LVM Version 2.0 Volume Groups in HP-UX 11i v3

Abstract

This whitepaper is an introduction to the LVM volume groups versions 2.0 and 2.1. The 2.0 volume groups became available with the March 2008 release of HP-UX 11i v3 (11.31) and the 2.1 volume groups with the 11i v3 2008 September release. Prior to the March 2008 release, only version 1.0 volume groups were available. In this document, 2.*x* denotes both 2.0 and 2.1 volume groups.

Audience

The document is intended for system administrators, operators, and customers who want to use and know about the LVM 2.*x* volume groups. It is assumed that the reader has a basic knowledge of LVM.

Introduction

LVM and MirrorDisk/UX now support 1.0 and 2.*x* volume groups. Version 1.0 is the version supported on all current and previous versions of HP-UX 11i. The procedures and command syntax for managing Version 1.0 volume groups are unchanged from previous releases, except for the enhancements described in this paper. When creating a new volume group, *vgcreate* defaults to Version 1.0.

What is a 2.*x* Volume Group?

A 2.*x* volume group is a volume group whose metadata layout is different from the one used for 1.0 volume groups. A 2.*x* volume group extends the limits of a 1.0 volume group.

How do I use a 2.*x* Volume Group?

A 2.*x* volume group is managed the same way as a 1.0 volume group using the same user interface. This interface is the same as in the HP-UX 11i v3 (11.31) initial release, but has been extended for 2.*x* version volume groups. These additions are listed in the "LVM Changes in the March 2008 Release of HP-UX 11i v3 " section.

What do I need to change to manage my 1.0 Volume Groups?

Nothing, unless you have scripts that parse the output of *lvdisplay*, *vgdisplay*, *pvdisplay*, or *vgscan*. The output of these commands is slightly changed and might impact parsing. The user interface to manage the 1.0 volume group did not change. The same LVM commands with the same options (as in HP-UX 11i v3 (11.31) initial release) are used to handle 1.0 volume groups. There are additional options that can be used on 1.0 volume groups.

New Limits

The limits remain the same for 1.0 volume groups. The following table illustrates the new upper limits with 2.*x* volume groups.

	Max Supported 1.0 volume groups limits	Max Supported 2.0 volume groups limits	Max Supported 2.1 volume groups limits
VG size	510 TB	2 PB	2 PB
LV size	2^{46} (64 TB) 16 TB supported	2^{48} (256 TB)	2^{48} (256 TB)
PV size	2^{41} (2 TB)	2^{44} (16 TB)	2^{44} (16 TB)
Number of volume groups	256	512	2048
Number of logical volumes per VG	255	511	2047
Number of physical volumes per vg	255	511	2047
Number of mirrors copies	2	5	5
Number of extents per volume group	2^{16} (64K)	2^{25} (32M)	2^{25} (32M)
Extent size	1 to 256MB	1 to 256 MB	1 to 256 MB
Stripe width	255	511	511

LVM Changes in the March 2008 Release of HP-UX 11i v3

Creation of a volume group

Automatic creation of the volume group directory and group file is available starting with the March 2008 release of HP-UX 11i. It is available for 1.0 and 2.x volume groups. The following examples show volume group creation using the legacy method (*mkdir* and *mknod*) and examples of automatic creation.

Creating a 1.0 volume group

To avoid manually creating the volume group directory and the group file, use *vgcreate*. In this case, *vgcreate* automatically creates the directory and group file if they do not exist for this volume group.

Example

```
# vgcreate -s 8 -l 3 -p 16 -e 63535 /dev/vg01 /dev/dsk/c3t4d0
```

To select a particular volume group number or if you want to create the volume group the same way as in releases before March 2008, first create the volume group directory and the group file.

Example

```
# mkdir /dev/vg01
# mknod /dev/vg01/group c 64 0x010000
# vgcreate -s 8 -l 3 -p 16 -e 63535 /dev/vg01 /dev/dsk/c3t4d0
```

Creating a 2.*x* volume group

- As with 1.0 volume groups, for 2.*x* volume groups, you can manually create the volume group directory and group file, or use *vgcreate* to automatically create them if they do not already exist.
- A new –V option specifies the version of the volume group to create. To create a 2.*x* volume group, specify "–V 2.0" or "–V 2.1" as an option on the command line. Version 1.0 volume groups are the default and –V 1.0 is also valid to specify a 1.0 version volume group.
- A new –S option must be used to create a 2.*x* volume group. It specifies the maximum size the volume group may reach. LVM uses this size to reserve enough space on disk to accommodate the metadata for a volume group of that size. Note this does not need to be the actual size of the volume group but how large the volume group can grow over time. The "How to Provision a 2.*x* Volume Group" section discusses provisioning strategies and tradeoffs.
- For a 2.*x* volume group, if the sum of the physical volume sizes passed on the command line is greater than the maximum size specified with the –S option, *vgcreate* automatically increases the maximum size of the volume group to the sum of the physical volumes sizes and displays an informational message.
- The maximum size specified requires a unit qualifier. The units are in Megabytes (2^{20}), Gigabytes (2^{30}), Terabytes(2^{40}), and Petabytes(2^{50}), represented by m, g, t, and p, respectively, on the command line
- A new –E option displays which volume group size can be reached for a given extent size, or which minimum extent size must be used for a given volume group size. Use this to determine the best fit of volume group size and physical extent size.
- The –s option (to specify the extent size) is mandatory for 2.*x* volume groups.
- The –e, –l, –p, and –f options are invalid when applied to a 2.*x* volume group. If used for a 2.*x* volume group, *vgcreate* fails.
  –e max_pe is not needed for 2.*x* volume groups because the 2.*x* metadata format is provisioned so that the number of physical extents can always grow to the maximum size specified with –S.

–l max_lv and –p max_pv are not needed for 2.*x* volume groups because any 2.*x* volume group is provisioned to handle the maximum supported number of logical volumes and physical volumes.
–f is not needed for 2.*x* volume groups because bad block handling is handled by all currently supported physical disk drives.

Example
To create a 2.0 volume group using 32 MB extents and a maximum size is 1 PB, enter:

```
# vgcreate -V 2.0 -s 32 -S 1p /dev/vg01 /dev/disk/disk50
```

Example
To create a 2.0 volume group using 4 MB extents and a maximum size is 8 TB, enter:

```
# mkdir /dev/vg01
# mknod /dev/vg01/group c 128 0x001000
# vgcreate -V 2.0 -s 4 -S 8t /dev/vg01 /dev/disk/disk49
```

Display of LVM Information

With the introduction of the new volume group version, the display commands display additional information. In addition, because 2.*x* volume groups support increased sizes, some of the display information can contain larger numbers than previous versions.

This section illustrates the difference for each display operation. As a general practice, when writing scripts to gather volume manager information, use the –F option introduced in 11i v3 to reduce the impact of future volume manager changes.

vgdisplay

In the March 2008 release of HP-UX 11iV3, the *vgdisplay* command displays three more lines than *vgdisplay* from the HP-UX 11i v3 initial release. In addition, the displayed number of physical extents might be larger on 2.*x* volume groups.

The new lines appear for both 1.0 and 2.*x* volume groups:

- The volume group version, as "VG version".
- The maximum size of the volume group, as "VG Max Size".
- The maximum number of physical extents the volume group can contain, as "VG Max Extents". This value is the ratio of volume group maximum size to extent size.

Example

```
# vgdisplay vgtestl2
--- Volume groups ---
VG Name                     /dev/vgtestl2
VG Write Access             read/write
VG Status                   available
Max LV                      511
Cur LV                      1
Open LV                     1
Max PV                      511
Cur PV                      70
Act PV                      70
Max PE per PV               2097152
VGDA                        140
PE Size (Mbytes)            8
Total PE                    1123123
Alloc PE                    40
Free PE                     1123083
Total PVG                   0
Total Spare PVs             0
Total Spare PVs in use      0
VG Version                  2.0
VG Max Size                 16t
VG Max Extents              2097152
```

} New lines starting with the March 2008 release

pvdisplay

In the March 2008 release of HP-UX 11iV3, the *pvdisplay* command displays one more line than *pvdisplay* from the HP-UX 11i v3 initial release.

When the –d option is used with *pvdisplay*, the new line appears for both 1.0 and 2.*x* volume groups. The new line is "Data End". "Data Start" and "Data End" are block numbers from the start of the disk. In this display, a block is always1024 bytes (no matter what the disk sector size is). "Data End" is the block number of the last block on the disk that may be used by LVM to store user data.

LVM only uses storage in multiples of extent size. In some cases, "Data End" – "Data Start" might not be a multiple of the extent size. In this case, the space between the last extent and the "Data End" does not contain any user data.

Example

```
# pvdisplay -d /dev/disk/disk148
--- Physical volumes ---
PV Name                     /dev/disk/disk148
VG Name                     /dev/vgtestl2
PV Status                   available
Allocatable                 yes
VGDA                        2
Cur LV                      0
PE Size (Mbytes)            8
Total PE                    16401
Free PE                     16401
Allocated PE                0
Stale PE                    0
IO Timeout (Seconds)        default
Autoswitch                  On
Data Start                  1024
Data End                    134358016   }   New line starting with the March 2008 release
Boot Disk                   no
Relocated Blocks            0
Proactive Polling           On
```

For 1.0 volume groups, the output of *lvdisplay* is the same as the HP-UX 11i v3 initial release. For 2.*x* volume groups, *lvdisplay* output changes if the number of mirrors is greater than 2. In that case, when *lvdisplay* –v is used, it displays extent mapping data for the additional mirrors on additional lines. This is illustrated in the following example.

Example

```
# lvdisplay -v  /dev/vgtestl2/lvol1
--- Logical volumes ---
LV Name                     /dev/vgtestl2/lvol1
VG Name                     /dev/vgtestl2
LV Permission               read/write
LV Status                   available/syncd
Mirror copies               3
Consistency Recovery        MWC
Schedule                    parallel
LV Size (Mbytes)            80
Current LE                  10
Allocated PE                40
Stripes                     0
Stripe Size (Kbytes)        0
Bad block                   NONE
Allocation                  strict
IO Timeout (Seconds)        default

   --- Distribution of logical volume ---
   PV Name                 LE on PV  PE on PV
   /dev/disk/disk79        10        10
   /dev/disk/disk80        10        10
   /dev/disk/disk81        10        10
   /dev/disk/disk82        10        10
   --- Logical extents ---
   LE        PV1                     PE1      Status 1 PV2                     PE2
Status 2 PV3                     PE3      Status 3
   LE        PV4                     PE4      Status 4
   000000000 /dev/disk/disk79        00000000 current  /dev/disk/disk80
00000000 current  /dev/disk/disk81        00000000 current
   000000000 /dev/disk/disk82        00000000 current
   000000001 /dev/disk/disk79        00000001 current  /dev/disk/disk80
00000001 current  /dev/disk/disk81        00000001 current
   000000001 /dev/disk/disk82        00000001 current
   000000002 /dev/disk/disk79        00000002 current  /dev/disk/disk80
00000002 current  /dev/disk/disk81        00000002 current
   000000002 /dev/disk/disk82        00000002 current
   000000003 /dev/disk/disk79        00000003 current  /dev/disk/disk80
00000003 current  /dev/disk/disk81        00000003 current
   000000003 /dev/disk/disk82        00000003 current
   000000004 /dev/disk/disk79        00000004 current  /dev/disk/disk80
00000004 current  /dev/disk/disk81        00000004 current
   000000004 /dev/disk/disk82        00000004 current
   000000005 /dev/disk/disk79        00000005 current  /dev/disk/disk80
00000005 current  /dev/disk/disk81        00000005 current
   000000005 /dev/disk/disk82        00000005 current
.
.
.
.
```

There is another difference in **lvdisplay(1M)**.

For 1.0 volume groups, the order in which the physical extents are displayed by *lvdisplay –v* can change across activation cycles. This is because when a 1.0 volume group is activated, LVM arranges the physical extents of a logical extent (LE) in order of increasing physical volume number.
For example, the display:

```
   --- Logical extents ---
   LE        PV1              PE1    Status 1      PV2            PE2    Status 2
000000003 /dev/disk/disk79 00000003 current  /dev/disk/disk80 00000014 current
```

might change to:

```
   --- Logical extents ---
   LE        PV1              PE1    Status 1      PV2            PE2    Status 2
000000003 /dev/disk/disk80 00000014 current  /dev/disk/disk79 00000003 current
```

after deactivation and activation again.

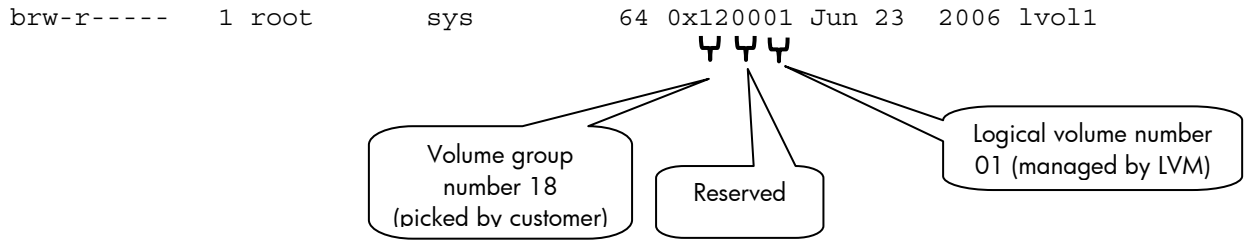
For 2.*x* volume groups, the order in the display does not change across activation.

LVM Device Special Files

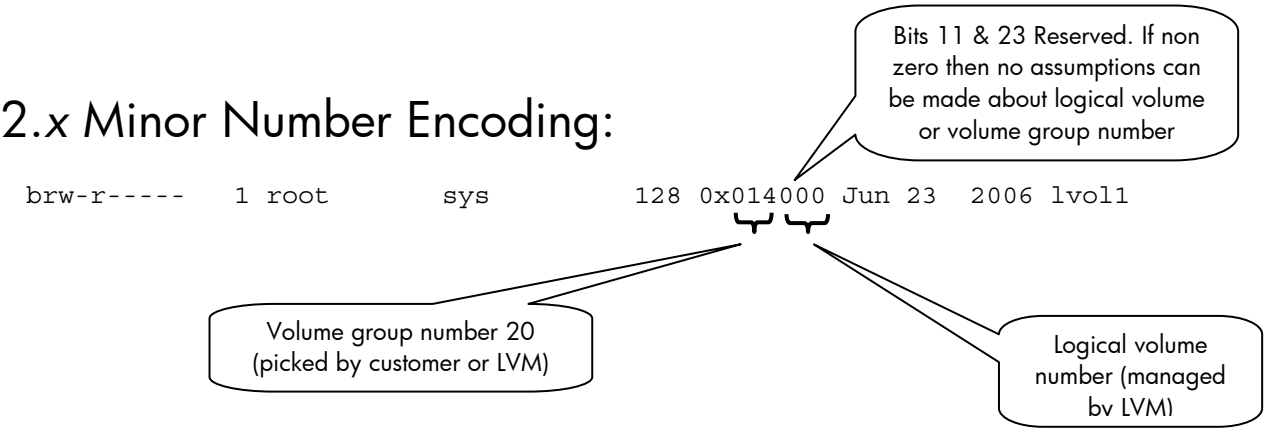
With volume group version 2.*x*, LVM device special files have changed. For version 1.0 volume groups, procedures do not need to change. For 2.*x* version volume groups, there are a few differences shown in the following figures.

- Device special files for Version 1.0 volume groups are unchanged
- Device special files for Version 2.*x* volume groups have a new major number (128)
- Device special files for Version 2.*x* volume groups have a different minor number encoding scheme

1.0 Minor Number Encoding:



2.*x* Minor Number Encoding:



lvmtab, lvmtab_p

LVM has a well known configuration file "/etc/lvmtab". With the introduction of version 2.*x* volume groups, LVM uses a new configuration file.

- /etc/lvmtab is a private binary file. It contains information related to 1.0 volume groups only. It is compatible with all supported HP-UX releases.
- /etc/lvmtab_p is a new private binary file. It contains information related to 2.*x* volume groups only. It has a different internal structure from lvmtab and contains additional information.

Example

```
# strings /etc/lvmtab_p
/dev/vgtestl2
A0000000000000004Sun Mar 16 03:02:19 2007241fa9de-dba3-11da-9cd2-23efa80dc3e7
/dev/disk/disk79
/dev/disk/disk80
/dev/disk/disk81
/dev/disk/disk82
/dev/disk/disk83
/dev/disk/disk84
/dev/disk/disk85
```

New Command (*lvmadm*)

lvmadm (1M)

The *lvmadm* command displays the supported limits for 1.0, 2.0, and 2.1 volume groups. It is not possible to create a volume group that goes beyond these limits.

Example

```
# lvmadm -t
--- LVM Limits ---
VG Version                      1.0
Max VG Size (Tbytes)            510
Max LV Size (Tbytes)            16
Max PV Size (Tbytes)            2
Max VGs                         256
Max LVs                         255
Max PVs                         255
Max Mirrors                     2
Max Stripes                     255
Max Stripe Size (Kbytes)        32768
Max LXs per LV                  65535
Max PXs per PV                  65535
Max Extent Size (Mbytes)        256

VG Version                      2.0
Max VG Size (Tbytes)            2048
Max LV Size (Tbytes)            256
Max PV Size (Tbytes)            16
Max VGs                         512
Max LVs                         511
Max PVs                         511
Max Mirrors                     5
Max Stripes                     511
Max Stripe Size (Kbytes)        262144
Max LXs per LV                  33554432
Max PXs per PV                  16777216
Max Extent Size (Mbytes)        256

VG Version                      2.1
Max VG Size (Tbytes)            2048
Max LV Size (Tbytes)            256
Max PV Size (Tbytes)            16
Max VGs                         2048
Max LVs                         2047
Max PVs                         2048
Max Mirrors                     5
Max Stripes                     511
Max Stripe Size (Kbytes)        262144
Max LXs per LV                  33554432
Max PXs per PV                  16777216
Max Extent Size (Mbytes)        256
```

Summary List of New or Obsolete Commands Options

The following table summarizes the command option changes with 1.0 or 2.*x* volume groups.

"No change" means that the option works as in the HP-UX 11i v3 (11.31) initial release.

"Optional" means that the option is not necessary to create or use the volume group.
"Invalid" means that the LVM command fails.
"Ignored" means that the LVM command ignores the option, but does not automatically fail.
"Obsolete" means that the LVM command ignores the option and displays a warning message, but does not automatically fail.
"New option" means that the option was first delivered in the March 2008 release of HP-UX 11i v3 (11.31).

LVM command	Option	New option	Effect on 1.0 volume group	Effect on 2.*x* volume group
vgcreate	–V	Y	Optional	Mandatory
	–S	Y	Invalid	Mandatory
	–E	Y	Invalid	Optional
	–s	N	No change	Mandatory
	–e	N	No change	Invalid
	–l	N	No change	Invalid
	–p	N	No change	Invalid
	–f	N	No change	Invalid
vgextend	–f	N	No change	Obsolete
	–z y	N	No change	Invalid
	–z n	N	No change	Obsolete
vgcfgrestore	–F	N	No change	Obsolete
	–v	Y	Optional	Optional
vgremove	–X	Y	Optional	Optional
pvchange	–z y	N	No change	Invalid
	–z n	N	No change	Obsolete
lvcreate	–r	N	No change	Ignored
pvcreate	–s	N	No change	Obsolete
lvchange	–r	N	No change	Ignored

Overview of Changes by Command

vgcreate (1M)

- *vgcreate* is changed. It can be used as before to create 1.0 volume groups. To create 2.*x* volume groups, you must use new options. For more information, see "Creation of a volume group" and "How to Provision a 2.x Volume Group".

vgextend (1M)

- The –f option is obsolete on a 2.*x* volume group. If used on a 2.*x* volume group, *vgextend* displays a warning.
- The –z y option is invalid on a 2.*x* volume group. If used on a 2.*x* volume group, *vgextend* fails.
- The –z n option is obsolete on a 2.*x* volume group. If used on a 2.*x* volume group, *vgextend* displays a warning.
- *vgextend* on a 2.*x* volume group fails if the volume group is already at the maximum size specified with *vgcreate*.
- *vgextend* on a 2.*x* volume group displays a warning if only part of the physical volume can be added to the volume group. If adding the whole physical volume results in exceeding the maximum size of the volume group, only a part of the physical volume is added.

- You cannot add bootable disks to a 2.*x* volume group.

vgcfgrestore (1M)

- The –F option is obsolete on a 2.*x* volume group. If used on a 2.*x* volume group, *vgcfgrestore* displays a warning.

vgimport (1M)

- Automatically creates the volume group directory (under /dev) and the group file if they do not already exist. This applies to 1.0 and 2.*x* volume groups.

vgremove (1M)

- For both 1.0 and 2.*x,* by default *vgremove* does not delete the volume group directory and group file. A new –X option has been added to *vgremove*. If used, –X deletes the volume group directory and group file. This option applies to 1.0 and 2.*x* volume groups.

pvchange (1M)

- The –z y option is invalid on a 2.*x* volume group. If used on a 2.*x* volume group, *pvchange* fails.
- The –z n option is obsolete on a 2.*x* volume group. If used on a 2.*x* volume group, *pvchange* displays a warning.

lvcreate (1M)

- Larger limits are allowed on 2.*x* logical volumes.
- The –r option is ignored on 2.*x* logical volumes.

lvextend (1M), lvreduce (1M)

- Larger limits are allowed on 2.*x* logical volumes.

pvcreate (1M)

- The –s option is obsolete on a 2.*x* volume group. When a PV that was created with –s is added to a 2.*x* volume group, *pvcreate* displays a warning and the –s is ignored.
- You cannot add bootable disks to a 2.*x* volume group.

lvchange (1M)

- The –r option is obsolete on a 2.*x* volume group. If used on a 2.*x* volume group, *lvchange* displays a warning.

vgdisplay (1M)

- Change in the display. See "Summary List of New or Obsolete Commands Options".

pvdisplay(1M)

- Change in the display. See "Summary List of New or Obsolete Commands Options".

lvdisplay(1M)

- Change in the display if using more than two mirrors. See “Summary List of New or Obsolete Commands Options”.

Other Differences Between Volume Group Versions 1.0 and 2.*x*

Sparing not supported on 2.*x* volume groups

The sparing function available on 1.0 volume groups is not available on 2.*x* volume groups.

Boot, dump and primary swap not supported on 2.*x*

A 2.*x* volume group cannot be the root volume group nor can it contain a boot disk. As a consequence, Ignite/UX does not allow the creation of a 2.*x* root volume group during a cold install and the *lvrmboot* and *lvlnboot* commands do not apply to 2.*x* volume groups.

A 2.*x* volume group cannot be used to save crashdumps. The *crashconf* command displays an error if used on a version 2.*x* volume group. For example:

```
# crashconf -s /dev/vgtest2_l2/lvol1
/dev/vgtest2_l2/lvol1: error: unsupported disk layout
warning: All dump devices may not be marked persistent
```

A 2.*x* volume cannot be used for primary swap, but can be used for secondary swap. The *swapon* command displays an error if used on a version 2.*x* volume group. For example:

```
# swapon -s /dev/vgtest2_l2/rlvol1
swapon: /dev/vgtest2_l2/lvol1: Invalid argument
```

Bad blocks not supported on 2.*x*

Version 2.*x* volume groups do not perform bad block relocation in software because modern disks and disk arrays handle such relocation in their own hardware.

Commands not supported on 2.*x* volume groups

vgmodify (1M), lvrmboot(1M), lvlnboot(1M), pvck(1M)

How to Provision a 2.*x* Volume Group

In the context of this document “provisioning” is the set of parameters used at the time of the volume group creation (*vgcreate*).

Provisioning is defined as how the volume manager reserves space on disk for future growth. Getting the provisioning correct is important because it is a trade off between ease of future growth and efficient disk space usage.

Simplified Provisioning

With 1.0 version volume groups, you provision the volume group with three parameters: max PVs, max extents, and max LVs. Furthermore, 1.0 volume groups keep their disk metadata within one physical extent. Both of these factors make it challenging to configure 1.0 volume groups can easily grow over time. With 2.*x* version volume groups, you provide only one maximum when provisioning. This is the maximum size of the volume group (new –S option). The size entered with –S is the size of the user data. LVM guarantees that for any 2.*x* volume group, you can later add physical volumes and logical volumes up to the 2.*x* supported limits. In addition, LVM disk metadata can be larger than one physical extent, thus giving much more flexibility on how to configure and provision a volume group.

To make room for LVM metadata, the actual size of the volume group is larger than what is specified with –S. For a 2.*x* volume group, you do not need to consider if the default maximum number of extents per physical volumes, the default maximum number of logical volumes, or the default maximum number physical volumes are sufficient. LVM manages this automatically.

Tip: When thinking about your volume group needs, consider how fast your storage requirements have grown over time. Estimate how fast and how long particular volume groups will exist. For example, if you have a database that doubles in size every two years, and you expect the application environment to last ten years, provisioning five times the current amount is a good starting point.

Example

- To create a 2.0 volume group provisioned for 1 petabyte, enter:

```
# vgcreate -V 2.0 -s 32 -S 1p myvg /dev/disk/disk149
```

There is a relationship between the maximum number of extents and the extent size when selecting the maximum volume group size. To help in selecting the extent size, you can preview it with *vgcreate* extent size or maximum volume group size.

After you know the volume group size you want to provision for, you can determine the minimum extent size required to achieve it. For that, use *vgcreate* with the –E option.

Example

- What is the minimum extent size to provision a volume group for 1 petabyte?

```
# vgcreate -V 2.0 -E -S 1p
Max_VG_size=1p:extent_size=32m
```

The maximum size for a volume group is displayed with *vgdisplay* in the "VG Max Size" field.

What is the Disadvantage of Over Provisioning?

The cost of over provisioning is reduced compared to 1.0 volume groups. It costs only disk space; it does not cost system memory. When a large 2.*x* volume group is provisioned, enough disk space is reserved for the metadata to handle any LVM configuration for a volume group of that size.

However, system memory is different. When this volume group is created or activated, the memory allocated is based on the extents allocated to logical volumes and not on the maximum size of the volume group. This is different from 1.0 version volume groups, in that LVM also allocates memory to match the provisioning on disk.

In other words, the system memory used by a 2.*x* volume group depends on how much of the volume group is used. A volume group generously provisioned uses little memory as long as not many extents are allocated to logical volumes. The memory usage grows roughly in proportion to the extents allocated to logical volumes.

As a consequence, the user can provision for very large volume groups without wasting system memory.

Metadata size on disk versus maximum volume group size

Figure 1 illustrates how much space is reserved on disk based on extent size and maximum volume group size.

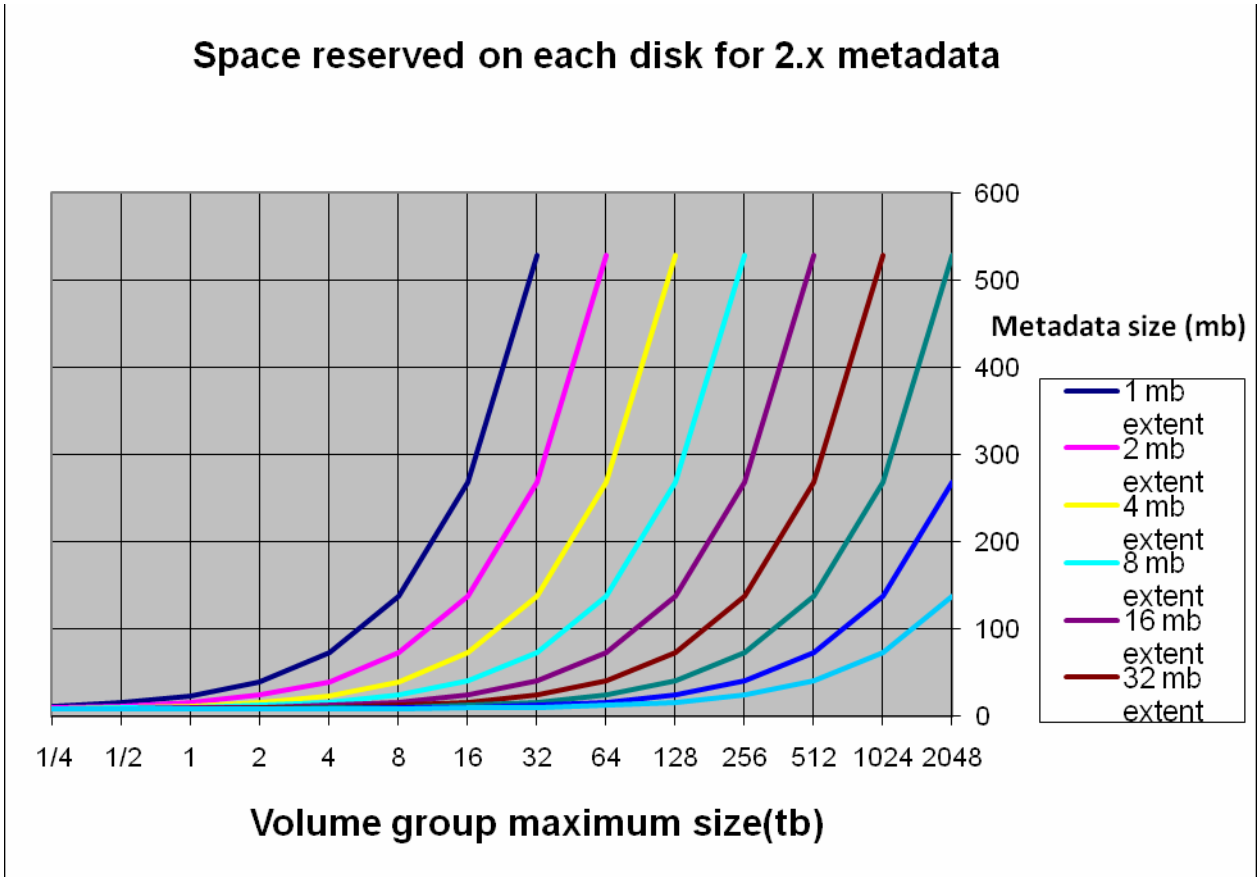


Figure 1

Notes:

- All lines stop at 518 MB on the Y axis because of the limit of 32 million extents per volume group.

What are the Advantages of Over Provisioning with 2.*x*?

Over provisioning does not increase the memory footprint of the volume group. Over provisioning guarantees you can grow the volume group up to its maximum size without having to create a second bigger volume group and copy the data over.

2.*x* can leave as much space for user data on disk as 1.0

For an equivalent provisioning (same extent size and same maximum volume group size), the size of the metadata on disk is larger for 2.*x* compared to 1.0. That is because the metadata is provisioned for more logical volumes, physical volumes, more extents per logical volume (or physical volume), and contains additional information such as the volume names.

However, because of the different structures of the 2.*x* metadata, 2.*x* can leave as much space for user data as 1.0.

2.*x* volume groups use the space at the end of the disk that cannot be used for user data. Often, the end of the last extent does not coincide with the end of the disk. 2.*x* volume groups try to use this left over space between the end of the last extent and the end of the disk to store metadata. As a consequence, even if the on disk 2.*x* metadata is bigger, the space available on disk for user data can be as large for 2.*x* compared to 1.0.

Example

A 1.0 volume group (vgtestl1) and a 2.0 volume group (vgtestl2) are created with the same provisioning (64 MB extent and maximum volume group size of 128 TB). They are created on disks of identical size to simplify the comparison. While the metadata of the 2.0 volume group is bigger, the space available for user data in each volume group is the same: 2050 extents (from *pvdisplay* output).

```
# diskinfo /dev/rdisk/disk148
SCSI describe of /dev/rdisk/disk148:
             vendor: COMPAQ
         product id: MSA1000 VOLUME
               type: direct access
               size: 134399790 Kbytes
   bytes per sector: 512

# diskinfo /dev/rdisk/disk149
SCSI describe of /dev/rdisk/disk149:
             vendor: COMPAQ
         product id: MSA1000 VOLUME
               type: direct access
               size: 134399790 Kbytes
   bytes per sector: 512

# vgcreate -V 1.0 -e 16384 -l 255 -p 128 -s 64 vgtestl1 /dev/disk/disk148
(16384*128*64MB = 128TB)
# vgcreate -V 2.0 -s 64 -S 128t vgtestl2 /dev/disk/disk149

# pvdisplay  /dev/disk/disk148
--- Physical volumes ---
PV Name                     /dev/disk/disk148
VG Name                     /dev/vgtestl1
PV Status                   available
Allocatable                 yes
VGDA                        2
Cur LV                      0
PE Size (Mbytes)            64
Total PE                    2050

# pvdisplay  /dev/disk/disk149
--- Physical volumes ---
PV Name                     /dev/disk/disk149
```

```
VG Name                     /dev/vgtestl2
PV Status                   available
Allocatable                 yes
VGDA                        2
Cur LV                      0
PE Size (Mbytes)            64
Total PE                    2050
```

Impact of over provisioning

Extent Size	Max VG Size "provisioned"	Actual VG size in allocated extents	Num PV's used	System Memory consumption[1]		Space reserved on each disk	
				2.x	1.0	2.x	1.0[2]
4 MB	500 GB	500 GB	2	6 MB	5 MB	9 MB	4 MB
4 MB	500 GB	250 GB	2	5 MB	5 MB	9 MB	4 MB
16 MB	30 TB	30 TB	120	35 MB	47 MB	38 MB	16 MB
16 MB	30 TB	1 TB	4	5 MB	10 MB	38 MB	16 MB
32 MB	100 TB	100 TB	255	59 MB	78 MB	58 MB	32 MB
32 MB	100 TB	50 TB	125	32 MB	46 MB	58 MB	32 MB
128 MB	1024 TB	1024 TB	500	140 MB	n/a	138 MB	n/a
128 MB	100 TB	100 TB	50	17 MB	21 MB	20 MB	128 MB
128 MB	1024 TB	100 TB	50	18 MB	n/a	138 MB	n/a

How to Pick an Extent Size

While the extent size has a similar impact on 1.0 and 2.*x* volume groups, this section discusses only 2.*x* volume groups unless 1.0 is explicitly mentioned.

[1] It is assumed that the logical volumes are not mirrored and the number of logical volumes is 255.
[2] The space is approximated to one extent because 1.0 wastes the space left after the last extent boundary. The actual space will vary based on extent size and disk/LUN size.

The extent size affects the following:

- The maximum volume group size you can select when creating a volume group.
- The amount of disk space reserved for the metadata.
- The memory footprint used by the activated volume group.
- The I/O performance (for the case where the logical volume is not striped).
- The resynchronization time of the mirrors in some particular cases.
- The minimum size of a logical volume.

As shown in Figure 2, "Maximum volume group size versus extent size", on page 21, the maximum size of a volume group dictates a minimum extent size. You can pick this minimum value or a larger value.

A larger extent size uses less disk space for metadata and reduces the volume group memory footprint. To understand the relationship, see the Figure 1, "Space reserved on each disk for 2.*x* metadata", on page 17 and Figure 4, "Maximal memory footprint, all extents allocated, max number of LVs and PVs", on page 24.

As an example, suppose you want a volume group that can grow to 128 TB. Figure 2, "Maximum volume group size versus extent size", shows that the minimum extent size is 4 MB. As a consequence, you can pick an extent size between 4 MB and 256 MB.

For the amount of disk space used for metadata (on each disk), See Figure 1, "Space reserved on each disk for 2.*x* metadata". The metadata disk space used is between approximately 520 MB for 4 MB extents and 20 MB for 256 MB extents.

For the memory footprint, see Figure 3, "Maximal memory footprint, all extents allocated, max number of LVs and PVs". The chart shows about 950 MB for 4 MB extents and 27 MB for 256 MB extents. However, these figures represent the full usage of 128 TB of the volume group. In the following year, you expect to use only half the capacity of the volume group (64 TB). Reading the chart using 64 TB as the volume group size instead of 128 TB, you can see about 500 MB for 4 MB extents and 20 MB for 256 MB extents. You can examine the chart for other extents sizes between 4 MB and 256 MB.

Notes:

- This maximum memory footprint chart gives the worst case memory footprint for a given volume group size and extent size. In addition to using all the extents and the maximum number of logical volumes and physical volumes, a specific configuration of the mirrors is needed to reach this worst case. Most likely the actual footprint in a user configuration will be smaller.
- The impact of the number of logical and physical volumes on the memory footprint is negligible.

For the example, you now have a good idea of how much disk and memory overhead you can incur depending on the extent size.

There is an incentive to use large extents because it reduces the LVM overhead in term of disk space and memory footprint. However, before selecting the extent size, you must consider three other factors.

The most important is the size of logical volumes. The minimum disk space allocation to a logical volume is one extent. As a consequence, it makes sense to select a small extent size for a volume group containing a lot of very small logical volumes. It avoids wasting disk space if a lot of small logical volumes are needed or used.

Another factor to consider is related to user I/O performance. If you want to use extent based striping (*lvcreate* –D y), smaller extents give better throughput. However, you can obtain equal or better I/O throughput with striped logical volumes (*lvcreate* –i). As a consequence, you can keep large extents and still get excellent I/O throughput by using striped logical volumes.

The last factor is related to resynchronization time. Small extent size can accelerate the resynchronization of a logical volume in the case where an application issues sparse I/Os on the logical volume at the time a physical volume is down. In this case, the amount of data to resynchronize is smaller. As a result, the resynchronization is faster. Note that if a resynchronization of a complete mirror of a logical volume is needed, the size of the extent has no affect on the resynchronization time.

In general, unless you need a lot of very small logical volumes, it is better to pick a large extent size and use striped logical volumes where you need high performance.

The following table illustrates how extent size limits maximum volume group size.

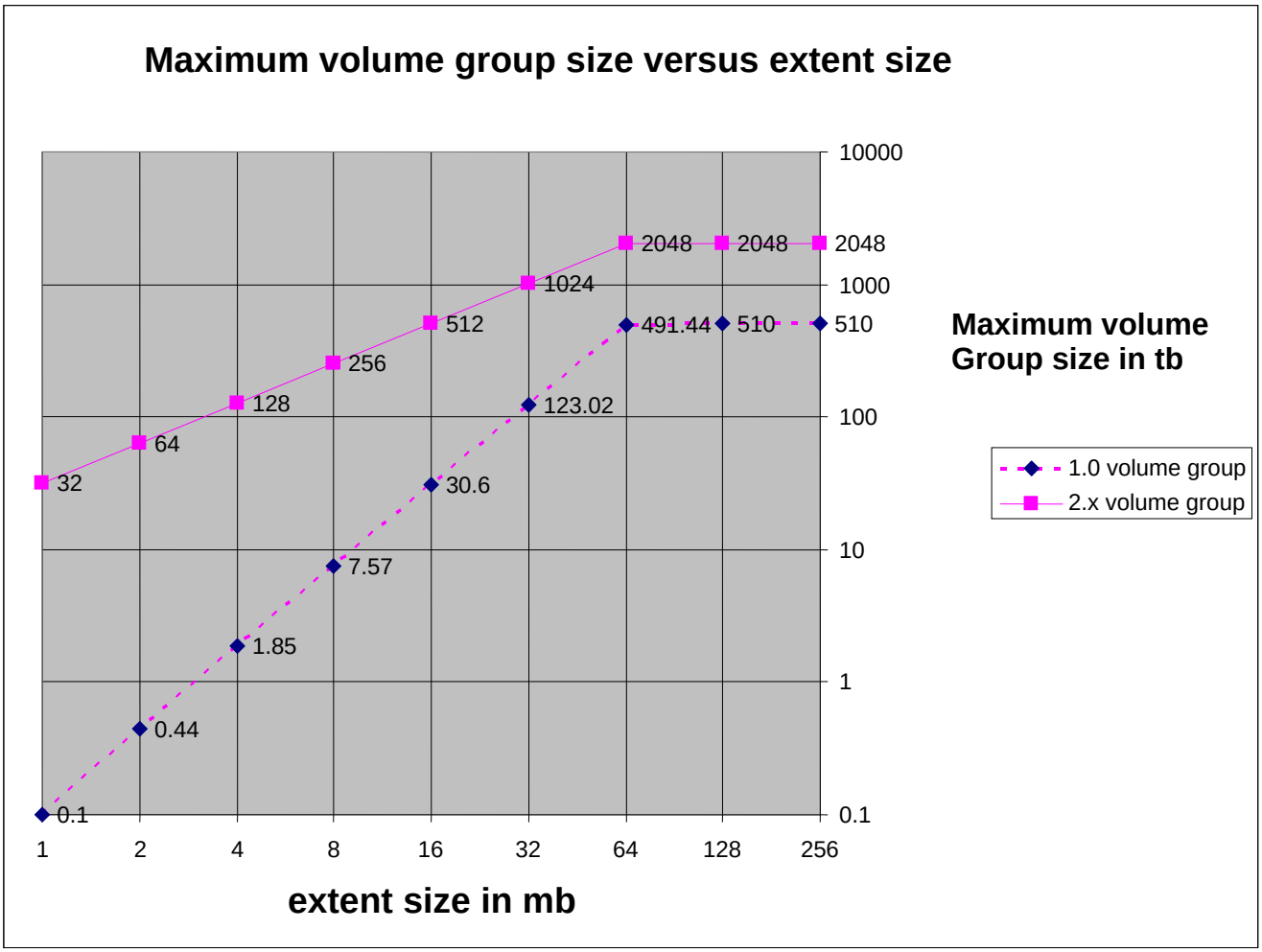


Figure 2

Notes:

- The 2.x graph stops at 2048 TB because of the limit of 32 million extents per volume group.
- The 1.0 graphs stops at 510 TB because the metadata must fit in one extent.

Memory Footprint

The memory footprints given in this chapter are estimates.

The memory footprint of a 2.*x* volume group provisioned for a given extent size and maximum volume group size is somewhere between the minimum and the maximum memory footprint as illustrated in Figures 3 and 4. The minimum and maximum memory footprints are shown in the "Minimum Memory Footprint" and "Maximum Memory " sections, respectively.

The 2.*x* volume group memory footprint starts very small when the volume group is created (independent of the provisioned maximum size of the volume group) and grows roughly proportional to the number of extents allocated to logical volumes.

For example, a 2.0 volume group provisioned for 128 TB with an extent size of 64 MB has a minimum footprint of 2 MB and a maximum memory footprint of 73 MB. The memory footprint of this volume group is 2 MB when the volume group is created and can eventually grow to 73 MB.

Minimum Memory Footprint

Figure 3 shows the memory footprint of 1.0 and 2.*x* volume groups when activated and with no extents yet allocated to logical volumes. This is called the "minimum footprint". This is the memory footprint just after a *vgcreate*, for example.

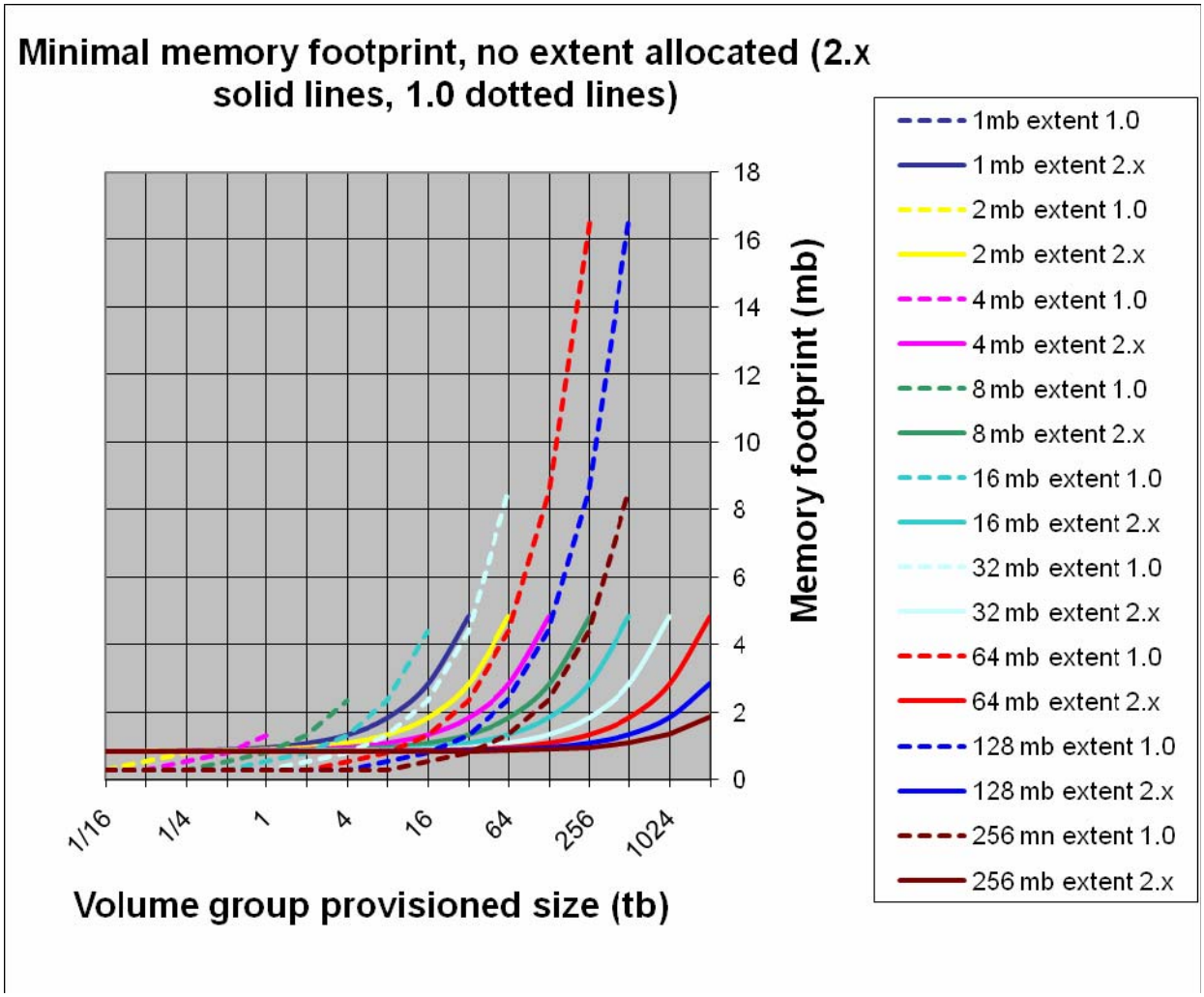


Figure 3

A 2.*x* volume group enables you to provision a very large size, but the minimal memory foot print is small. For example, the minimal memory footprint of a 2.*x* volume group provisioned for 2 PB with 64 MB extents is

approximately 5 MB while the minimal memory footprint of a 1.0 volume group with the same extent size provisioned for only 256 TB (8 times smaller) is 17 MB.

The minimum footprint of a 2.*x* volume group regardless of its provisioned size never exceeds 5 MB.

Maximum Memory Footprint

Figure 4 shows the memory footprint of 1.0 and 2.0 volume groups activated, fully populated, and with the worst case volume group configuration.

To get the 2.1 footprint in the same conditions, add 30 MB to the 2.0 footprint. The 30 MB difference is because in this maximum configuration, the 2.1 volume group contains 2047 physical and logical volumes while the 2.0 volume group contains only 511.

The worst case volume group configuration in terms of memory footprint is when all the extents of the volume group are allocated to logical volumes, the maximum number of logical volumes and physical volumes are used and everything is mirrored. Typically, users do not reach the maximum footprint even if all the extents are allocated to logical volumes.

Note that this is not a one-to-one comparison between 1.0 and 2.*x* because the volume groups in Figure 4 are assumed to contain the maximum number of logical and physical volumes, and the maxima for 2.*x* are higher. In other words, the memory footprint of a 2.*x* volume group includes more volumes than 1.0. For an exact comparison, see "Comparison of Volume Group Memory Footprints for the Same Number of Logical and Physical Volumes". However, the number of logical or physical volumes does not make a significant difference in the memory footprint.

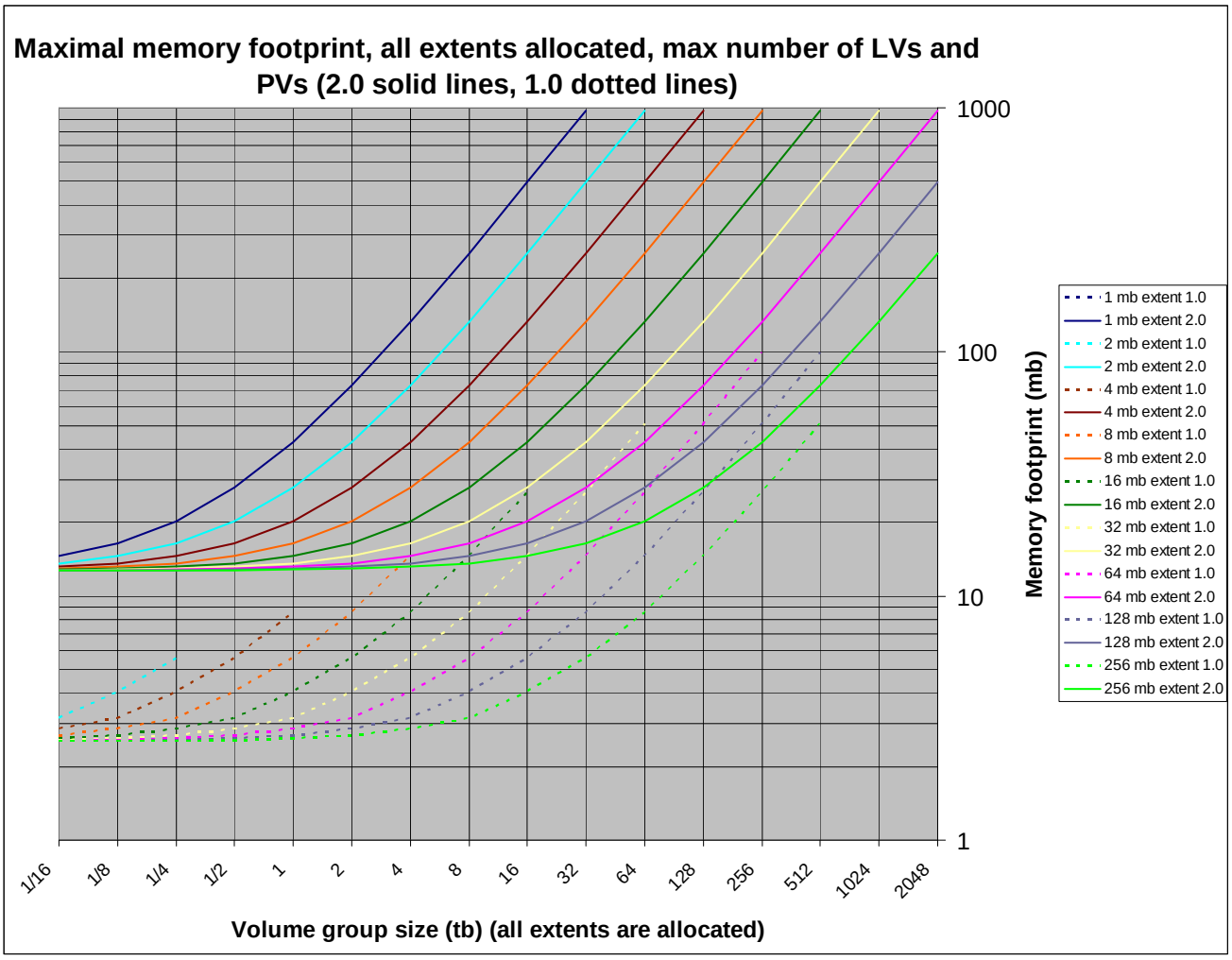


Figure 4

Notes:

- For a given extent size, the 2.0 line goes further right compared to the 1.0 line because a 2.0 volume group can be bigger.
- The 2.0 lines do not go farther up and right because of the limit of 32 million extents per volume group.

For the same volume group size and same extent size, a 2.0 volume group uses about 10 MB more than 1.0 because of the following reasons:

- A 2.0 volume group at the maximum contains 511 logical volumes and 511 physical volumes while a 1.0 volume group is limited to only 255 logical volumes and 255 physical volumes
- The10 MB increase enables the removal of the 1.0 limitations (for example, logical volume size and number of extents per physical volume or logical volumes) and enables new features such as storing the logical volume name in the metadata.

Comparison of Volume Group Memory Footprints for the Same Number of Logical and Physical Volumes

This comparison uses 255 logical volumes and 10 or 100 physical volumes per volume group. Figure 5 uses an extent size of 8 MB while Figure 6 uses 64 MB.

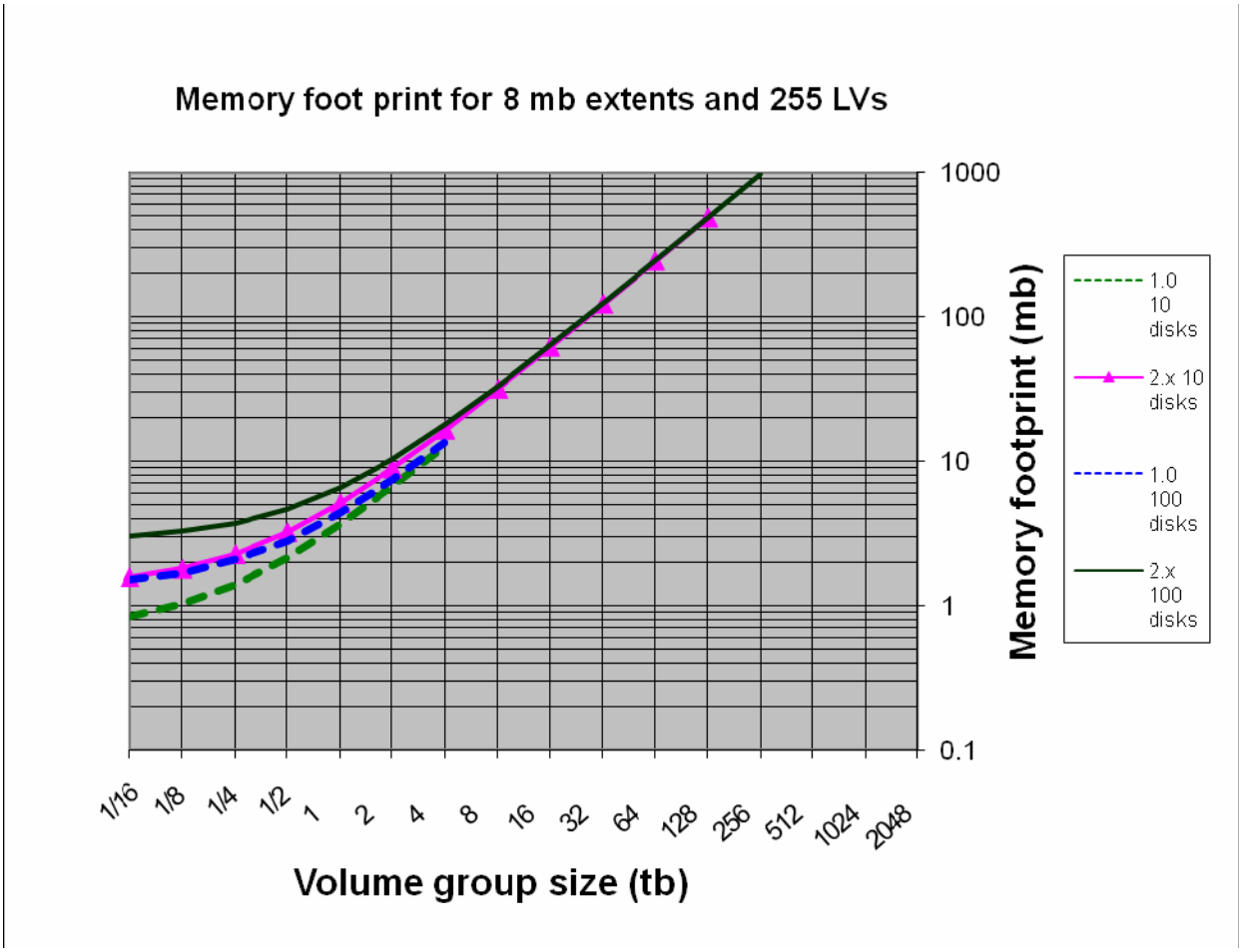

Memory foot print for 8 mb extents and 255 LVs
1000
100
10
1
0.1
Memory footprint (mb)
1.0 10 disks
2.x 10 disks
1.0 100 disks
2.x 100 disks
1/16
1/8
1/4
1/2
1
2
4
8
16
32
64
128
256
512
1024
2048
Volume group size (tb)

Figure 5

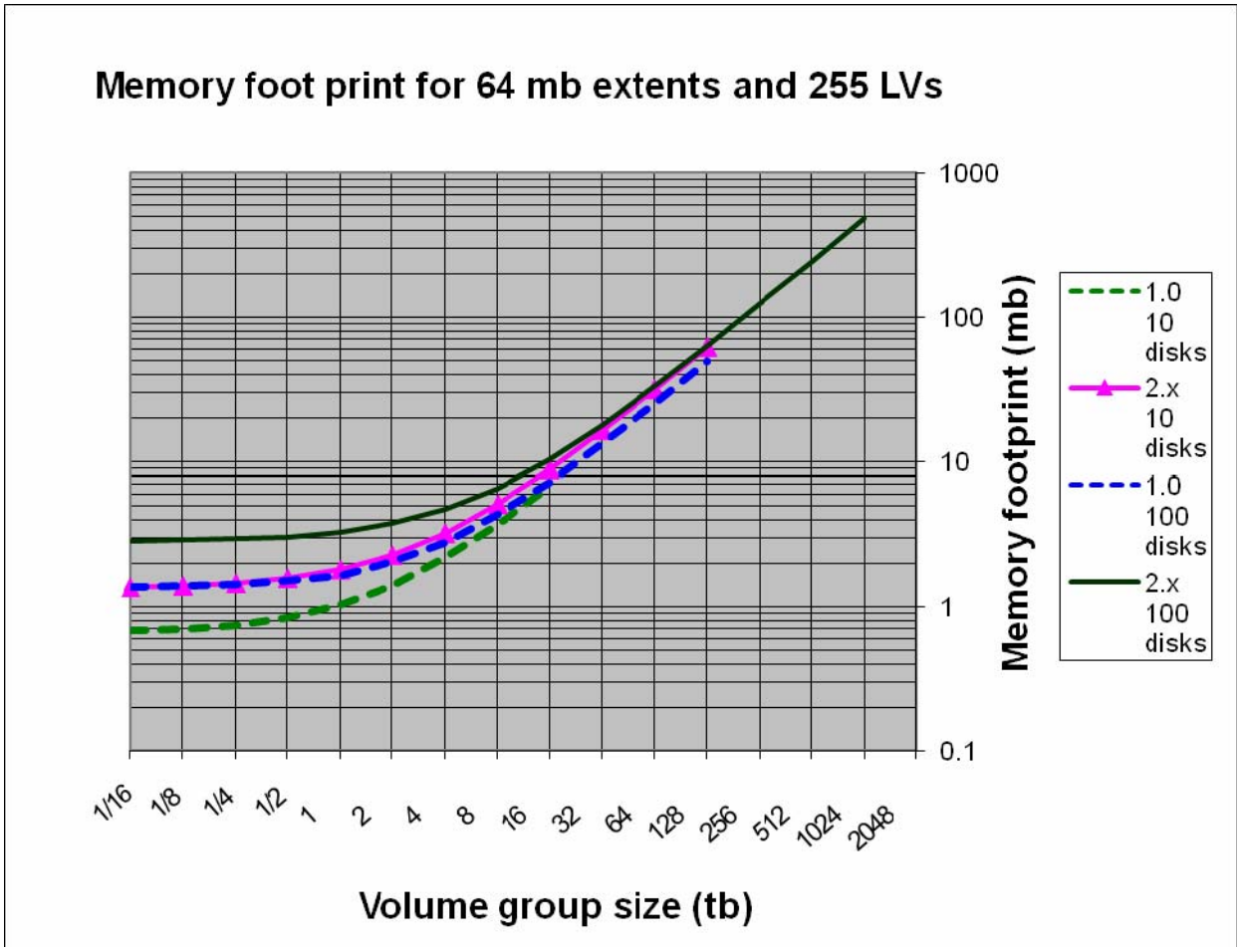


Figure 6

How to Configure the New Limits

Number of logical volumes or physical volumes in a volume group

There is nothing to do to on a 2.*x* volume group (at creation time or later) to reach the new limits for the number of physical volumes or logical volumes (511 for 2.0 and 2047 for 2.1). As indicated previously, version 2.*x* volume groups are provisioned in a way that LVM can always add physical volumes or logical volumes until the maximum capacity of the volume group is reached. Add physical volumes (*vgextend*) and logical volumes (*lvcreate*) as needed.

Maximum size of the volume group

The size limit is displayed in Figure 2 in the "How to Pick an Extent Size" section. You set the maximum size of the volume group when you create it (*vgcreate* option –S). If the volume group is not provisioned sufficiently, and you want to go beyond the volume group maximum size that was provisioned, with March 2008 HP-UX 11i v3 release, you must copy the data to a different, larger volume group.

For example, if you provision a volume group for 32 TB with an extent size of 4 MB (*vgcreate* –v 2.0 –s 4 –S 32t myvg /dev/disk/disk149), you cannot grow the volume group up to the limit allowed by 2.0, that is 128 MB for 4 MB extent. You can only grow the volume group up to 32 TB.

Size of logical volume or number of mirrors

To reach the new limits, increase the values passed in to *lvcreate*, *lvextend*, or FSWEB.

Migration Between1.0, 2.0, and 2.1

Why would a user migrate?

You might want to migrate from 1.0 to 2.1 because your1.0 volume group is too small. You can use *vgmodify* to extend the limits of a 1.0 volume group, but that might not work or might not be enough. HP does not recommend migration from 1.0 to 2.0, instead migrate to 2.1.

Migrating to a lower volume group version (2.*x* to 1.0 or 2.1 to 2.0) is impossible as soon as the limits of the destination version are exceeded. For example a 2.0 volume group containing 256 logical volumes (one more than the 1.0 limit) cannot be migrated to 1.0.

How to migrate?

At this time, there is no in-place migration available. You must migrate by copying data (for example, with *dd*) to another volume group created with the desired version.

You face the same situation when you cannot increase the size of a 1.0 volume group and *vgmodify* cannot do it either. You must copy over the data to a new volume group.

To speed up the copy, you can issue several *dd* commands in parallel on different volume groups or different parts of volume groups.

When to migrate?

Because the migration is done by copy, a good time to migrate is when you replace your storage.

High Availability

With version 2.*x* volume groups, you now have the ability to provide additional copies of user data, increasing the availability of your data. 1.0 volume groups are limited to two mirrors (three replicas of the data). 2.*x* volume groups can have up to five mirrors (six replicas of the data).

When Does it Make Sense to Use More Than Two Mirrors?

Using more than three replicas of the data makes sense as part of a setup for disaster recovery.
The following two examples show a logical volume whose data is replicated six times. Figure 7 shows a two site setup. In this case, if one site goes down the data stays highly available on the surviving site because it is replicated three times on that site.

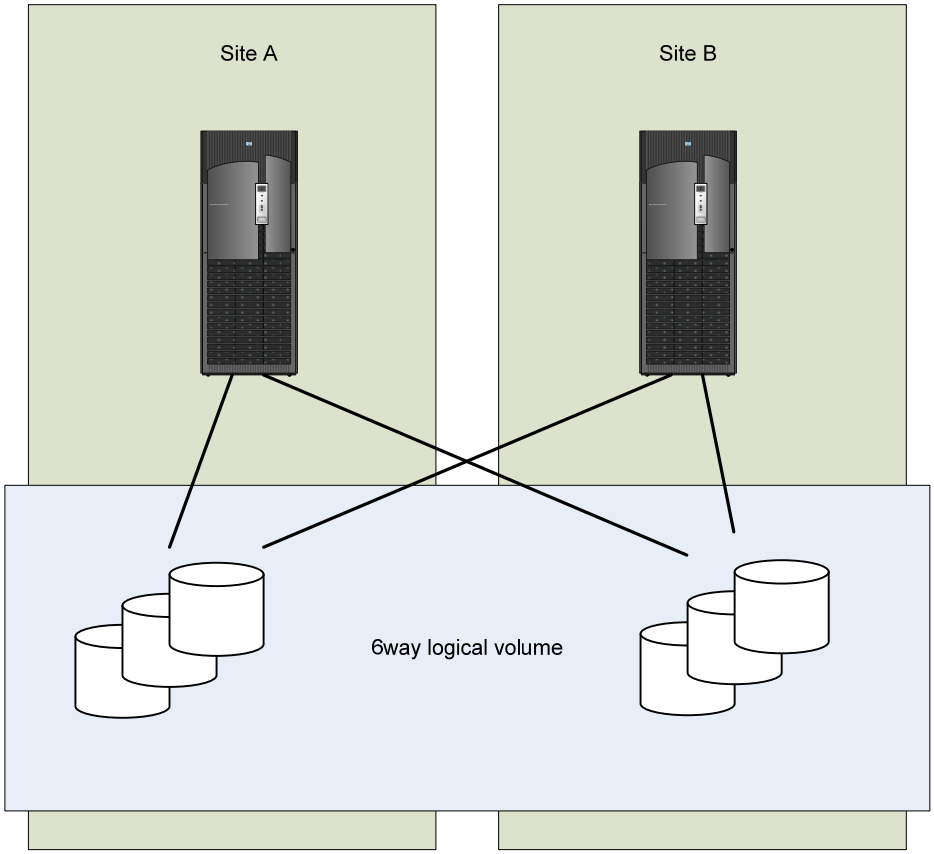


Figure 7

The advantage of having three mirrors at one site is when a backup strategy is deployed where a mirror logical volume is split off (*lvsplit*) to perform backups. In this example, either site can employ a backup strategy that does not jeopardize data availability during a site outage.

Figure 8 shows a three node cluster deployment In this example, each node can be geographically located together with two mirrors of data. This provides data redundancy at each site in the event of lost connectivity between the sites.

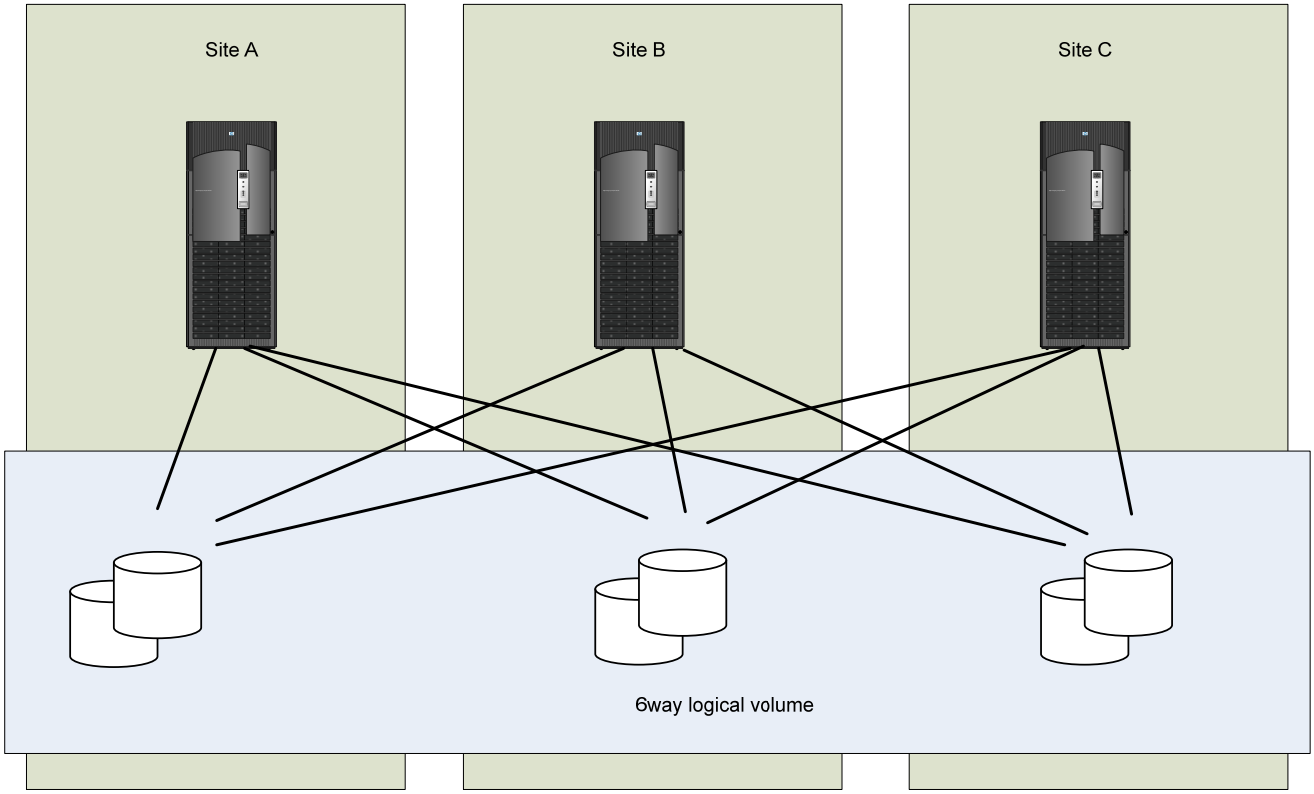


Figure 8

Glossary

DSF

Device Special File. A file associated with an I/O device. DSFs are read and written the same as ordinary files, but requests to read or write are sent to the associated device.

LUN

A SCSI logical unit. This refers to an end storage device such as a disk, tape, floppy, or CD. This is the logical unit itself and does not represent the path to the logical unit.

Metadata

The on-disk structures that LVM uses to manage a volume group. This space is not available for application data.

KB

A kilobyte unit of information equal to 2^{10} or 1024 bytes

MB

A megabyte unit of information equal to 2^{20} or 1,048,576 bytes

GB

A gigabyte unit of information equal to 2^{30} or 1,073,741,824 bytes

TB

A terabyte unit of information equal to 2^{40} or 1,099,511,627,776 bytes

PB

A petabyte unit of information equal to 2^{50} or 1,125,899,906,842,624 bytes

For More Information

To learn more about some of the LVM features, see the following document on HP documentation website:
http://docs.hp.com (Use search with the given name of the whitepaper)
http://www.docs.hp.com/en/oshpux11iv3#LVM%20Volume%20Manager

- SLVM Single-Node Online Reconfiguration (SLVM SNOR)
- LVM Online Disk Replacement (LVM OLR)
- When Good Disks Go Bad: Dealing with Disk Failures under LVM
- LVM Volume Group Dynamic LUN expansion (DLE)/vgmodify
- LVM Volume Group Quiesce/Resume
- HP-UX System Administrator's Guide Logical Volume Management
- HP-UX LVM Performance Assessment (The whitepaper will be available soon)

Call to Action

HP welcomes your input. Please give us comments about this whitepaper, or suggestions through our technical documentation feedback website: http://docs.hp.com/en/feedback.html.