**How to measure I/O Performance on Linux (Doc ID 1931009.1)**

**In this Document**

|  |  |
| --- | --- |
|  | [Goal](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l GOAL) |

|  |  |
| --- | --- |
|  | [Solution](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l FIX) |

|  |  |
| --- | --- |
|  | **[Introduction](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section21)** |

|  |  |
| --- | --- |
|  | [Idea of running any I/O stress testing](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section22) |

|  |  |
| --- | --- |
|  | [Before Start](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section23) |

|  |  |
| --- | --- |
|  | **[Tools](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section24)** |

|  |  |
| --- | --- |
|  | [dd](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section25) |

|  |  |
| --- | --- |
|  | [IOzone](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section26) |

|  |  |
| --- | --- |
|  | [IOmeter](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section27) |

|  |  |
| --- | --- |
|  | **[Usage](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section28)** |

|  |  |
| --- | --- |
|  | [dd](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section29) |

|  |  |
| --- | --- |
|  | [IOzone](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section210) |

|  |  |
| --- | --- |
|  | [Installation](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section211) |

|  |  |
| --- | --- |
|  | [Usage](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section212) |

|  |  |
| --- | --- |
|  | [IOmeter](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section213) |

|  |  |
| --- | --- |
|  | [Installation](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section214) |

|  |  |
| --- | --- |
|  | [Usage](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section215) |

|  |  |
| --- | --- |
|  | [Things to keep in mind](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section216) |

|  |  |
| --- | --- |
|  | **[Conclusion](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section217)** |

|  |  |
| --- | --- |
|  | [Disclaimer](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l aref_section218) |

|  |  |
| --- | --- |
|  | [References](https://support.oracle.com/epmos/faces/SearchDocDisplay?_afrLoop=162583654952210&amp;_afrWindowMode=0&amp;_adf.ctrl-state=158n3r7v18_4 \\l REF) |

IMG_256

**APPLIES TO:**

Linux OS - Version Oracle Linux 5.1 to Oracle Linux 7.0 [Release OL5U1 to OL7]  
 Linux x86-64

**GOAL**

 How to properly measure I/O performance on Linux using variety of tools

**SOLUTION**

Introduction

In this document Customer should be able to find most efficent/proper way to measure I/O performance on Linux.

Tools/commands which will be used in this document to check I/O performance:

* dd
* iozone
* iometer

Above tools will do great job when it comes to pre-prod where testing and benchmarking I/O is taking place as well.

Idea of running any I/O stress testing

The answer is simple - to prepare System to be 'ready' to take any heavy I/O operations. System is only one layer of Structure when it comes to I/O performance.

Things which are involved as well:

* Storage ( this should cover Storage unit and FC Switches )
* Network ( if  NFS, ISCSI is being used )
* App/DB Layer ( if used )

When I/O bottleneck is being observed mostly its getting observed under OS or APP/DB Layer but we need to know there are still some things on lower layer like Storage Unit/Network  
 Preparing system to stand any heavy I/O traffic is really important in pre-prod phase of system deployment, this document should help Storage/Sys Admin/APP-DB teams to check if system is able to match requested I/O.  
 Cooporating teams can simple test different scenarios like for example see if RAID-5 will be enough to meet APP/DB requirement or if RAID-10 should be considered.

System Administrators will be able to prepare different test scenario to even 'mimic' true Production Database stress test and come with the results, this can save some time while trace down the real problem, if for example above tests will prove that System is able to take High Load and benchmark data will show that system is able to run with maximum 'power' to meet I/O requests - then the issue might be somewhere else, maybe APP/DB layer is a true root cause here ( too many datafiles in tablespace, wrong query ? ) or simple wrong RAID setup on Storage or wrong Path selection policy eg. Active-Passive Load-Balance or Round Robin- This document should help in such scenarios.

Before Start

Please **NOTE** that above test tools like IOzone and IOmeter can bring system to his 'knees' as they can cause huge performance impact ( depends from pre-loaded test which will be used for example to mimic true Production Database activity )

Pay maximum attention to system performance before running test, it will be completely pointless to run IOzone and IOmeter if I/O performance is already bad on system, it will be better to prepare 'windows' for such testing and completely stop APP/DB so we can test system when its completely Idle to really check System I/O Potential and match it to real life scenarios.

Iozone / IOmeter may also **crash system / damage data** if both tools will be used in wrong way or used when system is already under high load.

Tools

Lets describe in quick words how above tools can be used and what they can measure

dd

Simple Linux command which can be used to measure I/O performance on disk or file-system system level

**pros:** easy to use, "out of the box" , can measure both disk ( Block ) and file-system level

**cons:** can only create simple read/write scenarios, unable to reproduce true real-life scenario ( like database access/activity test )

IOzone

Advanced File-System Benchmarking tool which uses sequential I/O access to actual files

**pros:** robust file-system performance tool, can mimic real-life I/O activity, can test multiple file-system at once, provide .xls data which can be used to prepare Graphic I/O Metrics,  can be used to test sequential/random synchronous/async operations,

**cons:** can cause high system utilization, may **damage** the data sitting on tested file-system, not so easy to use like dd command, used to perform only file-system based I/O scenario

IOmeter

Advanced RAW Device Benchmarking tool which uses asynchronous I/O to access block level

**pros:** robust RAW device ( LUN, Disks ) performance tool, advanced real-life scenarios, most advanced to mimic any kind of I/O access, provide .csv data which can be used to prepare Graphic I/O Metrics, can be used to test sequential/random async operations, Graphical UI

**cons:** can cause high system utilization, may  **damage** the data sitting on tested RAW DEVICE/File-System, most advanced tool which require some time to practice it, used to perform only RAW based I/O scenario

 Usage

dd

Simple examples for dd command usage to run variety of tests:

==WRITE TEST==

# time dd if=/dev/zero of=test bs=1M count=256

Above command will try to write 256MB to your Memory Cache, result should be really quick but you need to be aware that your disks are not 'touch' in this test, simple data will be written to cache and then sync to disk.  
 This test can simple prove if OS is able to make quick write to Memory cache.

# time dd if=/dev/zero of=test bs=1M count=256 conv=fdatasync

Above test is similar to previous one but here **conv=fdatasync** flag is being used. It simple says to dd command to flush the data to disk after it finish write to Memory Cache. This command should take more time as previous one as data will be synced to disk level.

**NOTE:**

Most important thing in write dd test is **bs** flag which is block-size please experiment with this setting to achieve better/real performance results mostly bs should be size of 4K 8K 256K 512K check what is the block-size for database or underlying LVM volume to match your bs size to your app/db/LVM block-size  
 Please also remember to change '**count**' flag so for example to create 246MB file with *bs=4K* please use *count=60000*

==READ TEST==

# dd if=test of=/dev/null bs=4096k

This is most basic read test, it again use bs flag , so please experiment with it to achieve more 'real' results.  
 Please pick-up any 'test' file from your local file-system or NFS share to test read performance.

**NOTE:**

Please remember to 'flush' cache before read test, as you might get speed from 'memory cache'

To do this please execute

# sync  
 # echo 3 > /proc/sys/vm/drop\_caches

IOzone

*Installation*

First install IOzone on testing machine

Download proper release package of IOzone eg. EL5 or EL6, in this document EL6 (OL6) is being used, click [here](http://pkgs.repoforge.org/iozone/iozone-3.424-2.el6.rf.x86_64.rpm /o IOzone rpm for OL6/EL6) to download it, for OL5/EL5 click [here](http://pkgs.repoforge.org/iozone/iozone-3.424-2.el5.rf.x86_64.rpm /o IOzone for EL5/OL5).

[root@solaris] # yum install iozone-3.424-2.el6.rf.x86\_64.rpm

We are using yum in case if any dependencies need to be installed.

Check if you can execute iozone command

[root@jenova] # iozone

Usage: For usage information type iozone -h

IOzone is now installed and ready to use

*Usage*

Common flags used by IOzone:

* *-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*
* *-i Used to specify which tests to run. (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=mixed workload, 9=pwrite/Re-pwrite, 10=pread/Re-pread, 11=pwritev/Re-pwritev, 12=preadv/Re-preadv)*
* *-I Use DIRECT IO if possible for all file operations. Tells the filesystem that all operations to the file are to bypass the buffer cache and go directly to disk*
* *-o Writes are synchronously written to disk. (O\_SYNC). Iozone will open the files with the O\_SYNC flag. This forces all writes to the file to go completely to disk before returning to the benchmark.*
* *-H Use POSIX async I/O with # async operations. Iozone will use POSIX async I/O with a bcopy from the async buffers back into the applications buffer*
* *-k Use POSIX async I/O (no bcopy) with # async operations. Iozone will use POSIX async I/O and will not perform any extra bcopys. The buffers used by Iozone will be handed to the async I/O system call directly*
* *-G Use msync(MS\_SYNC) on mmap files. This tells the operating system that all the data in the mmap space needs to be written to disk synchronously*
* *-r Used to specify the record size, in Kbytes, to test. One may also specify -r #k (size in Kbytes) or -r #m (size in Mbytes) or -r #g (size in Gbytes). Try always to match it with app/db/file-system block size*
* *-m Tells Iozone to use multiple buffers internally. Some applications read into a single buffer over and over. Others have an array of buffers. This option allows both types of applications to be simulated. Iozone’s default behavior is to re-use internal buffers. This option allows one to override the default and to use multiple internal buffers*
* *-e Include flush (fsync,fflush) in the timing calculations*
* *-+u Used to enable CPU statistics collection*
* *-+n No retests selected. Used to prevent retests from running*
* *-b Iozone will create a binary file format file in Excel compatible output of results*

 IOzone use many other flags, please check manual for iozone to get any other flags which can be used to test I/O performance

 ==READ/WRTIE TESTS==

[root@pandora]# iozone -l 2 -u 2 -r 16k -s 10M -F /nfsmount/tmp1 /nfsmount/tmp2

Above command will simple create 2 files if they don't exist with record size ( -r ) of **16K** and file size will be **10M**. IOzone will perform all tests on nfsmount share  
 Please also note that IOzone will write data to Cache so speed my appear high but above scenario can be used to verify if caching is working fine.

[root@pandora]# iozone -I -l 2 -u 2 -r 16k -s 10M -F /nfsmount/tmp1 /nfsmount/tmp2

Above command is similar to previous one with one difference, flag **-I** is being used so **data will be written directly to to disk** ( this will bypass Memory Cache so Direct I/O eg. O\_SYNC is used)  
 This result should bring more realistic output

[root@pandora]# iozone -o -l 15 -u 15 -r 8k -r 16K -s 512K -+u -F /nfsmount/tmp1 /nfsmount/tmp2 /nfsmount/tmp3 /nfsmount/tmp4 /nfsmount/tmp5 /nfsmount/tmp6 /nfsmount/tmp7 /nfsmount/tmp8 /nfsmount/tmp9 /nfsmount/tmp10 /nfsmount/tmp11 /nfsmount/tmp12 /nfsmount/tmp13 /nfsmount/tmp14 /nfsmount/tmp15

Above command will create **15** test files which are **128K** in size with record size of **16K**, IOzone will also use **-o** **Syncrhonous Access ( O\_SYNC )** mode and will report CPU stats for each test ( **-+u** flag )

Example output from IOzone:

Run began: Thu Oct 2 15:48:50 2014

Include fsync in write timing  
 Record Size 16 kB  
 File size set to 1024 kB  
 CPU utilization Resolution = 0.001 seconds.  
 CPU utilization Excel chart enabled  
 No retest option selected

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.  
 Min process = 15  
 Max process = 15  
 Throughput test with 15 processes  
 Each process writes a 1024 kByte file in 16 kByte records

Children see throughput for 15 initial writers = 40158.15 kB/sec  
 Parent sees throughput for 15 initial writers = 19398.05 kB/sec  
 Min throughput per process = 254.10 kB/sec  
 Max throughput per process = 5415.58 kB/sec  
 Avg throughput per process = 2677.21 kB/sec  
 Min xfer = 48.00 kB  
 CPU Utilization: Wall time 0.389 CPU time 0.031 CPU utilization 7.96 %

Children see throughput for 15 readers = 638236.86 kB/sec  
 Parent sees throughput for 15 readers = 412259.33 kB/sec  
 Min throughput per process = 29350.08 kB/sec  
 Max throughput per process = 54177.53 kB/sec  
 Avg throughput per process = 42549.12 kB/sec  
 Min xfer = 1024.00 kB  
 CPU utilization: Wall time 0.035 CPU time 0.011 CPU utilization 31.52 %

Output is really easy to understand, results are in KB/s , its always good to take Average ( Avg ) result and Child result ( this is what APP/DB will see for all processes )

IOzone can be used to test any file-system type like: EXT3/4, LVM, XFS, BTRFS, NFS, SAMBA etc. etc - so then we pick-up best one which will suit best app/db needs.  
 Also if more threads are being used with flag **-l** the more higher queue depth will become.

 NOTE:

Please remember that -F can be used to specify different mount-points so if we need to test for example two LVM mount-points in the same time then simple please provide different paths for each test file like /u01/test1.tmp and /u02/test2.tmp.  
 The more files will be added for testing purpose the more load CPU will need to take, same also go for file-size.  
 If only one test is required for example write test use -u flag example: -u 0 will test only write/rewrite.  
 IOzone has many flags which can be used to test synchronous/asynchronous random/sequential I/O access, please use them to test variety of scenarios.  
 Always try to match sector size eg. -r flag with App/DB/LVM block size  to achieve best performance results.  
 IOzone can be used to bypass system memory cache by using flag -I ( so it will use DirectI/O ) or don't use it to include Memory Cache mechanism

*Please pay maximum attention while running IOzone tests as they can cause high CPU or I/O activity, also keep in mind File-System space ( do not exceed file-system size with test files)*

IOmeter

*Installation*

First lets setup Server,  32bit installer which also comes with Import Wizard tool converting .csv to graphical metric ( MS Access )  is [here,](http://sourceforge.net/projects/iometer/files/iometer-stable/1.1.0/iometer-1.1.0-win32.i386-setup.exe/download /o IOmeter Installer) 64bit version of 1.1.0 Release is [here](http://sourceforge.net/projects/iometer/files/iometer-stable/1.1.0/iometer-1.1.0-win64.x86_64-bin.zip/download /o IOmeter archive with IOmeter Server package only) ( without Import Wizard ) or older 1.1.0-RC1 is  [here](http://sourceforge.net/projects/iometer/files/iometer-devel/1.1.0-rc1/iometer-1.1.0-rc1-win64.x86_64-bin.zip/download /o IOmeter 1.1.0-rc1)

Run installer and choose what component to install after that we should get IOmeter Server UI:

IOmeter

*Installation*

First lets setup Server,  32bit installer which also comes with Import Wizard tool converting .csv to graphical metric ( MS Access )  is [here,](http://sourceforge.net/projects/iometer/files/iometer-stable/1.1.0/iometer-1.1.0-win32.i386-setup.exe/download /o IOmeter Installer) 64bit version of 1.1.0 Release is [here](http://sourceforge.net/projects/iometer/files/iometer-stable/1.1.0/iometer-1.1.0-win64.x86_64-bin.zip/download /o IOmeter archive with IOmeter Server package only) ( without Import Wizard ) or older 1.1.0-RC1 is  [here](http://sourceforge.net/projects/iometer/files/iometer-devel/1.1.0-rc1/iometer-1.1.0-rc1-win64.x86_64-bin.zip/download /o IOmeter 1.1.0-rc1)

Run installer and choose what component to install after that we should get IOmeter Server UI:

IMG_257

NOTE:

Linux can also be used as IOmeter Server but WINE will be necessary in this case - in this example Windows is being Used

Now lets configure our Client ( OL6 box )

NOTE:

Linux can also be used as IOmeter Server but WINE will be necessary in this case - in this example Windows is being Used

Now lets configure our Client ( OL6 box )

Download Linux Client package [here](http://sourceforge.net/projects/iometer/files/iometer-stable/1.1.0/iometer-1.1.0-linux.x86_64-bin.tar.bz2/download /o IOmeter Linux Client Latest) ( latest/final but require 2.14 GLIBC ) or go for older 1.1.0-RC1 [here](http://sourceforge.net/projects/iometer/files/iometer-devel/1.1.0-rc1/iometer-1.1.0-rc1-linux.x86_64-bin.tar.bz2/download /o IOmeter 1.1.0-rc client-linux) - *note that you can Use Client Linux Release 1.1.0-rc1 with 1.1.0 Server package without any problems*.

Now un-tar package and cd to directory and execute dynamo

[root@kaneda] # tar -xvf iometer-1.1.0-rc1-linux.x86\_64-bin.tar.bz2  
 [root@kaneda] # cd iometer-1.1.0-rc1/  
 [root@kaneda] # ./dynamo -i IOmeter\_Server\_IP -m Linux\_client\_IP

Similar output to below one should appear after successful connection:

[root@kaneda]# # ./dynamo -i 127.1.1.10 -m 127.1.1.10

Dynamo version 1.1.0, Intel/AMD x64 64bit, built Nov 8 2010 22:23:00

Command line parameter(s):  
 Looking for Iometer on "10.167.250.206"  
 New manager name is "kaneda"

Sending login request...  
 kaneda  
 10.167.242.76 (port 59638)  
 Successful PortTCP::Connect  
 - port name: 10.167.250.206

\*\*\* If dynamo and iometer hangs here, please make sure  
 \*\*\* you use a correct -m <manager\_computer\_name> that  
 \*\*\* can ping from iometer machine. use IP if need.  
 Login accepted.  
 Reporting drive information...

Physical drives (raw devices)...  
 Reporting TCP network information...  
 done.

IOmeter Server should now detect Linux Client:

IMG_258

NOTE:

Please make sure to open port 59638 from both ends and make sure if IOmeter Server and IOmeter client can ping each other otherwise connection will fail and IOmeter Server will simple 'hang'

*Usage*

Now as connection is established and Linux Client is detected we can choose physical disks from Client side to be tested like in above IOmeter Server screen test will be performed on sdb and sdc ( use ctrl key to choose multiple disks )

First choose what test will be performed:

IMG_259

Each access/test can be edit - so manual changes can be made if necessary

IMG_260

Also its possible to edit 'Test Setup' where many other things can be changed like interval/cycling of tests/Run time etc. etc.

IMG_261

Results can also be observed 'live'

IMG_262

To start the test simple click 'Green Flag' , IOmeter will ask where to save .csv result which can be then used to prepare some graphic results

After all tests will be finished we will get summary page with average IOPS

IOmeter can also use configuration files with pre-loaded tests so it will be easy later on to run various tests to simple mimic different scenarios like: Web Server access, Database Acces, Workspace Access, Server Access etc. etc.

To load configuration file simple click folder icon and choose IOmeter .icf file with preloaded test

Pre-loaded tests will hold their own Access patern and as well their own Test Setup, below example show Access Patern to test Database Access ( OLTP )

IMG_263

Example configuration files with pre-loaded tests are available [here](https://support.oracle.com/epmos/main/downloadattachmentprocessor?parent=DOCUMENT&amp;sourceId=1931009.1&amp;attachid=1931009.1:IOMETER_CONFIG_FILES&amp;clickstream=yes \\o IOmeter_config_files_pre-loaded_tests). Many of IOmeter tests can also be found on External Webpages

Iometer is designed for RAW device performance, it can be used for file-system as well but this is not so widely used.

Iometer is most advanced tool to test RAW disk performance, so it can simple test any FC/ISCSI/FCOE LUN or directly attached disks to Server or any JBOD/RAID disks, Iometer can mimic plenty of real life scenarios, it can be used to test LUN which will be then used as ASM disk so throughput can be obtain for such device.

Also its wise to check results with Storage Vendor and verify if actual Iometer results meets Storage Unit performance, Storage Vendor should be kind and helpful to get IOPS which can be achieved on tested Storage unit compare those with Iometer result.

Iometer test its also useful when it comes to look for any performance issue, as it tests RAW Device perfomance we are sure than nothing else is being involved like: Paritioning/File-system/ASM/LVM etc. etc. so we can prove that actual RAW device performance is performing properly and issue might be somewhere else on other 'layers'

As already been said for IOzone, please always try to match blocksize with APP/DB/RAID-STORAGE STRIPE to achieve best performance and most realistic metrics

IOmeter is also capable to be run test with **different Queue Depth** and that's really useful feature so we can actually test different queue depth with multiple workers and this should bring even more IOPS to picture. Queue Depth value can be tuned on OS/Storage. As default IOmeter use queue depth of **1** ( *Outstanding I/O per target* in '**Disk Targets**' View ) but it will be good to change it as it won't bring realistic metric.

**NOTE:**

**As Iometer perform RAW device benchmark - pay MAXIMUM attention while choosing device for testing as tested disk will be wiped/modified.**

Iometer results are easy to understand as they simple provide IOPS/MB/s for each device/test so its easier then to use .csv file and make graphical metrics for example:

IMG_264

To create such graph please use build-in Import Wizard for IOmeter Server 1.1.0 Installer ( Windows ) but it will require MS Access and Excel

Linux perl script can be also used to achieve graphical metrics, its included in IOmeter Linux 1.1.0 Final Release - graph.pl, it can be also downloaded [here](http://fossies.org/linux/privat/iometer-1.1.0-src.tar.gz/iometer-1.1.0/misc/graph.pl?m=b \\o graph.pl for Linux), after that simple provide .csv file from IOmeter Server and conversion process will start.

Things to keep in mind

In above tests following keywords were used: Blocksize, Queue Depth, Stripe-Size / Stripe-Width, Async/Sync, Direct IO  here is explanation of them:

*Blocksize - Block: The smallest unit writable by a disk or file system. Everything a file system does is composed of operations done on blocks. A file system block is always the same size as or larger (in integer multiples) than the disk block size. Block size is commonly used in DB/APP terminology as it represent size of block which will be written to file-system or directly to disk, for example Oracle Database is using by default 8K block-size. Please keep this in mind while doing IOzone or IOmeter tests so block-size which will be used in in IOzone/IOmeter will match your file-system/APP/DB block-size requests for best performance results ( just make sure that block-size in IOzone/IOmeter is at least same size of APP/DB/FS block )*

*Queue Depth - This topic is already covered in KM DOC: [1579548.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=1579548.1)*

*Direct IO - This topic is already covered in KM DOC: [462072.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=462072.1)*

*Asynchronous Access - This topic is already covered in KM DOC: [462072.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=462072.1)*

*Synchronous Access - This topic is already covered in KM DOC: [462072.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=462072.1)*

*Raid Levels - This topic is already covered in KM DOC:* *[30286.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=30286.1 \\o Raid Levels for Oracle Database)*

*Stripe-size / Stripe Width (Array) - Stripe Size is size of the stripes written to each disk in Array, Stripe width refers to the number of parallel stripes that can be written to or read from simultaneously, this equal number of disks in array ( so a six-disk striped array will have a Stripe Width of 6 )  
Below image explain Stripe-Size and Stripe-Width:*

IMG_265

*Stripe-Size play his position when it comes also to I/O Performance, for example bigger stripes should be used when APP/DB is doing large amount of small read/write transactions, and use smaller stripes when app/db is needs to read quickly larger files with small amount of transactions. Stripe-Size plays major factor when it comes to RAID levels, as for example RAID with Striping with Parity will require extra reads/writes to maintain integrity.*

*Keep in mind that busy Database which perform relatively small read operations like 4-8K bigger Stripe-Size should be used so only one disk will do simple seek operations rather than causing multiple disks to perform seek operation which will equal latency. Many Storage Vendors are using larger Stripe-Size for their Storag Units something between 128K - 256K or even bigger, some of them are for example designed to bring I/O performance in OLTP where small size of block 4K is being used to constantly update Datbase pages bigger Stripe-Size will be used with RAID-10 as each write on for example RAID-5 array will cause parity adjustment penalty.  
Please keep this all in mind while doing IOmeter test as Stripe-Size on Array will play major factor there, as IOmeter can test variety of block-size and percentage of read/write operations this will impact overal IOmeter results.  
Its wise to test different Stripe-Size and see how IOmeter will perform so results will help to build optimal I/O Enviorment which will suit APP/DB needs.*

Conclusion

Using above tools like IOzone and IOmeter should help to answer some of the I/O performance statistics. Each tool has been designed to do something else than other so IOzone will measure file-system based performance and IOmeter will test RAW Device. Combining them both should bring some useful stats and testing for any Pre-Prod Enviorment which will be used for any stress test. Above tools are also great tools for any Production system struggling I/O issues so they can be simple reproduced by variety of tests. dd command its just added as a basic 'tool' to measure I/O performance but as it can be observed it really limited when it comes to deep dive into I/O benchmarking. Each tool can be used as 'exclude' tools any kind of I/O problems which are mostly observed on last layer like APP/DB/End-user, using above tools can help to 'remove' some aspects like file-system or RAW device from being Root Cause of I/O performance issues. Customers should also feel free to contact Storage Vendor if they want to know if results are meeting Storage Unit performance so things like IOPS/Throughput can be confirmed with Storage Vendor - or simple check Storage Unit Document and look for possible IOPS results with it. IOzone and IOmeter can be used as a 'comparsion' tools so if Customer will be migrating from one Storage Model/Vendor to new one, both tools will be great place to start and compare I/O Performance which can be achieved in real-life scenarios on both units and see which one will perform better in different tests ( as each Storage Unit might be designed to serve different purpose )

IOmeter and IOzone should be used by Sys-Admins who have experiance in OS and as well in Storage field, as some results might be miss-leading and will cause unnecessary confusion.

If HBA Cards performance is also concerned while performing above tests, KM doc: " [How to measure HBA ( Host Bus Adapter ) Performance/Utilization on Linux](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=1627808.1) "might be useful to check as it explain how to obtain HBA health-status/throughput/performance/config/errors using variety of tools and options to achieve it.

I't might be also good to install iotop command which is always useful to have and its added to OL5/6 channels to bring sys-admins some I/O monitoring tool which might be helpful to monitor any I/O benchmark test.

To install iotop simple execute:

[root@discovery-one]# yum install iotop

Then simple run it to start monitor I/O traffic

Disclaimer

Please note that Oracle Linux/VM team will be unable to help or support in case of IOzone/IOmeter queries or problems, those are 3rd party Open Source tools and they are not supported/maintained by Oracle.  
 Oracle will not take any responsibility of any inappropriate usage which will cause system to hang/crash or simple damage any Production/Test file-system or Disk/LUN.  
 Above tools were provided as a good start to test/meassure I/O performance but they require some basic knowledge in OS/Storage field.

More information around configuration, usage, example tests or config files for IOzone / IOmeter can be found on External Webpages.

**REFERENCES**

[NOTE:462072.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=462072.1) - File System's Buffer Cache versus Direct I/O  
 [NOTE:1579548.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=1579548.1) - What is the HBA Queue Depth and How to Check the Current Queue Depth Value?

[NOTE:1627808.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&amp;sourceId=1931009.1&amp;id=1627808.1) - How to measure HBA ( Host Bus Adapter ) Performance/Utilization on Linux