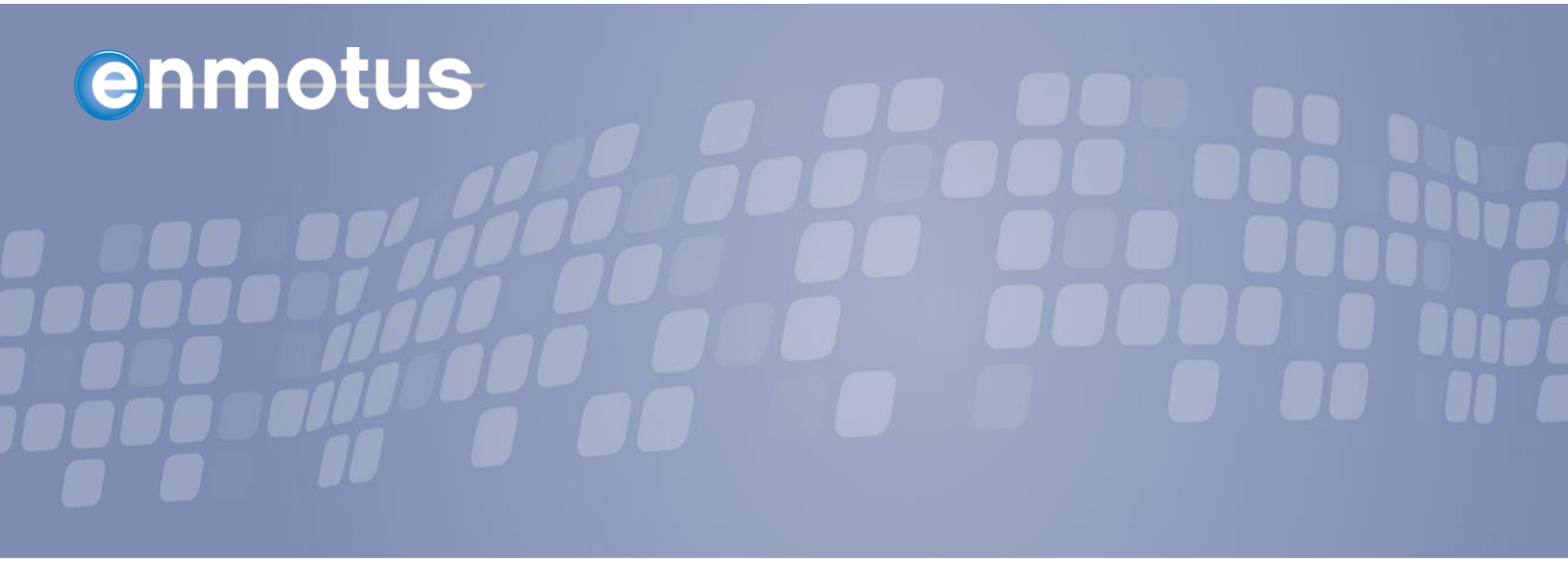

VirtualSSD™ for Server

Red Hat/CentOS 7.X, SUSE 12 SP3

Installation and User Guide

Software Version: 1.4.0

Date: 12th March, 2019

Contents

Tables

Installation and User Guide

1.0 Introduction to VirtualSSD Virtual Disks

The Enmotus VirtualSSD software uses a storage hypervisor approach to present virtual block devices to the operating system that are made up of one or more pools of physical storage devices such as solid state disks (SSDs) or traditional hard drives (HDDs). The Enmotus virtual disks are implemented entirely in the kernel driver layers, so any standard disk or storage management tools will work with an Enmotus virtual disk as it appears to the operating system as a normal physical device.

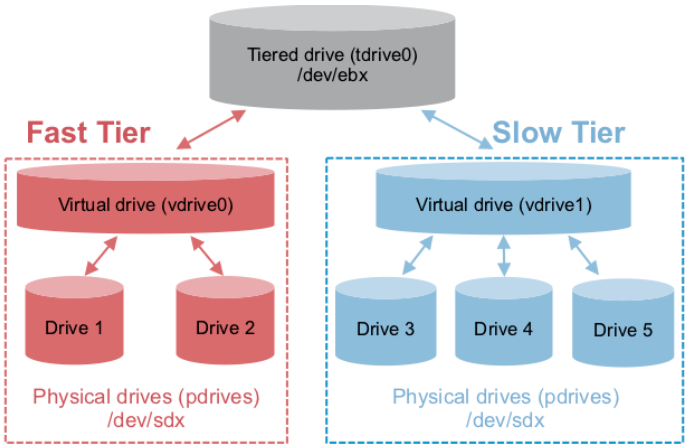

Figure 1: Enmotus Device Hierarchy

1.1 Virtual Hierarchy

As illustrated in Figure 1, there are four layers between the VirtualSSD presented to the user and the low level disks. Each physical disk, or Drive, is allocated to a fast or slow tiered virtual disk, or vDrive, which can accommodate up to sixteen (16) Drives in a single storage tier or pool. Note, a physical Drive may also be a virtual volume from a hardware, a host driver based or OS RAID stack. VirtualSSDs (or tDrives), are formed from a fast and slow vDrive pair and attached to a host block device (shown here as /dev/ebX where X=a,b,c …).

1.2 VirtualSSD vs. Caching

Enmotus VirtualSSD utilizes a storage hypervisor technology with fast virtualization and intelligent data migration that differs in many aspects from cache look-alike products. SSD caching alternatives make temporary **copies** of data onto a dedicated SSD device that persist for a finite period. This typically accelerates read traffic only.

Enmotus VirtualSSD on the other hand, treats the fast media as a primary storage device (rather than a look-aside device) and data is **moved** or relocated to the fast media where it can be accessed at very near the full read and write performance of the underlying fast storage devices.

Furthermore, the capacity of the fast tier can be made available as usable storage, as the Enmotus architecture does not require the capacity of the fast media to be hidden or reserved.

2.0 Installation

2.1 System Requirements

- Red Hat/CentOS 6.x/7.x 64 bit Linux Distribution
- OpenJDK v1.7 or higher (tested up to 11) if using the UI tools
- Intel or AMD compatible server system
- Min 4G system memory recommended

2.2 Software Installation

Download or copy the VirtualSSD Linux installation binary to a temporary installation directory. The filename/command line typically uses the form:

- `sudo tar xvzf ./VirtualSSD_RedHat7_XXXXX.tar.gz`

Where XXXXX indicates the version number of the installation package (no sudo necessary if logged in as root). Everything needed to install the Enmotus software is compacted within this file, included a default trial period mode if installing for the first time.

Decompress the .tar file if provided in this form and execute as sudo the INSTALL.sh installer script.

Open a terminal either with root or sudo privileges, and execute the above installation file. The Linux installation, if supported, will be correctly determined and the core modules and applications installed.

Standard RPM installation is used for CentOS and Red Hat.

2.3 Installing the License

After the software is installed, a trial period begins (typically 30 days) after which, the software will enter a state where it will no longer tier hot data to the fast device. In order to continue using the software, an activation key must be obtained from Enmotus or one of its representatives to continue to use the software.

The license may be activated one of two ways: either via an online activation process that handshakes with an Enmotus license server via the Internet, or an offline process if the system is being activated without an Internet connection.

In either case, the form of the activation key is ABCD-EFGH-IJKL-MNOP.

2.3.1 Online Activation Process

This is the simplest method for activating the license and requires an active Internet connection to the machine to the server being licensed:

```
> ecmd --license ABCD-EFGH-IJKL-MNOP
```

Once complete, the license status may be checked as follows:

```
> ecmd --license
```

2.3.2. Offline Activation Process

For offline activation, the following steps are required:

1. At a root or sudo level console on the machine being licensed, use the command:

   ```
   > ecmd --license ABCD-EFGH-IJKL-MNOP –offline
   ```

2. A local file will be generated and must be copied to an Internet connected computer and the file emailed to Enmotus or their representative.
3. A file will be returned which must be copied back to the machine being licensed.
4. The license may then be activated by typing the following command:

   ```
   > ecmd --license <pathname/file_name_returned> --activate
   ```

2.3.3 Checking License Status

Once successfully installed, the license status may be checked as follows:

```
> ecmd --license
```

2.3.4 Returning a License

To unlicense the server and return the license so that it can be used on a different server, the license may be returned to the license server using the following:

```
> ecmd --license return
```

Use the --offline option and follow the offline activation emailing instructions if no Internet connection is available.

2.4 Updating the Enmotus Software

The software is updated using the same installer program used for initial installation and will automatically update the appropriate components of the software according to the type of license deployed on the system.

3.0 Creating Tiered and Virtual Disk Volumes

Tiered volumes are created using raw disks, and in some cases, formatted volumes where a non-destructive transform may be supported.

The general command --create is used to make intermediate virtual volumes (vdrives) that form the tiers as well as the tiered volumes (tDrives). The general format of the command is as follows:

```
> ecmd --create [disk-type] <device1> [device2] … [mode]
```

Where:

disk-type = tdrive, vdrive[1]

device1,2… = the devices to be used to create the virtual device

mode = type or initial mode of the virtual volume (fast_last, fast_first, reserve…)

3.1 Creating a Simple 2-disk Tier

When combining a single pair of fast and slow devices together into a tDrive, there is no need to create intermediate vdrive devices and a single command line may be used:

```
> ecmd --create <device1> <device2> [mode]
```

```
Example:
```

where device 1 is the fast-device to be used in the fast tier (e.g. an SSD block or RAID level device) and device2 as the slow-device in the slow tier (e.g. a hard disk or second RAID level device). Note, the fast device should be provided first to ensure the tier correctly recognizes it as the target fast tiered device.

If no mode is specified, a reserve_first configuration will be created by default, meaning that the fast device will be located starting at the low order LBA range of the virtual tiered block device.

Example single disk tier tDrive create commands:

```
> ecmd --create /dev/sdb /dev/sdc fast_first
```

```
> ecmd --create /dev/md0 /dev/md1 fast_split
```

```
Table of configuration options”
```

[1] tdrive and tdisk, vdrive and vdisk may be used interchangeably to allow script compatibility with the Windows version of ecmd

fast_first	fast device will be located starting at the low order LBA range of the virtual tiered block device.	Full capacity is available to the O/S
fast_last	fast device will be located starting at the high order LBA range of the virtual tiered block device.	Full capacity is available to the O/S
fast_split	fast device is split between the beginning and end of the LBA range of the virtual tiered block device.	Full capacity is available to the O/S
reserve_first	fast device will be located starting at the low order LBA range of the virtual tiered block device, and fast capacity is hidden from O/S	Slow capacity is available to the O/S (Default)
reserve_last	fast device will be located starting at the high order LBA range of the virtual tiered block device, and fast capacity is hidden from O/S	Slow capacity is available to the O/S
copy_tier	fast device will be located starting at the low order LBA range of the virtual tiered block device, and fast capacity is hidden from O/S.	Acts as read cache, all writes are written to slow tier. Can lose fast tier and maintain data integrity

3.2 Creating Multi-disk Tiered Disks

The create command can use two internal Enmotus virtual disks (vdrives) as it arguments instead of raw block devices. In this case, the vdrives needs to be pre-configured prior to being placed into a tier.

The general command for creating the tDrive from vdrive components is:

```
> ecmd --create vdrive<p> vdrive<q> [mode]
```

Where vdrive p and q are vdrives created using the vdrive create command described below, and mode is the same as the single disk per tier i.e. fast_first, fast_last or fast_split.

Example multi disk tier tDrive create commands:

```
> ecmd --create vdrive0 vdrive2 fast_first

> ecmd --create vdrive3 vdrive4 fast_last
```

The index number of the vdrives to be used may be established using the --list vdrives or --list pdrives ('V' column) described earlier.

3.3 Creating vdrives

The tiering engine supports pooling of physical disk drives in either a linear concatenation or stripe mode. Striping is implemented around tiering page boundaries with a stripe element size the same as the page size setting for the tDrive (default 4M).

The command for creating a vdrive from pDrive components is:

```
> ecmd --create vdrive <device1> [device2] … [mode]
```

Up to a maximum of 16 devices may be combined in a pool. Valid modes include linear (concatenated) and striping.

Example multi disk vdrive create commands:

```
> ecmd --create vdrive /dev/sd[b-d,g] linear
```

creates a 4-disk linear concatenated vdrive pool using devices sdb,c,d and g

```
> ecmd --create vdrive /dev/sdj
```

creates a single disk vdrive using sdj.

4.0 Applications Guidelines

4.1 Automated Real Time, Block Level Tiering

VirtualSSDs are designed to operate automatically at sub-file level in real time i.e. they have no direct awareness of files in a system, only storage IO accesses, and move data within seconds of it being detected as active until the system is load balanced. Statistics are gathered continuously, so once the software has learned where to best place data, it then continues to adjust data placement as storage IO patterns change or new files are added to or deleted from the system.

Real-time block based tiering has several advantages over traditional batch or scheduled tiering:

- Only the portions of the files that need to be on the fast storage are moved
- Data is moved within seconds or minutes of being accessed as opposed to daily or weekly
- Once the system has tuned to the storage IO patterns, it continues to adjust where active data is stored

Real time tiering, especially given the increased availability of more cost effective SSDs, makes a big difference to daily server operation given today's highly dynamic data environments and increasing use of unstructured data in everyday business decisions.

4.2 Tuning VirtualSSDs

There are several tunable parameters which the user may use to change the behavior of the tiering software to suit a particular application.

The two primary high level tunable parameters are ***promote rate*** and ***promote policy*** which load a set of pre-defined values into the tiering engine and may be set for each individual tiered volume at create time using the options menu or modified later (see installation section earlier). For users wishing to customize these settings directly, see the Advanced Settings section later.

4.2.1 Setting Promote Rate

The promote rate setting decides how often data is moved and at what activity thresholds. Depending on the activity level of the server, this setting may be tuned depending on the anticipated loading levels of the server. Three levels are provided as shown in Table 1.

Table 1: Promote Rate Aggressive, Normal and Slow Parameters

User Setting	Scan Cycle Time	Promote Threshold per Page IOs	Promote Threshold per Page Mbytes	Max Page Promotes Per Cycle
Aggressive*	2s	1 IOs	1M	32
Normal	8s	4 IOs	4M	16
Slow	60s	50 IOs	50M	2

** Default Setting*

4.2.2 Setting Promote Policy

Statistics about hosts IO made to the tiered volume are recorded by the VirtualSSD and decisions are made if and when to move data from from the slow to the fast storage tier based on a the policy setting for that particular VirtualSSD. These statistics are maintained on a per-page basis (see page size settings in installation or later sections).

Table 2: IO and MB Policy Settings

Policy Setting	Description
Read Only IO*	Promotes data based on the frequency of read IO requests per page
Read-Write IO	Promotes data based on the frequency of read AND writes IO requests per page
Read Blocks	Promote data based on amount of data read per page
Read-Write Blocks	Promote data base don amount of data read AND written per page

** Default Setting*

4.3 Recommended Settings by Application

Table 3 illustrates the recommended settings that may be used for different types of applications. Typical application profiles include database, email and web server configurations.

Table 3: Recommended Settings by Application

Application Profile	Traffic Type	Promote Rate	Promote Policy
*Small Business Server - General	Random Unstructured	Aggresive	Read IO
Database Server - Read Intensive	Random Unstructured	Normal	Read IO
Database Server - Frequent Updates	Random Unstructured	Normal	Read-Write IO
Database Server - Medium Large File	Sequential/Random Mix	Normal	Read-Write MB
Exchange/Sharepoint Server	Random Unstructured	Normal	Read IO
VDI Server	Medium-Large Streaming	Aggressive	Read MB
General Host Virtual Server	Medium-Large Streaming	Normal	Read MB
Large File/Video Server	Large Streaming	Aggressive	Read-Write MB
Big Data Storage-Server	Random Unstructured	Normal	Read IO
Unified Storage Server (Windows)	Random Unstructured	Normal	Read IO

** Default Setting*

4.4 Setting Page Size

4.4.1 General Settings

At create time, it is possible to change the default page size for the VirtualSSD. The size of the page determines a number of operating parameters within the tiering engine such as:

- Level and number of statistics that are maintained across the volume
- The level of activity for the amount of data read or written
- The amount of data moved per promote cycle
- The amount of system RAM required to maintain the runtime statistics tables

The default setting is the maximum 4Mbytes per page and should be sufficient for most applications outlined above. However, it may be desirable to change the default to a smaller amount or some multiple of underlying stripe sizes if tiering RAID devices.

The general guidelines in Table 4 are provided, however please note these are dependent on the user environment and hardware and may require some experimentation for the particular environment the server and software is operating in.

Table 4: Page Size Setting Recommendations

Page Size (M,K bytes)	General Usage Guideline
4M*	• Volumes >= 50TB • Large File Based Applications e.g. Video Servers or Virtual Desktop Servers
1-2M	• Volumes 10-50TB • Medium to Large File Based Applications
512K	• Volumes < 10TB • Small-Medium File Based Applications
64K-256K	• Volumes < 5TB • Small File Based Applications

** Default Setting*

4.4.2 Aligning with RAID Stripe Sizes

To achieve maximum performance when used with a third party RAID device, it is highly recommended that the pagesize be set to some multiple of the fast tier RAID full stripe size. This ensures that VirtualSSD pages are always aligned with RAID stripes.

When creating a VirtualSSD tiered disk, the default setting for the page size is normally 4Mbytes. Once a tier is created this value cannot be changed without recreating the tier (i.e. delete and recreate it). The default can be changed prior to creating a tiered disk using the ecmd --pagesize <size> command (or via the econfig utility) where <size> can be bytes, Kbytes or Mbytes.

Examples of well aligned systems are as follows:

a) 8 drive RAID 5, stripe element per drive 256K

Number of data drives = 8 - 1 parity = 7
Full stripe size is 7x256K = 1792K

Recommended VirtualSSD pagesizes: 3584K or 1792K

b) 6 drive RAID 6, stripe element per drive 128K

Number of data drives = 6 - 2 parity = 4
Full stripe size is 4x128K = 512K
Recommended VirtualSSD pagesizes: 4M (default), 2M, 1M or 512K

Table 5: Page Size Recommendations for RAID Systems

RAID Stripe Size →			64KB	128KB	256KB
Total Drives in RAID6 set	Total Drives in RAID5 set	Data Bearing Drives (RAID0/1)	Enmotus Page Size (KB)	Enmotus Page Size (KB)	Enmotus Page Size (KB)
4	3	2	4096*	4096*	4096*
5	4	3	4032	3840	3840
6	5	4	4096*	4096*	4096*
7	6	5	3840	3840	3840
8	7	6	3840	3840	3072
9	8	7	4032	3584	3584
10	9	8	4096*	4096*	4096*
11	10	9	4032	3456	2304
12	11	10	3840	3840	2560
13	12	11	3520	2816	2816
14	13	12	3840	3072	3072
15	14	13	3328	3328	3328
16	15	14	3584	3584	3584
17	16	15	3840	3840	3840
18	17	16	4096*	4096*	4096*

** Default Setting*

4.5 Virtual Map Initialization Settings

As briefly described earlier, during the create process, the user has the ability to define how the initial tiered disk map may be constructed in terms of fast and slow device ordering in terms of the tiered disk logical block addressing. A tiered volume maps host IO requests on the fly into physical device IO requests by consulting a RAM based mapping table.

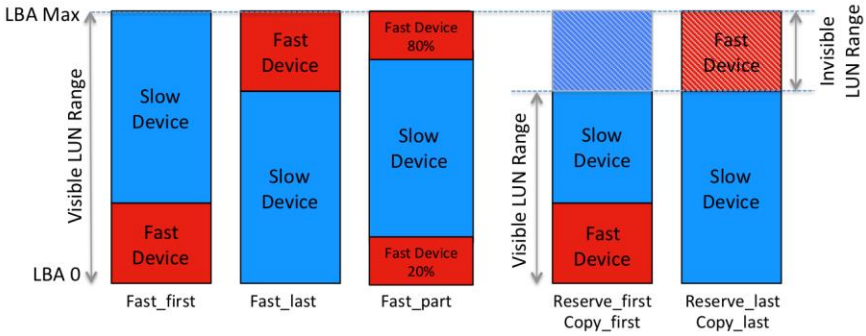

Figure 2: Initial Tier Mapping Options

From an application standpoint, these settings determine the initial starting performance levels of the volume and are dependent on the nature and data access patterns. For example, if we use fast first (default) setting, as the operating system copies data to the volume, it typically copies/moves files into

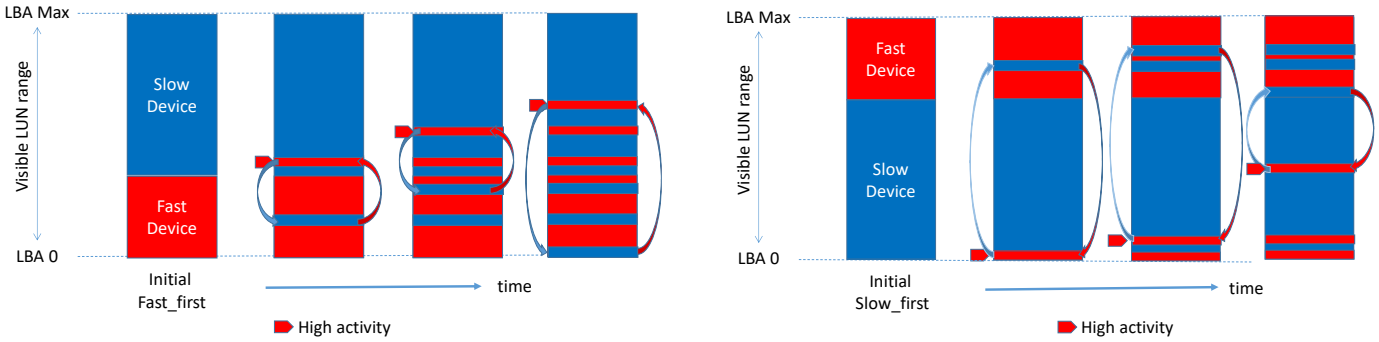

Figure 3: Fast First vs. Slow First Initial Mapping Over Time

the lower addressable area of any drive, hence for a fresh tiered volume, initial data will be copied to the SSD first for example. If we used slow first, initial data would be copied/moved to the HDD first for example. The former means the data is pre-warmed or tiered immediately at data copy, create or move time. The latter means the data is stored on the cold tier initially and will only be promoted if accessed later.

Table 6: Initial Tiered Map Setting

Initial Map Setting	Description	Typical Usage
Fast First	Fast tier (e.g. SSD) is mapped to the first logical block of the tiered volume, occupying the lower addressible range of the tiered disk.	Fast warming, applications where it is desired to have SSD performance immediately until the capacity of the SSD has been fully utilized. Works well when significantly less data than the SSD size is initially being placed onto the tiered volume.
Fast Last	Slow tier (e.g. HDD) is mapped to the first logical block of the tiered volume. SSD is placed at the end of the disk in this case.	Depending on the policy setting, all data is first copied to the slow tier (HDD) and only migrated to the SSD if subsequently read. Works well with larger data sets than the SSD size and the user wishes to selectively promote only the active portions of the data after it is copied to the tiered drive.
Fast Split	20% of the fast tier is mapped to the first logical block, followed by the complete slow tier, with the remaining 80% mapped to the end of the tiered volume.	A mix of the above, and especially useful for larger SSDs where only part of the data needs to be pre-warmed after initial create and the larger portion going to the hard drive. Also has the effect of setting an initial 80% of the SSD as a resource pool of fast storage that gets assigned to the active data as it is used.
Reserve First*	The fast tier capacity is hidden allowing it to be later removed if necessary. Fast tier (e.g. SSD) is mapped to the first logical block of the tiered volume, occupying the lower addressible range of the tiered disk.	For situations where the fast tier device may be frequently swapped out or removed. Note, it is possible to convert to this mode from any of the above modes. See --convert option in command line overview.
Reserve Last	The fast tier capacity is hidden allowing it to be later removed if necessary. Slow tier (e.g. HDD) is mapped to the first logical block of the tiered volume. SSD is placed at the end of the disk.	For situations where the fast tier device may be frequently swapped out or removed. Note, it is possible to convert to this mode from any of the above modes. See --convert option in command line overview.

** Default Setting*

Note, these setting only has an impact during the initial use period of the tiered volume, as over time, the automated load balancing engine will redefine the tied map based on the relative activity and promote settings.

4.6 Manual Command Line Promote Setup

For expert or experimental users, the following command line options exists to allow the aforementioned parameters to be individually tuned. It is recommended that any experimentation with these parameters be done on an offline or non-production test server whereever possible to ensure minimal disruption to end users before using on a production server.

To run the following commands, open a DOS Command Line or Windows PowerShell window with ***Administrator Priviledges*** and use the commands listed in Table 5. Also see Enmotus Command Line section later in this document.

Table 7: Advanced Command Line Configuration Options

Description	Command Line Format	Example
Promote Scan Time	ecmd --scantime <time> {s|m|h} where s=secs, m=mins, h=hrs	ecmd --scantime 5m
Promote Rate	ecmd --promote {aggressive|normal|slow} driveT where T is the number of the tiered volume in disk manager e.g. Drive3	ecmd --policy aggressive
Promote Threshold	ecmd --promote N driveT where N is the threshold value in IOs or Sectors (depending on the promote policy setting)	ecmd --promote 100 drive3 (100 IOs per page if rdio or 100*512=51,200 bytes)
Promote Policy	ecmd --policy {rdio|rwio|rdblock|rwblock} driveT	ecmd --policy rdblock drive3

4.8 Monitoring Tiered Volumes

There are several options available for monitoring storage activity of tiered volumes as well as assess how the fast tier (e.g. SSD) is mapped in relation to current and past activity levels. One of the most important views to see how well the fast tier has adapted to activity levels on the volume, and also quickly assess the locality information of data with respect to the fast-slow tiers, is the eLive Monitor utility region view. Using this view, the fast regions of the volume are shown in a highlighted color that represents the percentage of fast pages mapped to that region.

4.9 Testing Enmotus Tiered Volumes

Real-time tiering volumes behave differently to conventional caching or big iron tiering solutions. Tiers treat their different classes of storage as primary storage which is directly mapped to user storage allowing full performance of the media to be utilized for both reads and writes.

However, initial testing using many popular benchmarking programs can provide misleading results, principally because they tend to assume a single class of physical is being tested not a dynamic, mixed storage media device such as that enabled by tiering. Hence, broad sweeping benchmarks that exercise an entire all at once will create a result that looks not much better than the performance of the slow media device. This is because a tier blends the IOs from the two tiers and given the slow tier is usually significantly dominates the mix of fast and slow, the result is marginally better than slow.

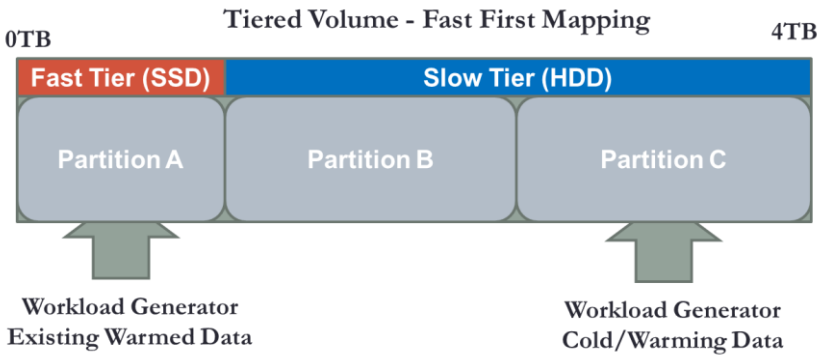

Figure 2: Test Partitions for Tier Analysis

In real world applications, only a small portion of any volume is active at any one time in 80%+ of all applications. Hence, when testing a tiered volume it is far more meaningful to test data in its two primary modes of operation:

- Pre-warmed or existing tiered data performance
- Cold or data that is being warmed

The easist way to do this is set up a test tiered volume using the fast first (default) configuration, then create three file partitions on the volume as follows:

- One partition at the beginning of the volume, roughly 5G smaller than the size of the raw fast device (e.g. SSD) - Partition A
- A second partition to fill the “middle” portion of the tiered volume - Partition B
- A final partition that is again smaller than the fast tier size - Partition C

Once the partitions are setup, then use an off the shelf benchmark program or file exerciser on partition A to measure what the performance of the fast tier is. This should be close to the raw performacne of the device used to form the fast tier.

To test the slowest performance of the tier, use the same exercisor or workload generator and target partition C which using the default setup, should be mapped to the slow portion of the tier. It is suggested that the benchmark or exerciser should be run for a short while, then left for an equal length

of time to allow the tier to move data without host io activity. The initial performance run should show slow media rates e.g. HDD IO or MByte rates. Each subsequent run after than should show increasingly better performance until it reaches the fast tier rates.

Note, in some cases, the maximum fast tier IO access rates may not be fully achieved unless the test is run with a read-write promote policy instead of the default read-only policy. It may also require the test to be run with promote rate set to aggressive. This is because some benchmarks may write to a location but not subsequently access a tiered page sufficiently to trigger a promote event to occur. Using the aggressive rate ensures that only a small amount of activity is necessary to trigger a promote and will attempt to move the data as quickly as possible (see earlier sections on promote policy and rate settings).

5.0 Enmotus Command Line

The Enmotus software may be managed entirely by command line without the use of graphical utilities using the Enmotus ecmd command line utility. All commands require a command line with root privileges.

5.1 Listing System Devices

The following command allows the currently available devices to be listed in the system and will also display if a device is currently ***captured*** by the Enmotus software i.e. it has Enmotus metadata on the disk.

```
>ecmd --list
```

A block device marked with * is the boot drive. This drive will be unavailable for use by the tiering engine. Devices marked with a '>' indicate that the drive is captured by the VirtualSSD software.

Additional options available for the --list command are:

`ecmd --list vdrives`	lists any intermediate (non-visible) vDrives configured
`ecmd --list pdrives`	lists just the captured devices loaded into the tiering engine
`ecmd --list lun`	lists the attached tDrives that are attached/mounted

5.2 Creating and Deleting a VirtualSSD

To create a tiered disk, use the following general commands:

```
>ecmd --create <fast drive> <slow drive>
```

or

```
> ecmd --create <hybrid drive> ssd=<ssd start lba> hdd=<hdd start lba>
```

Up to 7 tiered volumes may be created in this manner. To remove all tiered disks and return all disks to the operating system, use the following general command:

```
> ecmd --delete_all
```

or

```
> ecmd --delete /dev/ebX
```

to delete a specific tDrive (X=the block device letter identified in –list e.g. 'a' for eba).

Important: the delete command will erase all tiering and user volume information, so use carefully.

For example, creating a basic 2-drive tier using drive1 and drive2 from the --list command, use the following:

```
> ecmd --create /dev/sda /dev/sdb
```

where

/dev/sda is the fast tier device e.g. SSD or RAID device

/dev/sdb is the slow tier device e.g. HDD or RAID device

Note: always ensure that the fast drive is listed first so that in the event two similar device types are used, the tiering software will default to using the first one as the fast device.

5.3 Creating a Single Drive VirtualSSD

A single drive VirtualSSD may be created as follows:

```
> ecmd --create single /dev/sda
```

The VirtualSSD may then be upgraded later to a full tiered device by adding a fast tier device or SSD. VirtualSSD's created this way will default to reserve mode.

5.4 Creating Multi-disk VirtualSSDs

Multi-disk tiers allow multiple disks to be combined in either striped (performance) or concatenated modes directly within the VirtualSSD software without the use of an external RAID controller or software. The user must first create the individual virtual drives before tiering them.

The create command can use two internal Enmotus virtual disks (vdrives) as it arguments instead of raw block devices. In this case, the vdrives need to be pre-configured prior to being placed into a tier.

The general command for creating a tiered volume from vdrive components is:

```
> ecmd --create vdrive<p> vdrive<q> [mode]
```

Where vdrive p and q are vdrives created using the vdrive create command described below, and mode is the same as the single disk per tier i.e. fast_first, slow_first or fast_part, etc.

Example multi disk tier tDrive create command:

```
> ecmd --create vdrive0 vdrive2 fast_first
```

creates a new tDrive using vdrive 0 and 1, with vdrive 0 as the fast tier starting at LBA 0

The index number of the vdrives to be used may be established using the --list vdrives or --list pdrives ('V' column in the --status output) described earlier.

The command for creating a vDrive from pDrive components is:

```
> ecmd --create vdrive <device1> [device2] …  [mode]
```

Up to a maximum of 16 devices may be combined in a pool. Valid modes include linear (concatenated) and striping.

Example vdrive create command:

```
> ecmd --create /dev/sd[b-g] linear
```

creates a 6-disk linear concatenated vdrive using block devices sdb to sdg.

5.5 Creating Non-Default Maps

In all the above examples, the SSD or fast tiered device will always be mapped by default to the new tiered disk's LBA 0 address at create time i.e. it is placed at the start of the new device's address range (referred to as ***fast_first*** - see Figure 8). As a disk usually gets filled up first from its LBA 0 address, this is usually the most optimal way to use the tiered drive as most data will start out on the SSD until it has been filled up. This of course changes as the drive is used and the automated tiering adjusts the map.

In some cases, it is useful to place the SSD at the end of the virtual tiered volume's LBA range or split at the end and start. In this case, add the option ***fast_last*** or ***fast_split*** to the create command line e.g.

```
> ecmd --create /dev/sdb /dev/sdc fast_last
```

For both fast and slow first initialization, the fast and slow devices capacities are fully utilized e.g. if a 1TB SSD is tiered with an 8TB HDD, then the resulting size is 9TB (minus approximately 2G per disk for

metadata storage). If the SSD is expected to be removed at some point in the future and replaced with an updated version (or simply removed altogether), the ***reserve*** option ensures that only the slow tier is counted toward the total usable capacity so the user volume is unaffected by the removal.

Create using the reserve mode as follows:

```
> ecmd --create /dev/sdb /dev/sdc reserve
```

Note, it is possible to convert from the full capacity modes to the reserve mode and visa versa. See the convert option described later.

5.6 Version and General Status Information

The following is used to display eitehr version or status information about any currently configured tiers in the system:

```
> ecmd --version

> ecmd --status
```

Using our example from earlier, the following status output is produced.

```
$ ecmd --status
Host MicroTiering Engine, Version 1.5.1.19859
  Tiered Disks        : 1 (2 max)
  Physical Disks      : 2 total, 2 used (127 max)
  Global Tiering      : Enabled
  Time Running        : 0d:0h:48m:45s
  Global Scan Period  : 2 seconds
  Total Capacity      : 230 GiB/256 TiB (Available: 255 TiB)
  Fast Capacity       : 474 GiB/32 TiB (Available: 31 TiB)
  Page Map Memory     : 23 MBytes (24067 KBytes)
  Default Page Size   : 4M

Host Luns:
 ID ST TYPE      NAME     PRODUCTID        SIZE     SECT RAM   FAST  RSVD  PAGE
                                           GiB      SIZE CACHE PAGES PAGES SIZE
 ================================================================================
 0  OK tDrive 0  /dev/eba T00 VirtualSSD   230      512  OFF   205%    51%    4M

tDrives:
 ID ST NAME             SIZE  M   VD   IN PLCY ----------- PROMOTE --------
                        GiB       F,S            THR NEW     COUNT       PPS DIV
 ================================================================================
 0  OK T00 VirtualSSD   230   T   0, 1 RF RDIO    1  0       0           16  0

vDrives:
 ID ST CFG T  D  SECTORS     SECTORS     SIZE     M  STR RQ CONFIG
                AVAIL       CONFIG      GiB
 ================================================================
 0  0  2   0  0  3b5e0000    3b5e0000    474      15  13  0 Tiered-Fast
 1  0  2   0  1  1cdc4000    1cdc4000    230      15  13  0 Tiered-Slow

pDrives:
 ID ST T  V  NAME     PRODUCTID        REV  SECTORS     SECTORS     SIZE
                                            AVAIL       USED        GiB
 ================================================================================
 0  2  0  0  nvme0n1  SM951NVMeSAMSU   5D0Q 0           3b5e0000    476
 1  2  0  1  sda      WDCWDS250G2B0A-0 X611 0           1cdc4000    232
```

Most of the information is self explanatory. The status information codes for the tDrives are follows:

- **ID**: Internal number used by the tiering software for the tiered volume .e.g tDrive0
- **ST**: Status: 1 - mounted and ok, F0 - failed, FE - detached from lun
- **M**: tDrives Mode: T - tiering mode, X - tiering turned off, S - Single Tier Mode. vdrives: DDF RAID mode
- **VD**: Vdrives used in tier: x,y - fast vdrive ID, slow vdrive ID
- **IN**: Init Mode (mode the tiered was created in): FF - fast first, SF - slow first, FP - fast part (20% fast first), CU - Custom Map, RF – Reserved Fast
- **PLCY**: Tiering policy: RDIO - read IOs, RWIO read+write IOs, RBLK read - blocks, RWBK - read + write blocks
- **THR**: Promote Threshold. IO if IO policy set or Tiered Blocks/Sectors if block based policy is set
- **NEW**: New promote pages on the slow tier identified in the last scan or partial scan (may fluctuate depending on mode and bredth of scanning
- **COUNT**: Total number of pages promoted for the tier so far (multiply by page size to get total amount moved)
- **PPS:** Current page promotes permitted per scan.
- **DIV:** Scan time divider. 0-same as global scan time, 1-skip 1 cycle, 2-skip 2 cycles, etc

5.7 Changing Promote Policies

The default promote policy is to promote on activity based on read IOs only. This is sufficient for most applications. However, it is often necessary to change this to accommodate either write activity or block level activity. See the Applications Guidelines for more details.

The following commands are available:

```
ecmd --policy rdio /dev/ebX

ecmd --policy rwio /dev/ebX

ecmd --policy rdblock /dev/ebX

ecmd --policy rwblock /dev/ebX
```

where ebX is the drive letter shown in --list for the tiered volume.

Example usage: Change the policy for `ebb` to a read plus write IO activity based decision, use:

```
ecmd --policy rwio /dev/ebb
```

5.8 Changing Promote Rates

As described earlier in the Applications Guidelines, the promote mode may be configured independently for each tiered drive in the system. Available commands are:

```
ecmd –promote aggressive /dev/ebX

ecmd –promote normal /dev/ebX

ecmd –promote slow /dev/ebX

ecmd –promote off /dev/ebX

ecmd –promote on /dev/ebX
```

In addition, it is possible set raw threshold values for the promote engine which correspond to specific IOPs or block activity thresholds. The general command is:

```
> ecmd --promote P /dev/ebX
```

Where ‘P’ is the specific threshold value to start promoting data and depends on the policy setting. For example, if the policy is set to read only, then P specifies the total number of IOs that must be achieved for a page to promote or if set to a block based policy (see later), then P is the number of sectors for that page.

For example, to change a VirtualSSD (/dev/ebb) with read IO policy set to normal, the following commands are used:

```
> ecmd --promote normal /dev/ebb
```

5.9 Displaying Statistics

The tiering engine divides the tiered volume (tDrive) into a series of pages which are setup during the create process. Valid values for setting the page size are 64K, 128K, 256K, 512K, 1M, 2M and 4M.

Given there are often several millions of pages per tiered volume, it is not practical to show statistics on a per page basis. At command line level, the tiering engine allows the user to display information on a summarized page-region basis where each region represents 1/64th of the overall tiered volume. For each region, which may consist of several thousand pages, the use is able to display read, write or block activity as well as determine current mapping information e.g. determine which parts of the tiered volume an SSD based tier is mapped to.

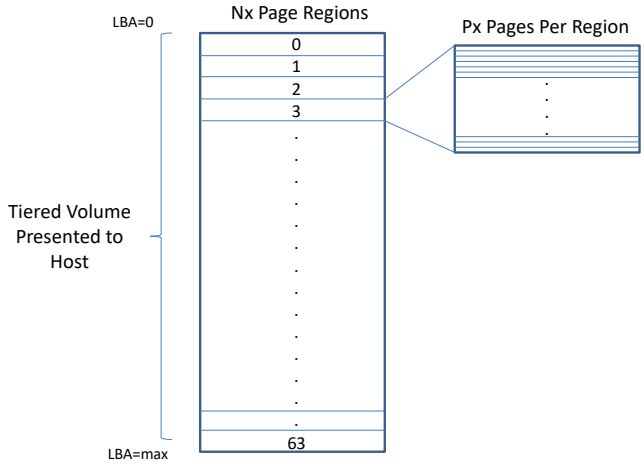

Figure 3: Page Regions for Displaying Stats

The following commands are available to support display of key operating statistics:

```
ecmd --stats promote_queue /dev/eba - displays the current prioritized
page activity list for tiered volume

ecmd --stats promote_hist /dev/eba  - displays recent promote activity

ecmd --stats count /dev/eba    - displays high level read, write and
                        block counts across the tiered volume page
                        regions

ecmd --stats map /dev/eba      - displays high level fast and slow tier
                        distribution across the tiered volume page
                        regions

ecmd --stats hostio /dev/eba   - displays high level host io statistics

ecmd --stats off /dev/eba      - turns off the statistics counters
                         (useful during formats)

ecmd --stats on /dev/eba       - turns on the statistics counters
```

`ecmd --stats convert /dev/eba - display status of any current convert operation`

An example of the stats map command is shown below:

```
$ ecmd --stats map drive5
Ecmd will now do drive discovery, this can take a while.
Getting statistics for drive5 (tDrive 0)
Enmotus tDrive[0] vdrive Pagemap Log:
  Page Regions        : 64
  Page Region Size    : 7913 pages (30 GiB)
  Promote Pool Region : 0
  Pool vPage0 Minimum : 468 (1d4h)
          ---- pages ----                        ---- pages ----
RGN   %    Tier0   Tier1    WT          RGN    %   Tier0   Tier1    WT
================================================================================
[ 0]+ 38%> bc7     1322     17c         [ 1]+ 27%  865     1684     10e
[ 2]+ 33%  a75     1474     14a         [ 3]+ 19%  5f7     18f2     be
[ 4]+ 18%  5ae     193b     b4          [ 5]+ 22%  715     17d4     dc
[ 6]   0%  10      1ed9     0           [ 7]   0%  1       1ee8     0
[ 8]   0%  1       1ee8     0           [ 9]+  6%  1f2     1cf7     3c
[10]   5%  1a4     1d45     32          [11]   0%  27      1ec2     0
[12]   0%  15      1ed4     0           [13]+  9%  2f9     1bf0     5a
[14]+ 10%  318     1bd1     64          [15]   3%  fa      1def     1e
[16]   4%  154     1d95     28          [17]   2%  d3      1e16     14
[18]+ 27%  898     1651     10e         [19]+ 17%  58e     195b     aa
[20]+  7%  233     1cb6     46          [21]   4%  15f     1d8a     28
[22]   3%  11e     1dcb     1e          [23]+  8%  2ae     1c3b     50
[24]+ 19%  60c     18dd     be          [25]   0%  0       1ee9     0
[26]+ 13%  40a     1adf     82          [27]+ 20%  670     1879     c8
[28]+ 16%  528     19c1     a0          [29]   4%  17e     1d6b     28
[30]+ 20%  671     1878     c8          [31]   0%  1       1ee8     0
[32]   0%  3       1ee6     0           [33]   5%  1b8     1d31     32
[34]   0%  0       1ee9     0           [35]   0%  2       1ee7     0
[36]   0%  3       1ee6     0           [37]   0%  0       1ee9     0
[38]   0%  0       1ee9     0           [39]   0%  2       1ee7     0
[40]   0%  3       1ee6     0           [41]   0%  0       1ee9     0
[42]   0%  0       1ee9     0           [43]   0%  2       1ee7     0
[44]   0%  3       1ee6     0           [45]   0%  0       1ee9     0
[46]   0%  0       1ee9     0           [47]   0%  2       1ee7     0
[48]   0%  3       1ee6     0           [49]   0%  0       1ee9     0
[50]   0%  0       1ee9     0           [51]   0%  2       1ee7     0
[52]   0%  3       1ee6     0           [53]   0%  0       1ee9     0
[54]   0%  0       1ee9     0           [55]   0%  2       1ee7     0
[56]   0%  3       1ee6     0           [57]   0%  0       1ee9     0
[58]   0%  0       1ee9     0           [59]   0%  2       1ee7     0
[60]   2%  ed      1dfc     14          [61]   0%  0       1ee9     0
[62]   0%  0       1ee9     0           [63]   2%  e2      1e07     14
```

5.10 Displaying Host and Metadata IO Activity Summary

A useful command for checking total activity to each virtual disk within the tier (i.e. fast/tier 0 and slow/tier1) is the --stats hostio option. An example output for an active tier is shown in the table following and for a tiered disk "/dev/eba", shows the total number of reads and writes made to each of

```
$ ecmd --stats hostio /dev/eba
Ecmd will now do drive discovery, this can take a while.
Getting statistics for drive5 (tDrive 0)
tDrive[0] Host-Metadata access counts:
                  Reads                          Writes
Fast Tier (0), /dev/nvme0n1:
  Host IOs      : 124004                         22173
  Host MBs      : 7668028                        968096
  Metadata IOs : 18621                           13230
  Metadata MBs : 2700364                         2584260

Slow Tier (1), /dev/sda:
  Host IOs      : 3479                           1150
  Host MBs      : 102798                         194555
  Metadata IOs : 18606                           13230
  Metadata MBs : 2696437                         2584260

tDrive[0] Host Total IOs by Size:
                  Reads                          Writes
 <=4K             59895                          13190
 <=32K            34597                          5254
 <=128K           32140                          4773
 <=1M             0                              0
  >1M             0                              0

tDrive[0] Host Total Trim Commands: 0
```

the tiers at the block layers.

From this, we can see clearly where the data is being written to the device, including hidden/metadata traffic (i.e. non-user) as well as a summary of the total requests by size. Furthermore, it is also possible to see the total number of trim commands written to the device where supported.

5.12 Setting Page Size

The tiering engine uses a global page size setting of 4M bytes, unless specified otherwise. To setup a tiered disk with a different page size, PRIOR to creating the tiered disk, reset the global page size using:

```
> ecmd --pagesize <value><unit>
```

where valid value is any value that is a multiple of 64K between 128K and optional unit may be K or M.

Example:

```
ecmd --pagesize 1M
```

All tDrives created from this point on will use the new value. Existing or previously created tDrives will continue to use their old value. It is not possible to convert between page sizes once a tDrive has been created.

5.13 File Pinning Utility (efile)

The tiering engine is often best left automatically promoting. However, it is often preferable to manually pin files to either tier directly. Note, this feature may not be available on all versions of the Enmotus software or OEM versions, but maybe available as a licensable upgrade.

The utility efile is used for promoting or demoting files between the tiers manually and the help menu is obtained by simply typing the following on the command line at a root level or sudo command prompt:

```
> efile --help

  efile --version    Prints out version information
  efile --promote  <file(s)> Promotes one or more files
  efile --demote   <file(s)> Demotes one or more files
  efile --release  <file(s)> Releases one or more files
  efile --mapping  <file(s)> Checks the file mapping of one or more
                       files
  efile --tracking <file(s)> Checks the tracking status of one or more
                       files
```

For example, to check if a file ***example.txt*** on a mounted tiered volume is located on the fast or slow tier, use the following:

```
> efile --mapping example.txt
```

The amount of the file that resides on the fast tier will be shown as a percentage e.g. 80% if partially promoted, 0% if entirely on the slow tier or 100% if entirely on the fast tier. To promote a file that is partially or entirely on the slow tier to the fast tier, use the following:

```
> efile --promote example.txt
```

If the file is partially or fully on the fast tier, to demote to the slow tier use the following:

```
> efile --demote example.txt
```

5.14 Example Tiered Volume Setup

Note: As the Enmotus engine is a block based tiered engine, it will monitor all basic IOs that is sees to the physical volumes it is managing. For singular events such as formatting the tiered volume, while it doesn't affect long term operation, it is sometimes beneficial to turn off promotes while formatting the

volume if aggressive policies and/or *readwrite* are in effect and turn them back on afterward. This prevents unnecessary promotes for non-application activity.

The following example illustrates how to setup a tier using a single SSD and single hard drive, presented to the system as /dev/sdb and /dev/sdc respectively. It assumed the user has sudo level privileges or can log in as root user.

Creating the tiered volume:

(a) Check the Raw Physical Devices are Available

```
$ ecmd --list
List system devices:
Device       Vendor       ProductId             Type       Rev        GiB
-------------------------------------------------------------------------
 sdd          WDCWD30      EZRZ-00Z5HB0          SATA HDD   80.00A80   2794
 sdc          ST1000DM     010-2EP102            SATA HDD   CC43       931
*sdb          WDCWD10      01X06X-00SJVT0        SATA SSD   01.01A01   111
 sda          WDCWDS2      50G2B0A-00SM50        SATA SSD   X61130WD   232
 nvme0n1      NVMe         SM951NVMeSAMSU        PCIE SSD   5D0Q       476
 nvme1n1      NVMe         P50D7020A120....      PCIE SSD   08K0       1863
```

(b) Load the Tiering Engine Manually (only necessary if the service is not started)

```
$ sudo ecmd --load /dev/sd[bc]
Loading Enmotus Tiering Module
Loaded OK
Adding /dev/sdb as pDrive[0]
Configuration Detected:  Physical: 0  Virtual: 0  Tiered: 0

Adding /dev/sdc as pDrive[1]
Configuration Detected:  Physical: 0  Virtual: 0  Tiered: 0

Scanning 2 disk(s) for existing configuration ...
pDrive port[0] size=0x1000000, name=/dev/sdb, 8GiB
pDrive port[1] size=0x1800000, name=/dev/sdc, 12GiB
Total disks used = 0, available = 2
```

(c) Creating the Tiered Volume

```
[root@localhost ~]# ecmd --create /dev/sda /dev/sdd
Testing drive performance...
  /dev/sda read 1024 Kbytes in 9 ms at 111.11 Mbytes/sec
  /dev/sdd read 1024 Kbytes in 1004 ms at 0.99 Mbytes/sec
Drive added successfully
Drive added successfully
Creating new tiered drive
Initializing pDrive[0]
Initialized pDrive[0] /dev/sda ok
Creating vDrive: mode=15 (left=2) f=0
  Assigned Single-disk vDrive number 0
  Setting vDrive[0] size to 484196352 sectors (0x1cdc4000) [pDrive left=0llx])
  Starting vDrive[0] at 0 sector (0x0) offset, size=0x1cdc4000)
  vDrive[0]: partition 0 added to pDrive[0] /dev/sda
  vDrive[0] Size : 230GiB, 484196352 Sectors
     Page Size   : 8192 (2000h)
     Total Pages : 59106 (E6E2h)
     Sector Size : 512
Initializing pDrive[1]
Initialized pDrive[1] /dev/sdd ok
Creating vDrive: mode=15 (left=1) f=0
  Assigned Single-disk vDrive number 1
  Setting vDrive[1] size to 5856337920 sectors (0x15d10a000) [pDrive left=0llx])
  Starting vDrive[1] at 0 sector (0x0) offset, size=0x15d10a000)
  vDrive[1]: partition 0 added to pDrive[1] /dev/sdd
  vDrive[1] Size : 2792GiB, 5856337920 Sectors
     Page Size   : 8192 (2000h)
     Total Pages : 714885 (AE885h)
     Sector Size : 512
Creating tDrive: mode=32
Setting up the tDrive using sector size 512 page size 8192 sectors ...
  vDrive[0] uses SSD in fast tier
  tDrive[0]: 2792 GiB sectors=5856321536 (15D106000h) Sectorsize=512 Mode=32
          : Active Pages=714883 (AE883h) Max Pages=773989 (BCF65h)
  Total Memory Allocated for tDrive[0]: 45616552 bytes, 43MB
Initializing vDrive Page Maps for tDrive[0], Init Mode = 7 ...
Updating metadata on all initialized disks...
Registered hlun[0] with host system:
  ID          : ENMOTUS  T00 VirtualSSD
  Serialnum   : b25dc3b2
  Rev         : 1.5
  Size        : 2792GiB, 5856321536 Sectors
  Sector Size : 512  Physical Sector Size : 4096
Attached tDrive[0] as block device eba
Ecmd will now do drive discovery, this can take a while.
T00VirtualSSD, /dev/eba successfully created
```

(d) Unloading the tiered volume:

```
$ sudo ecmd --unload
Unloading Tiering Module
Unmounting eba    Enmotus tDrive0   Tiered Disk   15
umount: /dev/eba: not mounted
Unloaded OK
```

(e) Reloading the tiered volume – specific disks:

```
$ sudo ecmd --load /dev/sdb /dev/sdc
Loading Enmotus Tiering Module
Loaded OK
Adding /dev/sdb as pDrive[0]
Configuration Detected:  Physical: 1  Virtual: 1  Tiered: 1

Adding /dev/sdc as pDrive[1]
Configuration Detected:  Physical: 2  Virtual: 2  Tiered: 1

Scanning 2 disk(s) for existing configuration ...
pDrive port[0] size=0x1000000, name=/dev/sdb, 8GiB
pDrive port[1] size=0x1800000, name=/dev/sdc, 12GiB
Loading tiered volume configuration from metadata
lun[0] => tDrive[0] 33552384 sectors 512 ssize 15GiB
Registered hlun[0] with host system:
  ID        : ENMOTUS  T00 Enmotus Disk   Rev: A006
  Size      : 15GiB, 33552384 Sectors
Total disks used = 2, available = 0
```

(f) Reloading tiered volumes using autoscan

Autoscan option automatically scans all system devices and load devices detected with Enmotus metadata automatically:

```
$ sudo ecmd --autoload
```

6.0 Advanced User Commands

6.1 Attach and Detach

It is sometimes useful for debug purposes to forcibly attach or mount a broken tDrive to a host lun for diagnostics purposes or simply detach/unmount it from the file system without destroying the tier.

The following commands are used:

```
> ecmd --attach t=0 {force}   attach tDrive0 as a /dev/ebx device if
                              currently offline. Use the force option if
                              the disk is in an error'd state.

> ecmd --detach t=4           detach the tDrive4
```

6.2 Loading and Initializing Tiered Volumes Manually

WARNING: Some of the following command(s) will remove or delete configuration information. Backup all important data before using.

The load, create and delete commands described earlier are the usually sufficient to setup or tear down a single tiered volume. However, it is sometimes necessary to manually setup one or more tiered volumes in discrete steps to support specific setup options.

To setup a tiered volume manually, several additional commands are used that first add disks one at a time, initialize the disks with a new metadata structure, then create a tiered volume from the component disks.

The following commands are available:

```
> ecmd --add /dev/<dev>             add a disk <dev> to the tiering
                                    engine driver

> ecmd --clear_meta {/dev/<dev>}    clear metadata off all the added
                                    pDrives or a specific pDrive

> ecmd --init /dev/<dev> {force}    initialize disk with an undefined
                                    metadata structure and force it to
                                    override any legacy disk protection
                                    if 'force' specified

> ecmd --create tDrive /dev/<fast-device> /dev/<slow-device>
                                     create tiered volume

> ecmd --rescan_meta                forces the driver to reload it's
                                    configuration from metadata all
                                    newly added pDrives
```

For example, to load a prior configuration on sdb and sdc without bringing the tier on line and clear it of all metadata, use the following:

```
> sudo ecmd --load
> ecmd --add /dev/sdb
> ecmd --add /dev/sdc
> ecmd --clear_meta
> sudo ecmd --unload
```

To load a tier manually one disk one at a time to verify a disk is loading ok, use:

```
> sudo ecmd --load
> ecmd --add /dev/sdb
> ecmd --add /dev/sdc
> ecmd --rescan_meta
```

6.3 Resetting a Loaded Tiered Volume Maps and Statistics

To reset a tiered volume's statistics gathering only without changing the data configuration on the tDrive, use the following:

```
> ecmd --reset stats /dev/ebX    resets a tDrive's statistics only.
                                 This is a non-destructive command.
```

The above command may be used on a running tiered volume as the virtual mapping is not changed by the reset command.

WARNING: The following command(s) will remove or delete configuration information. Backup all important data before using and also dismount the tiered volume from the file system.

6.4 Check Integrity of tdrive Metadata

The tiering engine automatically checks the state of the metadata on each drive in the system. As this can take a long time for virtual pooled disks, it cycles around each of the member pDrives in the vdrives or pools and compares the stored metadata tables with the active tables in RAM.

To force an immediate manual run of all or one of the disks in the tiered volume, use the following:

```
> ecmd --check /dev/ebx        checks tDrive 'n' metadata for each of
                               the member pDrives
```

6.5 Reset Error Flags

The status display will show if there are any errors immediately after the global status if a global error, or under the individual tDrive listing if related to a single tDrive.

To reset these error flags, use the following:

```
> ecmd --reset error            resets the global VSP level error flags

> ecmd --reset error /dev/ebX   resets the tDrive /dev/ebX error flags
```

See Appendix B for the error code meanings.

6.6 Manually Change the Scan Timer

It is possible to manually set the scan timer which tells the tiering engine how often to check for promote candidates and kick off a promote cycle. The ecmd --promote options described earlier sets up default values, however the default scan timer value may be overwritten using the following:

```
> ecmd --scantime <value> <s|m|h|d> t=N   sets the global scan timer
                                          e.g. --scantime 10s = 10 seconds;
                                          maximum is 7 days. If t=N
                                          (N=tDrive number) present, sets
                                          just for a particular tDrive.
```

6.7 Freezing Promotes and Stats Engine

It is often beneficial to freeze the state of the promote engine or particular tiered volume while performing maintenance functions or formatting the volume for the first time.

To stop a tiered volume from moving data automatically or preventing file pinning operations, use the following:

```
> ecmd --promote off /dev/ebX       # use t=N in place of /dev/ebX
```

To also freeze the statistics gathering for a particular tier use:

```
> ecmd --stats off /dev/ebX
```

6.8 Resetting a Tiered Map in Reserve Fast Mode

The current software supports the ability to reset a reserve fast mode VirtualSSD's map so that all active pages are moved off the fast tier back to the slow tier device(s).

This is accomplished using the following:

```
> ecmd --reset vmap driveN
```

Depending on the number of active pages that exist on the fast tier, this may take a while.

APPENDIX A: VirtualSSD Error Codes

Global Error Bits:

Bit	Hex Val	Description
0	1h	Fatal Error - cannot recover
1	2h	VMAP error detected on a tDrive
2	4h	Memory error during tDrive create or load
3	8h	tDrive failed
4	10h	tDrive vmap errors were recovered
5	20h	RSVD
6	40h	RSVD
7	80h	DDF I/O Error
8	100h	DDF CRC error was detected
9	200h	Maximum capacity error (license parameter exceeded)
10	400h	License keep alive timer has expired
11	800h	License 24 hour grace period timer has expired (demotes begin)
12	1000h	eboot FEh code error (client/bootable only)
13	2000h	pDrive I/O error
14	4000h	promote error detected
15	8000h	License parameter error (non-capacity)
15+		RSVD

tDrive Error Bits:

Bit	Hex Val	Description
0	1h	DDF I/O error
1	2h	Promote error
2	4h	Out of range error
3	8h	VMAP compare error
4	10h	VMAP checksum error
5	20h	VMAP error but was recovered
6	40h	VMAP flags error
7	80h	Pageregion error
8	100h	Memory allocation error
9	200h	Capacity max error (license parameter exceeded)

APPENDIX B: Additional Installation Instructions

If you received your software distribution as standard distribution packages, then use the following installation steps:

For Red Hat and CentOS 6.x distributions, type the following to install the kernel level tiering drivers using yum:

```
> sudo yum install kmod-en-tier-XXXX-Y.enmotus.x86_64.rpm
```

and the following to install the user level utilities.

```
> sudo yum install entierd-XXXX-Y.enmotus.x86_64.rpm
```

where XXXX-Y is the Enmotus version number. For example:

```
> sudo yum install entierd-4694-1.enmotus.x86_64.rpm kmod-en-tier-4694-
                        1.enmotus.x86_64.rpm
```

Alternatively, use rpm to install as follows:

```
> sudo rpm -ihv entierd-XXXX-1.enmotus.x86_64.rpm kmod-en-tier-XXXX-
                        1.enmotus.x86_64.rpm
```

Once installed successfully, start the Enmotus tiering engine service using:

```
> sudo service entierd start
```

Note: OEM versions may differ slightly in name e.g. entierd-{revision}-{oemname}.x86_64.rpm.

APPENDIX C: Creating VirtualSSD and Preserving Data on one Drive

VirtualSSD software supports creation of a VirtualSSD while preserving data on one drive. This is helpful in the situation where a HDD is running slower than desired and needs more performance. Another application is when an SSD is running out of space and needs more capacity, but only a portion of the SSD is being actively used.

The data to be preserved can be either the fast or slow portion of a VirtualSSD. This feature is performed by using the "ecmd –create" command. Note, this operation uses a different command line syntax "ecmd" instead of "ecmd."

The drive that contains the partition to be preserved will be recognized by the presence of the partition. The other drive in the VirtualSSD must not contain a partition. The VirtualSSD creation process will detect the partition that is to be retained and the user will be prompted to confirm the partition is to be preserved.

See example command and console response below:

This example adds an SSD to the existing HDD with a pre-existing partition

```
[root@centos-6u5-test19 Btest]# sudo ecmd --create /dev/sde /dev/sdb
Ecmd will now do drive discovery, this can take a while.
*** Drive /dev/sdb has a partition and Drive /dev/sde has none.
You can create a tier and preserve the data on /dev/sdb?
Create tier preserving data? (y/n) > y
Create tier preserving drive /dev/sdb data
                   O
                   O
                   O
Tier successfully created and data preserved.
```

`/dev/sde` **is the fast device**
`/dev/sdb` **is the slow device**

Before FuzeDrive Creation:

```
[root@centos-6u5-test19 Btest]# ecmd -list
Ecmd will now do drive discovery, this can take a while.
System drives:
 Device       Vendor       ProductId             Type       Rev         GiB
------------------------------------------------------------------------------
 sdb          TOSHIBA      MK2002TSKB            SATA HDD   MT2A        1863
 sde          KINGSTON     SH103S3120G           SATA SSD   524ABBF0    111
 sdd          TOSHIBA      THNSNC256GBSJ         SATA SSD   CJGA0202    238
 sdc          HGSTHDN      724040ALE640          SATA HDD   MJAOA5E0    3726
*sda          Hitachi      HDS721010KLA330       SATA HDD   GKAOA70F    931

Enmotus pDrives:
 ID ST  T  V NAME     PRODUCTID        REV  SECTORS     SECTORS     SIZE
                                            AVAIL       USED        GiB
 ============================================================================
```

* **Indicates the Boot Drive**

\> **Indicates the Drive is used in a FuzeDrive**

After FuzeDrive Creation:

```
[root@centos-6u5-test19 Btest]# ecmd -list
Ecmd will now do drive discovery, this can take a while.
System drives:
 Device       Vendor       ProductId             Type       Rev         GiB
------------------------------------------------------------------------------
>sdb          TOSHIBA      MK2002TSKB            SATA HDD   MT2A        1863
>sde          KINGSTON     SH103S3120G           SATA SSD   524ABBF0    111
 sdd          TOSHIBA      THNSNC256GBSJ         SATA SSD   CJGA0202    238
 sdc          HGSTHDN      724040ALE640          SATA HDD   MJAOA5E0    3726
*sda          Hitachi      HDS721010KLA330       SATA HDD   GKAOA70F    931
 eba          ENMOTUS      T01FuzeDrive          VDD  TDD   P001        1970

Enmotus pDrives:
 ID ST  T  V NAME     PRODUCTID        REV  SECTORS     SECTORS     SIZE
                                            AVAIL       USED        GiB
 ============================================================================
  0  2  1  0 sde      KINGSTONSH103S31 524A 0           db94000     111
  1  2  1  1 sdb      TOSHIBA MK2002TS MT2A 0           e8a08000    1863
```

* **Indicates the Boot Drive**

\> **Indicates the Drive is used in a FuzeDrive**

APPENDIX D: Manually Starting and Stopping VirtualSSD Core Components

The tiering engine will automatically load if installed using the standard Enmotus installer as described earlier. Alternatively, it may also be started, stopped or queried using the standard service system commands (shown here for Red Hat/CentOS distributions):

```
> sudo service entierd stop              stop the Enmotus service

> sudo service entierd start             load and start the Enmotus
                                         service

> sudo service entierd status            show status of the service
```

To aid in system evaluation, it is also possible to load and unload the tiering engine using ecmd. If required, use the following commands as system admin or via sudo:

```
> ecmd --autoload             scans the system for devices with Enmotus
                              metadata and auto loads them

> ecmd --load <device list>   loads with specific devices ignoring any
                              other devices
```

Example load command:

```
> ecmd --load /dev/sdb /dev/sdc

> ecmd --load /dev/sd[b-f]
```

If an existing tiered disk configuration exists on the component devices, it will automatically load the configuration from metadata stored on the drives and the tiered volume will be registered and mounted with the host storage block device handler as /dev/eba (or /dev/ebb, ebc if multiple tDrives are defined). Note, when loading one or more ***existing*** tiered volumes defined in metadata, the order of the devices does not matter i.e. the disks will automatically be assembled in the correct order.

Once all the appropriate devices have been manually loaded, to load the tier, use the following:

```
> ecmd --autoscan
```

If no devices are specified i.e. `ecmd --load`, the tiering engine will simply load into the system without creating any tiered devices and they must be manually added as described later.

To manually unload the tiering engine entirely, as system admin or via sudo use:

```
> ecmd --unload
```

This will unload all configured tiered volumes unless they are still mounted, then unload the tiering engine modular drivers. Tiered volumes unloaded this way will be available for use later and will retain complete state information, including through a system power cycle and may be reloaded using the --load <fast-device> <slow-device> option described earlier.