As we discussed in our [Linux performance monitoring introduction](http://www.thegeekstuff.com/2011/03/linux-performance-monitoring-intro/) article, measuring IO subsystem performance is very important.

If someone is complaining that a database (or any application) running on one server (with certain filesystem, or RAID configuration) is running faster than the same database or application running on another server, you might want to make sure that the performance at the disk level is same on both the server. You can use iozone for this situation.  
  
If you are running your database (or any application) on certain SAN or NAS environment, and would like to migrate it to different SAN or NAS environment, you should perform filesystem benchmakring on both the systems and compare it. You can use iozone for this situation.

If you know how to use iozone, you can pretty much use it for various filesystem benchmarking purpose.

Download and Install IOZone

[Iozone](http://www.iozone.org/) is an open source file system benchmarking utility.

Follow the steps below to download and install iozone on your system.

wget http://www.iozone.org/src/current/iozone3\_394.tar

tar xvf iozone3\_394.tar

cd iozone3\_394/src/current

make

make linux

What does IOzone utility measure?

IOzone performs the following 13 types of test. If you are executing iozone test on a database server, you can focus on the 1st 6 tests, as they directly impact the database performance.

1. Read – Indicates the performance of reading a file that already exists in the filesystem.
2. Write – Indicates the performance of writing a new file to the filesystem.
3. Re-read – After reading a file, this indicates the performance of reading a file again.
4. Re-write – Indicates the performance of writing to an existing file.
5. Random Read – Indicates the performance of reading a file by reading random information from the file. i.e this is not a sequential read.
6. Random Write – Indicates the performance of writing to a file in various random locations. i.e this is not a sequential write.
7. Backward Read
8. Record Re-Write
9. Stride Read
10. Fread
11. Fwrite
12. Freread
13. Frewrite

**10 IOZone Examples**

1. Run all IOZone tests using default values

-a option stands for automatic mode. This creates temporary test files from sizes 64k to 512MB for performance testing. This mode also uses 4k to 16M of record sizes for read and write (more on this later) testing.

-a option will also execute all the 13 types of tests.

$ ./iozone -a

The first setion of the iozone output contains the header information, which displays information about the iozone utility, and all the iozone options that are used to generate this report, as shown below.

Iozone: Performance Test of File I/O

Version $Revision: 3.394 $

Compiled for 32 bit mode.

Build: linux

Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins

Al Slater, Scott Rhine, Mike Wisner, Ken Goss

Run began: Sat Apr 23 12:25:34 2011

Auto Mode

Command line used: ./iozone -a

Output is in Kbytes/sec

Time Resolution = 0.000001 seconds.

Processor cache size set to 1024 Kbytes.

Processor cache line size set to 32 bytes.

File stride size set to 17 \* record size.

The second section of the output contains the output values (in per second) of various tests.

* 1st column KB: Indicates the file size that was used for the testing.
* 2nd column reclen: Indicates the record length that was used for the testing.
* 3rd column until the last column: Indicates the various tests that are performed and its output values in per second.

random random bkwd record stride

KB reclen write rewrite read reread read write read rewrite read fwrite frewrite fread freread

64 4 495678 152376 1824993 2065601 2204215 875739 582008 971435 667351 383106 363588 566583 889465

64 8 507650 528611 1051124 1563289 2071399 1084570 1332702 1143842 2138827 1066172 1141145 1303442 2004783

64 16 587283 1526887 2560897 2778775 2366545 1122734 1254016 593214 1776132 463919 1783085 3214531 3057782

64 32 552203 402223 1121909 1388380 1162129 415722 666360 1163351 1637488 1876728 1685359 673798 2466145

64 64 551580 1122912 2895401 4911206 2782966 1734491 1825933 1206983 2901728 1207235 1781889 2133506 2780559

128 4 587259 1525366 1801559 3366950 1600898 1391307 1348096 547193 666360 458907 1486461 1831301 1998737

128 8 292218 1175381 1966197 3451829 2165599 1601619 1232122 1291619 3273329 1827104 1162858 1663987 1937151

128 16 650008 510099 4120180 4003449 2508627 1727493 1560181 1307583 2203579 1229980 603804 1911004 2669183

128 32 703200 1802599 2842966 2974289 2777020 1331977 3279734 1347551 1152291 684197 722704 907518 2466350

128 64 848280 1294308 2288112 1377038 1345725 659686 1997031 1439349 2903100 1267322 1968355 2560063 1506623

128 128 902120 551579 1305206 4727881 3046261 1405509 1802090 1085124 3649539 2066688 1423514 2609286 3039423

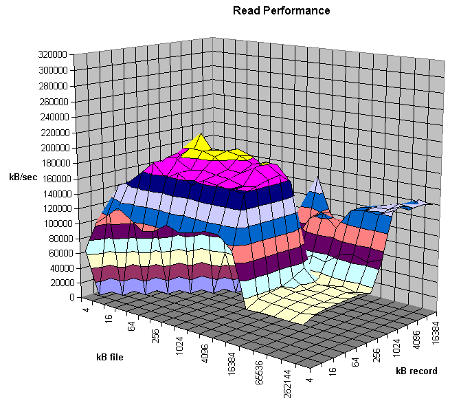
...

2. Save the output to a spreadsheet using iozone -b

To save the iozone output to a spreadsheet, use the -b option as shown below. -b stands for binary, and it instructs iozone to write the test output in binary format to a spreadsheet.

$ ./iozone -a -b output.xls

Note: The -b option can be used with any of the examples mentioned below.

From the data that is saved in the spreadsheet, you can use the create some pretty graphs using the graph functionality of the spreadsheet tool. The following is a sample graph that was created from iozone output.  


**Fig**: IOZone graph

3. Run only a specific type of test using iozone -i

If you are interested in running only a specific type of test, use the -i option.

Syntax:

iozone -i [test-type]

The test-type is a numeric value. The following are the various available test types and its numeric value.

* 0=write/rewrite
* 1=read/re-read
* 2=random-read/write
* 3=Read-backwards
* 4=Re-write-record
* 5=stride-read
* 6=fwrite/re-fwrite
* 7=fread/Re-fread,
* 8=random mix
* 9=pwrite/Re-pwrite
* 10=pread/Re-pread
* 11=pwritev/Re-pwritev
* 12=preadv/Re-preadv

The following example will run only the write tests (i.e both write and rewrite). As you see from the output the other columns are empty.

$ ./iozone -a -i 0

random random bkwd record stride

KB reclen write rewrite read reread read write read rewrite read fwrite frewrite fread freread

64 4 353666 680969

64 8 477269 744768

64 16 429574 326442

64 32 557029 942148

64 64 680844 633214

128 4 187138 524591

**Combine multiple iozone test types**

You can also combine multiple test types by specifying multiple -i in the command line.

For example, the following example will test both read and write test types.

$ ./iozone -a -i 0 -i 1

random random bkwd record stride

KB reclen write rewrite read reread read write read rewrite read fwrite frewrite fread freread

64 4 372112 407456 1520085 889086

64 8 385574 743960 3364024 2553333

64 16 496011 397459 3748273 1330586

64 32 499600 876631 2459558 4270078

4. Specify the file size using iozone -s

By default, iozone will automatically create temporary files of size from 64k to 512M, to perform various testing.

The 1st column in the iozone output (with the column header KB) indicates the file size. As you saw from the previous output, it starts with 64KB file, and will keep increasing until 512M (by doubling the file size every time).

Instead of running the test for all the file sizes, you can specific the file size using option -s.

The following example will perform write test only for file size 1MB (i.e 1024KB).

$ ./iozone -a -i 0 -s 1024

random random bkwd record stride

KB reclen write rewrite read reread read write read rewrite read fwrite frewrite fread freread

1024 4 469710 785882

1024 8 593621 1055581

1024 16 745286 1110539

1024 32 610585 1030184

1024 64 929225 1590130

1024 128 1009859 1672930

1024 256 1042711 2039603

1024 512 941942 1931895

1024 1024 1039504 706167

5. Specify the record size for testing using iozone -r

When you run a test, for a specific file size, it tests with different record sizes ranging from 4k to 16M.

If you like to do I/O performance testing of an I/O subsystem that hosts oracle database, you might want to set the record size in the iozone to the same value of the DB block size. The database reads and writes based on the DB block size.

reclen stands for Record Length. In the previous example, the 2nd column (with the column header “reclen”) indicates the record length that should be used for testing IOzone. In the previous example outout, for the file size of 1024KB, the iozone testing used various record sizes ranging from 4k to 16M to perform the write test.

Instead of using all these default record length sizes, you can also specify the record size you would like to test.

The example below will run write test only for record length of 32k. In the output, the 2nd column will now only display 32.

$ ./iozone -a -i 0 -r 32

random random bkwd record stride

KB reclen write rewrite read reread read write read rewrite read fwrite frewrite fread freread

64 32 566551 820553

128 32 574098 1000000

256 32 826044 948043

512 32 801282 1560624

1024 32 859116 528901

2048 32 881206 1423096

6. Combine file size with record size

You can also using both -s and -r option to specific a exact temporary file size, and exact record length that needs to be tested.

For example, the following will run the write test using a 2M file with a record length of 1M

$ ./iozone -a -i 0 -s 2048 -r 1024

random random bkwd record stride

KB reclen write rewrite read reread read write read rewrite read fwrite frewrite fread freread

2048 1024 1065570 1871841

7. Throughput test using iozone -t

To execute the iozone in throughput mode, use -t option. You should also specify the number of threads that needs to be active during this test.

The following example will execute the iozone throughput test for writes using 2 threads. Please note that you cannot combine -a option with -t option.

$ ./iozone -i 0 -t 2

Children see throughput for 2 initial writers 1= 433194.53 KB/sec

Parent sees throughput for 2 initial writers = 7372.12 KB/sec

Min throughput per process = 0.00 KB/sec

Max throughput per process = 433194.53 KB/sec

Avg throughput per process = 216597.27 KB/sec

Min xfer = 0.00 KB

Children see throughput for 2 rewriters = 459924.70 KB/sec

Parent sees throughput for 2 rewriters = 13049.40 KB/sec

Min throughput per process = 225610.86 KB/sec

Max throughput per process = 234313.84 KB/sec

Avg throughput per process = 229962.35 KB/sec

Min xfer = 488.00 KB

To perform throughput for all the test types, remove the “-i 0″ from the above example, as shown below.

$ ./iozone -t 2

8. Include CPU Utilization using iozone -+u

While performing the iozone testing, you can also instruct iozone to collect the CPU utilization using -+u option.

The -+ in front of the option might look little strange. But, you have to give the whole -+u (not just -u, or +u) for this to work properly.

The following example will execute all the test, and include the CPU utilization report as part of the excel spreadsheet output it generates.

$ ./iozone -a -+u -b output.xls

Note: This will display separate CPU utilization for each and every test it performs.

9. Increase the file size using iozone -g

This is important. If your system has more than 512MB of RAM, you should increase the temporary file size that iozone uses for testing. If you don’t, you might not get accurate results, as the system buffer cache will play a role in it.

For accurate disk performance, it is recommended to have the temporary file size 3 times the size of your system buffer cache.

The following example will run the iozone by increasing the maximum file size to 2GB, and run the automatic iozone testing for write tests.

$ ./iozone -a -g 2G -i 0

random random bkwd record stride

KB reclen write rewrite read reread read write read rewrite read fwrite frewrite fread freread

64 4 556674 1230677

64 8 278340 441320

64 16 608990 1454053

64 32 504125 1085411

64 64 571418 1279331

128 4 526602 961764

128 8 714730 518219

...

10. Test multiple mount points together using iozone -F

By combining several iozone options, you can perform disk I/O testing on multiple mount points as shown below.

If you have 2 mounts points, you can start 2 different iozone threads to create temporary files on both these mount points for testing as shown below.

$ ./iozone -l 2 -u 2 -r 16k -s 512M -F /u01/tmp1 /u02/tmp2

* -l indicates the minimum number of iozone processes that should be started
* -u indicates the maximum number of iozone processes that should be started
* -F should contain multiple values. i.e If we specify 2 in both -l and -u, we should have two filenames here. Please note that only the mount points need to exists. The file specified in the -F option doesn’t need to exists, as iozone will create this temporary file during the testing. In the above example, the mount points are /u01, and /u02. The file tmp1 and tmp2 will be automatically created by iozone for testing purpose.

Previous articles in the Linux performance monitoring and tuning series:

* [Linux Performance Monitoring and Tuning Introduction](http://www.thegeekstuff.com/2011/03/linux-performance-monitoring-intro/)
* [15 Practical Linux Top Command Examples](http://www.thegeekstuff.com/2010/01/15-practical-unix-linux-top-command-examples/)
* [7 Practical PS Command Examples for Process Monitoring](http://www.thegeekstuff.com/2011/04/ps-command-examples/)
* [10 Useful Sar (Sysstat) Examples for UNIX / Linux Performance Monitoring](http://www.thegeekstuff.com/2011/03/sar-examples/)