

Benchmarking the 3ware 9000 Controller Using Linux Kernel 2.6



Document v1.1

TABLE OF CONTENTS

Abstract	2
Introduction	2
Effective I/O Tuning and Benchmarking	2
Misconceptions	3
Tuning Test Case 1: Bonnie++ Benchmark: Hardware RAID 0	5
Tuning Test Case 2: Bonnie++ Benchmark: Hardware RAID 5	5
Explanation of Benchmark Output	5
RAID 0 Sample Benchmark Data	6
RAID 5 Sample Benchmark Data	13
Conclusion	20
Appendix – Glossary of Terms	20
Addendum	21

©2004 by Applied Micro Circuits Corporation. All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form by any means, electronic, mechanical, photocopying or otherwise, without the prior written agreement or permission of AMCC; 455 West Maude Avenue; Sunnyvale, CA 94085 USA.

AMCC is a registered trademark of Applied Micro circuits Corporation. 3ware, SwitchedRAID and 3DM are registered trademarks in the United States and StorSwitch is a trademark in the United States of Applied Micro Circuits Corporation. All other trademarks are the property of their respective holders.

Applied Micro Circuits Corporation assumes no responsibility for errors or omissions in this document, nor does AMCC make any commitment to update the information contained here in. Any information contained herein may change at any time without prior notice.

Abstract

This document discusses in detail several methods for effectively benchmarking the 3ware® 9000 Series RAID controllers using the Linux 2.6 kernel. Many variations for benchmarks are now available and this paper will assist with providing methods to yield consistent and accurate results for I/O performance. This document will clear misconceptions regarding benchmarking and also illustrate the benefits of system tuning parameters in detail.

This paper was written for Linux system administrators and AMCC Storage customers who are attempting to benchmark or tune their system to work with the 3ware 9000 controllers.

Introduction

Benchmarking and tuning can be difficult, time consuming, and frequently provide inconsistent and inaccurate results. Additionally, many misconceptions about Linux I/O benchmarking lead customers to invalid conclusions. However, with the proper system setup and tuning, I/O benchmarks can be achieved that are close to the maximum performance of the hardware. This white paper will provide some guidelines and assistance for benchmarking to try to help AMCC Storage customers measure the performance of the 3ware 9000 Series RAID controllers accurately.

Effective I/O Tuning and Benchmarking

In order to ensure successful benchmarking, fine-tune the following variables to achieve optimal performance:

Tune devices read-ahead cache settings

More asynchronous reads can be issued to the 3ware 9000 Series RAID controller by modifying the devices read-ahead cache settings. Having more asynchronous reads pending on the controller yields higher sequential read throughput. To increase the devices read-ahead cache settings, enter the following command (modifying the device, /dev/sda, as appropriate):

```
blockdev --setra 16384 /dev/sda
```

Perform I/O with a file size of 40X physical RAM size

Using only 2X physical RAM size can cause significant amounts of I/O to be read from the buffer cache. Running a benchmark with a file size of 40X physical RAM size will minimize the percentage of error as a result of I/O being read out of cache.

Multi-User Test Mode (run-level 3)

Ideally start system in single-user mode and then run the benchmark tools. However, in most real world situations, running at run-level 1 is less frequent. Hence, start system in run-level 3 with the basic module(s) support. Unload all redundant or unused modules and processes. Running concurrent processes can influence the benchmark tool and it will generate skewed results. Additionally, avoid loading X-windows. Use run-level 3 as it provides the use of networking and other functions that are closer to real world scenarios.

Run 'sync' to flush the buffer cache before the benchmark

There could be cached filesystem writes in the filesystem buffer cache to either the 3ware device or another device that need to be flushed before running the benchmark. Without flushing these writes, the benchmark could yield invalid results. This can be accomplished with the 'sync' command.

Check the process table for concurrent tasks

If there are other tasks running, such as 'updatedb' they will cause the benchmark tool to compete for CPU resources with the other tasks. The 'ps' or 'top' command will tell you whether anything else is using CPU cycles. Shutdown or kill any processes that appear to be competing for CPU resources before running the benchmark. Do not execute "ps" or "top" commands when the benchmark tool is running.

Avoid system intervention when tests are running

In most cases, the system under test should be left undisturbed until the running tests have completed. Do not run concurrent processes or programs because they will have a tendency to skew the results. Moving the mouse or pressing keys on the keyboard or pressing enter causes interrupts from the mouse or keyboard device. Entering a blank line into a shell will cause your benchmark to get scheduled out and your shell to schedule in for a small time slice, affecting your results.

Make sure no other drivers are loaded or busy

If possible make sure no other device drivers are loaded or busy utilizing CPU or memory bandwidth. This will negatively impact the performance results. Before running the benchmark, unload unnecessary drivers and kernel modules using the 'rmmod' command.

Misconceptions

There are many common misconceptions about I/O benchmarking under Linux.

Linux I/O benchmarks always give valid results

The accuracy of this statement is questionable. There are factors that may affect benchmark results. The system could be tuned poorly; there could be file system buffer cache hits, or background tasks could be running concurrently with the benchmark. There could also be dirty write buffers for a device that need to be flushed before the benchmark.

Linux I/O benchmarks avoid cache hits

This is not true. Even running benchmarks with an I/O size of 2X physical RAM size will result in some of the I/O being read out of cache. This will result in invalid benchmarking results.

Linux I/O benchmarks issue asynchronous I/O

Another misconception. Asynchronous reads, as seen by the device driver as multiple separate I/O, are not usually issued from a single benchmark process without the effect of tuning the devices read-ahead cache settings.

Filesystem benchmarks issue sequential I/O during sequential I/O tests

This is also not correct. Filesystem benchmarks usually use large files that end up being a sequence of data, metadata and possibly journal data. The LBA access pattern of the typical filesystem benchmark is not purely sequential; resulting in disk seeks which will result in invalid sequential I/O results.

All Linux filesystems perform the same

Different filesystems yield different performance results under a variety of different benchmarks and conditions. For more information on a comparison of the performance of various Linux filesystems, please see <http://oss.sgi.com/projects/xfs/papers/filesystem-perf-tm.pdf>

Multiprocessor systems results in higher I/O benchmark results

Parts of the device driver and Linux kernel SCSI layer are non re-entrant. Re-entrancy is prevented by usage of kernel primitives such as spinlocks and semaphores. Multiprocessor systems generally give the same I/O benchmark results as single processor systems. This is due to the fact that only one processor can be executing the non re-entrant parts of the OS at a time.

Memory bandwidth and CPU speed don't affect I/O benchmark results

Linux does a buffer copy from userspace to kernelspace and vice versa for all I/O except for specially compiled programs that use the O_DIRECT flag with the 'open' system call. This buffer copy uses CPU cycles and memory bandwidth. Systems with faster memory and CPU usually yield better benchmark results.

Linux does zero-copy I/O in a standard configuration

As stated above, Linux doesn't do zero-copy I/O from userspace to kernelspace and vice versa. The kernel copies data back and forth as it is DMA'd into kernel memory to/from the device. This causes higher CPU utilization than zero-copy I/O systems.

Tuning Test Case 1: Bonnie++ Benchmark: Hardware RAID 0

Sample Bonnie++ xfs filesystem results before adjusting the devices read-ahead settings were 203.3 MB/s writes and 88.1 MB/s reads (See RAID 0 sample benchmark data: Test #7, Trial #1). The devices read-ahead settings were then set to `blockdev--setra 16384/dev/sda`. The benchmark was re-run and these tuning settings resulted in a 466% increase in sequential read throughput from 88.1 MB/s to 410.7 MB/s (See RAID 0 sample benchmark data: Test #8, Trial #1).

Tuning Test Case 2: Bonnie++ Benchmark: Hardware RAID 5

Sample Bonnie++ xfs filesystem results before adjusting the devices read-ahead settings were 86.3 MB/s reads and 105.8 MB/s writes (See RAID 5 sample benchmark data: Test #7, Trial #1). The devices read-ahead settings were then set to `16384`. The benchmark was re-run and these tuning settings resulted in a 470% increase in sequential read throughput from 86.3 MB/s to 406.1 MB/s (See RAID 5 sample benchmark data: Test #8, Trial #1).

Explanation of Benchmark Output

dd

The use of `dd` is in combination with the `time` command. By running `time` before `dd`, we can determine how long it took a `dd` of a certain amount of data to complete. Using that time we can convert to normal benchmark units of megabytes per second.

Bonnie++

Output under the 'Machine' column represents the hostname of the computer on which the benchmark was run. Output under the 'Size' column represents the amount of I/O performed during the test. This is usually expressed in 'G' for gigabytes or 'M' for megabytes. Output under the 'Sequential Output' column is for sequential writes. This output is broken up into 3 types: 'Per Chr' (single character), 'Block' (large block), and 'Rewrite' (non-sequential re-write). Bonnie++ output is in Kbytes/sec. Output under the 'Sequential Input' column is for sequential reads. This output is broken up into 2 types: 'Per Chr' (single character) and 'Block' (large block). Output under the 'Random Seeks' column represents the number of random read or writes I/O operations performed per second. For each value there is a '%CP' (percent CPU usage). Any fields left blank indicate that particular test was not run.

iozone

'Record size' indicates the amount of data performed in each I/O. Output is in Kbytes/sec 'Initial write' output represents the data rate of a sequential write test. 'Rewrite' output represents the data rate of a non-sequential re-write test. 'Read' output represents the data rate of a sequential read test. 'Re-read' output represents the data rate of a sequential re-read test.

RAID 0 Sample Benchmark Data

3ware 9000 (RAID 0) performance testing for kernel 2.6

Summary of tests performed:

Test #1: Untuned raw block READ performance w/dd
Test #2: Untuned raw block WRITE performance w/dd
Test #3: Bonnie++ untuned w/ext2 filesystem
Test #4: Bonnie++ tuned w/ext2 filesystem
Test #5: Bonnie++ untuned w/ext3 filesystem
Test #6: Bonnie++ tuned w/ext3 filesystem
Test #7: Bonnie++ untuned w/xfs filesystem
Test #8: Bonnie++ tuned w/xfs filesystem
Test #9: lozone untuned w/ext2 filesystem
Test #10: lozone tuned w/ext2 filesystem
Test #11: lozone untuned w/ext3 filesystem
Test #12: lozone tuned w/ext3 filesystem
Test #13: lozone untuned w/xfs filesystem
Test #14: lozone tuned w/xfs filesystem

Goal of tests:

Maximum performance possible with 3ware 9000 Series RAID controller under Linux 2.6.

System configuration:

Processor: Xeon 2.4 Ghz (2)
RAID configuration: 12, 70 GB WDC WD740GD-00FLA0 drives in hardware RAID 0.
Stripe size: 64k
Kernel: 2.6.5
Controller: 3ware 9500S-12
Driver: 2.26.00.005
Driver cmds_per_lun setting: 254 (default)
Firmware: FE9X 2.02.00.009
OS Runlevel: 3
Bonnie++ version: 1.02c
lozone version: 3.203
Motherboard: SE7501 CW2
RAM: 512 MB
Amount of I/O performed: 40 X physical RAM in all tests: 20 GB

Test #1: Untuned raw block READ performance w/dd

Benchmark command line used:

```
sync; time `dd if=/dev/sda of=/dev/null bs=1M count=20480`
```

Trial 1: 177.143s 115.613 MB/s
Trial 2: 175.836s 116.472 MB/s
Trial 3: 182.671s 112.114 MB/s

Test #2: Untuned raw block WRITE performance w/dd

Benchmark command line used:

```
sync; time `dd if=/dev/zero of=/dev/sda bs=1M count=20480 && sync`
```

Trial 1: 100.736s 203.304 MB/s
Trial 2: 100.548s 203.684 MB/s
Trial 3: 100.756s 203.263 MB/s

Test #3: Bonnie++ untuned w/ext2 filesystem

Benchmark command line used:

```
sync; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b
```

Trial 1:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			200720	59	59157	18			88172	11	201.4	0

Trial 2:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			199708	59	59529	19			88183	11	198.6	0

Trial 3:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			200365	59	59965	19			88056	11	202.3	0

Test #4: Bonnie++ tuned w/ext2 filesystem

Tuning parameters used:

blockdev --setra 16384 /dev/sda

Benchmark command line used:

sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b

Trial 1:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			200339	59	106594	40			331021	68	197.1	1

Trial 2:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			200005	60	107294	40			331202	68	197.6	1

Trial 3:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			200415	60	107877	41			333682	68	196.2	1

Test #5: Bonnie++ untuned w/ext3 filesystem

Benchmark command line used:

sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b

Trial 1:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			160189	97	58237	24			88358	10	143.4	0

Trial 2:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			170746	97	58224	24			88396	10	141.9	0

Trial 3:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			167929	97	58109	24			88410	10	143.5	0

Test #6: Bonnie++ tuned w/ext3 filesystem

Tuning parameters used:

blockdev --setra 16384 /dev/sda

Benchmark command line used:

sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b

Trial 1:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			162251	96	106046	46			329289	67	142.5	1

Trial 2:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			163660	97	106522	46			330482	67	140.6	1

Trial 3:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			167395	97	106293	46			331970	67	142.0	0

Test #7: Bonnie++ untuned w/xfs filesystem

Benchmark command line used:

sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b

Trial 1:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			203339	64	57828	18			88178	11	411.7	1

Trial 2:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			203253	63	56745	18			88298	11	412.5	1

Trial 3:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			203514	62	57053	18			87889	11	416.2	1

Test #8: Bonnie++ tuned w/xfs filesystem

Tuning parameters used:

blockdev --setra 16384 /dev/sda

Benchmark command line used:

sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b

Trial 1:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			202836	62	126809	53			410799	88	383.9	2

Trial 2:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			202132	61	124160	51			409706	90	376.3	2

Trial 3:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			203999	65	123795	51			410073	91	384.4	2

Test #9: lozone untuned w/ext2 filesystem

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid0_ext2_untuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	201270.48
Rewrite	138652.64
Read	88552.74
Re-read	87495.85

Test #10: lozone tuned w/ext2 filesystem

Tuning parameters used:

```
blockdev --setra 16384 /dev/sda
```

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid0_ext2_tuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	201067.88
Rewrite	117948.62
Read	327641.25
Re-read	328280.19

Test #11: lozone untuned w/ext3 filesystem

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid0_ext3_untuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	161158.78
Rewrite	137409.03
Read	87890.70
Re-read	87770.06

Test #12: lozone tuned w/ext3 filesystem

Tuning parameters used:

```
blockdev --setra 16384 /dev/sda
```

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid0_ext3_tuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	168194.44
Rewrite	134735.62
Read	331440.41
Re-read	331421.66

Test #13: lozone untuned w/xfv filesystem

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid0_xfs_untuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	210373.81
Rewrite	203732.16
Read	88383.92
Re-read	88435.57

Test #14: lozone tuned w/xfv filesystem

Tuning parameters used:

```
blockdev --setra 16384 /dev/sda
```

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid0_xfs_tuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	209266.03
Rewrite	204313.34
Read	415544.47
Re-read	417052.16

RAID 5 Sample Benchmark Data

3ware 9000 (RAID 5) performance testing for kernel 2.6

Summary of tests performed:

Test #1: Untuned raw block READ performance w/dd
Test #2: Untuned raw block WRITE performance w/dd
Test #3: Bonnie++ untuned w/ext2 filesystem
Test #4: Bonnie++ tuned w/ext2 filesystem
Test #5: Bonnie++ untuned w/ext3 filesystem
Test #6: Bonnie++ tuned w/ext3 filesystem
Test #7: Bonnie++ untuned w/xfs filesystem
Test #8: Bonnie++ tuned w/xfs filesystem
Test #9: lozone untuned w/ext2 filesystem
Test #10: lozone tuned w/ext2 filesystem
Test #11: lozone untuned w/ext3 filesystem
Test #12: lozone tuned w/ext3 filesystem
Test #13: lozone untuned w/xfs filesystem
Test #14: lozone tuned w/xfs filesystem

Goal of tests:

Maximum performance possible with 3ware 9000 Series RAID controller under Linux 2.6.

System configuration:

Processor: Xeon 2.4 Ghz (2)
 RAID configuration: 12, 70 GB WDC WD740GD-00FLA0 drives in hardware RAID 5.
 Stripe size: 64k
 Kernel: 2.6.5
 Controller: 3ware 9500S-12
 Driver: 2.26.00.005
 Driver cmds_per_lun setting: 254 (default)
 Firmware: FE9X 2.02.00.009
 OS Runlevel: 3
 Bonnie++ version: 1.02c
 lozone version: 3.203
 Motherboard: SE7501 CW2
 RAM: 512 MB
 Amount of I/O performed: 40 X physical RAM in all tests: 20 GB

Test #1: Untuned raw block READ performance w/dd

Benchmark command line used:

```
sync; time `dd if=/dev/sda of=/dev/null bs=1M count=20480`
```

Trial 1: 182.424s 112.26 MB/s
 Trial 2: 178.506s 114.73 MB/s
 Trial 3: 178.518s 114.72 MB/s

Test #2: Untuned raw block WRITE performance w/dd

Benchmark command line used:

```
sync; time `dd if=/dev/zero of=/dev/sda bs=1M count=20480 && sync`
```

Trial 1: 180.143s 113.69 MB/s
 Trial 2: 182.760s 112.06 MB/s
 Trial 3: 181.405s 112.89 MB/s

Test #3: Bonnie++ untuned w/ext2 filesystem

Benchmark command line used:

```
sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b
```

Trial 1:
 Version 1.02c

		-----Sequential Output----				-- --Sequential Input--				--Random--			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			101075	30	39229	12			86360	10	191.3	0

Trial 2:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			101687	30	39039	12			86848	10	190.0	0

Trial 3:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			102127	30	39094	12			86540	10	192.7	0

Test #4: Bonnie++ tuned w/ext2 filesystem

Tuning parameters used:

```
blockdev --setra 16384 /dev/sda
```

Benchmark command line used:

```
sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b
```

Trial 1:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			100167	29	59249	22			331844	66	190.9	1

Trial 2:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			102001	30	59225	22			333762	67	189.3	1

Trial 3:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			103546	30	59026	22			331295	67	189.7	1

Test #5: Bonnie++ untuned w/ext3 filesystem

Benchmark command line used:

```
sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b
```

Trial 1:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			67948	42	36929	15			86567	10	137.3	0

Trial 2:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			67639	42	36809	15			86736	10	137.4	0

Trial 3:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			67837	42	37021	15			86800	10	139.1	0

Test #6: Bonnie++ tuned w/ext3 filesystem

Tuning parameters used:

blockdev --setra 16384 /dev/sda

Benchmark command line used:

sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b

Trial 1:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			67324	42	55863	23			325503	67	139.1	0

Trial 2:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			68258	42	58858	25			318736	69	137.2	0

Trial 3:
Version 1.02c

		-----Sequential Output----						-- --Sequential Input-				--Random-	
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			67975	42	58792	24			319172	68	138.6	0

Test #7: Bonnie++ untuned w/xfs filesystem

Benchmark command line used:

```
sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b
```

Trial 1:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			105889	31	44421	14			86393	11	386.5	1

Trial 2:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			104792	30	41070	13			86087	10	385.1	1

Trial 3:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			106473	31	40967	13			86518	11	382.1	1

Test #8: Bonnie++ tuned w/xfs filesystem

Tuning parameters used:

```
blockdev --setra 16384 /dev/sda
```

Benchmark command line used:

```
sync ; bonnie++ -n 0 -u 0 -r 512 -s 20480 -f -b
```

Trial 1:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			104715	31	82579	32			406124	90	363.7	2

Trial 2:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			106815	32	83099	32			407527	88	366.8	2

Trial 3:

Version 1.02c		-----Sequential Output----				-- --Sequential Input-				--Random-			
		-Per Chr-		--Block--		-Rewrite-		-Per Chr-		--Block--		--Seeks--	
Machine	Size	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	K/sec	%CP	/sec	%CP
linux4	20G			103872	30	82385	32			407186	90	360.3	2

Test #9: lozone untuned w/ext2 filesystem

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid5_ext2_untuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	100836.31
Rewrite	59858.33
Read	85921.48
Re-read	85976.95

Test #10: lozone tuned w/ext2 filesystem

Tuning parameters used:

```
blockdev --setra 16384 /dev/sda
```

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid5_ext2_tuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	101230.36
Rewrite	62749.98
Read	330300.38
Re-read	330942.94

Test #11: lozone untuned w/ext3 filesystem

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid5_ext3_untuned.xls
```

Record size = 64Kbytes
Output is in Kbytes/sec

Initial write	62091.05
Rewrite	57356.84
Read	85972.98
Re-read	85880.22

Test #12: Izone tuned w/ext3 filesystem

Tuning parameters used:

blockdev --setra 16384 /dev/sda

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid5_ext3_tuned.xls
```

Record size = 64Kbytes
Output is in Kbytes/sec

Initial write	61900.16
Rewrite	57989.56
Read	327009.88
Re-read	326700.16

Test #13: Izone untuned w/xfs filesystem

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid5_xfs_untuned.xls
```

Record size = 64Kbytes
Output is in Kbytes/sec

Initial write	117609.62
Rewrite	115533.13
Read	86769.66
Re-read	87462.46

Test #14: Izone tuned w/xfs filesystem

Tuning parameters used:

blockdev --setra 16384 /dev/sda

Benchmark command line used:

```
./iozone -s 20480m -r 64 -i 0 -i 1 -t 1 -b raid5_xfs_tuned.xls
```

Record size = 64Kbytes

Output is in Kbytes/sec

Initial write	117144.79
Rewrite	116371.01
Read	411931.59
Re-read	416373.00

Conclusion

Linux I/O benchmarks don't automatically provide expected results. With careful setup and tuning, running Linux I/O benchmarks on the 3ware 9000 Series RAID controller will yield close to the published results.

Appendix – Glossary of Terms

daemon – Program that is always running in the background of a system.

DMA – Dynamic memory access.

I/O – Input/Output operation of a device.

kernel – Core of the Linux operating system.

kernel space – Execution environment inside the kernel.

LBA – Logical block access.

RAID – Redundant array of inexpensive/independent disks.

user space – Execution environment outside the kernel.

Bdflush – Linux buffer cache flush daemon

VM layer – Linux virtual memory subsystem

Dirty cache or buffers – Dirty cache buffers contain data that has not yet been written to disk.

Addendum

For Linux 2.4 kernel systems with any RAID configuration, use the following parameters to enhance system performance:

Tune VM layer read-ahead cache settings

More asynchronous reads can be issued to the 3ware controller by modifying the VM layer read-ahead cache settings. Having more asynchronous reads pending on the controller yields higher sequential read throughput. To set these values, enter the following commands:

```
sysctl -w "vm.min-readahead=2048"
```

```
sysctl -w "vm.max-readahead=2048"
```

For Linux 2.4 kernel systems with RAID 5 configurations, use the following parameter to enhance system write performance:

Tune bdflush kernel daemon setting for RAID 5

The bdflush default settings impede with the 3ware RAID 5 write caching algorithms' scheme for flushing writes to disk. This can be avoided by changing the bdflush kernel daemons settings with the following command:

```
sysctl -w "vm.bdflush=0 500 0 0 500 3000 0 20 0"
```

The relevant values in the above command that should not be changed are the 1st and 7th fields, both of which are "0". The rest can be left the same or relevant to the system being tested. They represent the amount of dirty buffer cache to activate the bdflush daemon synchronously and asynchronously, respectively. For more information on these settings refer to `/usr/src/linux2.x/Documentation/sysctl/vm.txt` in the Linux kernel source

