3PAR Thin Conversion: Linux and Unix

Migrating from Legacy Storage to 3PAR Utility Storage

Document Abstract:

3PAR® Thin Provisioning software reduces storage costs by using copy-on-write technology that allocates physical disk blocks at the point they are actually written, thus eliminating the need for pre-planning, manual provisioning, and large up-front capacity purchases typical of legacy storage. The 3PAR Gen3 ASIC with Thin Built In™ works with 3PAR Thin Provisioning and data stored on legacy storage to enable the conversion of “fat” file systems to thinner, more efficient volumes on the 3PAR InServ® Storage Server. This document shows how a Linux or Unix system administrator can prepare for and perform this conversion.

Part Number: luf2t-wp-09.1

Table of Contents

1 Introduction

3PAR® Thin Provisioning software is a green technology that dramatically cuts capacity, energy, and related costs while substantially alleviating storage and system administration overhead. By allowing the safe overallocation of physical capacity, 3PAR Thin Provisioning lets organizations provision virtual capacity to applications once and then purchase physical capacity only as applications truly require it for written data. Since its introduction, 3PAR Thin Provisioning has continued to set the industry standard against which all other "thin" approaches are measured.

With 3PAR Thin Provisioning, capacity is dedicated and configured autonomically and in small increments from a single, reservationless, comprehensively scalable reservoir, so organizations can provision storage efficiently and without waste. However, starting "thin" has not been an option with many legacy storage vendors. So what about the petabytes of storage that currently reside on expensive and underutilized legacy storage? This technical paper discusses how organizations can address this problem of inefficient and underutilized storage in Linux and Unix environments by migrating from "fat" legacy storage to "thin" 3PAR Utility Storage.

2 3PAR Thin Technologies: Start Thin, Get Thin, Stay Thin

In an ideal world, all of your storage would start thin. But for organizations that have been using legacy solutions instead of 3PAR Thin Provisioning, starting thin may not have been an option. 3PAR Utility Storage allows organizations to not only start thin, but also to get thin and to stay thin.

Working in conjunction with 3PAR Thin Provisioning, 3PAR InServ® Storage Servers with Thin Built In™ feature built-in zero detection[1] which enables organizations to intelligently leverage data migration[2] from legacy arrays onto more efficient thin volumes on the InServ while preserving service levels and without disruption or performance impact. 3PAR is the first in the industry to offer storage systems with this zero detection capability designed into the hardware architecture of its arrays to enable wire-speed fat-to-thin conversions. The revolutionary 3PAR Gen3 ASIC is the only processor that provides an efficient, silicon-based mechanism for converting traditional "fat" volumes from other platforms to more efficient "thin" volumes on 3PAR arrays with on-chip, on-the-fly detection of all-zero blocks. With 3PAR, getting thin is now just as simple as starting thin.

[1] The zero-detection features require a 3PAR T- or F-Class InServ Storage Server and InForm Operating System 2.3.1.
[2] 3PAR Thin Conversion is migration-method agnostic. Data migration may be performed using tools provided with the host operating system or with third-party migration tools.

3 3PAR Thin Conversion Process for Linux and Unix

3PAR Thin Conversion is a three-step process for migrating data from a fully-provisioned volume (or LUN) on a legacy storage array to a Thin Provisioned Virtual Volume (TPVV) on a 3PAR InServ Storage Server. This can be done using tools provided with the Linux or Unix operating system, or with third-party migration tools, With either approach, the process starts with a planning step that identifies and quantifies free or unused space in order to determine whether or not there is significant benefit to “thinning” the volume. A preparation step is next, during which the volume is cleaned up and unused space overwritten with zeros. The final step is the migration itself, during which the InServ array detects blocks of zeros and does not allocate physical storage for them. Each of these steps is briefly described below. A detailed explanation of each step then follows.

Step 1: Assessment

During the first step of 3PAR Thin Conversion, the administrator quickly identifies whether or not the volume might be amenable to thinning based on file system free (or unused) space. In most cases, volumes which are good candidates for thinning can be identified in a minute or two.

There may be opportunities to create additional free space at this point by shrinking databases, emptying trashcans, deleting temporary files, archiving unused files, and performing other types of file system housekeeping. Specific instructions for these actions depend on the applications which use the file system and other details pertaining to how the file system is used. These instructions are beyond the scope of this document. File system cleanup can free up substantial amounts of storage in most file systems, leading to significant benefits from thinning even if the preliminary assessment indicates otherwise.

Step 2: Preparation

The preparation step consists of writing zeros to the free or unused space in the file system. Writing zeros does not free the space, but zeroed space detected by the 3PAR Gen3 ASIC during the migration step will not be allocated physical storage, so this step is necessary for thinning the volume.

Step 3: Migration

The final step of volume thinning is migration of the file system, which is the act of copying (or migrating) the volume containing the “fat” file system with zeroed free space to a TPVV on the InServ array.

The following subsections describe in detail the actions required to accomplish these three steps (including examples) for the following operating systems:

- Red Hat® Enterprise Linux® 5 (RHEL 5) using the EXT-3 file system
- Solaris 10
- AIX 6.1
- HP-UX 11i v3

3.1 Assessment

File systems contain user data, metadata used to manage the storage space, and free space. Free space includes storage blocks which have never been used plus blocks from deleted files. In most cases, when a file is deleted, its blocks are simply added to the pool of free space without zeroing or otherwise erasing them.

The core of the assessment step is to determine the number of physical blocks in the file system which are free or unused—in other words, they are not allocated to user or application data or file system metadata.

Once this initial determination has been made, clean the file system to recover as much free space as possible. Actions to consider include shrinking databases[3], emptying trashcans, deleting temporary files, and archiving unused files. File system cleanup is a crucial step in order to maximize the space savings from the conversion and may lead to significant benefits from thinning, even if the preliminary assessment indicates otherwise.

3.1.1 Quantifying Free Space

The `df` command will identify the amount of free space in the file system. The command is widely available across Linux and Unix variants, but the options and formatting of the output varies. The following example shows how `df` looks on Red Hat Enterprise Linux:

```
[root@rhel5 /] # df –T /vcpvv600
Filesystem    Type   1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg--f2t-vcpvv600
              ext3     9978408   2872004   6599524  31% /vcpvv600
```

In this example, the file system is about 9.5 GB (9,978,408 KB) in size but only 2.7 GB (2,872,004 KB) or 31% is in use; 6.3 GB is available (free).

Here is how `df` looks on Solaris 10:

```
bash-3.00# df -k /zfs_test
Filesystem            kbytes    used   avail capacity  Mounted on
zfs_test             10257408 3139184 7118112    31%    /zfs_test
```

[3] Consult the documentation for your database software for specific instructions on how to shrink the database.

The next example shows `df` on AIX 6.1:

```
# df -k /mnt_400
Filesystem    1024-blocks      Free %Used    Iused %Iused Mounted on
/dev/vcpvv400    10436608   7390452   30%       21     1% /mnt_400
```

The final example shows `df` on HP-UX 11i v3:

```
# df -k /vcpvv
/vcpvv         (/dev/vg-f2t/cpvv      ) :  9509874 total allocated Kb
                                           6774068 free allocated Kb
                                           2735806 used allocated Kb
                                                29 % allocation used
```

3.1.2 Choosing a Migration Mechanism

Several methods exist to perform online migrations of file systems on Linux or Unix operating systems:

- Mirroring using volume manage software
- External migration appliances

The effectiveness of Thin Conversion is independent of the migration method, so the choice may be made solely based on site preferences and policies.

Mirroring assumes the file system resides in a volume managed by a volume manager instead of a raw volume. This paper uses examples based on volume management software provided with the operating system, but third-party products (such as VxVM, a component of Veritas Storage Foundation) may also be used. The existing volume is mirrored to a TPVV on the 3PAR array, then the original (fat) volume is removed from the mirror set. The mirroring process is transparent to applications using the file system.

> **NOTE:** The ZFS file system is integrated with volume management using virtual storage pools called *zpools*. Because the pool management functions are aware of the file system, they are able to copy only used blocks when creating a mirror. Therefore, when migrating a ZFS volume on Solaris 10 using ZFS pool mirroring, there is no need to zero free space. In this case. portions of this document which discuss zeroing of free space, including the preparation step in Section 3.2, may be skipped.

External migration appliances can also be used to migrate volumes without service interruption until the migration appliance is removed from the SAN.

3.1.3 Determining the Benefit of Zeroing Free Space

The potential benefit of zeroing free space prior to migrating a file system depends on how much allocated physical space is not used by live data—in other words, the number of file system blocks on the file system’s free list. Obviously, if there is relatively little unused space, then there is little benefit to zeroing the free space to recapture this relatively small unused space. The reality is that file systems often have significant amounts of free space, all allocated to physical blocks. In this latter case, zeroing this significant amount of free space (to remove traces of deleted files) prior to file system migration will result in substantial storage savings.

3.2 Preparation

Preparation consists of writing zeros over remnants of old data in the free space. (This step can be omitted if ZFS pool management in Solaris 10 is used to migrate a ZFS file system. See Section 3.1.2 for more information.) The dd command may be used to zero free space on Linux and Unix systems. To minimize any impact on other processes, it is suggested that zeroing using dd be done incrementally rather than in a single step. For example, to zero ~7 GB of free space on Red Hat Enterprise Linux 5, use the following commands to zero one GB at a time:

```
[root@rhel5 /] # mkdir /vcpvv600/zerofree
[root@rhel5 /] # for ((i=0;i<7;++i)) do
> echo Generating file /vcpvv600/zerofree/$i ...
> nice dd if=/dev/zero of=/vcpvv600/zerofree/$i bs=1k count=1m
> done
Generating file /vcpvv600/zerofree/0 ...
1048576+0 records in
1048576+0 records out
Generating file /vcpvv600/zerofree/1 ...
1048576+0 records in
1048576+0 records out
…
```

Once the zeroing process completes, verify that all of the free space has be written with zeros:

```
[root@rhel5 /] # df –T /vcpvv600
Filesystem    Type   1K-blocks      Used Available Use% Mounted on
/dev/mapper/vg--f2t-vcpvv600
              ext3     9978408   9978408         0 100% /vcpvv600
```

Finally, delete the files of zeros:

```
[root@rhel5 /] # rm –rf /vcpvv600/zerofree
```

Except for the df command options and output formatting (illustrated in Section 3.1.1), the process is identical for the Unix operating systems discussed in this paper.

3.3 Migration

File systems are ready to be migrated once their free space has been zeroed. This paper illustrates the use of the volume mirroring approach described in Section 3.1.2 to perform the file system migration. If an appliance-based migration approach has been selected, documentation for the appliance should be used in lieu of the steps detailed in Section 3.3.2.

3.3.1 Enabling Zero Detection on the InServ

Before initiating the file system migration, it is necessary to enable zero detection on the InServ to which the file system(s) will be migrated. For example, the following command enables zero detection for a TPVV named `rhelthin1`:

```
t400 cli% setvv –pol zero_detect rhelthin1
```

3.3.2 Migrating the File System

Three simple steps are required to accomplish migration of a file system from fat storage to a TPVV using volume mirroring:

1. Add the TPVV as a mirror of the original volume.
2. Wait for the mirror to be synchronized.
3. Remove the old, fat storage from the mirror set.

While these steps are simple, the commands to accomplish this with a specific volume manager vary greatly. For this reason, an example is provided for each operating system covered by this paper using the volume management software included with the operating system. If you use a third-party volume manager such as VxVM (a component of Veritas Storage Foundation), you should consult the documentation for your product to determine the specific commands required.

3.3.2.1 Migrating the File System: Red Hat Enterprise Linux 5

In the following example, the Logical Volume Manager (LVM) component of Red Hat Enterprise Linux is used to migrate an EXT-3 file system from fat storage to a TPVV. In addition to the source and target volumes, a third, temporary volume is used to store the LVM mirror log that tracks which regions are in sync during the volume mirroring. This log can be kept in memory but Red Hat recommends keeping it on disk to provide persistence across reboots.

In this example, the following physical volume (PV) mappings are assumed:

- `/dev/dm-36` is the old, fat storage, mounted as `/vcpvv600`.
- `/dev/dm-37` is the new TPVV.
- `/dev/dm-44` contains the LVM mirror log.

All three PVs are part of Volume Group (VG) `vg-f2t`.

First, `/dev/dm-37` is added as a mirror of `/dev/dm-36` (using `/dev/dm-44` as a mirror log):

```
[root@rhel /]# lvconvert -m1 /dev/vg-f2t/vcpvv600 /dev/dm-36
/dev/dm-37 /dev/dm-44
  /dev/vg-f2t/vcpvv600: Converted: 6.3%
  /dev/vg-f2t/vcpvv600: Converted: 12.7%
  /dev/vg-f2t/vcpvv600: Converted: 18.9%
  /dev/vg-f2t/vcpvv600: Converted: 25.5%
  /dev/vg-f2t/vcpvv600: Converted: 31.7%
  /dev/vg-f2t/vcpvv600: Converted: 38.1%
  /dev/vg-f2t/vcpvv600: Converted: 44.3%
  /dev/vg-f2t/vcpvv600: Converted: 50.5%
  /dev/vg-f2t/vcpvv600: Converted: 57.0%
  /dev/vg-f2t/vcpvv600: Converted: 63.4%
  /dev/vg-f2t/vcpvv600: Converted: 69.8%
  /dev/vg-f2t/vcpvv600: Converted: 76.3%
  /dev/vg-f2t/vcpvv600: Converted: 83.0%
  /dev/vg-f2t/vcpvv600: Converted: 89.4%
  /dev/vg-f2t/vcpvv600: Converted: 95.8%
  /dev/vg-f2t/vcpvv600: Converted: 100.0%
  Logical volume vcpvv600 converted.
```

After the mirror has been created, /dev/dm-36 must be removed from the mirror:

```
[root@rhel /]# lvconvert -m0 /dev/vg-f2t/vcpvv600 /dev/dm-36
Logical volume vcpvv600 converted.
```

The volume mounted on `/vcpvv600` now resides on the TPVV.

3.3.2.2 Migrating the File System: Solaris 10

The ZFS pool management tools of Solaris 10 are used in this example to migrate a ZFS file system from fat storage to a TPVV.

The following devices are used:

- `/dev/dsk/c6t50002AC0006400C9d0` is the old, fat storage which contains a non-redundant storage pool (zpool).

- `/dev/dsk/c6t50002AC000C800F1d0` is the new TPVV.

A file system mounted at `/cpvv100` is the only file system in the `f2t` zpool. Mirroring applies to the entire storage pool, so if the zpool contained multiple file systems all would be migrated en bloc.

The first step is to attach the TPVV to the zpool as a mirror, using the `zpool status` command to monitor progress as follows:

```
# zpool status
  pool: f2t
 state: ONLINE
config:

        NAME                           STATE     READ WRITE CKSUM
        f2t                        ONLINE       0     0     0
          c6t50002AC0006400C9d0    ONLINE       0     0     0
# zpool attach f2t /dev/dsk/c6t50002AC0006400C9d0 \
/dev/dsk/c6t50002AC000C800F1d0
# zpool status
  pool: f2t
 state: ONLINE
 scrub: resilver completed after 0h0m with 0 errors on Wed May 27
23:02:58 2009
config:

        NAME                       STATE     READ WRITE CKSUM
        f2t                        ONLINE       0     0     0
          mirror                   ONLINE       0     0     0
            c6t50002AC0006400C9d0  ONLINE       0     0     0
            c6t50002AC000C800F1d0  ONLINE       0     0     0
```

Once the mirror synchronization (known as *resilvering* in ZFS terminology) has been completed, the old, fat storage may be detached as follows:

```
# zpool detach f2t /dev/dsk/c6t50002AC0006400C9d0
```

The file system mounted at `/cpvv100` now resides on the TPVV.

3.3.2.3 Migrating the File System: AIX 6.1

In the following example, the AIX 6.1 Logical Volume Manager (LVM) is used to migrate a JFS file system from fat storage to a TPVV.

The following physical volume (PV) mappings are assumed:

- `hdisk4` is the old, fat storage which contains a Volume Group (VG) named `F2T`.
- `hdisk6` is the new TPVV.

The `F2T` VG contains one Logical Volume (LV) named vcpvv400.

First, `hdisk6` is added to the VG as a mirror as follows:

```
# mirrorvg F2T hdisk4 hdisk6
  0516-1804 chvg: The quorum change takes effect immediately.
```

The command does not complete until the mirror has been created and synchronized. Once the command completes, remove the old, fat volume from the VG as follows:

```
# unmirrorvg F2T hdisk4
0516-1804 chvg: The quorum change takes effect immediately.
```

The LV contained in `F2T` and mounted on `/vcpvv00` now resides on the TPVV.

3.3.2.4 Migrating the File System: HP-UX 11i v3

In the following example, the HP-UX Logical Volume Management (LVM) tools are used, via the System Management Homepage (SMH), to migrate a VxVS file system from fat storage to a TPVV.

The following physical volume mappings are assumed:

- `/dev/dsk/c6t14d6` is the old, fat storage which contains a volume group named `vg-f2t`.
- `/dev/dsk/c9t4d1` is the new TPVV.

The `vg-f2t` volume group contains one logical volume named `cpvv`, mounted on `/vcpvv`.

Start SMH as follows:

```
# /usr/sbin/smh
```

Add the TPVV to the volume group. From the SMH main menu, navigate to the **Add Mirror(s)** page and add `/dev/dsk/c9t4d1` to the `vg-f2t` volume group as follows:

1. Select **Disks and File Systems.**
2. Select **Logical Volumes.**
3. Select **Details.**
4. Select **Add Mirror(s).**
5. In the **Add Mirror(s) to Logical Volume** field enter `/dev/vg-f2t/cpvv.`
6. Under **Extent Allocation Options** select **Specify Physical Volume(s) for Allocation.**
7. Select the PV named `/dev/dsk/c9t4d1.`
8. Select **Add.**

Wait until SMH displays the following message:

```
Logical volume "/dev/vg-f2t/cpvv" has been successfully extended.
Command Successful
```

Finally, remove the old, fat volume from the volume group. From the SMH main menu, navigate to the **Remove Mirror(s)** page and remove `/dev/dsk/c6t14d6` from the `vg-f2t` volume group as follows:

1. Select **Disks and File Systems.**
2. Select **Logical Volumes.**
3. Select **Details.**

4. Select **Remove Mirror(s).**
5. In the **Remove Mirror Copies of Logical Volume** field enter `/dev/vg-f2t/cpvv.`
6. Under **Extent Allocation Options** select **Remove mirror on specified physical volume.**
7. Select the PV named `/dev/dsk/c6t14d6.`
8. Select **Remove.**

The logical volume contained in `vg-f2t` and mounted on `/vcpvv` now resides on the TPVV.

3.3.3 Verifying Storage Savings

After the conversion, the `showvv` InServ command can be used to verify the space savings as follows:

```
t400 cli% showvv rhelthin1
 Id       Name      Type CopyOf BsId Rd   State AdmMB SnapMB userMB
443  rhelthin4 Base,tpvv    ---  443 RW started   256   3584  51200
-------------------------------------------------------------------
  1   total LD                                    256   3584      0
    total virtual                                   -      -  51200
```

In this example (also illustrated in Figure 3), a 50 GB VLUN is presented to the Linux server (51200 MB of user space), but only 3.5 GB of usable physical storage has been allocated (3585 MB of snap space, not counting parity or mirror space for RAID) plus 256 MB of admin space to store the TPVV-to-physical block mappings.

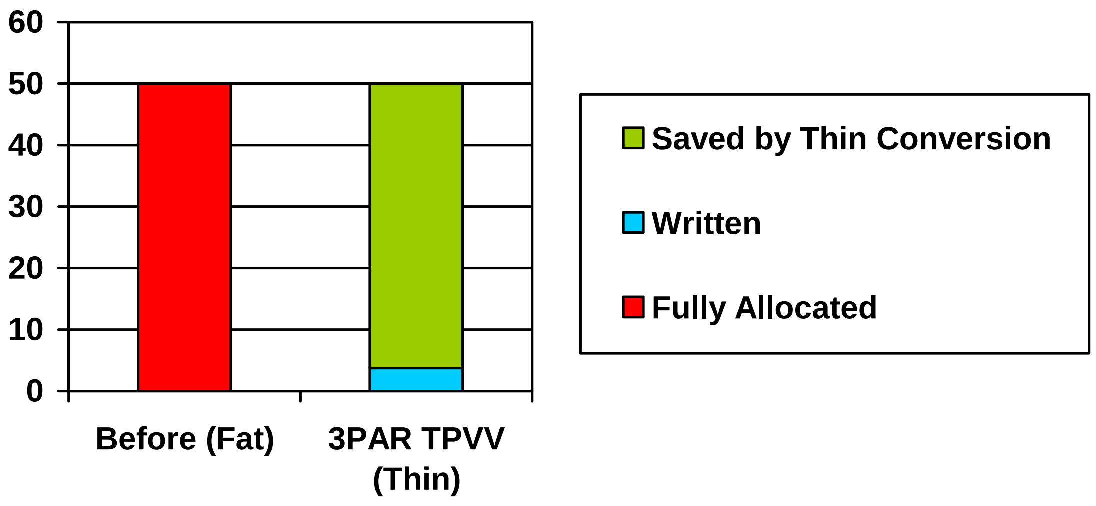

Figure 3: 50-GB LUN Before and After Thin Conversion.

4 Summary

As modern datacenters are constantly asked to do more with less—especially during times of tightening IT budgets—thin technologies are an ideal solution. 3PAR Thin Provisioning dramatically cuts capacity and related costs while substantially alleviating storage and system administration. With 3PAR Thin Provisioning and other 3PAR thin technologies, organizations can not only start thin, but can also get thin and stay thin. 3PAR InServ Storage Servers with Thin Built In™ and zero detection capability give organizations the ability to intelligently migrate data from traditional arrays onto more efficient thin volumes while preserving service levels and without disruption or performance impact. With 3PAR, getting thin has never been so simple.

About 3PAR

3PAR® (NYSE: PAR) is the leading global provider of utility storage, a category of highly virtualized and dynamically tiered storage arrays built for public and private cloud computing. Our virtualized storage platform was built from the ground up to be agile and efficient to address the limitations of traditional storage arrays for utility infrastructures. As a pioneer of thin provisioning and other storage virtualization technologies, we design our products to reduce power consumption to help companies meet their green computing initiatives, and to cut storage total cost of ownership. 3PAR customers have used our self-managing, efficient, and adaptable utility storage systems to reduce administration time and provisioning complexity, to improve server and storage utilization, and to scale and adapt flexibly in response to continuous growth and changing business needs. For more information, visit the 3PAR Website at: www.3PAR.com.