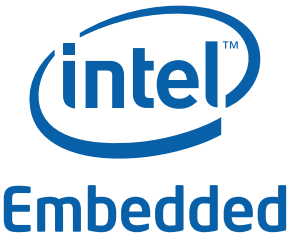


White Paper

Ramu Ramakesavan
(Technical Marketing Engineer)
Patrick Thomson
(Software Engineer)

Intel Corporation

Intel® Rapid Storage Technology (Intel® RST) in Linux*

August 2011

326024

Executive Summary

OEMs have been using Intel® Rapid Storage Technology (Intel® RST) in Microsoft Windows* client and server Operating Systems for many years. This software RAID solution has been used primarily on mobile, desktop, and workstation platforms and, to a limited extent, on server platforms.

The recommended software RAID implementation in Linux* is the open source MD RAID package. Intel has enhanced MD RAID to support RST metadata and OROM and it is validated and supported by Intel for server platforms. There is a growing interest at OEMs in having Intel extend the validation and support for RST on mobile, desktop and workstation platforms in a Windows and Linux dual-boot environment.

Intel enhanced MD RAID to support RST metadata and OROM and it is validated and supported by Intel for server platforms. There is a growing interest from OEMs in having Intel extend the validation and support for RST on mobile, desktop and workstation platforms in a Windows and Linux dual-boot environment. This paper describes the Intel® Rapid Storage Technology support in Linux.

This paper describes the support for Intel® Rapid Storage Technology in Linux.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel's tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. www.intel.com/embedded/edc.

Contents

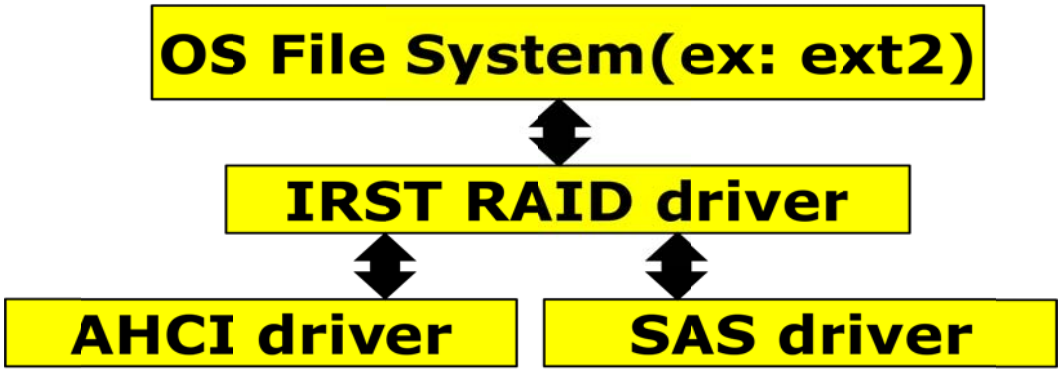
Intel® RST support in Linux*

The Linux implementations of software RAID support a number of RAID volume types including the Intel Rapid Storage Technology RAID volume type. The primary benefit of using Intel RST is in the presence of an Intel RST option ROM where the system can boot directly from any Intel RST RAID volume type instead of creating a dedicated partition or using a RAID superblock partition to store the bootloader.

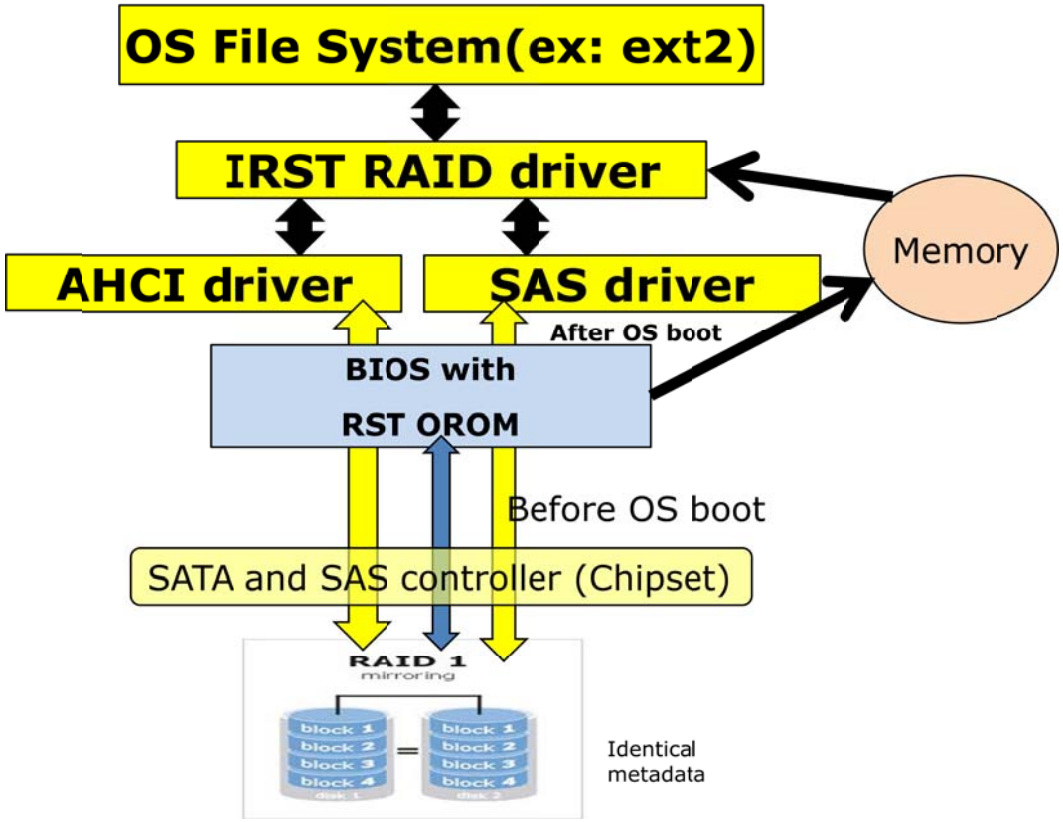
The following sections describe the different aspects of Intel RST support in Linux.

Support for RST Architecture in Linux

In simple terms, MD RAID in Linux is a block driver that filters data between the Linux File System driver, such as ext2 file system, and the low level hard disk drivers, such as the AHCI driver and SAS driver.



There is an option ROM (OROM) component in the BIOS that can create Intel RST RAID volumes and serves as the interface to the Intel RST RAID volumes in the pre-boot environment. Before the BIOS passes control to the system bootloader, the OROM leaves a copy of the features it supports, such as RAID 5, in system memory. This data can be read by mdadm to determine what features can be used when creating an Intel RST volume.



Linux Software RAID Implementations

There are two Linux Software RAID implementations: DM RAID and MD RAID. DM RAID is the legacy product and MD RAID is the newer implementation. Both implementations can support AHCI SATA and SCU controller in the Intel chipsets, and RAID metadata supported by RST OROM in the BIOS. However, all Intel development efforts are focused on MD RAID as it has many machine years of testing, and is widely accepted in the industry. Support for DM RAID is restricted to bug fixes.

Platforms and Linux Distributions Supported by Intel

MD RAID is supported by the PCG Linux team based in Poland. The Intel team validates and supports only server platforms using the Red Hat and SUSE Linux distributions. However, the MD RAID package is included in other Linux distributions also and can be used by Intel® Architecture mobile, desktop, and workstation platforms.

As of this publication, MD RAID is supported in the following distributions:

Distributions	Versions
Red Hat	RHEL 6.0, 6.1 and 6.2
Novell	SLES 11.0, 11.1 and 11.2

Comparison of DM RAID and MD RAID

DM RAID	MD RAID
Beginning with Linux* kernel version 2.6.18*, the dmraid* utility 1.0.0-rc15 supports RAID 0, RAID 1, and RAID 10.	Beginning with Linux kernel version 2.6.27*, the mdadm* utility 3.0 supports RAID 0, RAID 1, RAID 10, and RAID 5.
RedHat 5.3, 5.4, and 5.6 support DM RAID for 2009 and 2010 Intel server platforms.	RedHat 6.0, 6.1, and 6.2; SUSE SLES 11.0, 11.1, and 11.2 support MD RAID.
Lacking features.	Rich set of RAID features available. See section on MD RAID features for details.
Lacks synchronous metadata updates.	Supports synchronous metadata updates. Intel collaborated on external metadata architecture.
Only basic functionality support (RAID creation, deletion, simple array rebuild)	Wide range of supported features: creation, deletion, auto array rebuild, instant initialization, level migrations, capacity expansion, checkpointing etc.
Product deprecated. Support restricted to bug fixes.	Development efforts are focused on this product.

RST features in MD RAID

MD RAID implements the following features:

Feature	# of disks supported	Comments
Protocols supported		
AHCI	6	Allows Native Command Queuing and Native Hot Plug. Driver supported by Intel.
SAS	128	Driver supported by Intel
OROM/BIOS features supported		
Supports Intel metadata		Default mode. If the environment variable IMSM_NO_PLATFORM is set to ONE then MD RAID will skip all platform constraints and is used in non-Intel platforms.
MD RAID obeys constraints set in OROM		RST OROM copies BIOS settings to system memory that is read by the OS. MD RAID obeys all configuration data copied by RST OROM, such as “Device ID toggle for RAID5”

RAID features supported		
RAID 0	2-6	See http://www.intel.com/support/chipsets/iaa_raid/sb/CS-009337.htm for details.
RAID 1	2	See http://support.intel.com/support/chipsets/iaa_raid/sb/CS-009338.htm for details.
RAID 5	3-6	See http://support.intel.com/support/chipsets/imsm/sb/CS-020653.htm for details.
RAID 10	4	See http://support.intel.com/support/chipsets/imsm/sb/CS-020655.htm for details.
MBR/ GPT partitions; support for>2T drives		RST OROM and MD Driver support both MBR and GPT partitions. GPT partitions are required to supported drives larger than 2T.
Matrix RAID		Two RAID volumes on the same disk arrays.
Various RAID management functions are supported.		
RAID Creation & Removal		Creation and removal of RAID volumes by writing metadata onto the drives.
RAID Assembly		Starting (assembling) creates RAID volume from disks and exposes it as new device (in /dev directory, ex. /dev/md126)
Online Capacity Expansion (OLCE) allows reshaping of RAID volumes and migration of RAID level for a particular volume. With the "online" feature, the operation can be performed while a file system on top of the RAID volume is mounted. This allows the avoidance of having down time from taking the RAID volume offline for service. It consists of RAID Reshape.		
RAID Reshape		RAID capacity expansion using existing or spare drives.
The migration operation allows changing of the volume RAID level without loss of data stored on this volume.		
RAID level migration	2/3/4 RAID 0 to 3/4/5/6 RAID 5; 2 RAID 1 to 3/4/5/6 RAID5; 4 RAID 10 to 3/4/5/6 RAID 5	RAID level migrations do not require re-installation of the operating system. All applications and data remain intact. Only restricted migrations are allowed.
RAID Failure Management. Drives can be hot replaced or spare disks could be included in the volume for rebuild on failures.		
RAID Rebuild		A failed disk can be hot replaced and the software will automatically rebuild the RAID volume.

Array Auto rebuild		Same as RAID rebuild, but a spare drive in the array is used for rebuilding.
Bad Block Management		A limited number of bad block information is maintained at the end of the metadata. These are list of sectors that encountered media error during rebuild operation.
Read Patrol		Patrol read checks for physical disk errors that could lead to drive failure. These checks usually include an attempt at corrective action. See section on Read Patrol for details.
Checkpointing allows OLCE, Rebuild and Migration operation to continue after a failure such as a power failure, in the middle of the operation.		

Installation and Booting

Various Linux installers (Anaconda* in RedHat*, YaST* in SLES) have a specialized module that assembles RAIDs volumes using mdadm.

The following operations take place during booting:

1. OROM exposes the RAID volume as a separate device.
2. The BIOS reads the partition table, MBR or GPT and starts the boot loader such as GRUB.
3. The RAM file system is used to load the MD RAID driver and then RAID volumes are assembled.
4. The target root file system on the RAID volume is mounted and then OS is booted.

Managing RAID Volumes

The following sections describe commands used to create, assemble, and stop RAID volumes, as well as commands to manage failed drives.

Creating a RAID volume

RST support uses the concept of a CONTAINER device, which is a collection of devices that are managed as a set. The set of devices may contain up to two RAID arrays created on the same set of disks. For example, four devices in a 5-device set might form a RAID5 using first part of each device. The remaining space of devices might have a RAID0. One remaining device may be used as spare device. There is one set of metadata that describes all of the arrays in the container. So when mdadm creates a CONTAINER device, the device just represents the metadata. Other normal arrays can be created inside the container.

To create a RAID volume, use the following steps:

Warning: Creating a RAID volume will permanently delete any existing data on the selected hard drives. Back up all important data before beginning these steps.

1. First a container of Intel RAID metadata must be created.

```
mdadm  -C /dev/md/imsm /dev/sd[b-d] –n 3 –e imsm
Continue creating array? y
mdadm: container /dev/md/imsm prepared.
```

The command creates a RAID container with Intel® RST metadata format. The device node for the container will be /dev/md/imsm. In this example disks sdb, sdc, and sdd are used for this RAID set, and the total number of disks is 3.

2. Next we create the RAID volume `/dev/md/vol0`

```
mdadm -C /dev/md/vol0 /dev/md/imsm –n 3 –l 5
mdadm: array /dev/md/vol0 started.
```

The command creates a RAID volume `/dev/md/vol0` within the `/dev/md/imsm` container. Three disks are being used for the RAID volume creation. A RAID level 5 volume is requested.

#

The following command parameters may also be used in conjunction to give finer control for the creation of the RAID volume.

`-n` : Number of active RAID devices to be used in the volume.

`-x` : Specifies the number of spare devices in the initial array.

`-c` : Specifies the chunk (stripe) size in Kibibytes. The default is 512KiB.

`-l` : Specifies the RAID level. The options are 0, 1, 5, 10.

`-z` : Specifies the size (in Kibibytes) of space dedicated on each disk to the RAID volume. This must be a multiple of the chunk size. For example:#

#

```
mdadm –C /dev/md/vol0 /dev/md/imsm –n 3 –l 5 –z $((100*1024*1024))
```

The command above creates a RAID volume inside the /dev/md/imsm container with 100GB of data on each disk member.

Creating RAID configuration file

A configuration file can be created to record the existing RAID volumes. The information can be extracted from the existing RAID setup. The configuration file is typically stored at the default location of /etc/mdadm.conf. This allows a consistent assemble of the appropriate RAID volumes.

```
mdadm -E –s –-config=mdadm.conf > /etc/mdadm.conf
```

Volume Assemble

Inactive RAID volumes can be activated using the assemble option with mdadm.

The following command scans for the mdadm configuration file at /etc/mdadm.conf in order to assemble the RAID volumes: #

```
mdadm –As
```

Stopping the volumes

The following mdadm command can be used to stop all running RAID volumes provided they are not mounted:

```
mdadm –Ss
```

However, RAID volume names can be specified to stop the volume directly.

To fail an active drive

In order to mark an active drive as a failed drive (or set as faulty) manually, the following command can to be issued:

```
mdadm –f /dev/md/vol0 /dev/sdb
```

Remove a failed drive

To remove a failed drive, the following command needs to be executed. This only works on a container based RAID volume.

```
mdadm –r /dev/md/imsm /dev/sdb
```

Erasing RAID meta data

Having incorrect and bad RAID metadata can cause RAID volumes to be assembled incorrectly. The metadata can be erased with the following command to make sure the disk is clean. This operation does not attempt to wipe existing user data.

```
mdadm --zero-superblock /dev/sdb
```

Multiple disks can be specified to clear the superblock

Adding Spare disk to the RAID volume

Adding a spare disk allows immediate reconstruction of the RAID volume when a device failure is detected. Mdraid will mark the failed device as “bad” and start reconstruction with the first available spare disk. The spare disk can also be used to grow the RAID volume. The spare disks sit idle during normal operations. When using mdadm with IMSM metadata, the spare disk added to a container is dedicated to that specific container. The following command adds a spare disk to the designated container.

```
mdadm -a /dev/md/imsm /dev/sde
```

Online Capacity Expansion

Online Capacity Expansion (OLCE) allows reshaping of RAID volumes and migration of RAID level for a particular volume. With the "online" feature, the operation can be performed while a file system on top of the RAID volume is mounted. This allows the avoidance of having down time from taking the RAID volume offline for service. The file system on the volume should be expanded to the new volume size; however, not all file systems may support this functionality.

RAID reshape

With a running RAID volume, we can grow the size of the volume by expanding on existing unused disk space available to the RAID volume or by adding additional disks to the RAID container. With available disk space the following command can be issued to grow the RAID volume. This is assuming that we either have additional room to grow or additional disk has been added to the IMSM container.

```
mdadm –G /dev/md/vol0 –n 4
```

RAID migration

The migration operation allows changing of the volume RAID level without lost of data stored on this volume. The following table shows the available migration support with IMSM metadata.

Table 1 Migration capabilities with IMSM

Destination → ↓Source level	Non-RAID drive	RAID 0	RAID 1	RAID 10	RAID 5
Non-RAID drive	N/A	Yes	Yes	Yes	Yes
RAID 0	No	N/A	No	No	Yes
RAID 1	No	Yes	N/A	No	Yes
RAID 10	No	No	No	N/A	Yes
RAID 5	No	No	No	No	N/A

The following command allows the migration:

```
mdadm –G /dev/md/vol0 –l 5
```

Reporting RAID information

Report RAID details from BIOS

To see what Intel® RAID support is provided by the BIOS issue the command:

```
mdadm --detail-platform
       Platform : Intel(R) Matrix Storage Manager
        Version : 8.9.0.1023
    RAID Levels : raid0 raid1 raid10 raid5
    Chunk Sizes : 4k 8k 16k 32k 64k 128k
      Max Disks : 6
    Max Volumes : 2
 I/O Controller : /sys/devices/pci0000:00/0000:00:1f.2
          Port0 : /dev/sda (3MT0585Z)
          Port1 : - non-disk device (ATAPI DVD D  DH16D4S) -
          Port2 : /dev/sdb (WD-WCANK2850263)
          Port3 : /dev/sdc (3MT005ML)
          Port4 : /dev/sdd (WD-WCANK2850441)
          Port5 : /dev/sde (WD-WCANK2852905)
          Port6 : -no device attached -
```

Reporting RAID information

To print out details about a RAID container or volume, the following command can be used:

```
mdadm –D /dev/md/imsm
```

To print out RAID details about a member disk:

```
mdadm –E /dev/sdb
```

RAID monitoring

Mdadm provides the ability to monitor “metadata event” occurring such as disk failures, clean-to-dirty transitions, and etc. The kernel provides the ability to report such actions to the userspace via sysfs, and mdadm takes action accordingly with the monitoring capability. The mdmon polls the /sys looking for changes in the entries *array_state*, *sync_action*, and per disk *state* attribute files. This is meaningful for RAID1, 5 and 10 only.

Starting the monitor manually

The mdadm monitor, mdmon, is automatically started when MDRAID volumes are activated by mdadm through creation or assemble. Mdmon can also be started manually:

```
mdadm --monitor –daemonise –i /var/run/mdadm
```

The command above runs mdmon as a daemon to monitor all md devices. It also writes the process id (pid) of the mdmon daemon to `/var/run/mdadm` file.

There are additional command line parameters that can be passed to mdmon at startup.

Table 2 mdmon Parameters

Long form	Short form	Description
--mail	-m	Address to mail alerts or failures to.
--program or --alert	-p	Program to run when an event is detected.
--delay	-d	Seconds of delay between polling state. Default is 60s.
--config	-c	Specify a different config file.
--scan	-s	Find mail-address/program settings in config file.
--oneshot	-1	Check for degraded arrays and then exit.
--test	-t	Generate a TestMessage event against each array at startup.

mdmon will check the mdadm.conf config file to extract the appropriate entries for monitoring. The following entries we can set to pass to mdmon:

MAILADDR: This config entry allows an e-mail address to be used for alerts. Only one email address should be used.

MAILFROM: This config entry sets the email address to appear from the alert emails. The default from would be the “root” user with no domain. This entry overrides the default.

PROGRAM: This config entry sets the program to run when mdmon detects potentially interesting events on any of the arrays it is monitoring. There can be only one PROGRAM line in the config file.

Read Patrol

Patrol read checks for physical disk errors that could lead to drive failure. These checks usually include an attempt at corrective action.

You can enable Read Patrol scan by writing the string “check” to /sys/block/md###/md/sync_action, simply:

```
$ echo [check/repair] > /sys/block/md###/md/sync_action
```

where md### is MD device name (like /dev/md126)

Number of potential errors is stored in /sys/block/md###/md/mismatch_cnt. ‘repair’ setting does both checking and repair.

The above action performs Read Patrol only once. A cron job could be used to run Read Patrol periodically. Such a script is present in RHEL6.x releases.

Summary

MD RAID and DM RAID are both implementations of Intel® Rapid Storage Technology but MD RAID is richer in features and more widely adopted in the industry. No future features are planned to be implemented in DM RAID.

MD RAID implements a subset of features available in the Windows* implementation of Intel® Rapid Storage Technology using the same metadata. The Windows implementation has an administration UI, where administration in Linux is based on command lines. The Linux implementation has additional feature RAID Patrol, a RAS feature as it is primary used in server platforms. Since both Linux and Windows implementations use the same metadata, the RAID Arrays and Volumes can be accessed from both Linux and Windows in a dual-boot system.

The Intel® Embedded Design Center provides qualified developers with web-based access to technical resources. Access Intel Confidential design materials, step-by step guidance, application reference solutions, training, Intel’s tool loaner program, and connect with an e-help desk and the embedded community. Design Fast. Design Smart. Get started today. http://intel.com/embedded/edc.

Authors

Ramu Ramakesavan is a Technical Marketing Engineer with Datacenter and Connected Systems Group at Intel.

Patrick Thomson is a Software Engineer with Storage Group at Intel Corporation.

Acronyms

AHCI – Advanced Host Computer Interface

RST - Intel® Rapid Storage Technology

RAS – Reliability Availability and Serviceability

SAS – Serial Attached SCSI

SATA – Serial Advanced Technology Attachment

SCU – SAS Controller Unit

§