SR Performance Analysis

S. Hopkins, M. Ennis
Coraid, Inc.

Summary. This paper analyzes the performance of the SR appliances, comparing specific results from an SR1520 and an SR1521 in multiple configurations. The SR1521 appliance eliminates many of the hardware limitations in its predecessor, enabling load balancing of the network interfaces which has the effect of nearly doubling the access throughput. For large disk configurations, the access rate of the SR1521 outperforms the SR1520 by as much as 194%. Local access rates for parity initialization and disk reconstruction are also increased by as much as 244%.

SR AoE Throughput Rates

The following statistics were achieved by averaging the results of three independent runs of ddt[1] for the given SR configuration. `Ddt` is a simple tool that writes and reads to a file through a file system to determine the throughput capability of the file system and underlying storage. For a full description of `ddt`, please see *Appendix A*. Figures A and B present SR throughput statistics using jumbo[2] and standard Ethernet frames repectively. All throughput values are in units of KiB/s. As presented, the SR1521 outperforms the SR1520 in all large array configurations.

The Linux client used for these tests was stuart, a dual CPU (dual core), system with 1GiB of RAM. The Linux kernel was 2.6.18, and the `aoe` driver was aoe6-44. The `.config` file for stuart's kernel is available for download[3]. The client network cards used the Intel 82546GB controller. For each configuration above, an XFS file system was placed on the resulting AoE device. The file system was mounted, and `ddt` was run against this mount point.

1 See Appendix A for a full description of ddt.

2 For an explanation of why we use an MTU of 4200, please see the Linux *EtherDrive*® FAQ entry 5.29, "Why does the Linux AoE driver only use 4132 byte Jumbo Frames?"

3 See Appendix B for URL

Both SR appliances were tested running the 20070111 release. For the SR1521 tests, two network ports on stuart were directly connected to both ports of the SR1521. For the SR1520 tests, one network port on stuart was directly connected to the leftmost (first) port of the SR1520. Due to a hardware limitation on the SR1520, using the right port can actually decrease performance. For more information on this issue, please see the Application Note *SR Redundancy and Throughput* at the SR support page[1].

SR Local Rates

The rate at which the SR is capable of rebuilding RAID5 parity is an important metric for comparison. This rate is relevant for RAID5 initialization during first time creation as well as after any power failure. The `when` command reports the rate of all local reconstructions with an estimated time to completion. The rates stated below are the amount of total data that is being processed per second. The SR1521 initialization rate is 225% better than that of the SR1520.

The SR1521 is capable of a rate of 488,112 KB/s:

```
SR shelf 21> list -l
 0 1070.527GB offline
  0.0   1070.527GB raid5 initing 425501
    0.0.0  normal       82.348GB 21.0
    0.0.1  normal       82.348GB 21.1
    0.0.2  normal       82.348GB 21.2
    0.0.3  normal       82.348GB 21.3
    0.0.4  normal       82.348GB 21.4
    0.0.5  normal       82.348GB 21.5
    0.0.6  normal       82.348GB 21.6
    0.0.7  normal       82.348GB 21.7
    0.0.8  normal       82.348GB 21.8
    0.0.9  normal       82.348GB 21.9
    0.0.10 normal       82.348GB 21.10
    0.0.11 normal       82.348GB 21.11
    0.0.12 normal       82.348GB 21.12
    0.0.13 normal       82.348GB 21.13
SR shelf 21> when
0.0 488112 KBps 0:25:59 left
SR shelf 21>
```

The SR1520 is capable of a rate of 215,613 KB/s:

```
SR shelf 20> list -l
 0 1070.527GB offline
  0.0   1070.527GB raid5 initing 1194
    0.0.0  normal       82.348GB 20.0
    0.0.1  normal       82.348GB 20.1
    0.0.2  normal       82.348GB 20.2
    0.0.3  normal       82.348GB 20.3
    0.0.4  normal       82.348GB 20.4
    0.0.5  normal       82.348GB 20.5
    0.0.6  normal       82.348GB 20.6
    0.0.7  normal       82.348GB 20.7
    0.0.8  normal       82.348GB 20.8
    0.0.9  normal       82.348GB 20.9
    0.0.10 normal       82.348GB 20.10
    0.0.11 normal       82.348GB 20.11
    0.0.12 normal       82.348GB 20.12
    0.0.13 normal       82.348GB 20.13
SR shelf 20> when
0.0 215613 KBps 1:28:57 left
SR shelf 20>
```

Extending the example to a 10.5TB RAID5 array over 14 750GB disks, the SR1521 will complete parity initialization in approximately 5 hours and 52 minutes. The same task on an SR1520 will complete in 13 hours and 28 minutes.

Another important metric is the rate at which a disk can be reconstructed. During disk reconstruction the array is susceptible to a double failure. Faster disk reconstruction rates directly correlate to reduced exposure to this risk. The rates stated below are the amount of total data being processed per second. The SR1521 disk reconstruction rate is 244% better than that of the SR1520.

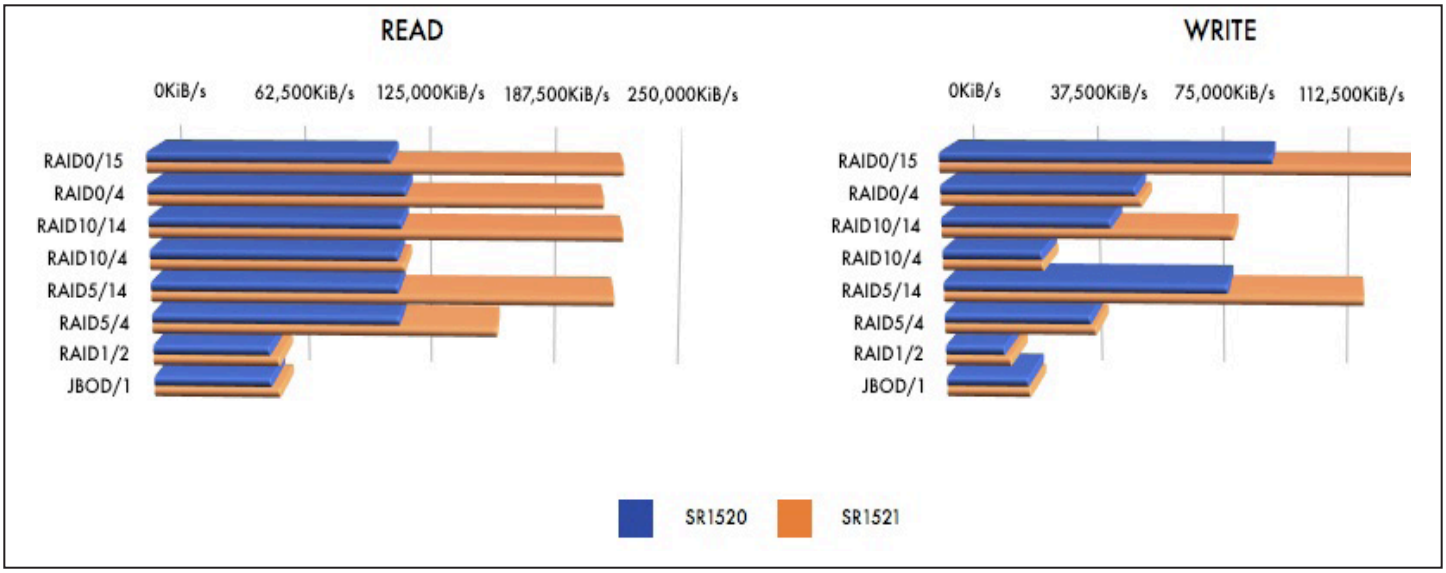
The SR1521 is capable of a rate of 563,347 KB/s:

```
SR shelf 21> list -l
 0 1070.527GB offline
  0.0   1070.527GB raid5 needrecover recovering degraded 1316
    0.0.0  normal       82.348GB 21.0
    0.0.1  normal       82.348GB 21.1
    0.0.2  normal       82.348GB 21.2
    0.0.3  normal       82.348GB 21.3
    0.0.4  normal       82.348GB 21.4
    0.0.5  normal       82.348GB 21.5
    0.0.6  normal       82.348GB 21.6
    0.0.7  normal       82.348GB 21.7
    0.0.8  normal       82.348GB 21.8
    0.0.9  normal       82.348GB 21.9
    0.0.10 normal       82.348GB 21.10
    0.0.11 normal       82.348GB 21.11
    0.0.12 normal       82.348GB 21.12
    0.0.13 replaced     82.348GB 21.13
SR shelf 21> when
0.0 563347 KBps 0:33:50 left
SR shelf 21>
```

The SR1520 is capable of a rate of 230,293 KB/s:

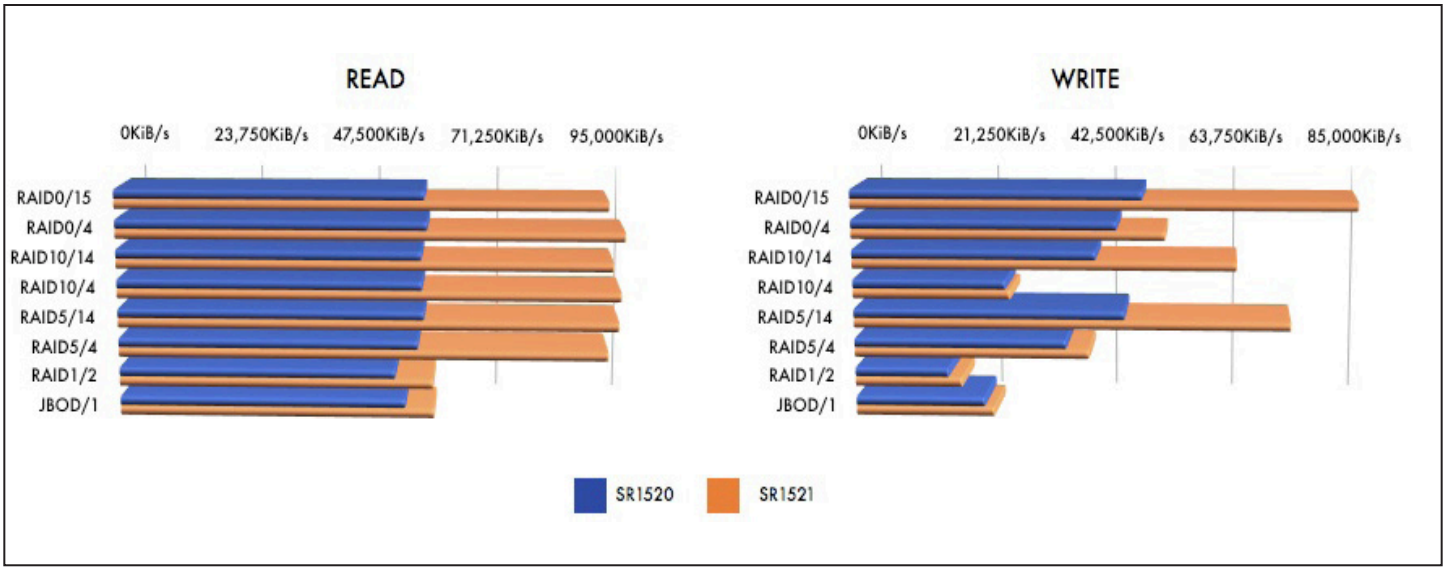
```
SR shelf 20> list -l
 0 1070.527GB offline
  0.0   1070.527GB raid5 needrecover recovering degraded 1015
    0.0.0  normal       82.348GB 20.0
    0.0.1  normal       82.348GB 20.1
    0.0.2  normal       82.348GB 20.2
    0.0.3  normal       82.348GB 20.3
    0.0.4  normal       82.348GB 20.4
    0.0.5  normal       82.348GB 20.5
    0.0.6  normal       82.348GB 20.6
    0.0.7  normal       82.348GB 20.7
    0.0.8  normal       82.348GB 20.8
    0.0.9  normal       82.348GB 20.9
    0.0.10 normal       82.348GB 20.10
    0.0.11 normal       82.348GB 20.11
    0.0.12 normal       82.348GB 20.12
    0.0.13 replaced     82.348GB 20.13
SR shelf 20> when
0.0 230293 KBps 1:23:19 left
SR shelf 20>
```

Extending the example as before, the SR1521 will complete recovery of a 750GB disk in a 10.5TB RAID5 array in approximately 5 hours and 10 minutes. The same task on an SR1520 will complete in approximately 12 hours and 40 minutes.



Model	KiB / s	RAID0 15 DISK	RAID0 4 DISK	RAID10 14 DISK	RAID10 4 DISK	RAID5 14 DISK	RAID5 4 DISK	RAID1 2 DISK	JBOD 1 DISK
1521	WRITE:	145,482.67	54,254.00	78,704.33	27,320.67	113,814.00	41,361.00	17,772.00	22,914.33
	READ:	212,323.00	203,633.67	212,401.33	115,040.33	208,849.67	156,546.00	57,612.00	57,706.33
1520	WRITE:	89,040.67	52,578.67	45,740.33	26,977.33	77,099.67	39,839.33	16,011.00	22,325.33
	READ:	109,062.33	115,582.00	113,621.67	111,786.67	111,989.33	112,440.33	53,302.67	53,674.00

Figure A. Performance Data: MTU 4200 (Jumbo)



Model	KiB / s	RAID0 15 DISK	RAID0 4 DISK	RAID10 14 DISK	RAID10 4 DISK	RAID5 14 DISK	RAID5 4 DISK	RAID1 2 DISK	JBOD 1 DISK
1521	WRITE:	81,922.00	50,971.67	62,564.00	25,701.33	71,408.67	38,548.67	17,551.67	22,823.67
	READ:	89,180.00	92,215.33	90,227.00	91,739.00	91,441.67	89,580.33	57,750.33	57,944.67
1520	WRITE:	47,299.67	43,103.00	39,672.00	24,935.67	44,337.33	34,794.00	15,300.00	21,240.00
	READ:	56,090.00	56,611.33	55,537.67	55,703.67	56,033.00	54,914.33	50,745.67	52,603.33

Figure B. Performance Data: MTU 1500 (Standard)

This page intentionally left blank.

Appendix A – Performance Analysis with ddt

Performance analysis of the SR series of appliances has in the past been performed using bonnie++ as the benchmark. When used to analyze SR throughput and client system resource usage, bonnie++ has limitations: it only reports the CPU utilization for the bonnie++ user process, it does not report utilization properly in the face of multiple CPUs, it inflates write throughput by omitting a data sync operation as part of the write test, and it does not give the SR time between the write and read stages to flush its dirty buffers to avoid having previous writes affect the reads. The last item is something for which bonnie++ can not be expected to account. We have written a new program, ddt, to overcome these limitations.

Essentially, ddt is dd with timing information. No attempt has been made to make ddt accept the same options or command line syntax as dd. To obtain accurate CPU utilization, ddt uses a 2.6 Linux kernel proc file. As a result, ddt may not run correctly on 2.4 Linux kernels.

```
[root@stuart ddt-1]# ./ddt
usage: ./ddt [-?] [-c count] [-b bs] dir
```

The ddt program only requires one argument, the directory to be used for performance testing. It will create a file in this directory and time the task of writing count blocks of size bs to the file. It will then time reading count blocks of size bs from the file. It then reports the results. By default count is 4Ki (2^{12}) and bs is 1Mi (2^{20}); the default settings will write and read a file of size 4GiB. The source of the writes is random data returned from a malloc(bs).

In its output ddt accounts for CPU utilization in read and write tests by using the counters in /proc/stat. The /proc/stat file accounts for time spent in the following areas:

- user: normal processes executing in user mode
- nice: niced processes executing in user mode
- system: processes executing in kernel mode
- idle: twiddling thumbs
- iowait: waiting for I/O to complete
- irq: servicing interrupts
- softirq: servicing softirqs

CPU utilization is calculated as follows. Idle and iowait are summed to calculate the time spent not performing I/O (m). The sum of all counters is calculated and stored (n). The usual percentage calculation then follows:

$$\%\text{CPU utilization} = (n - m) * 100 / n$$

For this calculation to be most accurate, the client machine must not be otherwise in use as the /proc/stat counters are for all processes systemwide.

For more information on /proc/stat in Linux, see Documentation/filesystems/proc.txt in your favorite 2.6 Linux kernel source tree.

The following is an example run of ddt. Three columns of data are output. The first column states the amount of data written / read and displays the row labels for the subsequent statistics. The second column lists the respective throughput rate in KiB/s. The third column presents the total CPU% utilization during each test.

```
[root@stuart ddt-1]# ./ddt /mnt/e21.0/
Writing to /mnt/e21.0/ddt.17999 ... syncing ... done.
sleeping 10 seconds ... done.
Reading from /mnt/e21.0/ddt.17999 ... done.
4096 MiB  KiB/s   CPU%
Write    147718      8
Read     213178     15

WRaw user=0 nice=0 system=1147 idle=19489 iowait=1237 irq=107 softirq=635
RRaw user=3 nice=0 system=841 idle=12127 iowait=1097 irq=93 softirq=1479
[root@stuart ddt-1]#
```

Note the raw counter numbers are reported for validation. Due to the way bonnie++ calculates process utilization, only the user and system counters above would have been reported.

The source for the ddt program is available from the SR support page at coraid.com[1].

1 Please see Appendix B for URL

Appendix B - References

The `.config` file for stuart is available at:

`http://www.coraid.com/support/sr/stuart.config`

The SR support page includes the SR firmware, user manual, and related docs:

`http://www.coraid.com/support/sr/`

Please e-mail `support@coraid.com` with any questions or comments.