Volume 4, Issue 2, February 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Workload Characteristics Impacts on file System Benchmarking

Naveenkumar Jayakumar , Farid Zaeimfar, Shashank D Joshi

Dept. Of Computer Engineering,

BVUCOE, Pune – 43, India

Abstract— File systems in real world, to measure their performance are quite complicated. The problem with the existing techniques for measuring the performance of file system only deals with the architecture and component view of the file systems. The evaluation process concentrates on simulating the file system and their components and interfaces in order to measure the performance. The processes do not take into consideration all the factors which vary the performance of file system in real world. This paper discusses the view of the File system performance impacted by the workloads and the underlying system components.

Keywords— Workload generator, Workload Characteristics, File system Attributes, File system performance, Performance Evaluations

I. INTRODUCTION

Workloads vary in their attributes which actually makes the file system to vary. File systems deployed in different environment will have different workloads and these factors if not considered will not yield accurate results while evaluating the file system performance. Some benchmarking process misses out couple of points to be considered while performing the evaluation. How well the file system is simulated under test which will represent the real time file system working? Secondly, what kind of workloads are supported and how well the performance of file system will be when subjected to various workloads? Which workload will perform well on file system? These questions should be answered by the process or the evaluation process of the file systems.

In systems, the data are stored in the form of files. The storing of files is the level of abstraction which is layered over the physical storage components like hard disks. The same file system provides an interaction layer to the client side in order to execute the functions and methods of the applications on the data. The file system can be viewed in two components one which represents the storing of data on storage devices i.e. Data Depict and the other is logical algorithms deployed which allows the client to store, retrieve and perform various other operations on the data. These set of algorithms can be varied without affecting the data depict. This allows developers to make changes and enhance the functionality of the existing file systems without affecting or changing the data depict.

Along with the data, the file systems also stores a lots of internal data which are much more complicated and without the help of these internal data, to perform an operation the data will be more difficult and close to not possible. The Meta data is regarded as data about data, which gives information about the data which stored on the disk. The Meta data provides information to the file system about what data has been requested by the client, what operation is to be executed and where to look for the memory or data to be placed or retrieved. File systems also use directories to provide a hierarchical name space. There are internal pointers (data structures) which are used to by the file systems to identify and use the files or folders requested by the clients. Each directory and files have attributes like ID, name, inode, Size etc... Each set of other files and directories along with some internal pointers packed together is referred as Directory. The directories makes it easy to find the files searched by the client, because they organize files in a tree like structures rather than providing with a flat name space. The basic unit of transfer between the application and the file system is byte in UNIX environment. So major focus is on maintaining the offset of the last byte transferred to or from the file. This should be tracked by the file system in all the scenarios whenever the application reads or writes data.

The UNIX kernel also supports object oriented approach to develop and deploy files and file system using the virtual file system interface. Using VFS, multiple request can be directed to the respective files system, since the pointers are maintained as per file by the UNIX kernel.

The file system on the storage device are very slow, so the performance also gets degraded by the speed of storage devices, hence the UNIX kernel tries to fetch the data from the hard disks and store it on the cache. Those data are kept in cache which are required by the application frequently. Also UNIX provide the Meta data cache which is also referred to name cache or directory name look aside cache. This buffer space stores the name look up operations which took place recently and an attribute cache which stores the Meta data and attributes. File systems are deployed in order to hide the working speed of the underlying hard disks.

II. PROBLEM DEFINITION

As the time passes the file system behaviour changes dramatically. As the applications perform various operations like creating, deleting and other functions on the file system with varying frequency and the file system gets filled in various

times, the performance attributes of the system also varies. Any researcher when tries to evaluate the file system, they take the file system which created newly and is not filled at the same time, they miss out the above performance characteristics factor which affects the performance. There are some demerits in the current trend of executing the process of evaluating the performance of file system. The current trend considers the file system to be created newly and which is not exposed to the frequent varying operations in the real world. These new file systems show very less changes in their working behaviour even though the test is executed for a day.

In real time implementation of the file systems, there are scenarios where the fragmentation of the free space leads to inconvenience in the contiguous allocation of the data. This will impact the throughput of the file system because, the more new files being created will be affected by the fragmentation and more fragmented files will be created. But while making evaluation, the files created are not fragmented. The data is read and write at optimal speeds by allocating physically contiguous disk blocks to logically sequential data which in turn enhances the throughput optimization.

The current trend of evaluating file system misses out, how the file system evolve with applying the workload. As the time passes and various stress is applied on the file system, there are chances that the file system may change its state and the way it will behave. There are various file system policies which are deployed and their effects on the file system is not considered or not measured while evaluation process takes place. Some of the policies do not get chance to be applied on the file system which is under test and their effects and impacts on the file system may not be recorded. The fragmentation does not affect sequential layout of each files data but also the proximity of related files, their locations and also the Meta data that describes it. Above all how the varying workload will affect all the factors discussed above for deploying and tuning the file systems.

III. APPROACH

It required that when the performance evaluation of the file system is carried out then the file system should not be empty in other words the file systems should represent the data and the all other features of real time file system. This way of analysing the file system performance will depict various patterns and of problems that do not show up when evaluating the new file systems. This paper proposes an approach called creating the matured file systems for evaluating the file system performance. The matured file system is created for test bed and is very much similar to the real time workload subjected file systems. The matured file system consists of sequence of records, and operations like read, write delete, modify file or directory etc. described. The approach also suggests to apply one kind of workload to different file systems and varying workload to different file systems and changing any one object at a time of test. Either the single workload applied to all file systems to be tested or the varying workload applied to one file system.

The above two approach will yield information about how a file system architecture gets affected in behaviour when subjected to varying workloads and in other hand, how well file system architecture performs to what kind of workload can be derived when varying workloads is applied to the file system. The real time workload consist of sequence of data or records which describes the functions like create , delete, modify, rewrite, reread etc. of files or directories. While creating the matured files system workload include only those functions that can sustain the long term state of file system.

The major operations performed on the file systems which affects the performance or helps in simulating the real time workload is read, writes creation and DE-allocation of files.

Using traces and snapshots for creating the matured file system workload will consume terabytes of space which the only hindrance of using these techniques. In order to generate the matured file system workload, create and note down snapshots from various file systems which are deployed in real time and active. The snapshots recorded be in a sequence so that some of the real time workload read in the snapshots can be simulated as it is for test bed file systems.

These sequence of snapshots will help in creating the matured file system workloads. These snapshots sequences are taken from long run real time file systems for months and years.

The approach for creating the matured file system workload follows here. Understand the operations which will setup the test bed file system which will resemble the file system state seen in the snapshots. Generate the sequence of operations which are used in the snapshots. The files operations can be created based on the snapshots inode change times of the files. Considering the time stamps and log from the snapshots try to align the actions being performed rather the snapshots do not provide alignment or the order of actions executed. So generate the workload outline by analysing the change times and inode listings between two consecutive snapshots.

For understanding the operations sequence, it is necessary to extract the timestamps of each operations which is toughest nearly impossible to find from the snapshots. So the sequence of these operations in this case is assumed. These change times related to inodes observed and noted down for each files will help you to find the respective time that file's Meta data is accessed and modified. So whenever there is a modification to Meta data, it is considered that disk blocks are allocated or new file is created. If two snapshots are carried out to have same inode but varying file attributes, then the file is either modified or replaced. These process of recording the inode change times from snapshot will benefit to measure the long lived files but what about files create and deleted for short durations or some operations which misses out i.e. not recorded between the snapshots. In order to fill the gap, the workload can be added up with set of create and delete operations which will be shifted with respect to time so that, the operations for short lived files is scattered for the whole workload. This approach will yield a new matured file system workload created out of observing the existing snapshots of the real time file system workload which is active. The file system stores all the logs and stores the behaviour of the file systems to each and every operation in the real time executed on it. It stores the time the operations started and ended, operation executed on the files, after each operations the meta data is accessed and the logs are updated so the complete information about what kind of workload was applied to the file system can be read or extracted

by studying the file system logs and operations executed on these file systems. The performance of files system depends on various factors like workload, block size, randomness of requests, no. of requests, buffer size etc. These metrics can be measured from the real time file system snapshots and create a similar workload which is referred to as matured file system workloads.

IV. WORKLOAD AND TESTBED

Since there is proliferation in applications and data sizes, there is increase in the complexity of the computing there by making performance measurement to the workload specific for a file system. It is important to know that what file system will support or perform well subjected to various types of workload. The analysis of the matured file system workload of different types applied on various file systems will yield a closer performance model of real time required performance. There exists many tools which will help in applying the workload to different set of file systems in order to derive the performance model of how well a file system suites the workload and perform well when subjected to that workload.

These tools are of open source and the tools come with the source code. In order to integrate the mature file system workloads to the existing tools, the API's and c files of the source code to be configured as per the real time file systems workload under consideration for the research being carried out. These tools have two parts one is the workload generator and second one is the analysis and performance predictors. The user can have the file system profile published by the micro benchmark tests which shares information about all the information about attributes and the file system functionality.

The below figure 1 explains the basic components of the tools which are used for analysing the performance of the file system.

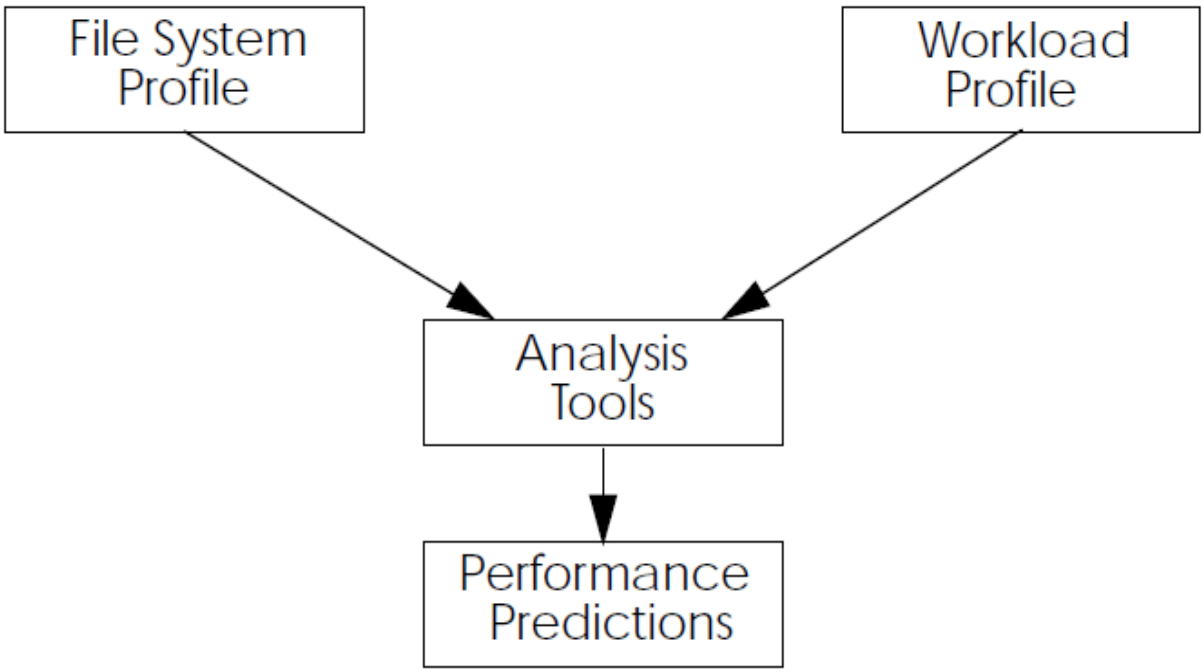

Fig 1: components of analysis tools

The tests have been perform on various file systems in ext4 and ext3 with the raw devices. The graphs shown below depicts the IOPS throughput, response time relations through executing read and write operations. The clustered environment maybe virtualized where the performance can get dropped down in the real time simulating the file systems i.e. deploying the matured file system workloads. The figure 2 shows the graphs which depicts the variations in the writes performed on the file system with respect to MBPS employed for the empty file system and matured file system workload.

There are many factors to be considered by performance is to be benchmarked. The factors which may impact the file system working in a clustered environment are block size, IO page size, capacity, cache size, frame size then you have IO alignment. If the IO size gets decrease while transferring that is a shoot up in IOPS and at the same time throughput decreases .while considering the response time the component utilization in clustered environment needs to be considered. The tests are tried to be executed in both environment with empty file system as well as matured file system workloads.

The figures 2, 3 and 4 represent the results of the tests performed on the both file system with empty file system scenario and other with the mature file system workload. The values in the case of throughput i.e. MBPS is also more compared to the empty file system as same with the case of response time. The response time in the empty file system is less than the matured file system workload. These are measured with respect to the time stamps. The system used for test beds are Intel core with hard disk of raw device of 8 GB and EXT 4 file system with 512 Bytes of block size. The IO size is mixture of 8KB to 100KB.

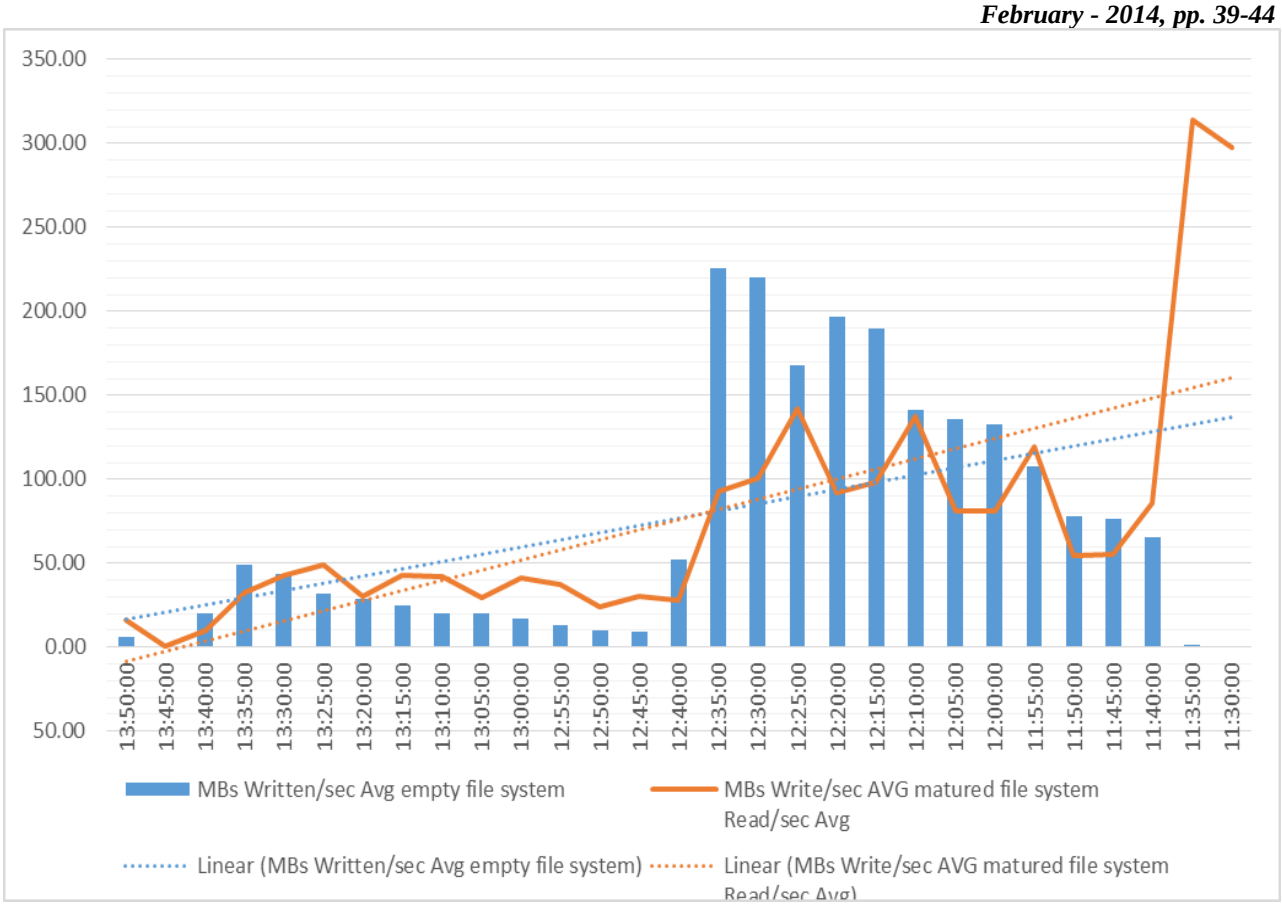

Fig 2: Empty File system and Matured file system throughput

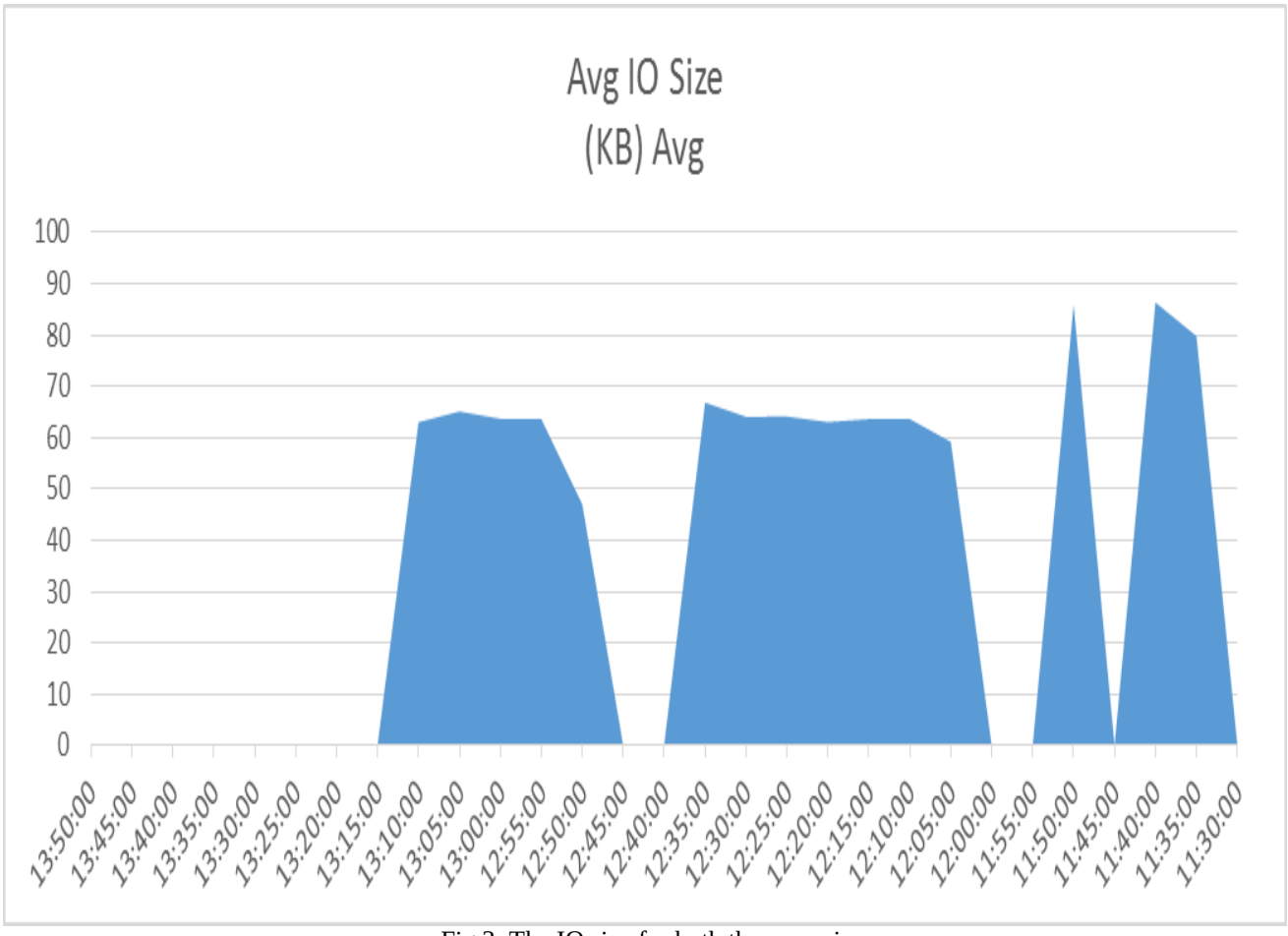

Fig 3: The IO size for both the scenarios.

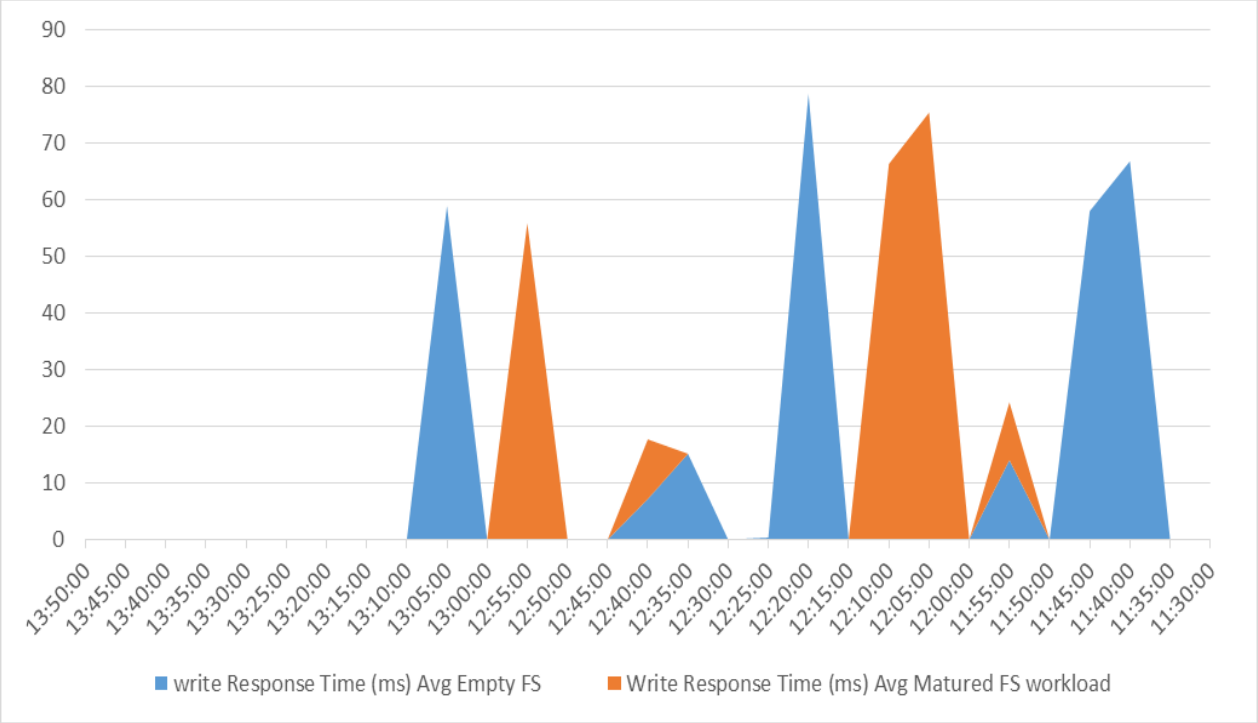

Fig 4: Response time for both the scenario tests

V. CONCLUSIONS

The performance of the file system will very well the clustered environment. File system benchmark needs to be derived before deploying it to real environment. System is created when you initialize or format your hard disk. It sets up the root directory and subsequent directories beneath it. That system is file system and allows you to create new files and folders, which are added to different parts of the "file tree" on your hard disk the same as programs, documents, pictures, music, and movie files. All these folders are organized by your computer's file system. Some of these folders are invisible to the user, but are recognized by the computer's file system also. It is nice to know that it is always working to keep your files organized. However, in any performance systems, high efficiency is always desirable, and many systems have specific efficiency requirements. Achieving high system efficiency involves measurement of system performance metrics and optimization based on estimated results. Whenever the performance evaluation of file system is to be done the file system workload should be of real time and the matured file system workload can be created as discussed in this paper and applied to the file system under test. This will yield the results or performance model closer to the real time expected performance results.

.

REFERENCES

[1] http://en.wikipedia.org/wiki/List_of_file_systems
[2] "Encina". Zois.co.uk. Retrieved 2012-06-15.
[3] "IBM - TXSeries for Multiplatforms, V7.1 - TXSeries for Multiplatforms - Software". 306.ibm.com. Retrieved 2012-06-15
[4] "Melio FS". Sanbolic. Retrieved 2012-09-21
[5] http://pcsupport.about.com/od/termsf/g/filesystem.htm
[6] http://pcsupport.about.com/od/termsf/g/fat.htm
[7] "Description of the exFAT file system driver update package". Support.microsoft.com- 2011-10-08- Retrieved 2012-06-15
[8] Michael Larabel (2011-10-05). "Samsung Introduces New Linux File-System: F2FS". phoronix.com-Retrieved 2012-12-07.
[9] "United States Patent: 5392427". Patft.uspto.gov. Retrieved 2012-06-15.
[10] http://hcc-embedded.com/products/file-sytems/flash-file-systems
[11] "gouffs". Sites.google.com. Retrieved 2012-06-15.
[12] "Embedded File System". RoweBots. Retrieved 2012-06-15.
[13] http://pcsupport.about.com/od/termsns/g/ntfs.htm
[14] http://linux.about.com/od/evm_guide/a/gdeevm164.htm
[15] http://pdos.csail.mit.edu/ivy/osdi02.pdf
[16] "Pastis P2P file systems". REGAL project. Retrieved May 30, 2013.
[17] "Nimbus filesystem". Retrieved May 30, 2013.
[18] http://en.wikipedia.org/wiki/Distributed_file_system#Distributed_file_systems

[19] [Baker91] Mary G. Baker, John H. Hartman, Michael D. Kupfer, Ken W. Shirriff, John K. Ousterhout. Measurements of a Distributed File System. Proceedings of the 13th Symposium on Operating Systems Principles (SOSP), pp. 198 – 212. Monterey, CA. October 1991.

[20] [Baker92] Mary Baker, Satoshi Asami, Etienne Deprit, John Ousterhout, Margo Seltzer. Non- Volatile Memory for Fast, Reliable File Systems. Proceedings of the 5th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-V), pp. 10 – 22. Boston, MA. October 1992.

[21] [Barve99] Rakesh Barve, Elizabeth Shriver, Phillip B. Gibbons, Bruce K. Hillyer, Yossi Matias, Jeffrey Scott Vitter. Modeling and optimizing I/O throughput of multiple disks on a bus. Proceedings of Sigmetrics '99, pp. 83 – 92. Atlanta, GA. May 1999.

[22] [Beck98] Michael Beck, Harald Böhme, Mirko Dziadzka, Ulrich Kunitz, Robert Magnus, Dirk Verworner. Linux Kernel Internals, Second Edition. Addison Wesley Longman Limited. Harlow, England. 1998.

