Scientific Linux and RAID performance on the SUN Fire X4500 "Thumper"

Frederik Orellana
DPNC, University of Geneva

29/10-2006

Installation of Scientific Linux 4.3

Scientific Linux 4.3 does not work out of the box on a Thumper because the driver for the Marvell SATA cards is not included in the distribution.

Instructions for Fedora Core 3 and 4 are available at

http://www.keffective.com/mvsata/.

There, you can also find some precompiled drivers. The driver source code is available at

http://www.keffective.com/mvsata/FC3/mvSata-Linux-3.6.1.tgz

In our case, it is a strict requirement to run Scientific Linux. Moreover, we wanted to run in 64 bit mode on the dual Opteron processors, that is, the architecture is SLC-4.3-x86_64 with kernel version 2.6.9-34.0.2. Thus, we had to compile the driver and produce a new installation CD. The compilation was done on a dual-processor machine already running SLC-4.3-x86_64.

The following links were useful in producing the new installation CD:

http://forums.gentoo.org/viewtopic.php?t=21327

http://www.cpqlinux.com/dig-bootnet.html

http://wiki.openvz.org/Modifying_initrd_image

For convenience, we provide the install disk image for download on

http://pcitapi34.cern.ch/~frederik/slc4-2.6.9-34.0.2.EL.cernsmp.iso

NOTICE (March 13, 2007): the kernel version in the SLC-4.3 RPM repository has changed, so the above image can no longer be used. You have to rebuild the mv_sata kernel module to match your actual current version. Do NOT use the sata_mv module of recent SLC versions: it does not give you the information to determine the controller to device name mapping and it crashes the machine when creating raid partitions.

Once the CD was created, we were able to boot the Thumper and install a system on a disk. The crucial thing to watch when installing is on *which* disk. Out of the 48 slots, only two are available as boot targets in the bios; disks 0 and 4 on controller 3 (see the table below).

To verify which devices have been created – in our case, we had only 22 disks inserted (a-v):

```
# ls -l /dev/sd* | grep "/dev/sd[a-v]$"
brw-rw----  1 root disk  8,   0 Sep  8 17:10 /dev/sda
brw-rw----  1 root disk  8,  16 Sep  8 17:10 /dev/sdb
brw-rw----  1 root disk  8,  32 Sep  8 17:10 /dev/sdc
...
brw-rw----  1 root disk 65,   0 Sep  8 17:11 /dev/sdq
brw-rw----  1 root disk 65,  16 Sep  8 17:11 /dev/sdr
...
```

To figure out the mapping /dev/sd[a-v] → controller number, disk number:

```
# grep HITACH `ls -t /proc/scsi/mvSata/*`
/proc/scsi/mvSata/12:1  4       0       0       2250308 488412117
HITACH          NCQ     1
/proc/scsi/mvSata/13:1  5       0       0       4045464 610776037
HITACH          NCQ     1
/proc/scsi/mvSata/16:2  0       0       0       2325313 488687581
HITACH          NCQ     1
...
/proc/scsi/mvSata/24:3  0       0       0       125974  4035638
HITACH          NCQ     1
...
/proc/scsi/mvSata/46:5  6       0       0       2618002 488557046
HITACH          NCQ     1
/proc/scsi/mvSata/8:1   0       0       0       2326364 488688423
HITACH          NCQ     1
/proc/scsi/mvSata/9:1   1       0       0       4470610 610776189
HITACH          NCQ     1
```

The devices are created in the order given by file names in /proc/scsi/mvSata. In our case, 8, 9, 12, 13, 16, ... Thus, the listing above tells you that /dev/sda is disk 0 on controller 1, etc. In particular, the first bootable disk (disk 0 on controller 3) is /dev/sdi, so this is where the system should be installed.

Ctrl 3		Ctrl 2		Ctrl 5		Ctrl 4		Ctrl 1		Ctrl 0	
Disk 3	Disk 7	Disk 3	Disk 7	Disk 3	Disk 7	Disk 3	Disk 7	Disk 3	Disk 7	Disk 3	Disk 7
Disk 2	Disk 6	Disk 2	Disk 6	Disk 2	Disk 6	Disk 2	Disk 6	Disk 2	Disk 6	Disk 2	Disk 6
Disk 1	Disk 5	Disk 1	Disk 5	Disk 1	Disk 5	Disk 1	Disk 5	Disk 1	Disk 5	Disk 1	Disk 5
Disk 0	Disk 4	Disk 0	Disk 4	Disk 0	Disk 4	Disk 0	Disk 4	Disk 0	Disk 4	Disk 0	Disk 4

Table 1: SATA disk slots and controllers.

Setting up software raid RAID

> *For performance, the thing to watch is to, as far as possible, have disks participating in a RAID array sit on different controllers.*

First, we partitioned one disk by hand, then copied the partition table to all other disks that were to be included in the storage array (notice that /dev/sdi is *not* included below).

```
fdisk /dev/sdj

sfdisk -d /dev/sdj > sfdisk.out

for i in a b c d e f g h j k l m n o p q r s t u v; do
sfdisk /dev/sd$i < sfdisk.out
done
```

Next, we set up three raid partitions:

/dev/md1: a raid-5 array of the 4 disks: /dev/sdm, /dev/sdp, /dev/sds, /dev/sdv.
/dev/md4: a raid-6 array of the 8 disks: /dev/sdb, /dev/sdd, /dev/sdf, /dev/sdh, /dev/sdl, /dev/sdp, /dev/sdr, /dev/sdu.
/dev/md5: a raid-50 array – that is, a raid-0 array of two raid-5 arrays:
/dev/md2: /dev/sda, /dev/sde, /dev/sdk, /dev/sdq and
/dev/md3: /dev/sdc, /dev/sdg, /dev/sdn, /dev/sdt.

After starting the creation of the raw 5 and 6 raid partitions,

```
mdadm --create --verbose /dev/md1 --level=5 --raid-devices=4
/dev/sdm /dev/sdp /dev/sds /dev/sdv

mdadm --create --verbose /dev/md2 --level=5 --raid-devices=4
/dev/sde /dev/sdq /dev/sdk /dev/sda

mdadm --create --verbose /dev/md3 --level=5 --raid-devices=4
/dev/sdc /dev/sdg /dev/sdn /dev/sdt

mdadm --create --verbose /dev/md4 --level=6 --raid-devices=8
/dev/sdb /dev/sdd /dev/sdf /dev/sdl /dev/sdh /dev/sdo /dev/sdr
/dev/sdu
```

we waited until the creation finishes before continuing to create the raid-50 partition and format the partitions; the status is checked with

```
less /proc/mdstat
```

Once the creation finished, we created the raid-50 partition – waited again - and wrote the configuration file:

```
mdadm --create /dev/md5 --verbose --chunk=4096 --level=0 --raid-
devices=2 /dev/md2 /dev/md3
mdadm --detail --scan > /etc/mdadm.conf
```

After the creation of the raid-50 partition had finished, we made an ext3 file system on all 3 partitions and created corresponding entries in /etc/fstab:

```
mkfs.ext3 -T largefile4 /dev/md1
mkfs.ext3 -T largefile4 /dev/md4
mkfs.ext3 -T largefile4 /dev/md5

# grep md /etc/fstab
/dev/md1        /raid/a5        ext3    defaults        1 2
/dev/md4        /raid/b6        ext3    defaults        1 2
/dev/md5        /raid/c50       ext3    defaults        1 2
```

Performance tests

IOZone on ext2/3

The tests were performed with the tool iozone, which is publicly available at

http://www.iozone.org/

A number of read and write tests were carried out, with different file and sector sizes. The maximum file size was set to 1 gigabyte.

```
/opt/iozone/bin/iozone -a -g 1G -i 0 -i 1 -i 2 | tee io.dat
```

This produced a number of tables (the file "io.dat"). To parse these and produce 3D plots, a small Mathematica program was written. We have made this program publicly available at

http://pcitapi34.cern.ch/~frederik/ioPlots.nb

Running the tests took a fair amount of time. For comparison, we first identical tests on two older systems:

- A dual 1 Ghz Pentium-3 machine with 1 GB of RAM and software (MD) raid-5 on 4 100 GB disks (one of which was dead). The disk used for the raw disk tests was a 40 GB IBM Deskstar IC35L040AVVN07-0, ATA 100, 7200 RPM. The raid partition in case was a 300 GB partition of 4 100 GB IBM Deskstar IC35L040AVVN07-0, ATA 100, 7200 RPM.
- A dual 2.4 GHz Opteron machine with 4 GB of RAM and a SUN StoreEdge 3511 disk array attached via fiber-channel. The internal disk used for the raw disk tests was a Seagate ST373207LC, SCSI, 10'000 RPM. The raid partition in case was a 1.92 TB partition of 400 GB, 7200 RPM serial ATA disks.

Then we moved on to run the tests on the root partition and the 3 raid partitions on the Thumper.

All the plots are given below.

Observations

- The raw disk performance of the Thumper is very good: it looks like we will be able to write 500 MB files sequentially at 1 Gb/s.
- The disk cache appears to be of 1 MB.
- On both of the two older systems the raw and the raid (software and hardware) performance are comparable.
- This is not the case for the Thumper.

A number of further tests were done to investigate the poor raid performance of the Thumper: it was found that

- The performance did not depend much on the number of controllers involved in the raid array, for a single write operation. This is expected to be different for multiple concurrent write operations. The performance of a raid5 array with 4 disks on the same controller was largely the same as with 4 disks on 4 different controllers.
- The performance did not depend much on the chunk size (4k or 64k).
- The performance does not depend much on the block size and stride options of ext3.

```
mkfs.ext3 -b 4096 -R stride=32
mkfs.ext3 -b 4096 -R stride=8 /dev/md2
```

- *Eventually it turned out that the reason for the poor raid performance as compared to the raw disk performance was not the raid itself, but rather the good performance of a raw disk with an ext2 file system on it – that is, on the* ***poor write performance of ext3****. (Having ext2 on the system disk was a pure coincidence.)*
- Mounting the ext partitions with the ordered and writeback journal options improved performance slightly, but it was still far from the ext2 performance and in our opinion at a too high price in terms of data security.

```
mount -o data=ordered /dev/sdc1 /mnt/sdc1
mount -o data=writeback /dev/sdc1 /mnt/sdc1
```

- Usenet: *"For the very simple benchmarks I was running, ext3's poor performance might be explained by the more conservative buffering used by ext3 (flushing buffers to disk every 5 seconds, not every 30 seconds like with ext2)."*

IOZone on xfs

Initially, the performance of xfs with its default configuration on a raw disk was tested and found to be comparable to that of ext2. See fig. D1.

Since, for ext3, we did not see any performance gains with our previous raid raid partitioning, a different partitioning was tried:

/dev/md1: a raid-0 array of the 4 disks: /dev/sda, /dev/sde, /dev/sdk, /dev/sdq.
/dev/md4: a raid-6 array of the 8 disks: /dev/sdb, /dev/sdd, /dev/sdf, /dev/sdh, /dev/sdl, /dev/sdp, /dev/sdr, /dev/sdu.
/dev/md5: a raid-50 array – that is, a raid-0 array of two raid-5 arrays:
/dev/md2: /dev/sdm, /dev/sdp, /dev/sds, /dev/sdv and
/dev/md3: /dev/sdc, /dev/sdg, /dev/sdn, /dev/sdt.

```
mdadm --create --verbose /dev/md1 --level=0 --raid-devices=4
/dev/sde /dev/sdq /dev/sdk /dev/sda

mdadm --create --verbose /dev/md2 --level=5 --raid-devices=4
/dev/sdm /dev/sdp /dev/sds /dev/sdv

mdadm --create --verbose /dev/md3 --level=5 --raid-devices=4
/dev/sdc /dev/sdg /dev/sdn /dev/sdt

mdadm --create --verbose /dev/md4 --level=6 --raid-devices=8
/dev/sdb /dev/sdd /dev/sdf /dev/sdl /dev/sdh /dev/sdo /dev/sdr
/dev/sdu
```

An xfs file system was created on all three partitions.

```
# mkfs.xfs -f /dev/md1
meta-data=/dev/md1               isize=256    agcount=32,
agsize=7631088 blks
         =                       sectsz=512
data     =                       bsize=4096   blocks=244194816,
imaxpct=25
         =                       sunit=16     swidth=64 blks,
unwritten=1
naming   =version 2              bsize=4096
log      =internal log           bsize=4096   blocks=32768,
version=1
         =                       sectsz=512   sunit=0 blks
realtime =none                   extsz=262144 blocks=0, rtextents=0
```

Subsequently, the same tests were carried out again. This time with a smaller maximum file size (512 MB); the performance anyway seems to flatten out with higher file sizes. See figures D.

```
/opt/iozone/bin/iozone -a -g 500M -i 0 -i 1 -i 2 | tee io.dat
```

Next, throughput tests were carried out, with 10 threads (see below). They yield a total write throughput of 1.4 Gb/s and a total read throughput of 1.1 Gb/s on raid6.

Moreover, a single test was carried out with a file size of 9 GB and a record size of 256 MB. This was in order to get over the RAM size of 8 GB. The observed write performance was 417 Mb/s, the observed read performance was 259 Mb/s.

```
/opt/iozone/bin/iozone -a -n 9G -g 9G -q 256 -y 256 -i 0 -i 1 -i 2
| tee io.dat
```

Observations

- The performance hit of raid5 and raid6 was small – as was observed on the older systems.
- Read performance was ~2.5 Gb/s. This must be the cache filling up.
- Write performance was ~1.2 Gb/s. This must be the cache filling up.
- With raid0 on 4 disks, performance was worse: read: ~2 Gb/s. Write: ~ 1 Gb/s.
- With raid6 on 8 disks, about the same performance was seen.
- With 10 parallel reads/writes, performance dropped with roughly a factor of 10 per transfer.

```
# /opt/iozone/bin/iozone -t 10 | tee io-x4500-raid6-xfs-t10.dat
Children see throughput for 10 initial writers  = 1418609.94 KB/sec
Parent sees throughput for 10 initial writers   =    3667.14 KB/sec
Min throughput per process                      =       0.00 KB/sec
Max throughput per process                      =  715965.50 KB/sec
Avg throughput per process                      =  141860.99 KB/sec
Min xfer                                        =       0.00 KB
.
.
Children see throughput for 10 readers          = 1105917.62 KB/sec
Parent sees throughput for 10 readers           =  174440.08 KB/sec
Min throughput per process                      =       0.00 KB/sec
Max throughput per process                      = 1105917.62 KB/sec
Avg throughput per process                      =  110591.76 KB/sec
Min xfer                                        =       0.00 KB
.
.
```

Copying data from a RAM disk on xfs

In order to have a "real-world" check, we also tried out reading a real ~500MB ATLAS data file from a RAM disk into a file on disk.

```
dd if=/dev/zero of=/tmp/ramdisk  bs=512k count=1024
mke2fs -F /tmp/ramdisk
mkdir /mnt/loop
mount /tmp/ramdisk /mnt/loop -o loop
time cp /mnt/loop/daq_SFI-51_combined_2100239_file02.data /raid/a0/
```

Observations

- When copying one file of 500 MB from a RAM disk to disk (raid0 or raid6), the performance was ~550 MB/s.
- When copying 10 identical files of 500 MB from a RAM disk to disk, the total performance was ~55 MB/s.
- When copying 10 identical files of 500 MB from disk to disk, the total performance was ~ 186 MB/s.
- When copying within the RAM disk, performance was poor, so the RAM disk seems not to be a good way to measure throughput.

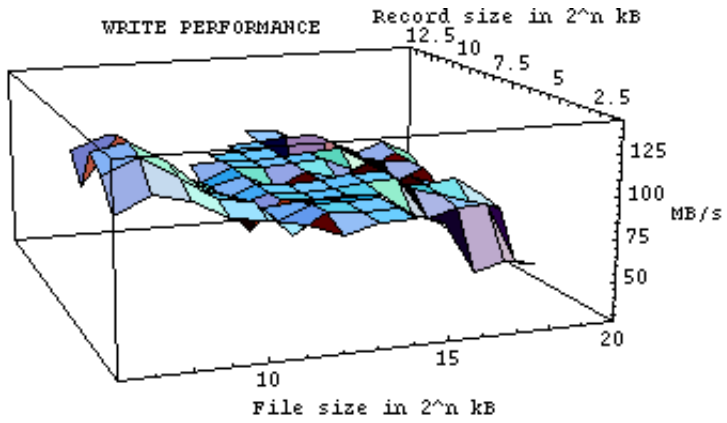
Conclusion

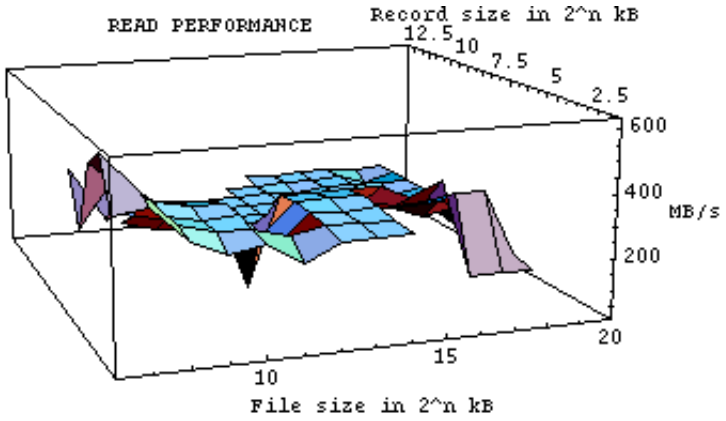
- The SATA disks of the Thumper seem to perform well under Linux with the Marvell driver. It would be interesting to make the same measurements under Solaris with zfs.
- The Marvell driver is brand new and can probably be optimised further.

- The ext3 file system has poor write performance compared to ext2 and xfs.
- This poor performance can probably be alleviated by enough tweaking; we did, however, despite some effort, not find out how.
- Since a journaling file system is needed for optimal uptime, xfs is recommended.
- With xfs and raid6 we saw a read performance of ~2 Gb/s and a write performance of 1 Gb/s with the benchmark tool IOZone, using file sizes below 1 GB.
- When reading 10 files in parallel, we saw a performance of ~110 MB/s per file with IOZone, using file sizes below 1 GB.
- When writing 10 files in parallel, we saw a performance of ~140 MB/s per file with IOZone, using file sizes below 1 GB.
- With a file size of 9 GB the write and read performance observed wit IOZone were 417 Mb/s and 259 Mb/s.
- Straightforwardly timing the copying of 500 MB files, more or less confirmed the performance measured with IOZone: we saw between 55 and 186 MB/s in total, when copying 10 files sequentially, and ~550 MB/s when copying one file.
- xfs allows many tweaks that should be investigated, but we don't expect them to change the results dramatically.
- Real-world stress testing needs to be performed along the lines of the document "Performance tests and tuning with SATA Linux file servers".

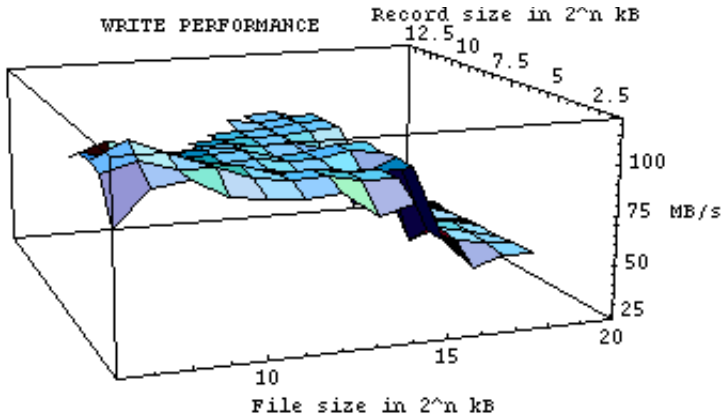
Performance plots

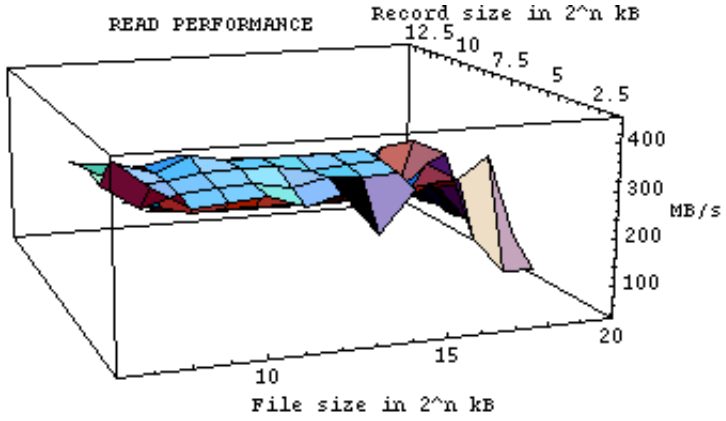
A1. IBM Deskstar 7200 RPM IDE raw disk I/O - ext3



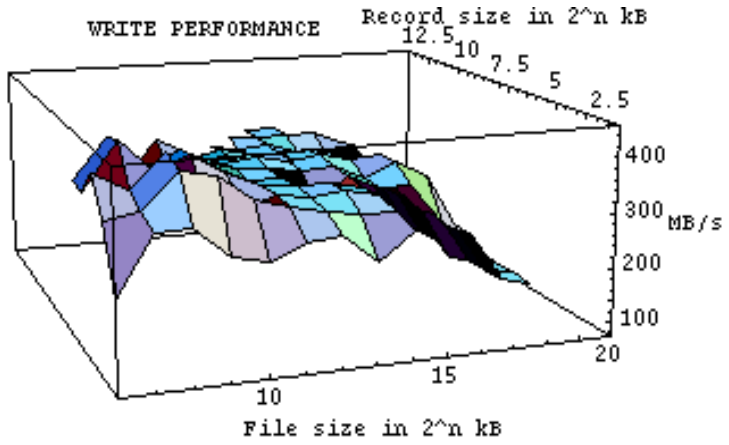


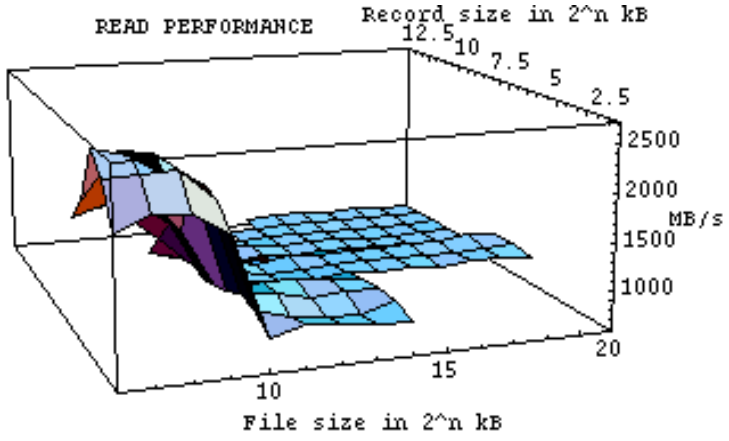
A2. IBM Deskstar 7200 RPM IDE Linux software raid – 3 disks - ext3



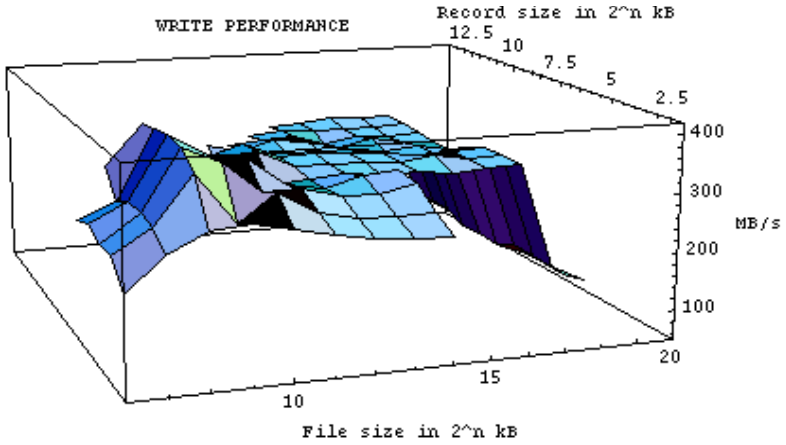


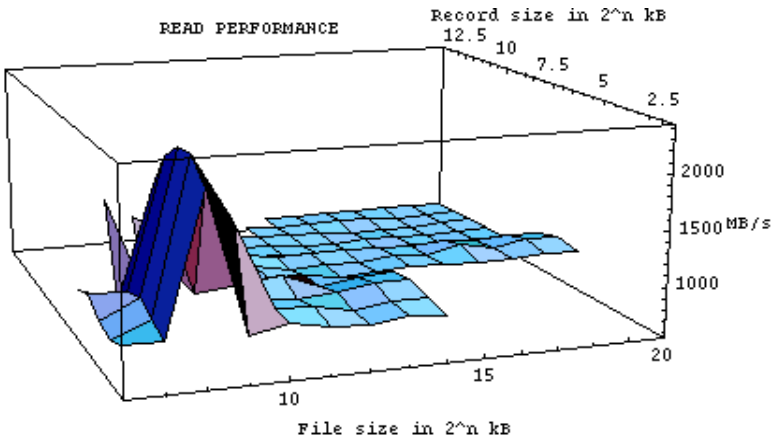
B1. Seagate 10'000 RPM SCSI raw disk I/O - ext3



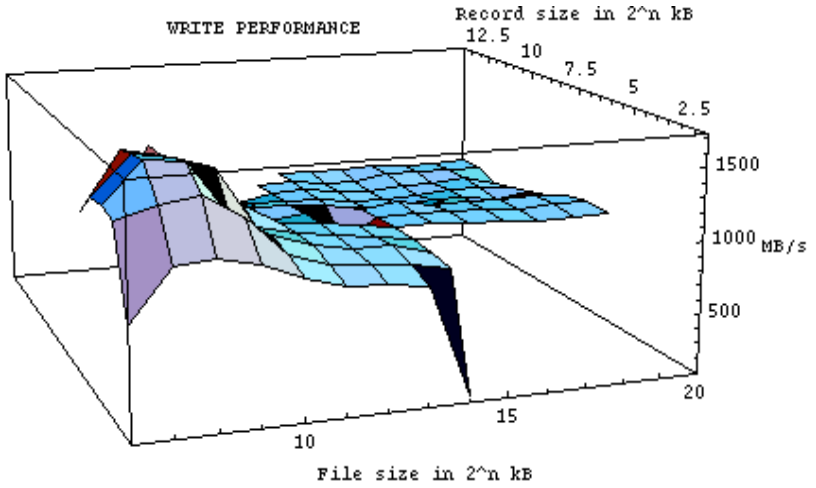


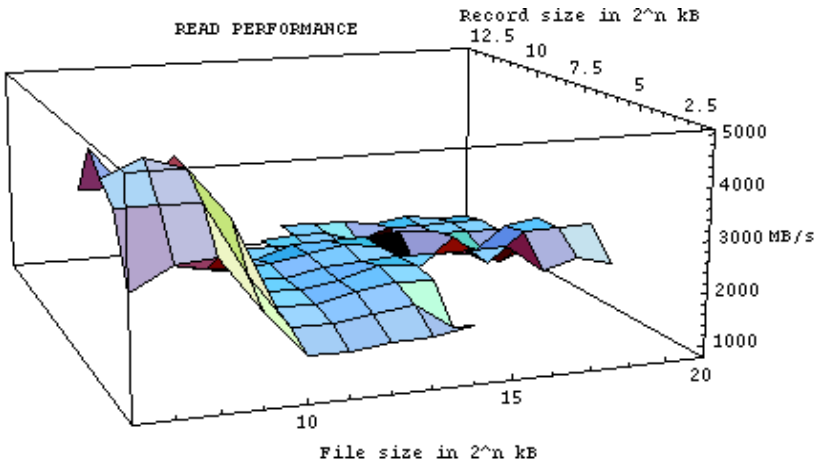
B2. SUN StoreEdge 3511 hardware raid on 7200 RPM serial ATA disks - ext3



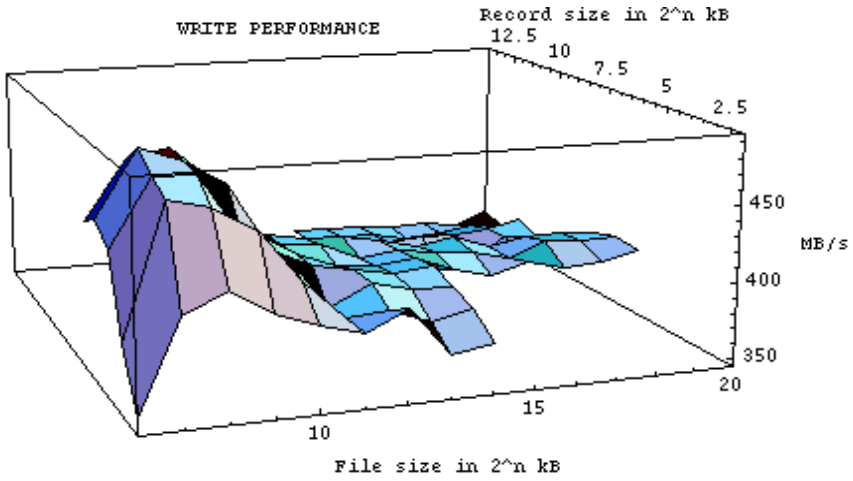


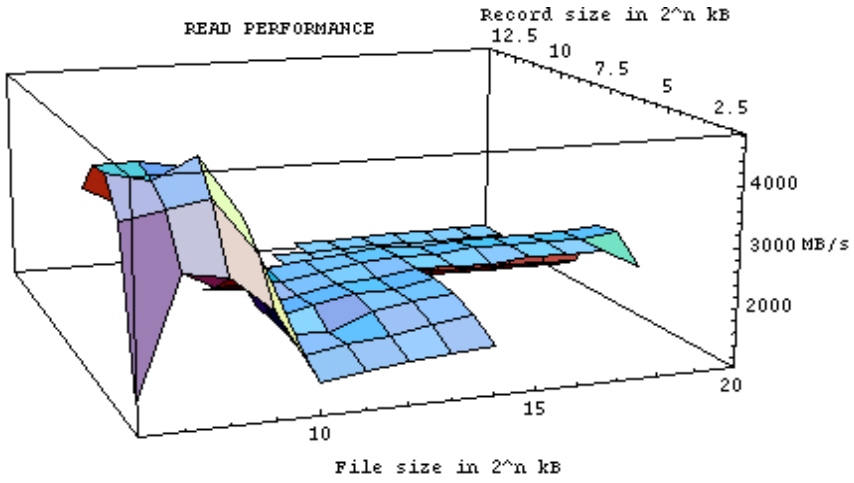
C1. Thumper raw disk I/O - ext2



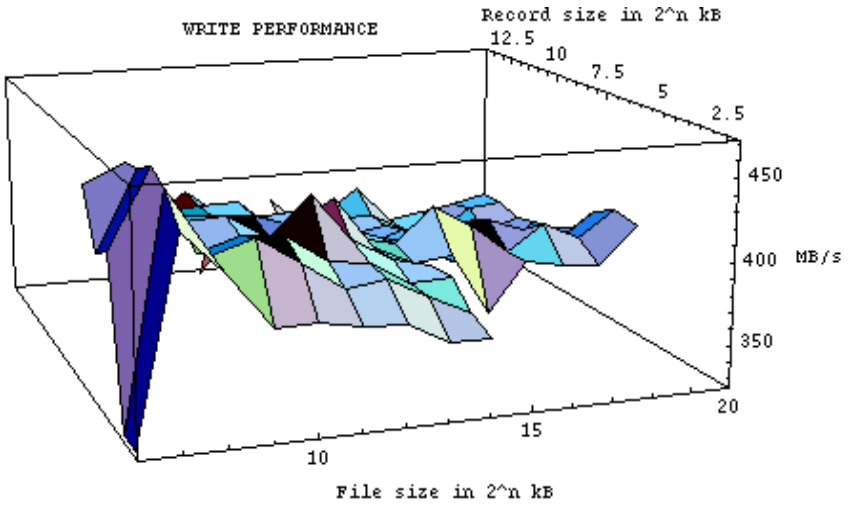


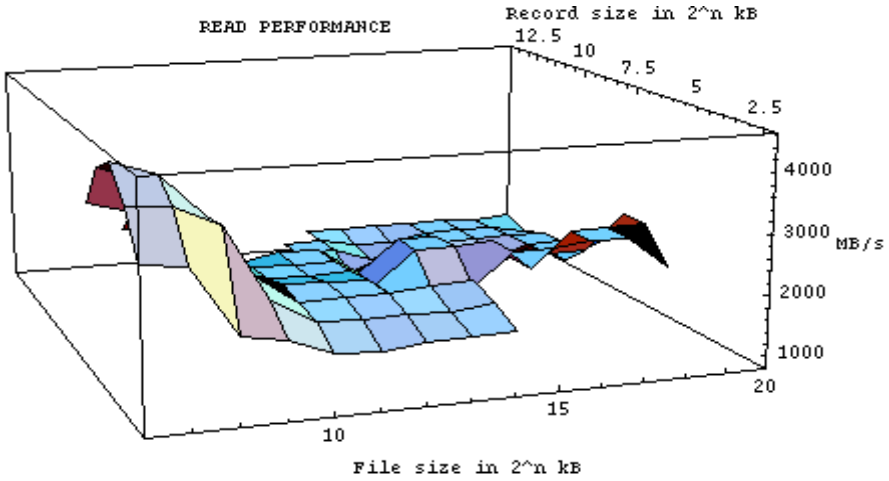
C2. Thumper raw disk I/O – ext3



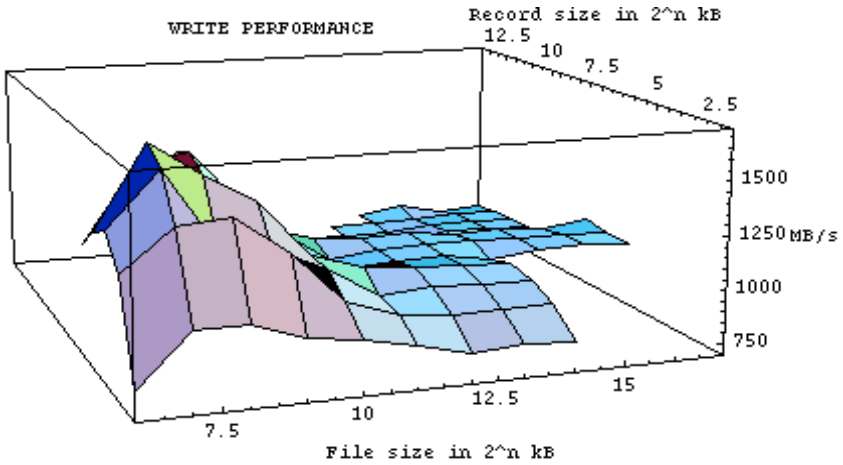


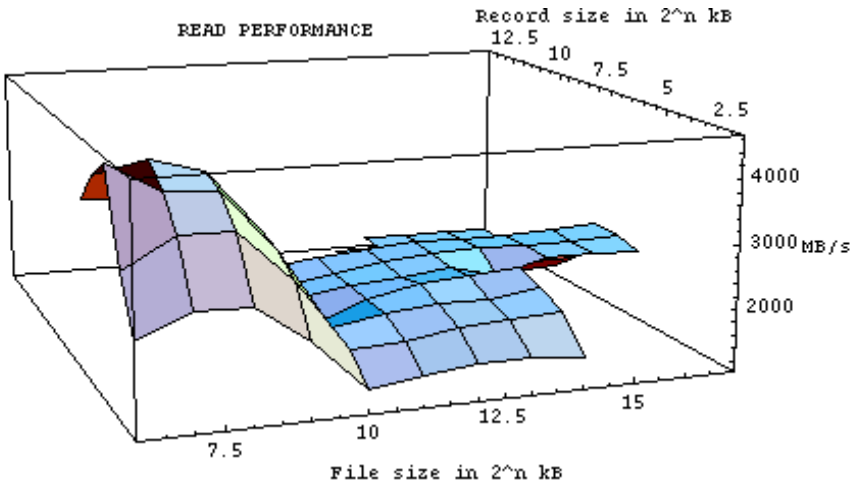
C3. Thumper raid-50, 8 disks on 4 controllers – ext3



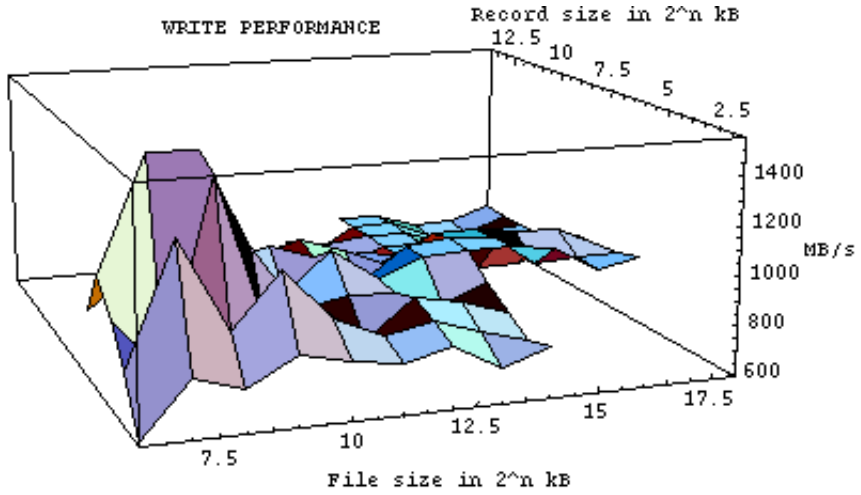


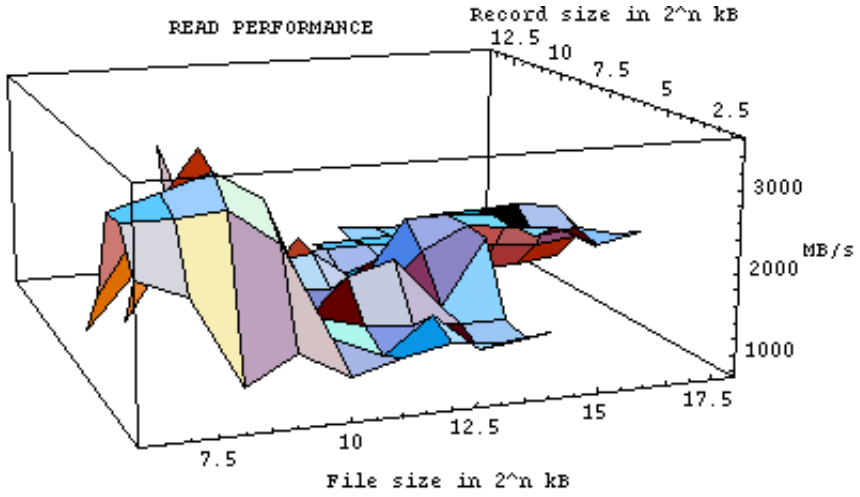
D1. Thumper raw disk – xfs



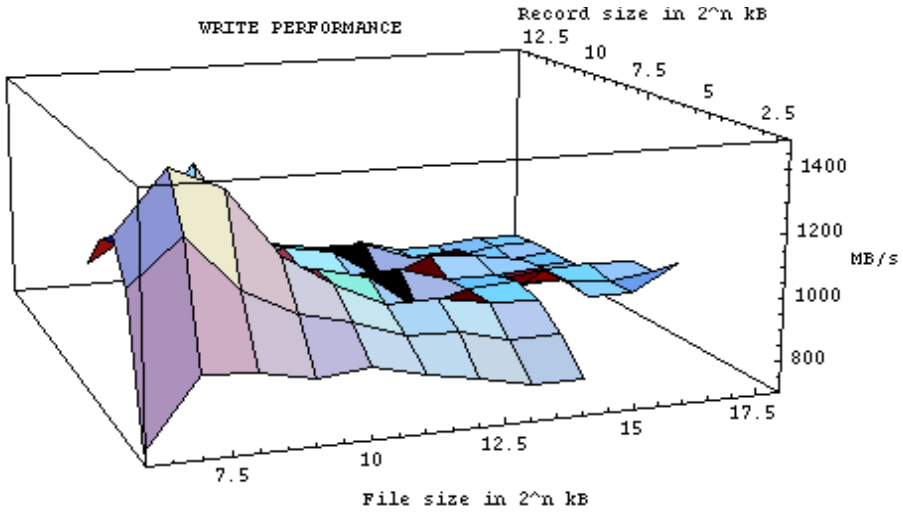


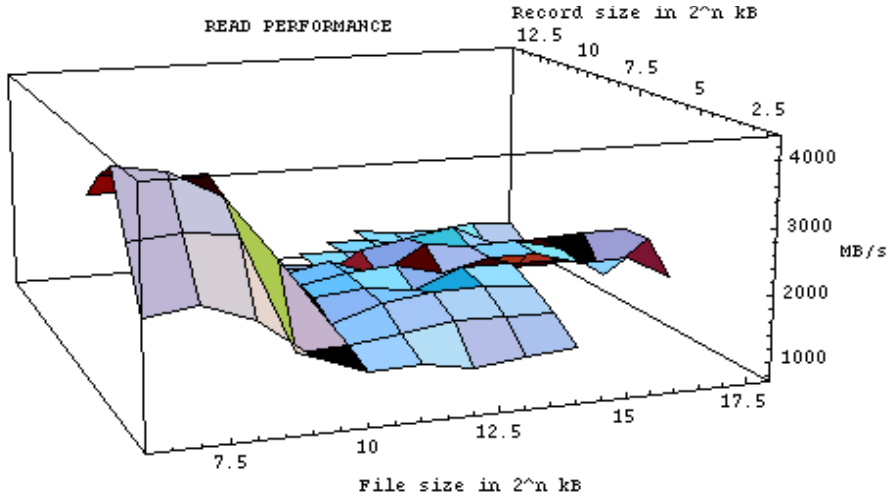
D2. Thumper raid0, 4 disks on 4 controllers – xfs





D3. Thumper raid6, 8 disks on 4 controllers – xfs




READ PERFORMANCE
Record size in 2^n kB
12.5
10
7.5
5
2.5
4000
3000
2000
1000
MB/s
7.5
10
12.5
15
17.5
File size in 2^n kB