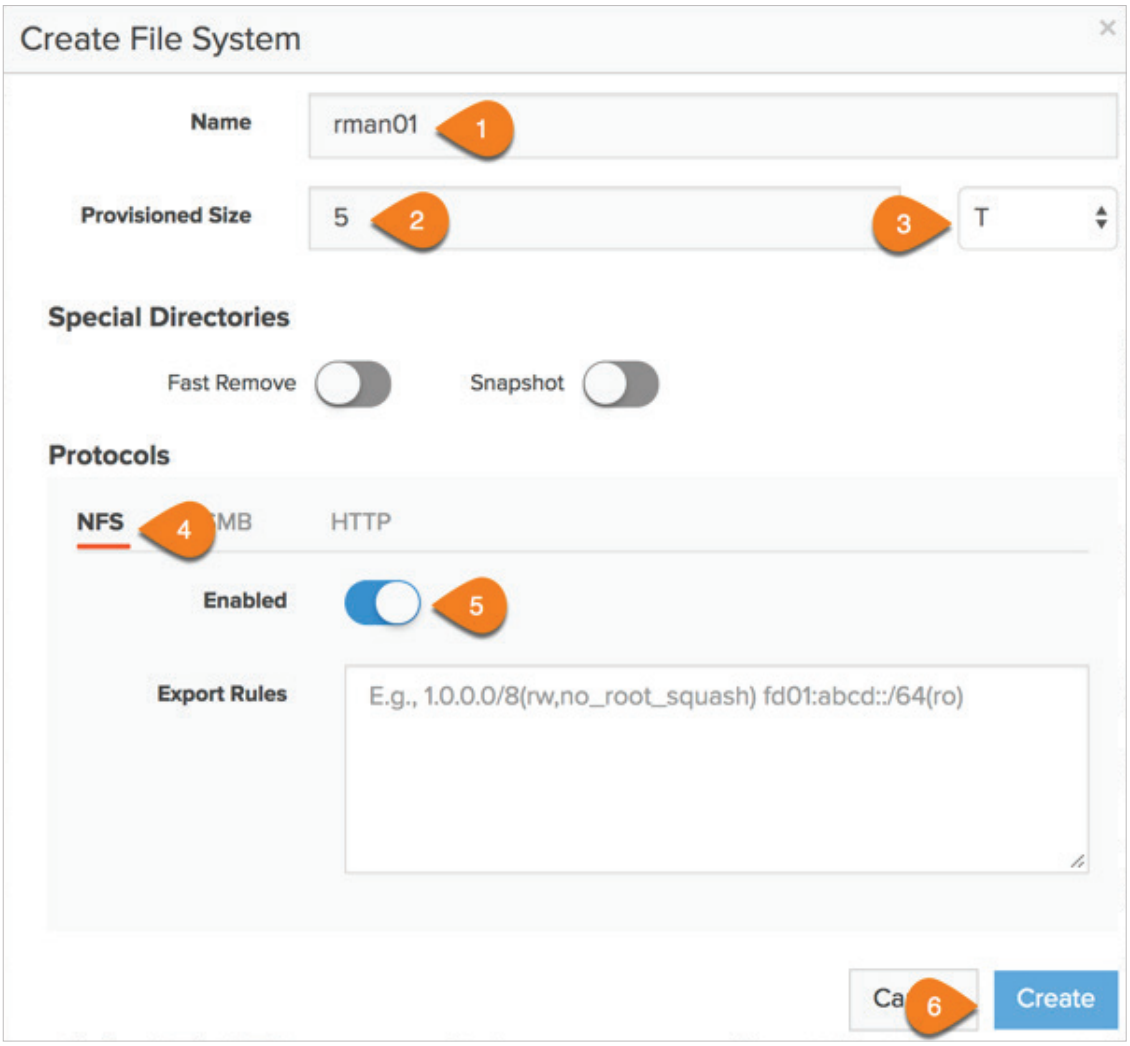

FIGURE 5. Create File System window

> Note: Be sure to size the NFS filesystem(s) to hold the RMAN backup. Ideally, the filesystems should be sized based on retention and frequency of full backups. Here, organizations will benefit from the built-in data reduction feature of FlashBlade, and the data reduction rate will vary based on the type of data. As the space is always thin-provisioned, there is no penalty for creating a larger filesystem.

Database Configuration

OLTP and data warehouse-type databases were setup on the source server and were hosted on a Pure FlashArray //M50. The databases were populated with **OLTP** and data warehouse-type data, and the allocated sizes of the databases were 1.2 TB and 1.8 TB, respectively.

SPACE USAGE OF THE OLTP DATABASE

```
SYS@OLTP> @showspace

Space Details
-----------------------------------------------
Allocated Space :    1,201.64 GB
Used Space      :    1,034.49 GB
```

SPACE USAGE OF THE DW DATABASE

```
SYS@DW> @showspace

Space Details
-----------------------------------------------
Allocated Space :    1,801.53 GB
Used Space      :    1,652.68 GB
```

Oracle dNFS Configuration

Oracle dNFS was enabled at the database level. Even though the source databases were hosted on SAN storage, the RMAN backup targets are on the NFS filesystems, and thus the **oranfstab** was updated to reflect the mount points of the filesystems that will host the backups (**rman01**, **rman02**, **rman03**, and **rman04**).

As one of the clone databases (**DWDEV**) will be hosted on the source database server, the **oranfstab** on the source database server includes the mount points that are relevant to the cloned database (**devdata01**, **devdata02**, and **devarch01**).

ORANFSTAB FILE ON THE SOURCE DATABASE SERVER

```
server:FlashBlade
local: 192.168.201.160
path: 192.168.201.100
local: 192.168.202.160
path: 192.168.202.100
local: 192.168.203.160
path: 192.168.203.100
local: 192.168.204.160
```

```
path: 192.168.204.100
nfs _ version: nfsv3
export: /rman01 mount: /b01
export: /rman02 mount: /b02
export: /rman03 mount: /b03
export: /rman04 mount: /b04
export: /devdata01 mount: /w02
export: /devdata02 mount: /w03
export: /devarch01 mount: /w05
```

The **oranfstab** file on the destination database server is similar to that of the source, but the local entries reflect the private IP addresses that are specific to the destination server.

```
server:FlashBlade
local: 192.168.201.161
path: 192.168.201.100
local: 192.168.202.161
path: 192.168.202.100
local: 192.168.203.161
path: 192.168.203.100
local: 192.168.204.161
path: 192.168.204.100
nfs _ version: nfsv3
export: /rman01 mount: /b01
export: /rman02 mount: /b02
export: /rman03 mount: /b03
export: /rman04 mount: /b04
export: /devdata01 mount: /w02
export: /devdata02 mount: /w03
export: /devarch01 mount: /w05
```

RMAN Duplication – Prerequisites

1. **RMAN backup** – This paper assumes you are already backing up your source Oracle databases to FlashBlade using RMAN. You can certainly mix full and incremental backups of individual data files, but a full backup of every data file is required.
2. **Recovery catalog** – Even though this is not a key requirement, most Oracle customers using RMAN for backups use Recovery catalog. The advantage with the recovery catalog is that it enables duplication without connection to the source database.
3. **Naming strategy for the duplicate files** – The simplest strategy is to configure the duplicate database to use the same names as the source database, assuming the following conditions are met:
   - If the source database uses ASM disk groups, the duplicate database must use ASM disk groups with the same names. Use **DB_CREATE_FILE_DEST** and **LOG_FILE_CREATE_DEST** parameters in the auxiliary instance.
   - If the source database files are Oracle Managed Files (OMF), then the auxiliary instance must set **DB_CREATE_FILE_DEST** to the same directory location as the source database.
   - If the names of the database files in the source database contain a path, this path name must be the same in the duplicate database.
4. **Oracle database software on the destination host** – Make sure the destination host has the same Oracle database software version as that of the source.
5. **Making backups accessible at the destination host** – RMAN uses metadata from the RMAN repository (recovery catalog or control file of the source database) to locate the backups and archived redo log files needed for duplication.
   - Given that the backups sent to FlashBlade are NFS mounted, it is very easy to mount the same backups on the destination host. Be sure to mount them with the same mount points on the destination host.
   - In case the backup is not available at the same location, use the **BACKUP LOCATION** option as part of the **DUPLICATE** command to point to the location of the backups.
6. **Initialization Parameter file** – Create a text-based initialization parameter file (**pfile**) for the auxiliary/destination instance with just one entry, **db_name**:
   - `db _ name = <duplicate db name>`
7. **Make sure the directory structures are populated** on the target server that will be hosting the duplicate database.
   - adump directory
8. **Password file for the auxiliary/destination instance is required** for active database duplication.
   For backup-based duplication, you can either create a password file or use OS authentication to connect to the auxiliary instance. In this paper, we used the OS authentication to connect to the auxiliary instance.

DATABASE DUPLICATION PROCESS

Duplicate Database on to a Remote Server

The backups of the source database (**OLTP**) include all the datafiles, control files, archived log files, and **SPFILE**.

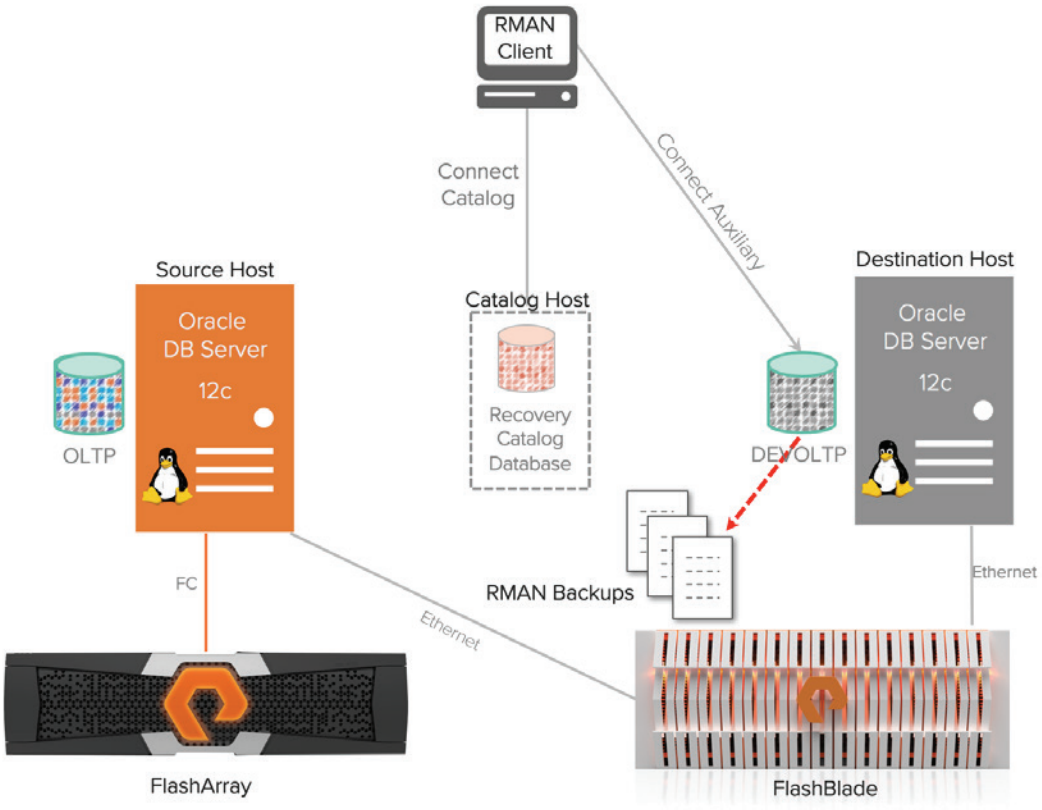

FIGURE 6. Remote server duplication

The backups from the source host (**oradb01**) were sent to four filesystems on FlashBlade mounted as **/b01**, **/b02**, **/b03**, and **/b04**.

```
[oracle@oradb01 ~]$ df -h /b0[1-4]
Filesystem              Size  Used Avail Use% Mounted on
192.168.201.100:/rman01  5.0T     0  5.0T   0% /b01
192.168.202.100:/rman02  5.0T     0  5.0T   0% /b02
192.168.203.100:/rman03  5.0T     0  5.0T   0% /b03
192.168.204.100:/rman04  5.0T     0  5.0T   0% /b04
[oracle@oradb01 ~]$
```

The following RUN block commands were performed through RMAN on the source database in order to perform the backup. To speed up the backups, we allocated 8 channels, enabling 8 streams of RMAN backup. In your case, you might already have a working RMAN setup for your periodic backups.

```
RUN
{
  ALLOCATE CHANNEL ch1 TYPE DISK FORMAT '/b01/orclbkp/oltp/%d _ D _ %T _ %u _ s%s _ p%p';
  ALLOCATE CHANNEL ch2 TYPE DISK FORMAT '/b02/orclbkp/oltp/%d _ D _ %T _ %u _ s%s _ p%p';
  ALLOCATE CHANNEL ch3 TYPE DISK FORMAT '/b03/orclbkp/oltp/%d _ D _ %T _ %u _ s%s _ p%p';
  ALLOCATE CHANNEL ch4 TYPE DISK FORMAT '/b04/orclbkp/oltp/%d _ D _ %T _ %u _ s%s _ p%p';
  ALLOCATE CHANNEL ch5 TYPE DISK FORMAT '/b01/orclbkp/oltp/%d _ D _ %T _ %u _ s%s _ p%p';
  ALLOCATE CHANNEL ch6 TYPE DISK FORMAT '/b02/orclbkp/oltp/%d _ D _ %T _ %u _ s%s _ p%p';
  ALLOCATE CHANNEL ch7 TYPE DISK FORMAT '/b03/orclbkp/oltp/%d _ D _ %T _ %u _ s%s _ p%p';
  ALLOCATE CHANNEL ch8 TYPE DISK FORMAT '/b04/orclbkp/oltp/%d _ D _ %T _ %u _ s%s _ p%p';
  BACKUP DATABASE filesperset 4;
  RELEASE CHANNEL ch1;
  RELEASE CHANNEL ch2;
  RELEASE CHANNEL ch3;
  RELEASE CHANNEL ch4;
  RELEASE CHANNEL ch5;
  RELEASE CHANNEL ch6;
  RELEASE CHANNEL ch7;
  RELEASE CHANNEL ch8;
}
RUN
{
  ALLOCATE CHANNEL ch1 TYPE DISK maxopenfiles 16;
  BACKUP CURRENT CONTROLFILE
  FORMAT '/b01/orclbkp/oltp/%d _ C _ %T _ %u'
  SPFILE
  FORMAT '/b01/orclbkp/oltp/%d _ S _ %T _ %u'
  PLUS ARCHIVELOG
  FORMAT '/b01/orclbkp/oltp/%d _ A _ %T _ %u _ s%s _ p%p';
  RELEASE CHANNEL ch1;
}
```

The above command reads the data from the source database that is hosted on FlashArray and writes the backup to FlashBlade.

```
select to_char(start_time,'mm/dd/yy hh24:mi:ss') st, to_char(end_time,'mm/dd/yy hh24:mi:ss')
et,
       INPUT_BYTES_DISPLAY, OUTPUT_BYTES_DISPLAY, INPUT_BYTES_PER_SEC_DISPLAY,
       OUTPUT_BYTES_PER_SEC_DISPLAY, TIME_TAKEN_DISPLAY
 from v$rman_backup_job_details
 where status = 'COMPLETED';

Start Time        End Time          INPUT_BYTE OUTPUT_BYT IP/SEC     OP/SEC     TIME_TAKEN
----------------- ----------------- ---------- ---------- ---------- ---------- ----------
11/09/17 10:40:14 11/09/17 10:48:58 1.01T      1.01T      1.98G      1.96G      00:08:44
```

> The backup size of 1.01 TB took 8 minutes 44 seconds at a throughput of 1.96GB/s, which is equivalent to 6.89 TB/hour.

The following screenshots show the bandwidth metrics across the source system (FlashArray) and target system (FlashBlade) during backup of the source database.

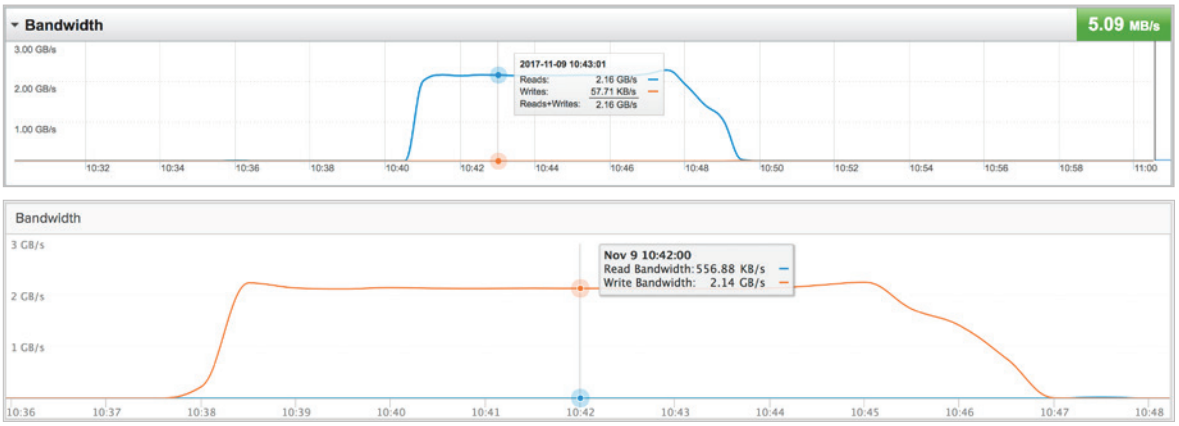

FIGURE 7. Bandwidth metrics on FlashArray and FlashBlade

DETAILED STEPS OF THE DUPLICATION PROCESS

The duplicate database (**devoltp**) will be created on the destination server (**oradb02**).

1. Make the backups available on the destination server, preferably at the same location as that of the source. Since the backups are on NFS filesystems from FlashBlade, we mounted them on the same mount points (**b01** – **b04**) on the destination server.

```
[oracle@oradb02 ~]$ df -h /b0[1-4]
Filesystem              Size  Used Avail Use% Mounted on
192.168.201.100:/rman01  5.0T  2.4T  2.7T  48% /b01
192.168.202.100:/rman02  5.0T  2.3T  2.8T  46% /b02
192.168.203.100:/rman03  5.0T  2.4T  2.7T  48% /b03
192.168.204.100:/rman04  5.0T  2.3T  2.8T  45% /b04
[oracle@oradb02 ~]$
```

2. Make the directory structure available to host the duplicate database. We used three filesystems **devdata01**, **devdata02**, and **devarch01** and mounted them on **/w02**, **/w03**, and **/w05** respectively.

```
[root@oradb02 ~]# df -h /w0[1-5]
Filesystem                Size  Used Avail Use% Mounted on
192.168.202.100:/devdata01  2.0T     0  2.0T   0% /w02
192.168.203.100:/devdata02  2.0T     0  2.0T   0% /w03
192.168.204.100:/devarch01  2.0T     0  2.0T   0% /w05
[root@oradb02 ~]#
```

3. Create an initialization parameter with just one entry **db_name**.

```
[oracle@oradb02 ~]$ more initdevoltp.ora
*.db_name='devoltp'
[oracle@oradb02 ~]$
```

4. Set the environment variables **ORACLE_HOME** and **ORACLE_SID** to reflect the corresponding values in the destination server. Make sure the Oracle binary on the destination is the same as that of source. You can accomplish this by either cloning the source Oracle home or installing the Oracle software on the destination.

```
[oracle@oradb02 ~]$ env |grep ORA
ORACLE_SID=devoltp
ORACLE_HOME=/u01/app/oracle/product/12.2.0/dbhome_1
[oracle@oradb02 ~]$
```

5. Start the destination database in **nomount** mode using the **pfile** created above.

```
[oracle@oradb02 ~]$ echo $ORACLE_SID
devoltp
[oracle@oradb02 ~]$ sqlplus / as sysdba

SQL*Plus: Release 12.2.0.1.0 Production on Thu Nov 9 12:35:38 2017

Copyright (c) 1982, 2016, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup nomount pfile='/home/oracle/initdevoltp.ora'
ORACLE instance started.

Total System Global Area  591396864 bytes
Fixed Size                  8623208 bytes
Variable Size             507513752 bytes
Database Buffers           67108864 bytes
Redo Buffers                8151040 bytes
SQL> exit
Disconnected from Oracle Database 12c Enterprise Edition Release 12.2.0.1.0 - 64bit Production
[oracle@oradb02 ~]$
```

6. Using the RMAN client, connect to the recovery catalog and to the auxiliary (duplicate) instance, and run the **DUPLICATE** command. Please provide the following parameters to reflect your environment, along with the **DUPLICATE** command and the **SPFILE** to restore the **SPFILE** from the backup. The **db_file_name_convert** and **log_file_name_convert** parameters are necessary for non-OMF, ASM files.

```
db _ file _ name _ convert – The first entry reflects the location on the source and the
second entry is the location on the destination
log _ file _ name _ convert – The first entry reflects the location on the source and the
second entry is the location on the destination
control _ files – Provide the location on the destination
log _ archive _ dest _ 1
db _ recovery _ file _ dest
audit _ file _ dest

run {
allocate auxiliary channel a1 device type disk maxopenfiles 8;
allocate auxiliary channel a2 device type disk maxopenfiles 8;
allocate auxiliary channel a3 device type disk maxopenfiles 8;
allocate auxiliary channel a4 device type disk maxopenfiles 8;
allocate auxiliary channel a5 device type disk maxopenfiles 8;
allocate auxiliary channel a6 device type disk maxopenfiles 8;
allocate auxiliary channel a7 device type disk maxopenfiles 8;
```

```
allocate auxiliary channel a8 device type disk maxopenfiles 8;
duplicate database oltp to devoltp spfile
 set audit _ file _ dest='/u01/app/oracle/admin/devoltp/adump'
 set db _ file _ name _ convert='/u02/oradata/oltp','/w02/oradata/devoltp', '/u03/oradata/oltp','/
w03/oradata/devoltp'
 set log _ file _ name _ convert='/u04/oraredo/oltp','/w02/oradata/devoltp'
 set control _ files='/w02/oradata/devoltp/control01.ctl','/w05/fra/devoltp/control02.ctl'
 set log _ archive _ dest _ 1='location=/w05/fra/devoltp/archivelog'
 set db _ recovery _ file _ dest='/w05/fra/devoltp'
 set db _ name='devoltp'
nofilenamecheck;
}
```

The **DUPLICATE** command will perform all activities, including restoring the **SPFILE**, copying datafiles to the new location, creating new control files, creating the new database, and changing DBID.

The restore part of the duplicate process took 18 minutes 49 seconds to restore 1.018TB at an average of 0.99GB/s read and 0.99GB/s write on FlashBlade.

FIGURE 8. FlashBlade bandwidth during restore

Duplicate Database on the Same Server

This process is similar to that of the previous one and, in this case, we duplicated the source database **DW** as **DWDEV** on the same server (**oradb01**). Even though we are duplicating on the same server, we are not connecting to the source database (**DW**) during the **DUPLICATE** process. Instead, we used the backup metadata from the recovery catalog database. If you do not have an RMAN catalog, you can certainly connect to the source database to get the backup details from the control file of the source database.

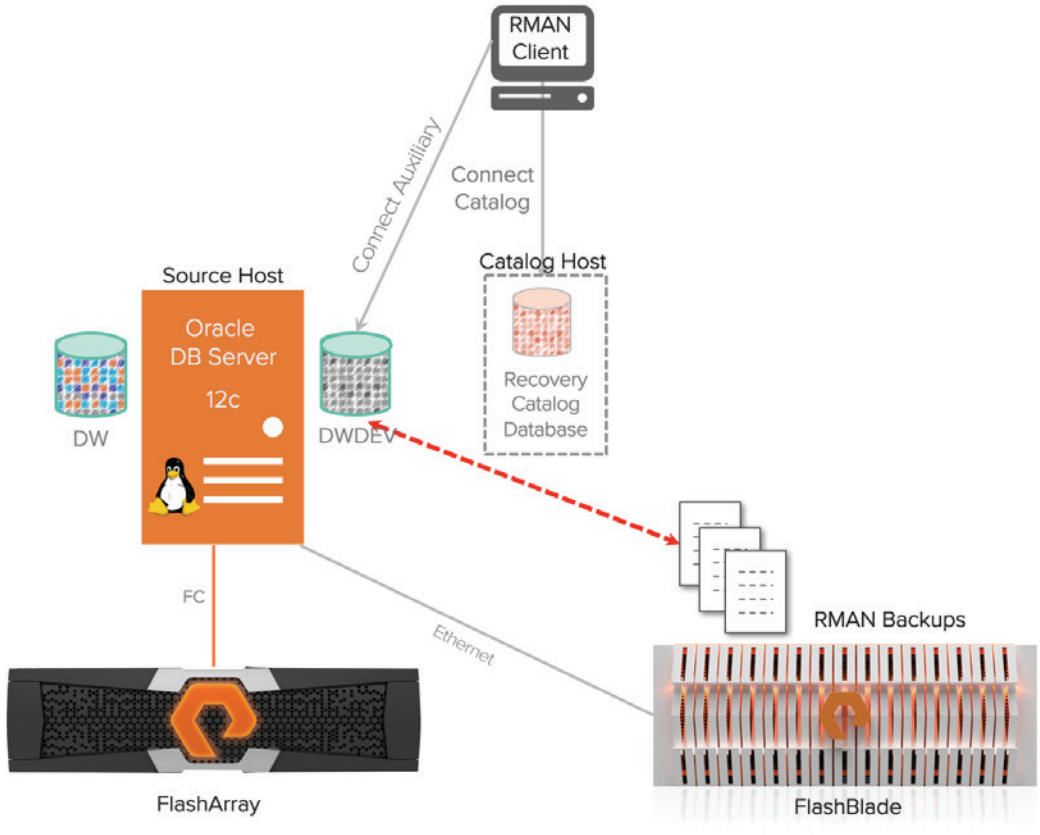

FIGURE 9. Same server duplication

The backup for this source database (**DW**) was also sent to the same NFS filesystems from FlashBlade that are mounted as **/b01**, **/b02**, **/b03**, and **/b04**.

```
select to_char(start_time,'mm/dd/yy hh24:mi:ss') st, to_char(end_time,'mm/dd/yy hh24:mi:ss')
et,
       INPUT_BYTES_DISPLAY, OUTPUT_BYTES_DISPLAY, INPUT_BYTES_PER_SEC_DISPLAY,
       OUTPUT_BYTES_PER_SEC_DISPLAY, TIME_TAKEN_DISPLAY
 from v$rman_backup_job_details
```

```
 where status = 'COMPLETED';

Start Time        End Time          INPUT_BYTE OUTPUT_BYT IP/SEC     OP/SEC     TIME_TAKEN
----------------- ----------------- ---------- ---------- ---------- ---------- ----------
11/09/17 14:01:35 11/09/17 14:15:30 1.61T      1.61T      1.98G      1.98G      00:13:55
```

It took 13 minutes 55 seconds to backup 1.61TB of data at 1.98GB/s write bandwidth to FlashBlade. FlashBlade is capable of performing 3.5GB to 4 GB/s of write – thus there is room for squeezing more write bandwidth out of FlashBlade. This intent of this paper is to show what is possible with FlashBlade, which is not a point solution but a consolidation environment that can host faster backups as well as other workloads like high bandwidth databases and analytical applications.

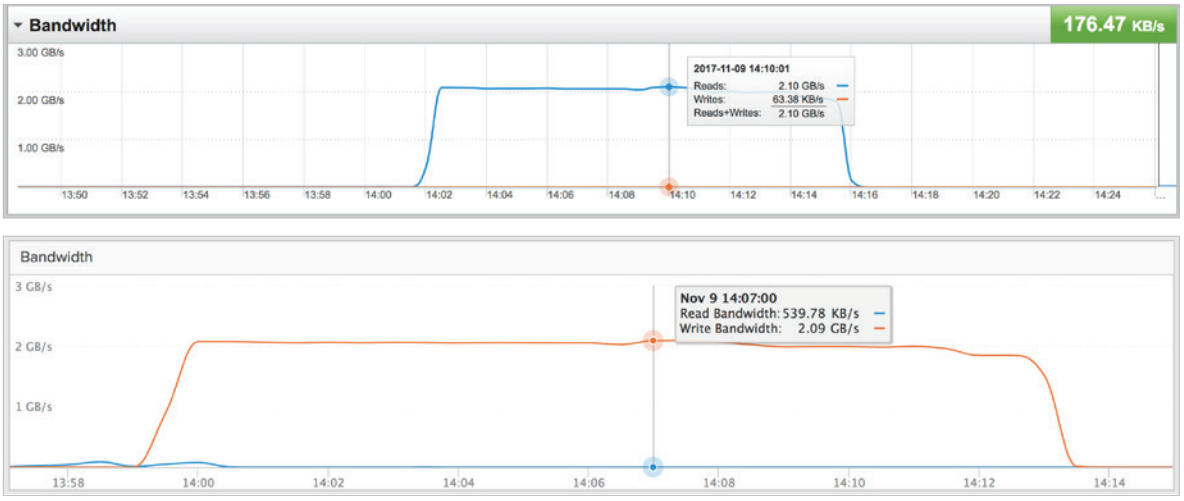

FIGURE 10. Bandwidth metrics on FlashArray and FlashBlade

DETAILED STEPS OF THE DUPLICATION PROCESS

The duplicate database (**dwdev**) will be created on the same server (**oradb01**) as the source database (**DW**).

1. Make the directory structure available to host the duplicate database. We used three filesystems, **devdata01**, **devdata02**, and **devarch01**, and mounted them on **/w02**, **/w03**, and **/w05**, respectively, on the same server which will host the duplicated database. We also created the **adump** directory in the source server.

```
[oracle@oradb01 rman]$ df -h /w0?
Filesystem                Size  Used Avail Use% Mounted on
192.168.201.100:/devdata01  2.0T  605G  1.5T  30% /w02
192.168.202.100:/devdata02  2.0T  601G  1.5T  30% /w03
192.168.204.100:/devarch01  2.0T  811M  2.0T   1% /w05
[oracle@oradb01 rman]$ ls -ltrd /u01/app/oracle/admin/dwdev/adump
drwxr-xr-x 2 oracle oinstall 12288 Nov  9 15:55 /u01/app/oracle/admin/dwdev/adump
[oracle@oradb01 rman]$
```

2. Create an initialization parameter with just one entry **db_name**.

```
[oracle@oradb01 rman]$ more initdwdev.ora
*.db_name='dwdev'
[oracle@oradb01 rman]$
```

3. Set the environment variables **ORACLE_HOME** and **ORACLE_SID** to reflect the corresponding values in the source server. As the source database and the duplicate database are on the same server, it is critical to set the environment variables to the right database. Also, if you have different versions of Oracle home installed on the same server, it is absolutely critical to point to the right Oracle software. In our case, we are pointing to the 12.1 oracle home as the source database (**DW**), set up on 12.1.

```
[oracle@oradb01 rman]$ env |grep ORA
ORACLE_SID=dwdev
ORACLE_HOME=/u01/app/oracle/product/12.1.0/dbhome_1
[oracle@oradb01 rman]$
```

4. Start the destination database in **nomount** mode using the **pfile** created above.

```
[oracle@oradb01 rman]$ sqlplus / as sysdba

SQL*Plus: Release 12.1.0.2.0 Production on Thu Nov 9 16:05:26 2017

Copyright (c) 1982, 2014, Oracle.  All rights reserved.

Connected to an idle instance.

SQL> startup nomount pfile='/home/oracle/rman/initdwdev.ora'
ORACLE instance started.

Total System Global Area  520093696 bytes
Fixed Size                  2926176 bytes
Variable Size             444598688 bytes
Database Buffers           67108864 bytes
Redo Buffers                5459968 bytes
SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, OLAP, Advanced Analytics and Real Application Testing options
[oracle@oradb01 rman]$
```

5. Using RMAN client, connect to the recovery catalog and to the auxiliary (duplicate) instance and run the **DUPLICATE** command. Please provide the following parameters to reflect your environment, along with the **DUPLICATE** command and the **SPFILE** to restore the **SPFILE** from the backup. The **db_file_name_convert** and **log_file_name_convert** parameters are necessary for non-OMF, ASM files.

```
db _ file _ name _ convert – The first entry reflects the location on the source and the
second entry is the location on the destination
log _ file _ name _ convert – The first entry reflects the location on the source and the
second entry is the location on the destination
```

```
control _ files – Provide the location on the destination
log _ archive _ dest _ 1
db _ recovery _ file _ dest
audit _ file _ dest

run {
allocate auxiliary channel a1 device type disk maxopenfiles 8;
allocate auxiliary channel a2 device type disk maxopenfiles 8;
allocate auxiliary channel a3 device type disk maxopenfiles 8;
allocate auxiliary channel a4 device type disk maxopenfiles 8;
allocate auxiliary channel a5 device type disk maxopenfiles 8;
allocate auxiliary channel a6 device type disk maxopenfiles 8;
allocate auxiliary channel a7 device type disk maxopenfiles 8;
allocate auxiliary channel a8 device type disk maxopenfiles 8;
duplicate database dw to dwdev spfile
 set audit _ file _ dest='/u01/app/oracle/admin/dwdev/adump'
 set db _ file _ name _ convert='/d02/oradata/dw','/w02/oradata/dwdev', '/d03/oradata/dw','/w03/
oradata/dwdev'
 set log _ file _ name _ convert='/d04/oraredo/dw','/w02/oradata/dwdev'
 set control _ files='/w02/oradata/dwdev/control01.ctl','/w05/fra/dwdev/control02.ctl'
 set log _ archive _ dest _ 1='location=/w05/fra/dwdev/archivelog'
 set db _ recovery _ file _ dest='/w05/fra/dwdev'
 set db _ name='dwdev'
nofilenamecheck;
}
```

The restore of 1.8TB (which is the total allocated space of the database, while 1.61TB is the used space) of DW database took 19 minutes 45 seconds which is equivalent to 1.5GB/s write bandwidth on FlashBlade. As the backup of the source database is hosted on FlashBlade, it also has to perform at a similar read bandwidth to extract the backups, which can be seen in the following screenshot from the FlashBlade GUI showing bandwidth metrics.

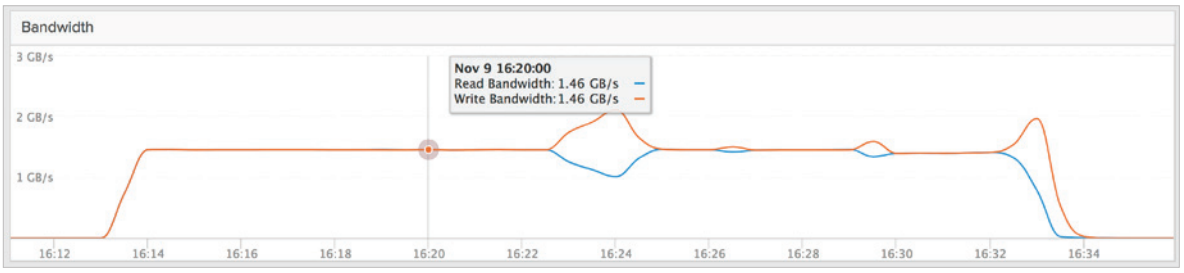

FIGURE 11. FlashBlade bandwidth during restore

Validation of the Duplicate Database

While database duplication on FlashBlade is faster and simpler, we also wanted to validate the cloned database. Thus we ran a query on the cloned database (**DWDEV**) that was hosted on the **oradb01** server and at the same time started another database duplication of the **OLTP** database onto **oradb02** server.

```
SYS@DWDEV> select /*+ full(l) parallel(l) */ count(*) from tpch.lineitem l;

  COUNT(*)
----------
9000021272

Elapsed: 00:07:08.01

Statistics
----------------------------------------------------------
        103  recursive calls
          0  db block gets
  145891713  consistent gets
  145416387  physical reads
          0  redo size
        546  bytes sent via SQL*Net to client
        552  bytes received via SQL*Net from client
          2  SQL*Net roundtrips to/from client
          1  sorts (memory)
          0  sorts (disk)
          1  rows processed
```

```
Oracle instance started

Total System Global Area    8254390272 bytes

Fixed Size                     8640288 bytes
Variable Size               1962936544 bytes
Database Buffers            6274678784 bytes
Redo Buffers                   8134656 bytes
allocated channel: a1
channel a1: SID=857 device type=DISK
allocated channel: a2
channel a2: SID=979 device type=DISK
allocated channel: a3
channel a3: SID=1101 device type=DISK
allocated channel: a4
channel a4: SID=1223 device type=DISK
allocated channel: a5
channel a5: SID=1345 device type=DISK
allocated channel: a6
channel a6: SID=1468 device type=DISK
allocated channel: a7
channel a7: SID=1589 device type=DISK
allocated channel: a8
channel a8: SID=1712 device type=DISK

Starting restore at 09-NOV-17

channel a1: starting datafile backup set restore
channel a1: restoring control file
channel a1: reading from backup piece /b01/orclbkp/oltp/OLTP_C_20171109_3psj4iok
channel a1: piece handle=/b01/orclbkp/oltp/OLTP_C_20171109_3psj4iok tag=TAG20171109T104851
```

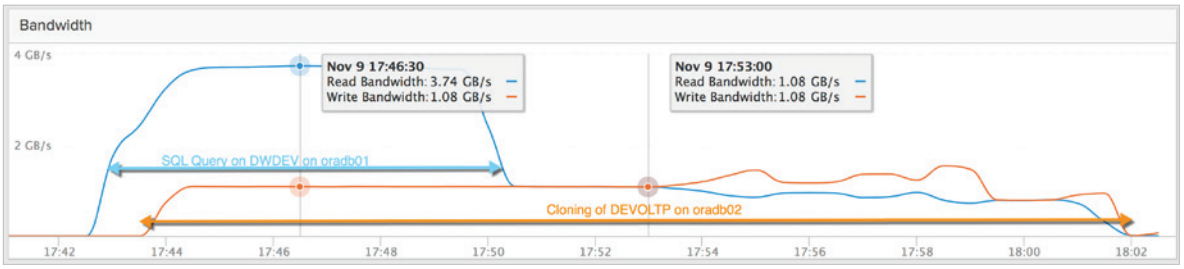

FIGURE 12. FlashBlade read/write bandwidth

As you can see from the screenshot of the FlashBlade GUI, FlashBlade was reading over 3.7 GB/s of data between the SQL query and the duplicate process where it has to read the backup from FlashBlade, while writing around 1 GB/s as part of the duplicate process.

The scale-out FlashBlade system still has room for additional performance, which means you can consolidate or add additional workloads that can accelerate your analytical applications or speed up your data warehouse ingest.

NFS FILESYSTEMS FOR ORACLE

Networked filesystems provide the flexibility to mount the same filesystem across multiple hosts to enable shared storage over TCP/IP. Thus, NFS filesystems are a viable option for an Oracle RAC environment that requires shared storage across all RAC nodes. Also, for customers who have standardized Ethernet-based connections between database hosts and storage instead of Fiber Channel, NAS storage systems like FlashBlade that provide networked filesystems would be the preferred choice. Make sure the NFS protocol is selected when creating the filesystem on FlashBlade.

USE DNFS OVER KERNEL NFS

To scale up bandwidth on FlashBlade, enable multiple connections from the client rather than a single connection. Oracle's Direct NFS creates a separate connection to the storage system for every server process, as opposed to a single connection per mount point via Kernel NFS.

ENABLE PARALLELISM

To increase read and write bandwidth on FlashBlade, use client-level parallelization techniques – like parallel queries and multiple RMAN channels based on CPU availability – on your host in conjunction with dNFS. This increases the number of connections to FlashBlade, especially with dNFS.

USE MULTIPLE NETWORK INTERFACES

To enhance network bandwidth, be sure to have multiple network interfaces on the client. These multiple interfaces can be configured on a single subnet or on multiple subnets.

SINGLE SUBNET

IO performance on a dNFS environment with multiple interfaces and a single subnet is limited to the speed of the first interface that is picked by the OS. This is because the OS returns the first path when multiple paths are available on the subnet and thus the traffic is always routed through the first path.

For example, in the setup below, FlashBlade has a single IP address on which the NFS filesystem will be mounted from the client, and the client has two interfaces.

```
Client interface       NFS server
ens7f0 10.21.108.193   10.21.108.10
ens7f1 10.21.108.194   10.21.108.10

[oracle@rlxora-b01-11-06 ~]$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.21.108.1     0.0.0.0         UG    0      0        0 ens7f0
10.21.108.0     0.0.0.0         255.255.255.0   U     0      0        0 ens7f0 <-- first route
10.21.108.0     0.0.0.0         255.255.255.0   U     0      0        0 ens7f1 <-- route ignored
```

As per the routing table, traffic can go through the first interface (**ens7f0**) as the destination and the mask fits for both routes; the OS will invariably choose the first route.

In this case, to enhance bandwidth, use NIC bonding at the client level. The relevant Oracle support document (Doc ID 833481.1) provides other means to enable multiple paths in a single subnet using static routing, but it doesn't address the issue of availability when a NIC fails or a network cable is pulled, as the routing table will not be updated.

MULTIPLE SUBNETS

"Direct NFS client best practices recommend always using multipaths in separate subnets."

Oracle recommends using a separate subnet for each interface, with support for up to four subnets. With multiple subnets, there is no need to bond the network interfaces to aggregate bandwidth across the available interfaces. The routing will be automatic in the case of multiple subnets.

```
Client interface       NFS server
ens7f0 10.21.108.193   10.21.108.10
ens7f1 10.21.107.194   10.21.107.10

[oracle@rlxora-b01-11-06 ~]$ route -n
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         10.21.108.1     0.0.0.0         UG    0      0        0 ens7f0
10.21.108.0     *               255.255.255.0   U     0      0        0 ens7f0
10.21.107.0     *               255.255.255.0   U     0      0        0 ens7f1
```

In this case, these are two dynamic routes and, based on the traffic, the route is selected automatically.

As such, if you decide to use multiple subnets, this should be configured on both the client and the FlashBlade side. Multiple subnets can be configured in the FlashBlade GUI, under **Network Settings**.

Be sure to update the **oranfstab** with the subnets and the mount point details. In RAC environments, all RAC nodes should have the appropriate **oranfstab** file configured.

NFS VOLUMES AND MOUNTPOINT REQUIREMENTS

It is not required to have as many NFS filesystems/volumes as the subnets for dNFS to be effective. Nor is it required to mount a single volume on to all subnets for dNFS to be effective. Oracle dNFS reads the **oranfstab** and, based on storage paths and mount details, it will create multiple paths when the database files are accessed.

For example, with two subnets and two mounts, dNFS would create four paths to the storage system for every server process.

LINUX MOUNT OPTIONS

For mounting the NFS filesystem on Linux, use the following mount options. Do not specify the **rsize** and **wsize** options, as the system can get the default offered by FlashBlade, which is 524288.

rw, bg, nointr, hard, tcp, vers=3, nolock, noac, actimeo=0

ENABLING AND DISABLING DIRECT NFS CLIENT

To enable the Direct NFS Client, the standard Oracle Disk Manager (ODM) library that supports Direct NFS Client should be used. It can be enabled as follows (starting with Oracle 11.2):

```
cd $ORACLE_HOME/rdbms/lib
make -f ins_rdbms.mk dnfs_on
```

To disable the Direct NFS Client, perform the following:

```
cd $ORACLE_HOME/rdbms/lib
make -f ins_rdbms.mk dnfs_off
```

VERIFYING THE USE OF DIRECT NFS CLIENT

- If dNFS is enabled, the **alert.log** will show the following entry when the database is started.

```
Oracle instance running with ODM: Oracle Direct NFS ODM Library Version 4.0
```

- 2) Check the dNFS server information from the **v$dnfs_servers** view.

```
SQL> select svrname, dirname, mntport, nfsport, wtmax, rtmax from v$dnfs_servers;
SVRNAME               DIRNAME                MNTPORT    NFSPORT      WTMAX      RTMAX
-------------------- -------------------- ---------- ---------- ---------- ----------
FlashBlade           /rman01                    2049       2049     524288     524288
```

> Note: Even though dNFS is enabled, Oracle only mounts the volume/filesystem and opens the files when they are accessed. If no datafiles are accessed, then the above view will return no rows.

CONCLUSION

Data is the new currency in the modern era. It must be analyzed, backed up, recovered, and iterated upon at the speed of modern business. While traditional backup appliances are optimized to store data, they are notoriously poor at restoring data, forcing IT to deploy additional storage systems for DBAs who want to iterate on older data.

Pure Storage FlashBlade provides a scalable, highly available, data reducible, and performant solution to consolidate general purpose backup workloads as well as test/dev operations. As shown in this paper, FlashBlade delivered over 3.7 GB/s of data read performance between the SQL query and the duplicate process where it has to read the backup from FlashBlade, while writing around 1 GB/s as part of the duplication process. These numbers are miniscule compared to what FlashBlade can offer: it still has performance headroom, which means you can consolidate additional workloads that can accelerate your analytics or significantly improve the performance of test and development systems.

Having FlashBlade as the backup target alone cannot provide these outstanding results. The source system should be capable of reading and writing data at required rates; if not, the source system becomes the bottleneck. Complementing FlashBlade, Pure's FlashArray product provides scalable read bandwidth for backup activity and comparable write bandwidth for restore activity.

REFERENCES

The following documents and links were referred to in preparing this document.

1. **Pure Storage Community pages**
   https://support.purestorage.com

2. **Oracle Support KB article "Direct NFS: FAQ Doc ID 954425.1"**
   https://support.oracle.com

3. **Duplicating Databases from RMAN Backup and Recovery User's Guide**
   https://docs.oracle.com/database/122/BRADV/rman-duplicating-databases.htm#BRADV010

APPENDIX A: VARIATION OF DUPLICATE COMMAND

The following are the variations of the backup-based **DUPLICATE** command.

1. Backup-based duplication without a connection to the source database but with the recovery catalog.

```
DUPLICATE DATABASE sourcedb TO duplicatedb
  SPFILE
    SET parameter_name='parameter value'
  NOFILENAMECHECK;
```

The **SET** command allows you to specify the initialization parameters that are relevant to the duplicate environment. The use of the **SPFILE** clause in the **DUPLICATE** command directs RMAN to use the server parameter file from the source database for the auxiliary instance.

> Note: If the source database does not use a server parameter file or RMAN cannot restore a backup of the server parameter file, you must manually create a text-based initialization parameter file.

2. Backup-based duplication with no recovery catalog but access to the source database and need to duplicate a database to a past point in time.

```
DUPLICATE DATABASE sourcedb TO duplicatedb
  SPFILE
    SET parameter_name='parameter value'
  NOFILENAMECHECK
  UNTIL TIME 'SYSDATE-7';
```

By default, the **DUPLICATE** command creates a duplicate database by using the recent backups of the source database and then performs recovery to the most recent consistent point available in the backups and archived redo logs. However, if you want to duplicate a database to a past point-in-time, you can use the **UNTIL** clause.

3. Backup-based duplication without a target or recovery catalog.

```
DUPLICATE DATABASE TO duplicatedb
  UNTIL TIME "TO_DATE('11/01/2017 15:00:00','MM/DD/YYYY HH24:MI:SS')"
  SPFILE
  BACKUP LOCATION '/backups'
    SET parameter_name='parameter value'
  NOFILENAMECHECK;
```

The use of the **BACKUP LOCATION** clause identifies this as a backup-based duplication with neither a target connection nor a recovery catalog. The **UNTIL TIME** clause is the only option permitted with the **BACKUP LOCATION** clause.

The source database name is not specified. This way, **DUPLICATE** obtains the database name and DBID from the backups.

The **NOFILENAMECHECK** option is necessary when the duplicate database files use the same names as the source database files.

ABOUT THE AUTHOR

Somu Rajarathinam is the Pure Storage Oracle Solutions Architect responsible for defining database solutions based on the company's products, performing benchmarks, and developing reference architectures for Oracle databases on Pure.

Somu has over 20 years of Oracle database experience, including as a member of Oracle Corporation's Systems Performance and Oracle Applications Performance Groups. His career has also included assignments with Logitech, Inspirage, and Autodesk, ranging from providing database and performance solutions to managing infrastructure, to delivering database and application support, both in-house and in the cloud.

Web: www.somu.us
Twitter: @purelydb

ps_wp35p_oracle-rman-duplicate-on-flashblade _ltr_01