SUN BLUEPRINTS APPROVED
[illegible]

Hardware Replication Challenges

Selim Daoud, Solution Technology Group, Switzerland

Sun BluePrints™ OnLine—November 2003

Sun Microsystems, Inc.
4150 Network circle
Santa Clara, CA 95045 USA
650 960-1300

Part No.: 817-4127-10
Revision A
Edition: November 2003

Please
Recycle

Adobe PostScript™

Hardware Replication Challenges

As the business value of the data increases, the replication technique becomes more important. Consequently, more and more information technology (IT) infrastructures rely on data replication for availability, access, and security of vital data. Data replication consists of cloning a set of data stored on a primary disk onto a secondary disk. Storage administrators set up data replication for various reasons:

- Disaster recovery plans—store a clone of the production data off site.
- Live data manipulation—treat data, while avoiding overkill of the production system.
- Online backups—back up online data without disrupting the production system

This article covers the following topics:

This article is intended for use by intermediate-level system administrators.

The I/O stack model provides a programmatic view of the different treatment of data during I/O operations. Different types of replication exist along the I/O stack. This article describes the challenging aspects of hardware replication. Commonly used in large environments, this type of replication is attractive as it does not overload a server and because it is operating system (OS) independent. However, accessing the replicated data is not straightforward. Accessing the replicated data without sacrificing the performance and flexibility of a Logical Volume Manager (LVM) requires additional steps. As an example, this article details how to access, on a single host, both primary and secondary (replicated) data, using ShadowImage on the Sun StorEdge™ 99x0 systems and the VERITAS Volume Manager (VxVM). The article is as generic possible, so that this example can apply to other platforms.

I/O Stack and Data Replication

When dealing with storage systems, it is important to keep in mind the I/O architecture within a server. This I/O architecture, better called the I/O stack, describes how data, stored on a physical media, is accessed by an application. The I/O path corresponds to the software stack an I/O request has to go through. Replication of data consists of copying an I/O request from a primary I/O stack to a secondary one. This replication can occur within different layers of the I/O stack, either local or remote. The following paragraphs briefly describe the I/O layers and their corresponding replication modes.

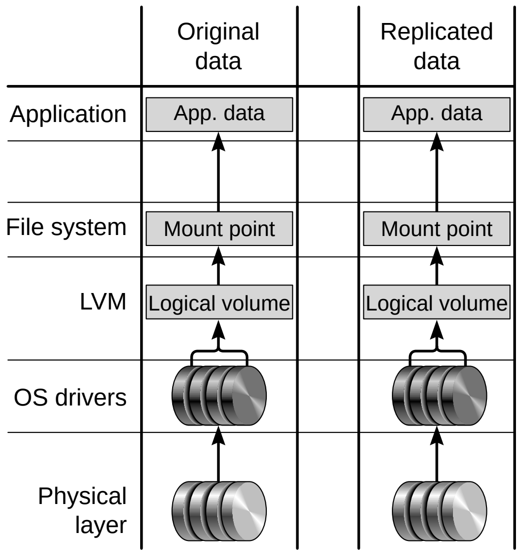

FIGURE 1 I/O Stack Layers

Layer 1: Physical Layer

This I/O layer includes all of the storage hardware components: storage subsystems, the connectivity, disk drives, and storage area network (SAN) switches. When the replication occurs at this level, it is called hardware replication, and it is handled by a physical component such as a disk array (Sun StorEdge 99x0 systems for example) or even a SAN switch (Sun™ PSX-100).

Layer 2: Device Drivers Layer

Device drivers are software components, specific for given hardware. There is a device driver for every hardware device the operating system interacts with (Controller bus, device type, ...). Device drivers make a physical disk drive visible to the OS as a disk device (`/dev/rdsk/c5t`, ...). No replication products operate at the device driver level.

Layer 3: Logical Volume Manager Layer

The role of the logical volume manager (LVM) is to organize and manage disk devices, so as to optimize performance, availability, and capacity of a storage space. This software layer determines which physical device an I/O request goes to. This decision is handled by different algorithms such as RAID-1, RAID-5, and so on. This layer is very complex and, in many cases, it is optional. Different products on the market offer a LVM-based replication type. This includes the Sun StorEdge™ Availability Suite Software and the VERITAS Replicator.

Layer 4: File System Layer

On top of a logical volume is the file system. This software layer provides an interface to block addressing. Indeed, in order to store a data, it is necessary to determine a suitable location on the media. The file system translates a block address to a logical address, such as a file or directory. The file system is also responsible for managing the storage space by offering different services such as metadata, error checking, and clean-up. Products that offer replication at this I/O layer include the Solaris™ Operating System's built-in `filesync` and GNU `rsync`.

Layer 5: Application Layer

This layer corresponds to the application stack running on a given server. Replication is sometimes handled directly by the application. ORACLE products and Lotus Notes include replication mechanisms for their own data.

Hardware Replication

Hardware replication, which is the replication method addressed in this article, is executed by a physical component involved in the storage system: the storage array. An enterprise class storage array holds firmware that contains codes to handle replication. Every time an I/O occurs on the primary disk, the modification is instantly copied onto the secondary disk. The storage array monitors every disk for modifications, and modifies the secondary disk accordingly. Consequently, the secondary disk is identical to the primary disk—it is cloned.

When an application accesses its data repository, it accesses, in fact, several physical disks located in the storage array. An important function of a storage array is to regroup those physical disk drives under one logical unit number (LUN). Through the software layers, an application reads its data from and writes its data to this LUN. In other words, a single I/O at the OS level implies several physical I/Os at the disk-drive level. These physical I/Os are copied onto the secondary disks during replication.

As stated previously, the physical layer is independent of the OS. Therefore, this replication is almost invisible to the OS and is controlled from physical components only. In a storage architecture, different components can control the replication: SAN switches (such as the Sun PSX-1000 switch) and disk subsystems (Sun StorEdge SE99x0 systems). This article concentrates on the disk subsystem-based replication, as this is commonly used in large storage environments. Although it is an attractive way of replicating, it brings on a number of challenges. The main challenge is: how can an application access the replicated data reliably?

Hardware Replication Problems

Hardware replication exposes a number of challenges when accessing replicated data. Accessing replicated data means that its I/O stack must be completely reconstructed. Indeed, because the hardware replication happens at a low level in the I/O stack, each I/O stack layer must be correctly created:

- Physical layer—ensure the replicated disks are consistent and accessible.
- Driver layer—detect the replicated disks.
- LVM layer—reconstruct the replicated logical groups and volumes.
- File system layer—make the replicated file system consistent.
- Application layer—make the data ready for the application.

A certain number of difficulties might occur in each one of these layers, as we described below.

Physical Layer

Consistency groups—as described earlier, a single I/O generated by an application is translated into several I/Os at the disk level. It is important that the storage array maintains the coherence of these multiple I/Os, in order to have a consistent replicated disk. Enterprise-class storage systems use the notion of consistency groups to insure such coherence and correct write ordering.

Splitting the pairs—before accessing the cloned disks, the replication must be suspended (split). This suspension can only be done when the primary and original disks are fully synchronized. The replicated disks will be accessible once the split is totally completed.

Driver Layer

Drivers are the software components that link the hardware to the OS. At this stage, no replication is taking place at this level. Therefore, the drivers must be at the correct levels and configured for the hardware. The goal is to correctly detect and access the cloned disks drives.

LVM Layer

When doing hardware replication on a logical volume, the entire content of the physical disks is cloned. This includes the configuration section (called the private region in VERITAS, or `metadb` in the Solaris™ Volume Manager (SVM), and the data section (also called the public region). However, this private region (or `metadb`) holds disk identifications parameters. Therefore, the cloned disks and original disks have the same ID. This is not a major issue if the replicated data is to be accessed on two different hosts, but it can be a difficult issue to solve if you want to access the replicated data on the same host.

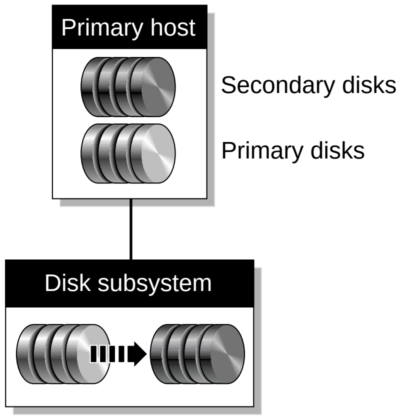

FIGURE 2 One-Host Configuration

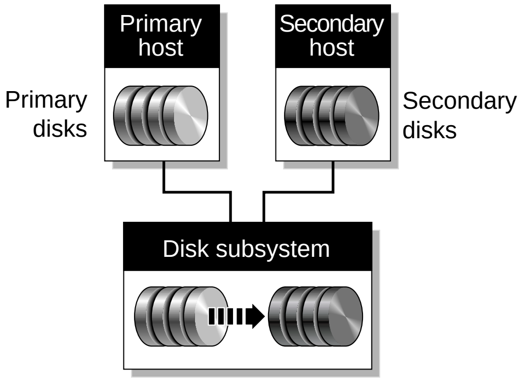

FIGURE 3 Two-Host Configuration

Replicated Data On a Different Host

Accessing the replicated data on a secondary host is equivalent to importing the logical group (or metaset) that contains the logical volumes you want to access. However, because the disks are cloned, the volume manager on the secondary host will believe this logical group is already imported on the primary host. This information is stored in the private region. It is necessary to clean up this information on every replicated disk (`clearimpor` under VxVM). It is then possible to import the replicated logical group, and access its volumes.

Replicated Data On the Same Host

In this case the situation becomes challenging. As described previously disk cloning implies duplication of diskid. If not properly supported, the LVM can get confused and, in the worst case scenario, this can lead to silent data corruption. The method described in this article to access the replicated volumes is:

- Re-create a new logical group and populate it with the cloned disks.
- Re-create every logical volume on this new disk group, using the configuration of the primary group.

File System Layer

Once the LVM layer is correctly done, you must avoid reformatting the volumes. It is necessary, however, to check the file systems for any corruption. Indeed, if a crash happened on the primary host, a file system corruption might happen, especially if the crash occurred during creation of a large file. This is why it is strongly advised that you use journalized file systems such as UNIX File System (UFS) with logging enabled, or the VERITAS File System (VxFS).

Application Layer

Finally, the application can use the replicated data. A final problem can occur, depending on the application. For example, if the application uses host-specific values (such as the node name, IP address, and mount points) you must reconfigure the application with the new values.

Example 1: Logical Device With SVM

To illustrate what has been presented, this section describes a method to access a replicated set of data, created using an enterprise class subsystem: the StorEdge SE99x0 systems. ShadowImage is used to replicate the LUNs. The data to access is configured as a metadevice using SVM patched at the latest level. The primary metadevice is called `d101` and is a soft partition created on top of metadevice `d100`: RAID-0 of four SE99x0 LUNs. All of these metadevices are part of the metaset `labset`.

In this example, the primary and secondary volumes (metadevices) are accessed from two different hosts (the primary host is `storage10` and the secondary host is `storage26`). In this situation, the primary host has access to the primary LUNs only, and secondary host sees only the secondary LUNs. This constraint forces you to reconstruct the metaset and metadevices on the secondary site before accessing the data. There is no possibility of importing or exporting the metaset from one host to the other (`take` and `release` ownership of a metaset implies that every disk is visible on both hosts).

SVM metaset	labset
SVM metadevice	`d100`, RAID 0 of 4 LUNs `d101`, softpartition on top of `d100`

Primary disks:
LUNS visible from `storage103`
	`c8t500060E8000000000000ED1600000200d0`
	`c8t500060E8000000000000ED1600000201d0`
	`c8t500060E8000000000000ED1600000202d0`
	`c8t500060E8000000000000ED1600000203d0`

Secondary disks:
LUNS visible from `storage26`
	`c3t500060E8000000000000ED160000020Ad0`
	`c3t500060E8000000000000ED160000020Bd0`
	`c3t500060E8000000000000ED160000020Cd0`
	`c3t500060E8000000000000ED160000020Dd0`

ShadowImage is set up so that LUNs are paired, under the consistency group LAB, as follows:

Primary Disk -->	Secondary Disk
`c8t500060E8000000000000ED1600000200d0 -->`	`c3t500060E8000000000000ED160000020Ad0`
`c8t500060E8000000000000ED1600000201d0 -->`	`c3t500060E8000000000000ED160000020Bd0`
`c8t500060E8000000000000ED1600000202d0 -->`	`c3t500060E8000000000000ED160000020Cd0`
`c8t500060E8000000000000ED1600000203d0 -->`	`c3t500060E8000000000000ED160000020Dd0`

As described in the previous section, to access the replicated data, you must insure that every layer of the I/O stack is correctly set up. In this example, the steps would be divided as follows:.

- Physical layer—ensure the replicated disks are consistent and accessible.
- Driver layer—detect the replicated disks.
- LVM Layer—reconstruct the replicated metasets and metadevices.
- File system layer—make the replicated file system consistent.
- Application layer—make the data ready for the application.

▼ To Ensure Consistent and Accessible Replicated Disks in the Physical Layer

- **Suspend the replication:**

  Before accessing the replicated LUNs, stop (suspend) the replication and make sure every LUN is in a suspended (`psus`) state:

```
root@storage103 # pairsplit -g LAB
root@storage103 # pairevtwait -g LAB -s psus -t 1800
```

Of course, `pairspli` must be issued when all the LUNs are already synchronized (state PAIR). Failing to do so will result in corrupted secondary devices

▼ To Detect the Replicated Disks in the Driver Layer

- **Scan the disks and verify that they are all visible and accessible from the secondary host.**

  This is achieved using the Solaris OS command `devfsadm`, which scans I/O buses for new devices and reads the partition table of each disk:

```
root@storage26 # devfsadm
```

▼ To Reconstruct the Replicated Metasets and Metadevices in the LVM Layer

Modify the primary metaset configuration to reflect the new devices, and apply the modified configuration to a newly created metaset.

1. **Create a metaset on secondary host:**

```
root@storage26 # metaset -s labset -a -h storage26
```

2. **Populate the new metaset with cloned disks:**

```
root@storage26 # metaset -s labset -a \
c3t500060E8000000000000ED160000020Ad0\
c3t500060E8000000000000ED160000020Bd0 \
c3t500060E8000000000000ED160000020Cd0 \
c3t500060E8000000000000ED160000020Dd0 \
```

3. **Create new configuration for the secondary metaset.**

   Start by obtaining the metadevice configuration of the primary host:

```
root@storage103 # metaset -s labset -p
labset/d101 -p labset/d100 -o 1 -b 10485760
labset/d100 1 4 c8t500060E8000000000000ED1600000200d0s0 \
c8t500060E8000000000000ED1600000201d0s0 \
c8t500060E8000000000000ED1600000202d0s0 \
c8t500060E8000000000000ED1600000203d0s0 -i 32b
```

4. **On the secondary host, create a metadevice configuration file called** `/etc/lvm/md.tab` **containing the previous output with the correct secondary LUNs.**

   The order of appearance must be respected:

```
root@storage26 # cat /etc/lvm/md.tab
labset/d101 -p labset/d100 -o 1 -b 10485760
labset/d100 1 4 c3t500060E8000000000000ED160000020Ad0s0 \
c3t500060E8000000000000ED160000020Bd0s0 \
c3t500060E8000000000000ED160000020Cd0s0 \
c3t500060E8000000000000ED160000020Dd0s0 -i 32b
```

5. **Apply the metadevice configuration file to the replicated host:**

```
root@storage26 # metainit -s labset -a
labset/d100: Concat/Stripe is setup
labset/d101: Soft Partition is setup
root@storage26 #
```

▼ To Make the Replicated File System Consistent in the File System Layer

1. **Check the consistency of the file system:**

```
root@storage26 # fsck /dev/md/labset/rdsk/d101
```

The `fsck` lists the corrupted files. Action must be taken to recover them. This operation makes sense in case of a crash. During a crash, some files might be corrupted (files in creation and modification mode).

2. **Mount the file system:**

```
root@storage26 # mount /dev/md/labset/dsk/d101 /mnt/LAB
```

▼ To Make the Data Ready for the Application in the Application Layer

At this stage, you can consider the replicated data accessible. Some application specific actions might take place, such as modifying configuration files, links, or other clean up and recover processes.

Example 2: Logical Device With VxVM

To illustrate what has been presented, the following paragraphs describe a method to access a replicated set of data created using an enterprise class subsystem: the StorEdge SE99x0 systems. ShadowImage is used to replicate the LUNs. The data to access is configured as a volume, using VERITAS Volume Manager 3.5 patched at the latest level.

The primary volume is called `labvol`. It is part of the disk group `labdg`, and is a RAID 0 stripe of five LUNs:

VxVM disk group	`labdg`
VxVM volumes	`labvol`, RAID 0

VxVM device name (LUNS)	HDS99100_0 HDS99100_1 HDS99100_2 HDS99100_3 HDS99100_4

This example covers the trickiest situation, accessing the primary and secondary data from the same host. In this situation, the primary and secondary LUNs are both visible from the same host. Because they are physically different, VxVM assigns them distinct device names:

VxVM device name	VxVM disk name
HDS99100_0	9910_0
HDS99100_1	9910_1
HDS99100_2	9910_2
HDS99100_3	9910_3
HDS99100_4	9910_4
HDS99100_10	9910_0_S
HDS99100_11	9910_1_S
HDS99100_12	9910_2_S
HDS99100_13	9910_3_S
HDS99100_14	9910_4_S

ShadowImage is set up so that LUNs are paired, under the consistency group LAB, as follows:

Primary Disk --> Secondary Disk
HDS99100_0 --> HDS99100_10
HDS99100_1 --> HDS99100_11
HDS99100_2 --> HDS99100_12
HDS99100_3 --> HDS99100_13
HDS99100_4 --> HDS99100_14

As described in the previous section, to access the replicated data, you must insure that every layer of the I/O stack is correctly set up. In this example, the steps are:

▼ To Ensure Consistent and Accessible Replicated Disks in the Physical Layer

- **Suspend the replication:**

  Before accessing the replicated LUNs, stop (suspend) the replication and make sure every LUN is in a suspended (psus) state:

```
root@storage103 # pairsplit -g LAB
root@storage103 # pairevtwait -g LAB -s psus -t 1800
```

Of course, pairsplit must be issued when all the LUNs are already synchronized (state PAIR). Failing to do so will result in corrupted secondary devices.

▼ To Detect the Replicated Disks in the Driver Layer

- **Scan the disks and verify that they are all visible and accessible from the secondary host.**

  This is achieved using the VxVM command, vxdiskconfig, which scans the I/O buses for new devices and reads the private region of every disk:

```
root@storage103 #  vxdiskconfig
```

At this stage, the private region of the cloned disks has the same entries as the primary disks, in particular for the *diskname* and *diskid* parameters.

▼ To Reconstruct the Replicated Logical Groups and Volumes in the LVM Layer

Modify the primary disk group configuration to reflect the new devices, and apply the modified configuration to a newly created disk group.

1. **Save the disk group configuration of the primary data:**

```
root@storage103 # vxprint -g labdg -m > labdg.vxprint
```

This command dumps the whole configuration of the disk group `labdg` into a file. This file contains more information than necessary. The information of concern is subdisks, plexes, and volumes, which can be extracted by running:

```
root@storage103 # i=`grep -n “sd “ labdg.vxprint | cut -d: -f1 | head -1`
root@storage103 # more +$i labdg.vxprint > labdg_S.vxprint.tmp
root@storage103 # mv labdg_S.vxprint.tmp labdg_S.vxprint
```

The file `labdg_S.vxprint` is the file to push in the disk group configuration of the replicated disks to re-create the volumes.

2. **Clean the cloned disk to re-initialize the disk ID of each cloned disk:**

```
root@storage103 # vxdisksetup -f -i \
9910_0_S=HDS9910_10 \
9910_1_S=HDS9910_11 \
9910_2_S=HDS9910_12 \
9910_3_S=HDS9910_13 \
9910_4_S=HDS9910_14
```

3. **Create the new disk group and populate it with the cloned disks:**

```
root@storage103 # vxdg init labdg_S 9910_0_S 9910_1_S 9910_2_S 9910_3_S 9910_4_S
```

4. **Modify** `labdg_S.vxprint` **to reflect the disk configuration of the replicated disk group by replacing every occurrence of the primary** *diskname* **and** *device name* **by the corresponding** *diskname* **and** *device name* **of the secondary disks.**

   a. **To ease this process, first create a file containing the device name of primary and secondary disks:**

```
root@storage103 # cat paires.txt
HDS99100_0 HDS99100_10
HDS99100_1 HDS99100_11
HDS99100_2 HDS99100_12
HDS99100_3 HDS99100_13
HDS99100_4 HDS99100_14
```

b. The string replacement is done by using the following Bourne Shell loop:

```
root@storage103 # while read pdev sdev
do
 pname=`vxdisk list | grep -w $p| awk ‘{print $3}’`
 sname=`vxdisk list | grep -w $s| awk ‘{print $3}’`
 cat labdg_S.vxprint | sed -e “s/$pname/$sname/g” -e “s/$pdev/$sdev/g”\
   > labdg_S.vxprint.tmp
mv labdg_S.vxprint.tmp labdg_S.vxprint
done < paires.txt
```

At this stage, the file `labdg_S.vxprint` reflects the configuration of the new disk group.

5. **Apply the volume configuration file to the new disk group:**

```
root@storage103 # vxmake -g labdg_S -d labdg_S.vxprint
```

6. **Start the new disk group:**

```
# vxvol -g labdg_S init active labvol1
```

▼ To Make the Replicated File System Consistent in the File System Layer

1. **Check consistency of the file system:**

```
root@storage103 # fsck /dev/vx/rdsk/labdg_S/labvol1
```

The `fsck` lists the corrupted files. Action must be taken to recover them. This operation makes sense in case of a crash. During a crash, some files might be corrupted (files in creation and modification mode).

2. **Mount the file system:**

```
root@storage103 # mount /dev/vx/rdsk/labdg_S/labvol1 /mnt/LAB_S
```

▼ To Make the Data Ready for the Application in the Application Layer

At this stage, you can consider the replicated data accessible. Some application specific actions might take place, such as modifying configuration files, links, or other cleanup or recover processes.

Conclusion

The hardware replication acts exclusively at the physical layer of the I/O stack. It offers advantages such as speed and platform independence. However, because this replication occurs at a low level in the I/O stack, the real work must be done at the OS level to fully access the replicated data. This can be challenging and many IT infrastructures consider a simple approach: removing the LVM layer. Using this approach, the major complexity of accessing the data disappears. However, to have performance file systems (as needed for advanced applications, databases, SAP...), system administrators require the use of LVM, partly because the LUN management system embedded in storage arrays is not sufficient for I/O intensive applications. The method described in this article shows it is possible to take advantage of a LVM in the hardware replication infrastructure.

About the Author

Selim Daoud is a recognized leader in data storage and backup technologies for open systems. He obtained an MSc in Computer Science at the University of Wales (U.K.), and an MSc in Applied Mathematics in computing at Toulouse University in France.

Over the course of his career in the computer industry, Selim gained valuable experience working with data backup technology, storage system design (mainly RAID implementations), and UNIX system administration. He managed a support organization (dealing with storage and backup technology) in London, served as a consultant specializing in backup systems in Paris, and was in charge of multiple migrations of backup systems and storage deployment for the European Organization for Nuclear Research (CERN) in Geneva, Switzerland.

Selim currently holds a project engineering position, specializing in computer storage and backup technology in the Sun Professional Services organization in Switzerland.

Ordering Sun Documents

The SunDocsSM program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

Accessing Sun Documentation Online

The docs.sun.com web site enables you to access Sun technical documentation online. You can browse the docs.sun.com archive or search for a specific book title or subject. The URL is http://docs.sun.com/. To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at:

http://www.sun.com/blueprints/online.html.