**Performance comparison of filesystems on Linux**

**AUTHORS**

Praveen Dareddy

Jayesh Tambe

**Table of contents**

1. **Introduction** 3
2. **Questions** 3
3. **Relevant Research Survey** 3
4. **Methodology** 4

4.1 Configuration 4

` 4.2 Testing tool 6

4.3 Classification of files 7

1. **Results** 7
2. **Conclusions** 11
3. **References** 12

**1. Introduction**

Choice of a file system is an important decision to be made while designing a computing system for a particular use. As an example, If we are planning to use a server as a File Server, file system used must be robust and high performant to meet expectations. In this paper, we are trying to determine the best file system for a particular use case, given an underlying Hardware. Use cases serve as the best method to analyze different file systems and case study can then be used as recommendation for choice of file system. Underlying hardware, such SSD or HDD for operating system can also make huge impact on performance of operations. In this paper, we have compared various file systems throughput using HDD and SSD Hardware. We will be comparing ext3, ext4, btrfs and xfs. This paper can be used to determine file system to fully exploit advantages of an underlying hardware.

**2. Questions**

1. Which Filesystem under Linux, using SSD hardware has best performance with respect to read and write operations for a set of large files and a set of small files?  
 2. Which Filesystem under Linux, using HDD hardware has best performance with respect to read and write operations for a set of large files and a set of small files?

**3. Relevant Research Survey**

Comparing file system's performance is quite difficult as there are multiple factors which could affect file I/O such as file name mapping, disk sector organization, block size, metadata, cache and memory buffer. A comparison based solely on technical specifications could be very misleading. For the purpose of our study, to choose the best file system for a given hardware, we have studied several papers which performed benchmarks tests.

Some factors like Record size are telling for performance, transfer rate increases approximately linearly with increasing block sizes [1]. The same study evaluated performance of several filesystems like FAT32, NTFS, Ext2, Ext3. In this study, they have compared filesystem performance on sequential operations, random operations, repeated operations and strided reads. They have concluded that differences in performance are much more dramatic across platforms rather than filesystem on a given platform, and that observed performance advantage is minimal in most cases compared to other parameters [3].

For this study, we wanted to choose one parameter which will effectively represent filesystem’s performance and can be used as defined yardstick. We have finalized on data transfer throughput as our parameter, where data is read from a location and then copied to a location. As we are essentially performing read and write using a single benchmark test, we felt results of our study will be simple, yet compelling for system administrators to use.

**4. Methodology**

**4.1 Configuration**

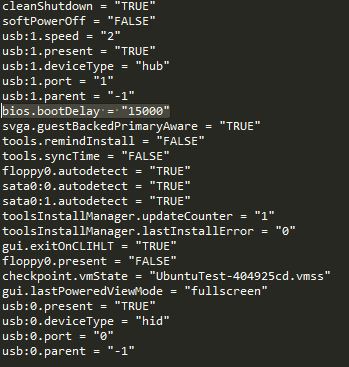
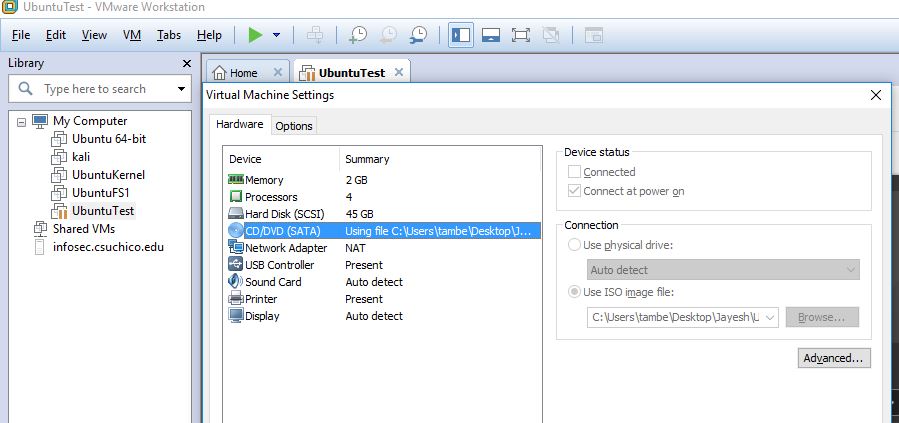
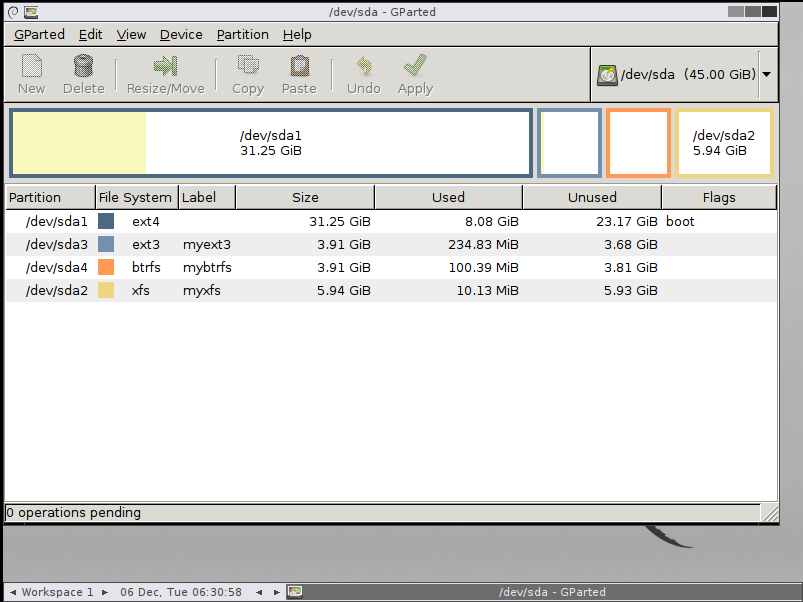
We used following system configuration with Ubuntu 64-bit OS using VMware Workstation.

1. Memory - 2GB RAM
2. No. of processors - 2
3. No. of cores/processor - 2 (Total cores = 4)

4. GParted tool to configure partitions for Ext4, Ext3, Xfs, Btrfs

One major concern to test file systems is to have an exactly same environment, in which we can test the behavior and performance. Best way to do that is to test those systems on a single machine and not create bunch of instances of Linux machines on VMware. To host all file systems under test on a single machine, we found a tool called GParted [4][5]. GParted allowed us to have multiple partitions on a single machine and within each partition we created file systems we wanted to test.

Following are steps to use GParted on Linux machine to create multiple file systems.

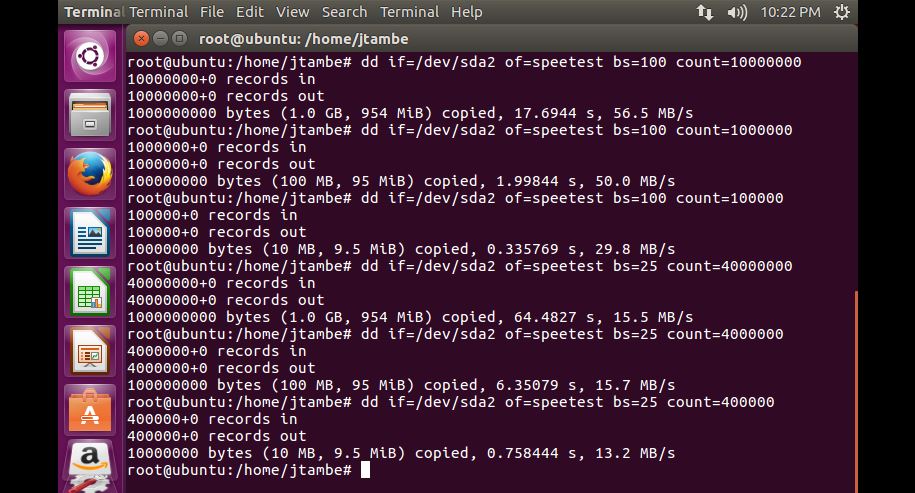
1. Download ISO file (gparted-live-0.25.0-3-i686) from <http://gparted.org/livecd.php>.
2. For the Linux machine, in which one is going to test, one needs to add   
   bios.bootDelay = "15000" attribute in VMware virtual machine configuration  
   
3. Provide the path of downloaded ISO file to VMware machine’s settings.  
   
4. When the system loads, using boot delay, one can enter into GParted and then configure the file systems that need to be tested.
5. Once user selects intended file systems, GParted shows partitions as shown in following image  
   

**4.2 Testing tool** Once the file systems were configured, another challenge was to identify most reliable tool with which we could test the throughput for our file systems. There are bunch of tools like Iozone and Bonnie++ that can be used. However, they are third party tools and are not shipped with linux OS systems.  
 So we decided to use a command which is already a part of Linux OS. ‘dd’ command allows one to test throughput of read and write speeds. Following is description of ‘dd’ command.

Example: dd if =dev/sda2 of=speedtest bs=100 count=10000000

1. If = read from input file instead of stdin
2. Of = write to given file instead of stdout
3. Bs = block size in bytes
4. Count = copy as many input blocks

Above command measures throughput for file of size (bs \* count) which is 1GB.

Following image shows the output from dd commands.  


Command from example can be seen executed first in this image. It creates a file of size 1GB with throughput of 56.5 MB/s.

From research papers, we knew that as block size increases, throughput also increases. Although it comes with fragmentation disadvantages. However, for scope of our research, fragmentation was not under scrutiny.

When we created 1GB file with block size of 25 bytes, we observe throughput of only 15.5 MB/s. This corroborated for us that, ‘dd’ command is reliable tool for our testing. Similar observations can be found for files of sizes 100MB and 10MB in image.

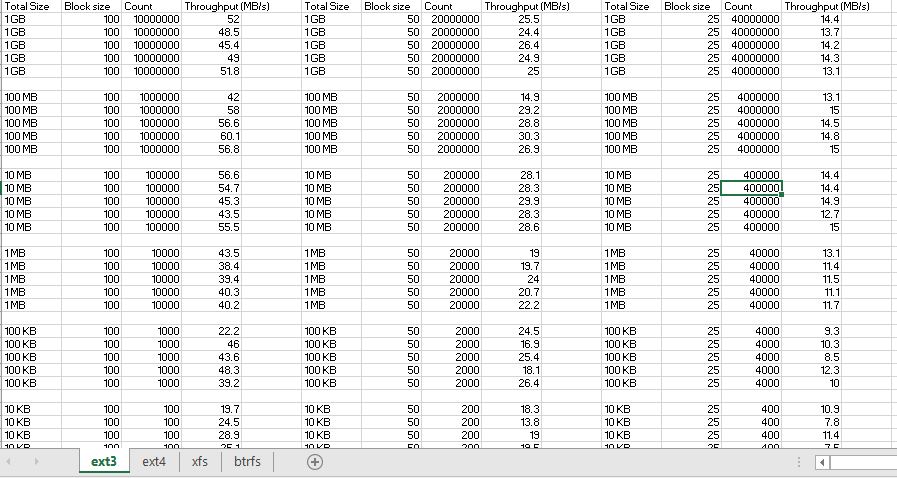
**4.3 Classification of files**

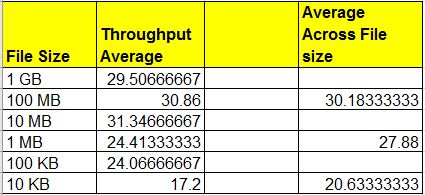
Since we were trying to observe behavior for large and small files, we needed to define size of large and small files under test. Because our testing was done on local machines and not on cloud/ distributed file system, we decided 1Gb-100MB as large file size, 10MB-1MB as moderate file size and 100KB-10KB as small file size.

For readings, we noted 5 readings of throughput data, using block size of 100, 50 and 25 bytes for each file size and aggregated throughput reading for each file size. Further on, we aggregated throughput readings for range of file. That is, we averaged throughput for 1GB-100MB, 10MB-1MB and 100KB-10KB. Details of data observed are further discussed in results section of this paper

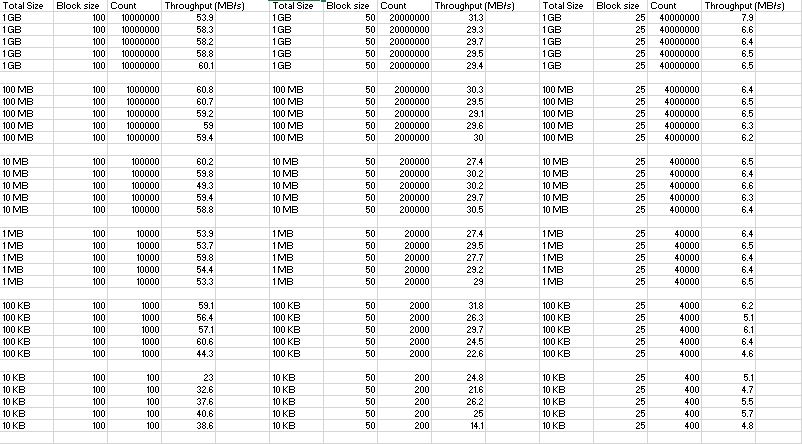
**5. Results**

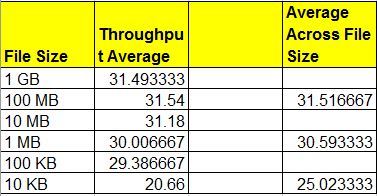
Following two images show data observed for ext3 file system observed on HDD.





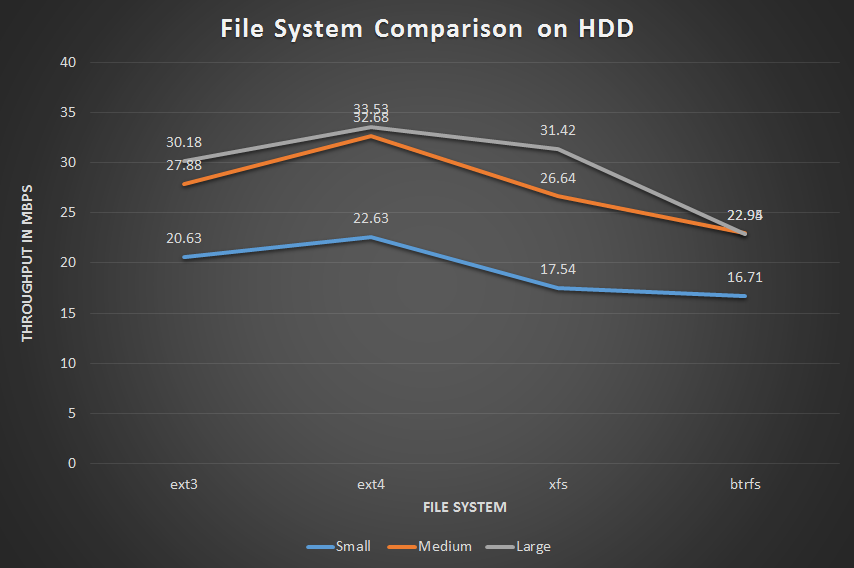
Following two images show data observed for ext3 file system observed on SSD.



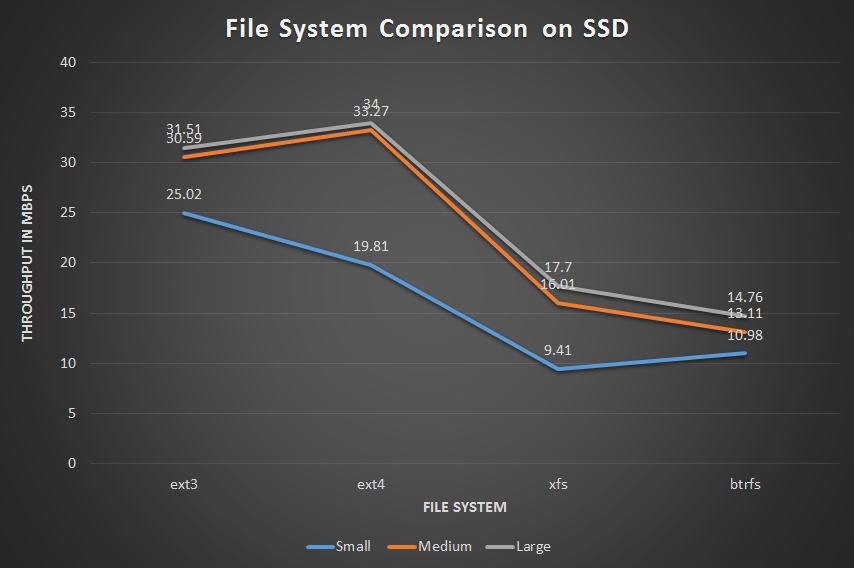


On similar lines, we tested each file system on HDD and SSD, and compared file systems separately, i.e. on HDD and SSD.  
After all data was recorded, we plotted two graphs for different file systems for HDD and SSD.

Following are images of those 2 graphs.



Above graph shows that on HDD, small size files displayed highest throughput, 22.63 MB/s on ext4 and large size files showed highest throughput, 33.53 MB/s on ext4 system, whereas btrfs showed worst throughput for both types of files, 16.71 MB/s for small and 22.95 MB/s for large files.



Above graph shows that on SSD, small size files displayed highest throughput, 25.02 MB/s on ext3 and large size files showed highest throughput, 34 MB/s on ext4 system, whereas btrfs showed worst throughput, 14.76 MB/s for large files and xfs showed worst throughput, 9.41 MB/s for small files.

**6. Conclusions**

In this section, we have provided answers for questions along with other observed findings. In general, we have found out that ext4 has better performance in both large and small files, while btrfs performed poorly in both small and large file classifications.

1. With HDD as underlying hardware, we have found that ext4 filesystem has better performance for large files.

2. With HDD as underlying hardware, we have found that ext4 filesystem has better performance for small files.

3. With SSD as underlying hardware, we have found that ext4 filesystem has better performance for large files.

4. With SSD as underlying hardware, we have found that ext3 filesystem has better performance for small files.

**7. References**

[1] Lanyue Lu, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau, Shan Lu

A Study of Linux File System Evolution <http://pages.cs.wisc.edu/~ll/papers/fsstudy.pdf>

[2] Ray Bryant, Ruth Forester, John Hawkes

Filesystem Performance and Scalability in Linux 2.4.17 <http://oss.sgi.com/projects/xfs/papers/filesystem-perf-tm.pdf>

[3] Matti Vanninen James Z. Wang On Benchmarking Popular File Systems

<https://people.cs.clemson.edu/~jzwang/pub/fbench.pdf>

[4] Gparted Manual

<http://gparted.org/display-doc.php%3Fname%3Dhelp-manual>

[5] Alexander Zeitler Resizing a VMware Workstation VM partition using GParted - get the swap partition out of my way!

<https://alexanderzeitler.com/articles/resizing-a-vmware-workstation-partition-using-gparted/>

[6] Henry Cook, Jonathan Ellithorpe, Laura Keys, Andrew Waterman Exploring File System Optimizations for SSDs

<http://web.stanford.edu/~jdellit/default_files/iotafs.pdf>

[7] Jelena Kljajić, Nada Bogdanović, Marko Nankovski, Marjan Tončev Performance Analysis of 64-bit ext4, xfs and btrfs filesystems on the Solid-State disk technology

<http://infoteh.etf.unssa.rs.ba/zbornik/2016/radovi/RSS-2/RSS-2-8.pdf>

[8] File System Performance: The Solaris™ OS, UFS, Linux ext3, and ReiserFS

<http://www.oracle.com/technetwork/systems/linux/fs-performance-149840.pdf>

[9] Dominique A. Heger Workload Dependent Performance Evaluation of the Btrfs and ZFS Filesystems

<http://www.dhtusa.com/media/IOPerfCMG09.pdf>