High-Level Design (HLD)

Revision 1.2
Last Updated: 1/21/2003 - 3:56 PM

SCSI RAS Tools
RAID-1 Enhancements for Linux

Primary Author(s): Andrew Cress

Key Contributors:

Legal Notices and Disclaimers

Disclaimers

Legal Notices

Abstract

This design document describes the SCSI RAS Tools RAID-1 Mirroring project for Carrier Grade Linux Enhancements using Linux 2.4 and later kernels.

Hard disks are the most common system element to be replaced, and are therefore a critical consideration in improving availability. Disk Mirroring (RAID-1) is the technique of using redundant disks to record multiple copies of the data so that a failure of one disk does not cause data loss. Linux currently provides some software RAID-1 functionality in the base kernel. These changes enhance the reliability, availability and serviceability of the various drivers that are commonly used in a Linux software RAID-1 configuration. A separate effort within Intel has been made to enable various common hardware RAID adapters and their drivers on Linux.

The implementation includes changes to the md, scsi, and aic7xxx kernel drivers. For the md (software RAID) driver, it adds additional hardening to increase reliability and enhancements to provide additional logging for serviceability. For the scsi driver, it adds additional hardening and enhanced logging and statistics capability to this layer. For the aic7xxx driver, additional hardening and enhanced logging capabilities are added. These enhancements also include added SCSI serviceability tools to update disk firmware, view & update mode pages, view defect lists, and perform diagnostic functions using Linux.

The Disk Mirroring enhancements reduce the amount of out-of-service time for disk problems via redundancy and in-service tools to diagnose problems and effect repairs to a failed disk while the system is operational under Linux.

Revision History

Revision	Date	Author	Reason for Changes
0.1	3/26/2002	Andy Cress	Release 1.2 Design for Disk Mirroring
0.2	4/12/2002	Andy Cress	More detail added
0.3	4/18/2002	Andy Cress	Title change, TOC, moved some links
0.4	5/15/2002	Andy Cress	More info on hot-insertion & RSM SCSI in section 4.10
0.5	6/07/2002	Andy Cress	Section 4.5.2: Updated sgraidmon.c methods Section 4.9: Show the statistics that rmscsi exposes
1.0	7/30/2002	Andy Cress	Edits for carrierlinux.org
1.1	9/12/2002	Andy Cress	Changed some terminology and links for CGLE sourceforge projects.
1.2	1/21/2003	Andy Cress	Cleaned up header/footer & added PCP reference

Table of Contents

1. Introduction

1.1 Purpose of this Document

The purpose of this document is to define the scope of the work to be done in improving the RAS features of the Linux Disk Mirroring Subsystem. The target environment for these improvements is for high availability systems using either SCSI or IDE disk drivers, and Linux software mirroring (RAID-1), although these improvements would also apply to other environments using the same improved components.

Since the disk subsystem is by far the most common hardware component to be replaced, any improvements that minimize outages for servicing these components will add significantly to the overall system Reliability, Availability and Serviceability (RAS).

It is our intention to make the targeted software improvements to the Disk Mirroring Subsystem freely available to the Linux open source community so that they may be included in future versions of Linux, and to make these improvements available to customers for their use as updates to existing versions of Linux in the meantime.

1.2 Document Scope

This design document deals with the implementation and improvement of software RAID capabilities within Linux 2.4. Anyone interested in understanding the Disk Mirroring internal design should read this document. This document does not deal with drivers for hardware RAID adapters, and these are addressed separately.

The Intel hardware RAID adapter drivers can be found on the support.intel.com site at the URLs in section 1.4. For other vendor hardware RAID adapter drivers, refer to either the Linux kernel source, or to the vendor web site.

1.3 Terminology

Acronym/ Abbrev.	Description
HLD	High-Level Design
CGLE	Carrier Grade Linux Enhancements
RAS	Reliability, Availability, and Serviceability
SCSI	Small Computer System Interface
RAID	Redundant Array of Inexpensive Disks
IDE	Integrated Drive Electronics (also known as ATA)

Acronym/ Abbrev.	Description
ATA	Advanced Technology Attachment (see IDE)
RSM	Resource Statistics Monitor (see project documentation in section 1.4)

Table 1-1: Definition of Acronyms used.

For additional terminology, see “Appendix A: Definitions”

1.4 Related Documentation

Document Name	Revision	Doc #
Carrier Grade Linux Enhancements from OSDL http://developer.osdl.org/		
CarrierLinux.org Project Repository http://www.carrierlinux.org		
Latest versions of Disk Mirroring documentation and beta software http://scsirastools.sourceforge.org/		
Driver Hardening and Manageability http://hardeneddrivers.sourceforge.net/	1.8	
Linux Resource Statistics Monitoring - Architecture Specification http://resourcemntrd.sourceforge.net/ See also: http://pcp4cgl.sourceforge.net	0.03	
Linux System RAS Projects on SourceForge: http://sourceforge.net/projects/systemras/		
SCSI Draft Standards: ftp://ftp.t10.org/t10/drafts/		
Justin Gibbs' Adaptec aic7xxx driver site http://people.freebsd.org/~gibbs/linux/		
General overview of RAID: http://www.pcguide.com/ref/hdd/perf/raid/index.htm		
Linux High Availability RAID http://www.linas.org/linux/Software-RAID/Software-RAID-9.html		
Linux SCSI Generic driver and utilities: http://gear.torque.net/sg/		
Adaptec SCSI Specifications (see esp. AIC-7899) http://www.adaptec.com/worldwide/product/prodtechindex.html?cat=%2fTechnology/SCSI&source=home		
LSI/Symbios SCSI Specifications (see esp. LSI 53C1030) http://www.lsilogic.com/products/stor_prod/oem/io/specs.html ftp://ftp.lsil.com/HostAdapterDrivers/linux/ (drivers)		
Linux SCSI Generic Interface http://www.torque.net/sg		
Neil Brown's mdctl & mdadm utilities http://www.cse.unsw.edu.au/~neilb/source/mdctl/		
Intel hardware RAID adapter drivers: http://support.intel.com/support/motherboards/server/srcu31/index.htm http://support.intel.com/support/motherboards/server/srcu311/index.htm http://support.intel.com/support/motherboards/server/srcu21/index.htm		

Table 1-2: Related Documentation

2. Methodologies and Notations

3. Assumptions and Dependencies

All of the code within the scope of this document is assumed to be open source. The kernel modules are under GPL license, and the user-space utilities are under BSD-style licensing.

Below is a list of maintainers related to this project from the Linux 2.4 MAINTAINERS list. The Linux Kernel Mailing List applies to all of these, at linux-scsi@vger.kernel.org.

Module Name	Maintainer Name	Email Address	Module Web Site
Adaptec aic7xxx driver	Justin Gibbs	gibbs@scsiguy.com	http://people.freebsd.org/~gibbs/linux/
sym53c8xx driver	Gerard Roudier	groudier@free.fr groudier@club-internet.fr	ftp://ftp.tux.org/pub/people/gerard-roudier http://www.lsilogic.com
SCSI Subsystem	Eric Youngdale	linux-scsi@vger.kernel.org eric@andante.org	http://www.kernel.org
SCSI SG Driver	Doug Gilbert	dgilbert@interlog.com	http://www.torque.net/sg
Software RAID (Multiple Disks) Support	Ingo Molnar Neil Brown	mingo@redhat.com neilb@cse.unsw.edu.au linux-raid@vger.kernel.org	http://linas.org/linux/raid.html http://www.cse.unsw.edu.au/~neilb/source/mdctl/
Logical Volume Manager	Heinz Mauelshagen	Linux-LVM@EZ-Darmstadt.telekom.de	http://linux.msede.com/lvm

Table 3-1 Linux Maintainers Related to Disk Mirroring

4. High-level Design

4.1 Design Decomposition

Below are the hardware and software components in the Linux Disk Mirroring subsystem.

User-space	**User Application, Tool, or Database**
Kernel	**Software RAID modules (md, raid1, lvm)**
Kernel	**SCSI Adapter driver**
Hardware	**SCSI Host Adapter**
Hardware	**SCSI Disk Device(s)**

Table 4-1 Disk Mirroring Components

The focus of this project is to improve the RAS qualities of the Linux Kernel modules that are used in Disk Mirroring for the target environment. The 'md' (multiple devices) and LVM (Logical Volume Manager) modules in Linux provide the RAID0, RAID1 or RAID5 capability. The most common SCSI host adapters for the target environment are the Adaptec SCSI 'aic7xxx' driver, and the LSI Logic (was Symbios) 'sym53c8xx' driver.

4.2 Component Descriptions

4.3 Internal Components

User-space	***other tools*** mdadm	***RSM scsi_event library***	***scsirastools*** sgdiag, sgmode, sgdskfl, sgdefects, sgraidmon
Kernel	***raid1*** (or ***raid5***, etc.)		
	md (multiple devices)		
	sg (SCSI Generic driver)	***sd*** (block driver for scsi disks)	
	scsi (mid-layer)		
	aic7xxx	***sym53c8xx***	
Hardware	***SCSI Host Adapter***		
	SCSI Disk Devices		

Table 4-2 Disk Mirroring Internal Components
The shaded areas are those covered by the Disk Mirroring feature.

The ***md*** kernel module is the core of the Linux software RAID feature. It provides redundancy and device mapping at the partition level for hard disk data. Software RAID also encompasses several other related modules: ***raid, raid0, raid1, raid5, lvm-mod, linear, multipath,*** and ***xor***. For the purposes of the scsirastools RAID-1 project, the critical modules are md and raid1.

Md	Core software RAID module in Linux, supports multiple physical spindles through a single logical device. Required for RAID and LVM.
Raid	Combine several hard disk partitions into a logical block device (/dev/md*).
raid0	striping data across multiple partitions/disks without any parity. Enhances performance, but there is no redundancy.
raid1	keeping a duplicate copy of one partition/disk on a standby partition/disk. Either disk can maintain data integrity if the other one is lost.
raid5	striping data across multiple partitions/disk including parity, so that removing any one partition/disk will not break the data integrity
Linear	mode that allows partitions to be appended together as logical devices.
Xor	checksumming parity logic used by the raid5 module.
lvm-mod	Logical Volume Manager provides a way to administer software RAID devices in logical volumes, and allowing the administrator to resize logical volumes.
multipath	Multipath-IO is the ability of certain devices to address the same physical disk over multiple 'IO paths'. The code ensures that such paths can be defined and handled at runtime, and ensures that a transparent failover to the backup path(s)

	happens if a IO errors arrives on the primary path. This is used in shared SCSI environments where there are multiple host adapters.

Table 4-3 Software RAID Kernel Modules

The ***scsi*** mid-layer kernel module encompasses the core scsi modules. This layer also contains these modules: ***sd*** for scsi disks, ***sg*** for SCSI generic access, and ***sr*** and ***st*** for CD-ROMs and tapes. These provide the framework for common SCSI functions used by all SCSI host adapters. Each host adapter driver provides device-specific routines in a format that the scsi module can use.

The ***aic7xxx*** kernel driver provides device-specific support for Adaptec SCSI chipsets in the AIC7xxx family. This driver supports all of Adaptec's PCI-based SCSI controllers, but not the hardware RAID controllers. It also supports the AIC7770-based EISA and VLB SCSI controllers. This is an Adaptec-sponsored driver written by Justin Gibbs. It is intended to replace the previous aic7xxx driver maintained by Doug Ledford since Doug is no longer maintaining that driver (labeled "aic7xxx_old" in Linux 2.4 kernels). See http://people.freebsd.org/~gibbs/linux/ for more information. Intel TSRLT2 server platforms use Adaptec 7899 chips on the motherboard, and this is a commonly used SCSI chipset.

The ***sym53c8xx*** kernel driver provides device-specific support for LSI/Symbios SCSI chipsets in the SYM53C8xx family. There are several versions of this driver: ***sym53c8xx_2, sym53c8xx,*** and ***ncr53c8xx***. For more information about these driver versions, see these files in the kernel source tree: drivers/scsi/sym53c8xx_2/Documentation.txt and drivers/scsi/README.ncr53c8xx. There are also drivers from LSI available at ftp://ftp.lsil.com/HostAdapterDrivers/linux/. One key target chipset to be used on some Intel server platforms will be LSI 53C1030. The ***sym53c8xx_2*** ver sym-2.1.17a driver, from the kernel.org source, is the latest version as of this writing, and it currently only supports up to LSI 53C1010.

The user-space ***tools*** shall be able to use the SCSI-generic driver to access SCSI devices directly. Of course, the utilities are run as needed and have no persistent resources. The SCSI Generic interface requires that the kernel configuration file have CONFIG_CHR_DEV_SG=y. The following utilities shall be implemented:

- **sgdskfl**
  This utility uses the SCSI Generic interface to update the disk firmware microcode for various types of SCSI disk and tape devices. The appropriate algorithm is chosen, depending on the vendor and product model of each SCSI device. A log file is created by default to track progress. The firmware microcode images are, by default, named so that they match the device model name, with the suffix ".lod". Safeguards are provided to scan the firmware image and verify that it matches the target device model. Measures are taken after the firmware is downloaded to make sure that the disk becomes ready again after it restarts with the new microcode (sometimes a "start_unit" command is needed). Depending on the type of device, this process takes approximately 90 seconds per device. The utility is intended to allow disk firmware to be upgraded while Linux is running. The device could be taken offline if it is part of a RAID, or left online if a 90-second delay is tolerable. Upgrading disk firmware has no impact on the data portion of the disk.

- **sgmode**
  This utility uses the SCSI Generic interface to read the SCSI device mode pages and also to write the mode pages. It first lists all SCSI devices in the system with all pertinent information. The user can choose one of these devices to read and/or write the desired mode page values. This utility is unique in that it reads all of the mode pages at once, and can set all of the mode pages from a mode format (*.mdf) file for each disk model. Thus, if mdf files exist for each disk model in the system, all of their mode pages could be set automatically in one pass. The logical capacity of the device can also be changed when writing the mode pages, and a special confirmation is required if a capacity change is about to occur, since a this can affect the data portion of the disk. By default, only mode pages are modified, so this utility can actually be run at any time during normal Linux operation. A log file is written by default, and the mode pages are always read before new values are written, so that the original values can be preserved if needed.
- **sgdefects**
  This utility uses the SCSI Generic interface to obtain the device defect lists. This is useful for analyzing the number of grown defects over time to predict when a device failure may occur. The utility can gather the number of factory and grown defects by querying the device, and optionally list each of the defects (in the log file). A log file is created by default for diagnostic purposes.
- **sgdiag**
  This utility uses the SCSI Generic interface to send specific SCSI commands to a given device for diagnostic, administrative, or testing purposes. A log file is created by default. Several different functions can be performed with this utility, such as:
  - Compose a SCSI command to send (for testing)
  - Reset SCSI bus (last resort for SCSI protocol errors)
  - Format SCSI disk (low level SCSI format, use with caution)
  - Test for common problems
- **sgraidmon**
  This utility uses the SCSI Generic interface to monitor the status of SCSI devices in the system and take action when a device is hot-removed or hot-inserted. It uses SAF-TE commands to obtain hot-insertion information from hot-swap backplanes, and polls the attached SCSI devices to look for any change in status. If a device is hot-removed, this utility invokes a script to remove the device's partitions from the software RAID. If a device is hot-inserted, a script is invoked to add it to the software RAID and remirror it, if that device's name is configured in /etc/raidtab. This utility can optionally run as a background daemon.
- **RSM scsi_event library**
  This is a library which reports SCSI statistics to a common Resource Statistics Monitoring daemon (StatSentry). This subsystem library is a loadable library which takes /proc/scsi/scsi statistics and some grown defect data (see sgdefects) and provides it in a common form to RSM. This library is merged in with that open-source project at http://sourceforge.net/projects/resourcemntrd.

4.4 Internal Data Structure Map

MODULE	DATA STRUCTURE REFERENCE
sgmode	SCSI-3 Primary Commands – section 8.3 Mode Parameters SCSI-3 Block Commands – section 7.1.3 Mode Parameters
sgdskfl	SCSI-3 Primary Commands – section 7.25 Write Buffer Command
sgdefects	SCSI-3 Block Commands – section 6.1.7 Read Defect Data
sgdiag	SCSI-3 Block Commands – section 6.1.1 Format Unit
sgraidmon	SCSI Accessed Fault-Tolerant Enclosures (SAF-TE) specification 1.0
sgerr.c	SCSI-3 Primary Commands – section 7.20 Request Sense Command
md	$kernel_source/Documentation/md.txt
scsi	$kernel_source/Documentation/scsi.txt $kernel_source/Documentation/scsi-generic.txt
aic7xxx	$kernel_source/drivers/scsi/README.aic7xxx
sym53c8xx_2	$kernel_source/drivers/scsi/sym53c8xx_2/Documentation.txt

4.5 Internal Methods

4.5.1 Scsiras kernel modules

Since each of the scsiras kernel components are existing modules in the kernel source, the scope of their methods is left to the kernel documentation above and the list of source files for each module are listed here.

md	drivers/md	linear.c, lvm.c, lvm-fs.c, lvm-snap.c, md.c, multipath.c, raid0.c, raid1.c, raid5.c, xor.c
scsi	drivers/scsi	scsi.c hosts.c scsi_ioctl.c constants.c scsicam.c scsi_proc.c scsi_error.c scsi_obsolete.c scsi_queue.c scsi_lib.c scsi_merge.c scsi_dma.c scsi_scan.c scsi_syms.c
sd	drivers/scsi	sd.c
sg	drivers/scsi	sg.c
aic7xxx	drivers/scsi/aic7xxx	aic7770.c aic7770_linux.c aic7xxx_93cx6.c aic7xxx.c aic7xxx_linux.c aic7xxx_linux_pci.c aic7xxx_pci.c aic7xxx_proc.c
sym53c8xx	drivers/scsi/sym53c8xx_2	sym_fw.c sym_glue.c sym_hipd.c sym_malloc.c sym_misc.c sym_nvram.c

4.5.2 Scsirastools user-space methods

Getmd.c	int findmatch(char *buffer, int sbuf, char *pattern, int spattern, char figncase) int getline(FILE * fd, char *buf, int len) int getmd(char *devpattn, int sdevpattn, char *mddev, char *mdpart) int main(int argc, char **argv)

Sgsub.c	int get_sense(int sts, uchar * buf) int sense_report(int sts, const char *errmsg, uchar * bufp) void i2h(uchar * chp, int len, char *str) void dumpbuf(FILE * fdout, uchar * bufp, int mlen) int get_serial(int sgfd, uchar * buf, int rlen) int scsi_inquiry(int sgfd, uchar * buf, int rlen) int test_unit_ready(int sgfd, char fshowmsg) int read_capacity(int sgfd, ulong * dsize) int write_buffer(int sg_fd, uchar * buf, ulong len, uchar mod, uchar bufid) int get_defects(int sgfd, uchar * buf, int len, char fplist) int mode_sense(int sgfd, uchar page, uchar * buf) int mode_select(int sgfd, uchar * buf, uchar len) int start_unit(int sgfd) int scsi_reset(int sgfd, int type) int scsi_format(int sgfd, uchar patt, int timeout, int noglist) int seagate_inquiry(int sgfd, uchar * buf, int rlen) int sn_inquiry(int sgfd, uchar pgcode, uchar * buf, int rlen) int send_scsicdb(int sgfd, uchar * wbuf, int wlen, uchar * rbuf, int rlen) int send_scsicdb1(int sgfd, uchar * wbuf, int wlen, uchar * rbuf, int rlen) int set_sg_debug(int sgfd, int dbglvl)

Sgcommon.c	void closefd(void);
	void closelog(void);
	void closeall(void);
	void quit(int rc);
	void itoh(uchar * chp, int len, char *str);
	int findmatch(char *buffer, int sbuf, char *pattern, int spattern, char figncase);
	uchar get_ndev(int first);
	uchar get_idev(void);
	uchar get_mdev(char *model);
	char get_func(void);
	void do_pause(void);
	void showit(char *buf);
	void dumpbufr(FILE * fdout, uchar * bufp, int mlen, char *hdr);
	void make_dev_name(char *fname, int k, int fnumeric);
	int get_ival(char *valstr, char fhex);
	int get_scsi_info(int sgfd, int idx, char *fname, int fservo);

Sgerr.c	void show_host_status(FILE * fdout, int host_status) void show_driver_status(FILE * fdout, int driver_status) void show_sense_err(FILE * fdout, const char *leadin, const unsigned char *sense_buffer, int sb_len) int sg_chk_err(FILE * fdout, const char *leadin, int masked_status, int host_status, int driver_status, const unsigned char *sense_buffer, int sb_len)
Sgdiag.c	int main(int argc, char **argv) int do_reset(int idx, int type) int do_sninquiry(int idx)

	int do_format(int idx, int noglist) int do_sendcdb(int idx) int beforegd(int idx, int *pnumrdy)
Sgdefects.c	void dumpdlist(FILE * fdout, uchar * bufp, int mlen) int main(int argc, char **argv) ulong cap_report(u_char * buf) void showmodebuf(FILE * fdout, uchar * bufp) int do_getdefects(int idx, char fvalues) int beforegd(int idx, int *pnumrdy)
Sgdskfl.c	int main(int argc, char **argv) ulong getimage(int idx, uchar ** pbuf, char fservo) int write_buffers(int sgfd, uchar * buf, ulong len, uchar fservo) int writeimage(int idx, uchar * imgbuf, ulong len, char fservo) int beforedl(int idx, int *pnumrdy) int afterdl(void)
Sgmode.c	int main(int argc, char **argv) ulong cap_report(u_char * buf) int do_modeselect(int idx) void showmodebuf(FILE * fdout, uchar * bufp) int do_modesense(int idx) int beforemd(int idx, int *pnumrdy) int aftermd(void)
Sgraidmon.c	int getmd(char *devpattn, int sdevpattn, char *mddev, char *mdpart) int read_safte(int sgfd, int mode, uchar *buf, int len) static int sg_cmd(int sgfd, uchar *cdb, int cdblen, uchar *data, int dlen) void make_dev_name(char *fname, int k, int do_numeric) int scanit(int bmode) int mkdaemon(int fchdir, int fclose) int main(int argc, char **argv)

4.6 System Dependencies & File Structures

The kernel configuration must have CONFIG_MD=y and CONFIG_BLK_DEV_MD=y. Root mirroring via software RAID will not work if md is built as a module. CONFIG_MD_RAID1 can be either =m or =y. Of course, most Linux kernels use a ramdisk image to initialize optional modules that may be required during boot, so CONFIG_BLK_DEV_RAM=y, CONFIG_BLK_DEV_INITRD=y, and CONFIG_BLK_DEV_LOOP =y or =m.

In the kernel configuration, CONFIG_CHR_DEV_SG must be =y to support the scsirastools. Also, the scsirastools support both numeric (/dev/sg0, /dev/sg1) and alphanumeric (/dev/sga, /dev/sgb) representation of sg devices. If only one representation is provided in a given Linux distribution, use the “-n” option with each tool switch device naming modes. RedHat supports both modes, while MontaVista supports only numeric mode. Hence, the default mode for the scsirastools will be numeric mode.

4.7 External Data Structures

The /proc interface provides a variety of data which is available to external user-space programs. Also, the mdadm utility provides the ability to examine the software raid superblock on a given disk partition. An example of the data that is available is shown below.

```
# cat /proc/mdstat
Personalities : [raid1] [multipath]
read_ahead 1024 sectors
md1 : active raid1 sdb1[1] sda1[0]
      72192 blocks [2/2] [UU]

md2 : active raid1 sdb5[1] sda5[0]
      136448 blocks [2/2] [UU]

md0 : active raid1 sdb6[1] sda6[0]
      8747264 blocks [2/2] [UU]

unused devices: <none>

# mdadm --examine /dev/sda1
/dev/sda1:
          Magic : a92b4efc
        Version : 00.90.00
           UUID : 2a069a6a:fabfbd17:869c5640:b1e930fc
  Creation Time : Fri Mar 22 11:58:40 2002
     Raid Level : raid1
           Size : 72192
     Raid Disks : 2
    Total Disks : 2
Preferred Minor : 1

    Update Time : Wed Apr 17 10:52:04 2002
          State : dirty, no-errors
  Active Drives : 2
 Working Drives : 2
  Failed Drives : 0
   Spare Drives : 0
       Checksum : 7fd237e6 - correct
         Events : 0.70


      Number   Major   Minor   RaidDisk   State
this     0       8       1        0      active sync   /dev/sda1
   0     0       8       1        0      active sync   /dev/sda1
   1     1       8      17        1      active sync   /dev/sdb1

# cat /proc/scsi/scsi
Attached devices:
Host: scsi0 Channel: 00 Id: 00 Lun: 00
  Vendor: IBM      Model: DPSS-309170N     Rev: S93E  Ser#: ZD139931
  Type:   Direct-Access                    ANSI SCSI revision: 03
  Tallies: timeouts 0 resets 0 par_errs 0 disk_errs 0 trans_errs 0 user_errs 1
Host: scsi0 Channel: 00 Id: 01 Lun: 00
  Vendor: IBM-ESXS Model: ST336605LW    !# Rev: B244  Ser#: 3FP0WHD4
  Type:   Direct-Access                    ANSI SCSI revision: 03
  Tallies: timeouts 0 resets 0 par_errs 0 disk_errs 0 trans_errs 0 user_errs 1

# cat /proc/scsi/aic7xxx/0
Adaptec AIC7xxx driver version: 6.2.4
aic7899: Ultra160 Wide Channel A, SCSI Id=7, 32/253 SCBs
Channel A Target 0 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
      Goal: 160.000MB/s transfers (80.000MHz DT, offset 63, 16bit)
      Curr: 160.000MB/s transfers (80.000MHz DT, offset 63, 16bit)
      Channel A Target 0 Lun 0 Settings
            Commands Queued 18921
            Commands Active 0
            Command Openings 253
            Max Tagged Openings 253
            Device Queue Frozen Count 0
Channel A Target 1 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
      Goal: 160.000MB/s transfers (80.000MHz DT, offset 63, 16bit)
      Curr: 160.000MB/s transfers (80.000MHz DT, offset 63, 16bit)
      Channel A Target 1 Lun 0 Settings
            Commands Queued 19305
            Commands Active 0
            Command Openings 64
            Max Tagged Openings 253
            Device Queue Frozen Count 0
Channel A Target 2 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 3 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 4 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 5 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 6 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 7 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 8 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 9 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 10 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 11 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 12 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 13 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 14 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 15 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)

# cat /proc/scsi/aic7xxx/1
Adaptec AIC7xxx driver version: 6.2.4
```

```
aic7899: Ultra160 Wide Channel B, SCSI Id=7, 32/253 SCBs
Channel A Target 0 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 1 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 2 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 3 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 4 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 5 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 6 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 7 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 8 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 9 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 10 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 11 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 12 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 13 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 14 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
Channel A Target 15 Negotiation Settings
      User: 160.000MB/s transfers (80.000MHz DT, offset 255, 16bit)
```

4.8 External APIs

```
sg device interface (see http://gear.torque.net/sg/ for more information):
      open(const char * filename, int flags)
      write(int sg_fd, const void * buffer, size_t count)
      read(int sg_fd, void * buffer, size_t count)
      close(int sg_fd)
      ioctl(int sg_fd, int command, ...)  [sg specific]
      poll(struct pollfd * udfds, unsigned int nfds, int timeout_ms)
      fcntl(int sg_fd, int cmd) or fcntl(int sg_fd, int cmd, long arg)
```

sd device interface (see scsiinfo for sample usage: ftp://tsx-11.mit.edu/pub/linux/ALPHA/scsi/)

```
fd = open(devicename, O_RDWR);
status = ioctl(fd, 1 /* SCSI_IOCTL_SEND_COMMAND */, buffer);
close(fd);
```

md interface used by mdadm (sample usage, see mdadm ref in section 1.4 for more information):

```
mdfd = open(dev, O_RDWR, 0);
ret = ioctl(mdfd, BLKGETSIZE, &size);
```

```
        ret = lseek64(mdfd, offset, 0);
        ret = read(mdfd, super, sizeof(*super));
        ret = write(mdfd, super, sizeof(*super));
        ret = fstat(mdfd, &stb);
        close (mdfd);
```

4.9 Design Strategies

SCSIRAS RAID-1 Mirroring enhancements included in release 1.0:

```
 * aic: Upgrade of the aic7xxx driver from version 6.1.7 to version 6.2.2
   See http://people.freebsd.org/~gibbs/linux/ for associated change history.
 * aic: Removed and handled 4 panic sites in aic7xxx
 * aic: Improved recovery from scsi parity errors
 * aic/scsi/md: Added calls to Enterprise Event Logging (via CONFIG_EVLOG).
 * scsi: Improved logging of check conditions and bus_resets in scsi.
 * scsi: Changed some error messages in scsi to be less informal
 * scsi: Added display of the device serial number to scsi messages.
 * scsi: Added a test_unit_ready retry after resets in scsi error handling.
 * scsi: Added serial number & tallies to /proc/scsi/scsi.
 * md: Added additional debug messages
 * md: improved raid1 error handling during resync/reconstruct
   The most significant one is that if you had a mirrored set in which all the devices failed, then write
   requests would never return, whereas they should return with an error.
 * md: fixed null pointer dereference oops in lvm code, where it referenced an invalid LV if /boot is not on
    the root fs.
 * md: more granular locking during md resync
 * md: fixed some unchecked pointer sites in md.c
 * md: improvements to resync code speed window
 * md: set md_notifier priority to 1 to avoid race condition with other notifiers it depends on in the stack
   with priority 0.
 * md: improved md error handling to not mark last disk faulty
 * md: Clarified resync message text
 * md: dont show a bug if we hotadd a disk in a new slot
 * md: if a faulty disk was removed, don't check it any more
 * tools: create sgdskfl, sgmode, sgdefects, and sgdiag tools for administering various disk models.
```

SCSIRAS RAID-1 Mirroring enhancements added in release 1.1:

```
 * aic: The driver in kernel 2.4.17 is now version 6.2.4 which includes some patches from TLT 1.0
 * scsi: add additional tallies to /proc/scsi/scsi
 * md: many of the patches backported into TLT 1.0 are already in kernel 2.4.17
 * md: validate superblocks before writing to avoid inconsistent values
```

SCSIRAS RAID-1 Mirroring features to be added in release 1.2:

- md/scsi: ***Support hot-insertion*** of disks into software RAID-1
  There is currently no support for this functionality in the Linux software RAID subsystem. This will entail leverage of some existing open-source tools as well as original code to implement the following functions:

  - *Hot-Insertion event recognition*. The software RAID subsystem can currently recognize removal events, but not insertion events. This will require utilization of the *hotplug* device interface and a monitoring function (*mdadm --follow*, *scsimon* and/or *sgraidmon*) to recognize the successful insertion of a SCSI disk. The monitoring function may need to utilize functions at either the scsi layer or the software RAID layer, or both. This may also require kernel fixes to allow the disk to be successfully hot-inserted.
  - Unique *device enumeration* (required for accurately recognizing transition events above). This could be accomplished leveraging the open-source *scsimap* utility, which also allows the SCSI serial number to be used to verify uniqueness.
  - *Preparation of the inserted disk* for remirroring (automatically set up partitions to match the disk to be mirrored). This will require a modified version of *fdisk* that supports saving partition information from one disk and restoring it to another disk. The *sfdisk* utility provides this functionality.
  - *Auto-initiating a remirror* on the inserted disk, based on certain criteria. This will utilize either the *mdadm* or *raidhotadd* existing open-source utilities.

  All of these functions should take place automatically by default. The goal is to make the software RAID functionality comparable to hardware RAID adapter functionality.

- sym: Support for ***LSI/Symbios SCSI Adapter*** chipset (sym53c8xx_2)
  There are three flavors of the Symbios driver currently in the Linux kernel source. The sym53c8xx_2 driver was chosen because it replaces the other two and includes the latest LSI chip support. There are only 5 calls to panic() in the current sym-2.1.17a version of this driver. It will be enhanced to hardening level 2, according to the Driver Hardening White Paper. The LSI 53C1030 chipset will be used on the motherboard of Tiger-4 and some future Intel platforms. Development and testing of this feature is dependent upon obtaining a system with the LSI 53C1030 chipset.
  UPDATE: This was postponed to release 1.3 due to scheduling issues.

- rm/scsi: Add a Resource Statistics Monitor ***scsi_event subsystem library*** to track errors and predict SCSI disk failures before they occur. There are a number of event tallies available via /proc/scsi/scsi from previous TLT changes. There is also a tool to extract the grown defect information from a disk drive (sgdefects). The scsi_event library will utilize these statistics to measure the change in events over time, and can provide a sample algorithm for empirically predicting when a disk device may fail from this data. When the library determines that a disk failure is imminent this will cause a Resource Statistics Monitor event (which should include enough information to uniquely identify the disk device and its symptoms), and the Resource Statistics Monitor will trigger whatever action the user has configured. Resource Statistics Monitor is a related open-source project at http://sourceforge.net/projects/resourcemntrd.
  The scsi_event library will utilize an algorithm in the RSM daemon to allow it to track trends and/or historical statistics (an RSM dependency). The statistics to be gathered for each disk device are: serial number, scsi timeouts, scsi resets, scsi parity errors, disk-related sense errors, transient sense errors, user-related sense errors, and number of grown defects on the disk. A fault management module (user-space) will then use a decision tree to determine if a failure is imminent, and if so, which part is the likely

cause. This fault management module will be an application which uses RSM, but will be separate from the RSM scsi_event library. The table below shows which statistics are most likely to trend up from which type of fault.

STATISTIC	SCSI Adapter fault	SCSI Cabling fault	Disk device fault	Software fault
SCSI Timeouts	Y	Y	*	Y
SCSI Resets	Y	Y	Y	Y
SCSI Parity Errs	Y	Y	Y	
Disk sense errors			Y	
Disk grown defects			Y	

Fault Indicator Matrix

* = only when accompanied by another statistic also.

RM subsystem libraries should not block, so that they do not affect the RM daemon. Therefore, the scsi_event library will do the following when it polls for various statistics:
1) read any waiting grown defect responses pending from the disk via scsi-generic.
2) send a new request for the number of grown defects to the disk via scsi-generic (non-blocking).
3) open/read/close /proc/scsi/scsi to obtain the other statistics (tallies & serial number).

Below are the statistics that the rmscsi subsystem library will expose for each device.

```
        ScsiEvent Subsystem
             No Resources: 2
                  rid: 0 /dev/sda
                  rid: 1 /dev/sdb
             No Statistics: 8
                  sid: 0 SCSI Timeouts
                  sid: 1 SCSI Resets
                  sid: 2 SCSI Parity Errors
                  sid: 3 SCSI Disk Errors
                  sid: 4 Grown Defects
                  sid: 5 Capacity
                  sid: 6 SCSI Transient Errors
                  sid: 7 SCSI User Errors
```

UPDATE: The RSM scsi fault management module portion of this was postponed to release 1.3 due to scheduling constraints.

- tools: ***standardize various pathnames*** for .mdf, .lod and .log files based on distributor best practices and the Linux standard at http://www.pathname.com/fhs/.

- tools: switch default /dev/sg* ***device mode*** from alpha to numeric in each tool. This will allow distributions such as MontaVista to function without specifying additional command line options.
- tools: add "***Send Diagnostic*** Command" to the sgdiag tool. This is based on SCSI-3 Primary Commands – section 7.23, and SCSI-3 Block Commands, section 7.1.1. The purpose of this is that the device will perform a self-test and report a sense error if any problems are detected.

4.9.1 Product Installation Strategy

When the kernel is installed, either from a binary rpm or from kernel source, the scsi and md modules will be part of the kernel. The adapter driver kernel configuration parameters should be enabled by default as modules for maximum flexibility, so that whichever adapter chipset is used by the target platform can be loaded. The initrd image allows these drivers to be loaded from the boot filesystem into a ramdisk so that these drivers are available when at boot time when the root disks need them.

In order to set up a software root mirror configuration on a Linux system, some distributors (such as RedHat) provide options in the standard GUI installation to set up a set of RAID-1 disk partitions. However, any Linux distribution can be configured for root mirroring by using the procedure outlined in the scsirastools UserGuide (see http://scsirastools.sourceforge.net/docs/UserGuide). Note that some values in the example procedure may need to be changed, depending on disk size, and whether other partitions are also present, such as the service partition.

4.9.2 Initialization and Shutdown Strategies

Upon initialization, there are two phases, first md loads itself into memory before the disks are online, then after the disks and associated drivers are ready, the md driver scans the partitions on all drives to see if any of them have a partition type of 0xfd, which signifies that it is intended for Linux raid autodetect. This check is made during the initial bringup of the Linux OS, so that the root filesystem (which may be Linux raid) can be mounted. The md driver then reads the superblock on each raid partition to see the status of the raid. If the superblock on one or more of the partitions is invalid or out of date, md uses the other disk (latest valid superblock), and marks the older/invalid partition as part of the raid, but "failed". It can be remirrored later, as the administrator desires, using utilities like *raidhotremove*, *raidhotadd*, etc.

During shutdown, md is invoked by the kernel at the end of the shutdown process to make sure that writes to all raid partitions are synchronized and all active devices have valid superblocks.

4.9.3 Interoperability and Compatibility Support

This software raid functionality should be compatible with any disk drive supported under Linux, and any SCSI driver supported under Linux. For the purposes of this project, some of the disk drives and SCSI adapters that are commonly used in servers are targeted.

SCSI Host Adapters targeted:

- Adaptec aic7xxx family (esp. 7899)
- LSI/Symbios 53c8xx family (esp. 53C1030)

SCSI disks targeted:

- Seagate ST39173 disks
- Seagate ST318452 and ST336752 disks
- Fujitsu MAN3184MP and MAN3367MP disks
- Hitachi DK32DJ-18MW and DK32DJ-36MW disks
- IBM DPSS series disks (DPSS-309170N)

4.9.3.1 Target hardware/software environment

The target platform for this software is a Linux 2.4 OS on an Intel-Architecture server. This software makes use of the SCSI bus, and assumes that SCSI devices are the default for internal hard disks. A Carrier Grade Linux Enhancements kernel is built upon the Linux 2.4.18 kernel.org base.

There are several tested hardware platforms for the SCSI RAS Tools feature:

- Intel TSRLT2/TSRMT2 rack servers (with Adaptec 7899 SCSI) that have space for 2 disk drives.
- Intel Itanium rack servers (with LSI 53C1030 SCSI) that will have space for 2 or more disk drives.
- Servers that have an externally attached disk unit with hot-plug capability.
- Other servers that have multiple internal SCSI disks

4.10 Key Design Decisions and Alternatives

The Resource Statistics Monitor SCSI Event subsystem library could have been implemented within the kernel, or in a special daemon, however, this function fit nicely into the Resource Statistics Monitor design, and the statistics had already been made accessible to user-space previously in scsirastools R1.0 and R1.1 enhancements. Therefore, this library will provide a consolidated view of the SCSI statistics over time, and sample application algorithms to analyze the data and predict when a given disk will fail. This approach gives the maximum flexibility to this functionality for customers who may have other factors or other algorithms used to determine when they want to replace a disk. Implementing it in user-space also does not impact the performance of the scsi subsystem in the kernel. The RSM SCSI Event subsystem library will be packaged with the Resource Statistics Monitor Daemon for the sake of convenience. A standard RSM application to utilize these statistics will be provided in scsirastools release 1.2, and a custom SCSI application with a pre-defined fault prediction algorithm will be provided in scsirastools 1.3.

The hot-insertion feature could be provided in the kernel or as a user-space daemon. The considerations are that adding a new device needs to be reflected in the kernel data structures for scsi and md modules, but that there may need to be customization of the decisions to be made when a disk is inserted into the machine. The automatic assumption is that the disk is meant to be formatted and remirrored, but there may be cases in which a disk is inserted with data that the customer wishes to preserve. If a daemon is used, the reformatting and remirroring could easily be disabled, if desired. Also, several existing tools (raidhotadd, mdadm, mapscsi) make it possible to interact with the kernel to make insertion of a new disk possible from user-space. Therefore, since it is possible, it is preferred to implement the hot-insertion feature in a user-space daemon. Further analysis indicates that it is feasible to do all of the required hot-insertion functions from a user-space daemon (e.g. sgraidmon). This daemon can enumerate the devices, and by keeping a list of their serial numbers, can detect when a SCSI device has been removed or inserted. It can also use SAF-TE commands to query the hot-swap backplane, if present, to show if there are any insertion events. The daemon then can match the sg/sd device name that was removed/inserted with a raid (md) device name, and invoke a script to partition and remirror the disk if it was inserted.

The support for LSI 53C1030 chips in the sym53c8xx_2 driver is not included as of this writing. It should not be too different from the LSI 53C1010 chips that are supported in that driver today. The 53C1030 support will probably be added by the driver maintainer before the release 1.3 schedule requires this to be done, but if not, and the schedule allows, additional work can be done to provide the support for 53C1030 prior to implementing the driver hardening features.

4.11 Product Requirements Document (PRD) Correlation

Below are the required items in the release 1.2 schedule as part of the SCSI RAS Tools project:

- SCSI Driver Hardening (includes fixes, POSIX Event Logging, Statistics, etc)
- Adaptec aic7xxx Driver Hardening
- RAID-1 (md) Driver Hardening
- Hot-plug disk support (including SAF-TE)
- Developer validation on IPF
- Intel TSRLT2 Platform Support
- User Documentation

Here is how I see those relating to the key added features in SCSI RAS Tools in release 1.2:

- Support for software RAID **hot-insertion** events
  includes device enumeration, bug fixes, mapping, and auto-formatting
  RSM **scsi_event subsystem library** to predict SCSI disk failures
  This driver hardening feature has two aspects: reporting, and predictive analysis. These two functions can (should) be implemented as discrete modules.
- Support of **LSI/Symbios SCSI driver** for other platforms, since this is the second-most common embedded SCSI chipset.

Requirements	**Design Implementation**
Itanium™ Processor Family Support – Features All features exist in open-source releases shall be available in CVS for these 64-bit platforms in additional to all new features developed for relase 1.2.	Itanium (64-bit) support will be utilized on the Intel server platforms, and should be enabled during the release 1.2 timeframe to prepare for new platform releases. Most software raid and scsi modules are not anticipated to have significant problems with 64-bit porting.
Platform Support – Platforms This project shall provide the support for the following upcoming carrier-grade server releases: • Intel TSRLT2 (2U DP Tualatin moving to DP Prestonia Q4'02) • Intel TSRMT2 (1U DP Tualatin moving to DP Prestonia Q4'02) • Intel Itanium2 servers (supporting 870 Chipset) This shall include: • Hardware enabling features, including processor feature, chipset, adapter drivers, etc. • Key enabling software features needed for the platforms or for key customers considering the platform	The aic7xxx driver is used for the 7899 chip on most of these platforms, but the LSI 53C1030 chip will be used on some of the Itanium2 platforms and will require a separate development/hardening effort.
Hardened Driver Support – Implementation The purpose of this device drivers hardening requirement is to add the enhancement to a existing device driver code to provide the HA capability such as:	**SCSI Driver Hardening** In Release 1.0 the AIC7XXX, SCSI, and MD drivers were Level 2 compliant, except

• Harden the driver to withstand the abnormal and stressed operation, • Provide robust operation in the case of failure, • Provide error reporting in the case of failure, • Provide system monitoring functions to this device driver for diagnostic or resource monitoring purpose, Each device driver listed above shall implement all level 1 and level 2 hardening tasks as specified here. Level 3 tasks shall be implemented if they are applicable to a specific device driver. These 3 levels are: **Level 1** • Comply with Coding Practices – all device driver code shall comply with the coding standards from the DH Whitepaper • Fault Injection Tested – all device drive shall be tested and passed with fault injection testing as specified in the Document "Fault Injection Testing Specifications". **Level 2** • Event Logging and Error Reporting (Use POSIX IEEE Std. 1003.25 service) • Statistic Reporting: o Provide hardware statistics data to support Ravenel resource monitoring feature via standard APIs. o Provide Real Time software statistics data to support Ravenel resource monitoring feature via standard APIs, o Supporting various statistic data types – counters, watermarks, and thresholds. o Provide configurable statistics data including configuration file options, runtime flags, runtime commands, or loading of an alternative drivers with compile time flags for diagnostic purpose. • Diagnostics Support o Routines/APIs to perform self test and report result o Routines/APIs to support of online diagnostics and report result o Routines/APIs to support of offline diagnostics and report result **Level 3** • Hot Swap o Comply with PICMG Compact PCI Hot Swap Infrastructure Interface Specifications, • Fault Recovery o Automated detect device driver fault, report	for the statistics not yet being reported via an RSM library. In Release 1.2, these drivers shall become Level 3 compliant.

errors, and recovery from the fatal error. ○ APIs to perform re-initialization and recovery the device driver error. • Dynamic configuration ○ The device driver shall be a dynamically loadable module so that device driver can be load/unload at the run time without re-boot the system.	

Table 4-9: Correlation of requirements to high-level design specification.

Appendix A Definitions

A. Definitions of Terms and Acronyms

Term	Definition
GPL	GNU Public License. The default license for Linux software, required for most Linux kernel modules.
RSM	Resource Statistics Monitor, as defined by the SourceForge project in the references.

Table A-1: Definitions

Appendix B Design Description Techniques

B. Execution Scenarios

B.1 Scenario 1 – Disk bad spot develops

If a bad spot develops on the disk media, it will start to show up as a sense error showing a successful (recovered) read/write with retries (e.g.: key=03/asc=17/ascq=01). If the mode pages are set properly (see sgmode), the disk will probably automatically reallocate the bad spot when it becomes an unrecovered error (e.g.: key=03/asc=17/ascq=06). If not, then the disk will report this unrecovered error to the application (e.g.: key=03/asc=11/ascq=00), and the log messages, along with the /proc/scsi/scsi information, will uniquely identify which disk has the bad spot, and where it is (info field of the logged sense report gives the LBA on the disk). The redundancy of the root mirror also comes into play at this point.
At this point the administrator would want to check the number of grown defects with sgdefects, and may choose to reformat the disk to reallocate any logged defects. He would do this by removing the disk from the mirror (mdadm, raidhotremove), then format the disk with sgdiag. The sgdefects tool would show that the bad spot had been added to the grown defect list afterwards. The disk could then be remirrored again, and the entire procedure occurred while the system was operational in Linux. Note that the remirror process is designed to only occupy a minimal amount of processing time so that critical user processes take precedence over it.

B.2 Scenario 2 – Systematic defect is discovered in disk configuration or firmware

If there have been several disk errors and the logs provided to the OEM vendor indicate a systematic problem, it is often resolved with either a mode page configuration change or a disk firmware upgrade. For the mode page changes, sgmode can be run with the new settings on all disks in the system automatically without taking the disks out of operation. For a disk firmware upgrade, it is recommended that the disks be taken out of the mirror in turn and sgdskfl should be used to upgrade the disk firmware. Again, no downtime is experienced by the system for this correction. Without these added tools and logging, however, each system would have to be taken out of service and require 30-60 minutes of downtime, with a great deal of service cost.
For each disk (2 per system) a script could be executed to:

1. Remove the disk from the active software RAID-1 (via raidhot* or mdadm commands)
2. Run sgdskfl to download the firmware to the disk.
3. Add the disk back into the software RAID-1 (via raidhot* or mdadm commands)
4. When the remirror is complete (check with /proc/mdstat), repeat for the other disk.

B.3 Scenario 3 – A disk fails to function

If a disk fails to function properly, sgdiag can be used to test its internal diagnostics for clues. Even if it fails to respond completely, since the disks are mirrored, no downtime is experienced, and the disk can be analyzed later for faults, along with the syslog or event log messages to determine the root cause of the failure. When the disk is replaced (either with normal service, or via hot-plug if supported), it can be remirrored in background (raidhotadd), while the system continues to operate. A further enhancement is shown when the scsi_event RSM library predicts a disk failure event before it happens, allowing administrators time to plan for replacement of the disk at their convenience.