Analysis based on Linux file system performance

Zhi Li

ID:1718663

*Abstract*—an Operating System consists of many parts, including security system, I/O device, file system was one of those . It is responsible for storing files and managing them, which does have many responsibilities. However, the speed could reflect the performance of file system. The performance depends on the parts of it : category structure, file managing structure. This project aims to find out the reason of performance reducing in EXT2/3 file system and give out a proper suggestion for improvement.

Keywords—Operating System,EXT , Linux, performance, plan(key words)

1. Introduction

Linux system has been a popular system since it was invented in 1991 due to many features such as open-source, high compatibility to many softwares [1].However, the file system of Linux has longer history than the system itself. The predecessor of Linux file system is Minix[2]. It is more than a file system but an Operating system, which was invented before Linux was born. Minix file system was once a successful file system at its time but an apparent weakness frustrated it. The partition is less than 64MB[3], but the limited spaces could not even fulfill the need of one file at current age. After Linux was invented, the urge of a new file system has prompted EXT’s born. In 1992 , the first version of EXT( EXT1) was launched in Linux[4]. Up to now, EXT the fifth version EXT5 has been published and it is still the main file system of Linux[4]. This report will give result and analysis mainly on EXT2/EXT3 version.

1. Background

Data Structure of EXT2/3

Linux system supports many kinds of file systems[4]. To discuss that, it is necessary to learn about the structure. Linux file system consists of two parts: virtual file system and concrete file system[5]. Among them, virtual system is the entrance and user interface of file system while the other one is concrete file systems such as EXT2, Minix, sysV. The category structure was the subset of the concrete file system. The main object of this report is EXT2/3.Thus the category structure in this report refers to the category structure of EXT2/3 file system. The core of Ext2/3 is two kinds of inner data structure, among which one is Superblock while the other is Inode[6]. Superblock is a form which contains important information of file system such as tags, capacity, numbers of index nodes(Inode). It is the global description of file system structure. In EXT2/3, Inode is the basic structure. Each node could be viewed as a file, the node contains information which concludes the description and the content of this file. All the nodes were kept inside a table called index table, and each item in the table would be arranged an unique identification called index number. Thus to find out the file is to find out the index in the index table.

Moreover, the category of EXT2/3 is a special file. Thus as a file, it is an Inode, but in this Inode it contains all the category items in this category, and each item is an Inode, too. The structure of Inode and superblock was shown in Figure 1.

Inode

As mentioned above, Inode is the basic data structure of EXT2/3. In Linux file system, all the files are Inodes even including the category. All the Inodes are unique, which means there are not identical Inodes in EXT2/3 file system.Inside an Inode there are 15 block-pointers, among which 12 are used as the direct pointers, another 3 pointers are used as indirect pointers. Data is kept in block called data block. The pointers will lead the file system to find out the data block which it searches for .Figure 2 shows out the relationship between the pointers and the data block of file.



Figure1.Ext2 data structure

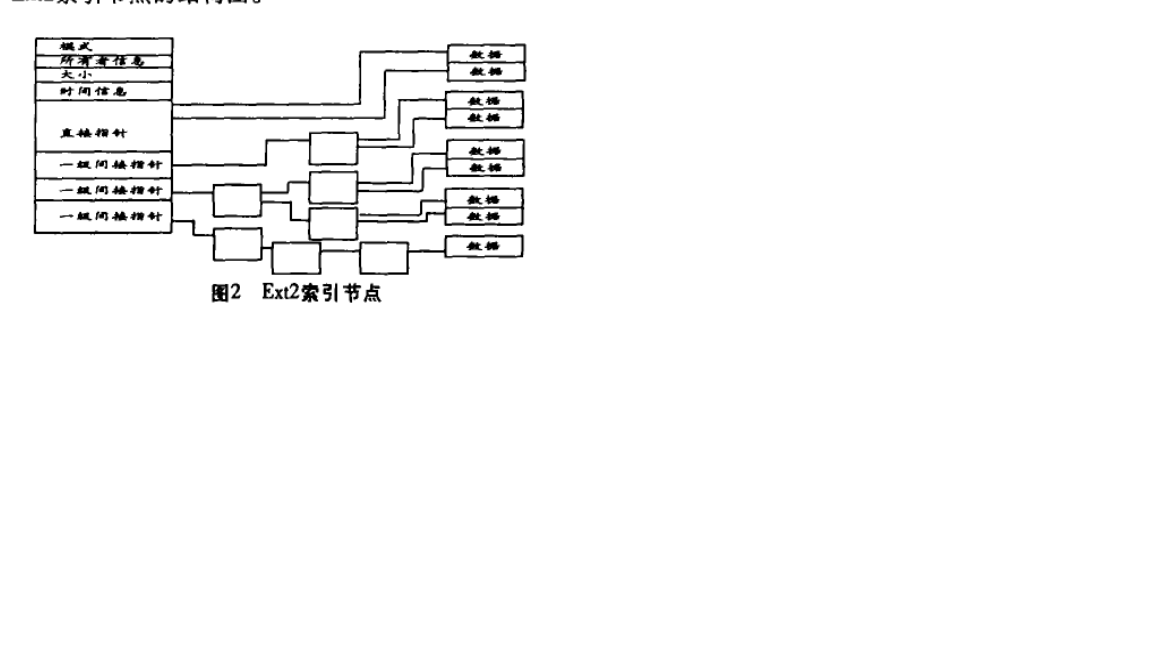


Figure 2 .Ext2 relationship between the pointers and the data of file

1. Analysis of reason for lower performance

No matter what the file system it is, the substance of storing data is to use the data structure of this file system to arrange the files and data. Thus to analyze the reason for the lower performance is to analyze the data structure of this file system. Assume a data block is 1024 bytes, which means it could store 1024 bytes data. Using the former 12 pointers, the maximum direct storing space can reach 1024\* 12 = 12kb. If we keep some data in this block, and if the data is less or equal to 12kb, then it will be so quickly to reach and open them, because they can be visited directly by the Inode. However, if the amount of data is more than 12kb, Inode will use a first-level indirect pointer to point at the first-level indirect block to assist store data. The first-level indirect storing space can store 256\*1024 =268KB data. If the first-level is still not enough, this Inode will use the second-level pointer and so on till the third-level pointer being used. However, the visiting speed will decrease with the indirect level increasing[7]. In addition, large amount of data will cause the lower performance of EXT2/3 file system.

Ⅳ. Comparision:

NTFS

NTFS file system stands for “New Technology File System”, which was born with Windows NT Operating System[8]. In NTFS, the core of category structure is MFT(main files table). MFT could be seen as an array, every file in the system could be a record in this array[8]. Each record only takes 1KB space, and was ordered from 0. Thus in NTFS, to get a file is to find out the order of this file. This central alignment of files is quite different with EXT2/3. Firstly, in EXT2/3 a section of disk will be portioned out to store the Inodes, while in NTFS MFT is an ordinary file so that NTFS will not portioned out a section to store this file. Secondly, EXT2/3 will have more levels as the files capacity goes up. As mentioned above, if the data is more than 12KB, EXT2/3 will use indirect pointer to point at another data block. This will cause the level of category increase, for example, a 12 KB data will be stored only in one level of category, but a 200Kb data will be stored with two levels of category [9]. Once the category level goes up, the visiting speed and the file performance will decrease in a large scale. No matter in category arrangement or storing strategy, NTFS’s category structure theoretically has better performance than EXT2/3.

Ⅴ. Experiment

The experiment aims to compare the two file systems performance which is in the same hardware condition. NTFS file system test was carried out in Windows 10 System while EXT2/3 test was carried in Linux centOS7. The test including the writing speed into a category and reading speed from the same category. The category structure of two systems are different, thus we could only choose to set the test file in a similar depth of files, For example in Windows E://testdir/test and in Linux /tmp/test. The test result was shown below:

Test account of data is 8G, in EXT3 writing speed is 1.1GB/s and reading speed Is 2.2GB/s, but in NTFS the read speed is 3.5GB/s and write speed is 1.2G/s. Excluding the difference of testing software and operating system. The performance gap can be contributed to the file system and the category structure.

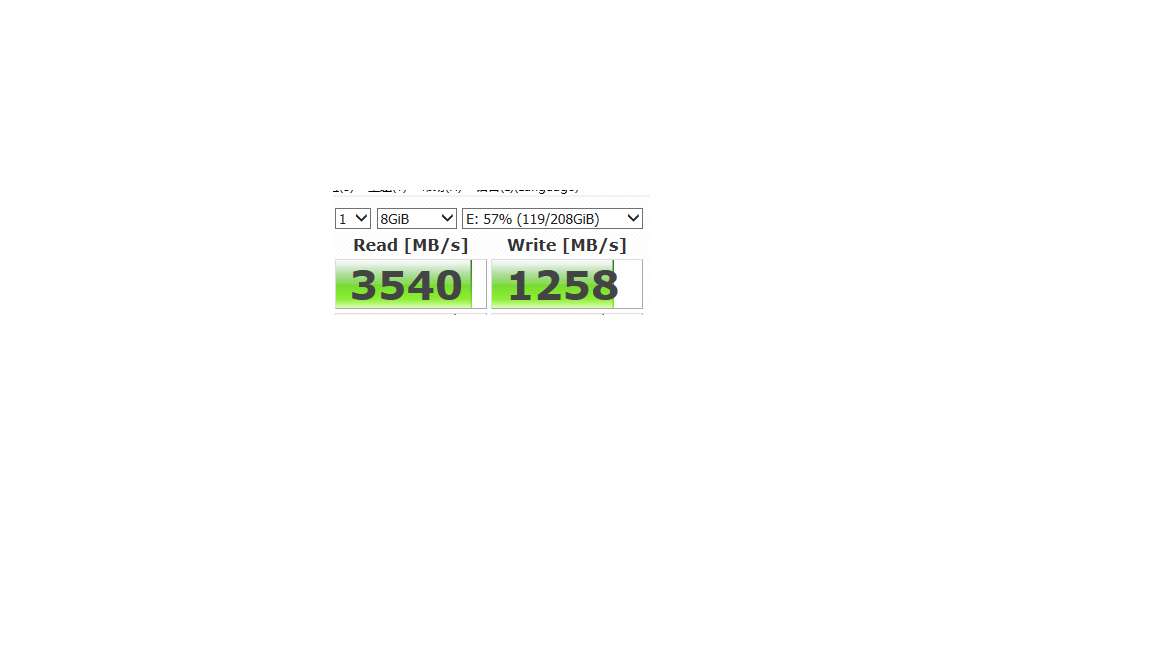


Figure 3: NTFS read/write speed



Figure 4:EXT3 read speed



Figure 5: EXT3 write speed

Ⅴ. Evaluation

There are two points the testing does not perform:

1. Measuring group is very limited, the sample could not stands for the average and final result. Error of this experiment does exist
2. Excluding the system, there are still many factors which could influence the result were not taken into consideration.

However, this testing process is still worth referring. In data amount control, a large amount of data was chosen to test(8G), this amount could typically reflect the weakness of EXT2/3 file system has. Meanwhile, to reflect the performance which file system does, the category structure level was kept consensus, which could reduce the difference error which category structure brings.

Ⅵ. Conclusion and Suggestion

Through the experiment and theory analysis, the reason of EXT2/3 performance reducing is when large amount of data needs to be stored, Inode will use many levels of indirect pointers to point to the large data blocks. It is similar to the ‘jump’ process, which means when a higher place needs to be reached, the people needs many times of jumping, thus the process naturally spent more time than directly reaching the destination. To solve this problem, structure of NTFS file system was taken into consideration. It is suggested that in EXT2/3 system could use a file to store the Inode like an array, the index of this array could be the Inode’ s unique number. In this way , Inode could save much time from unnecessary indirect redirection to find out the data block.

Ⅶ. References

[1] V.Steven (2011,April.13) *Twenty Years of Linux according to Linus Torvalds* [Online].Available:

https://www.zdnet.com/article/twenty-years-of-linux-according-to-linus-torvalds/

[2] P.Hall (Apr.9, 2000) *MINIX is now available under the BSD license* [Online].Available:

http://minix1.woodhull.com/faq/mxlicense.html

[3] M.M EI Din Mahmoud and A.EI-kadi, “A DOS/Linux extensible file system

”, inProceedings Second IEEE Symposium on Computer and Communications. Alexandria, Egypt, July 1997, pp.306-319.

[4] Q.Chen .”Linux kernel file system analysis and research”, Ph.D.dissertation, Dept.Comput.Sci., Univ.Nanjing normal university , NanJing, JiangSu, 2009.(in Chinese)

[5] B.Moshe.*Linux File System.* Beijing. Tsinghua university press, 2003

[6] X.Guo et al. ,"Ext2 File System of Linux” in *Computer application research.* [Online]. \vol.18, no.5 pp.128-130 May.2001.Available:

https://www.ixueshu.com/document/86ebf3c16188b229318947a18e7f9386.html (in Chinese)

[7] JiLe.Li. “File system performance optimization technology research” Doctoral dissertation,Dept.Comput.Sci.,Univ.China University Of Petroleum,China,2014(in Chinese)

[8] Yang L et al. “Analysis and Implementation of UFS File System Based on Computer Forensics,*” Applied Mechanics and Materials,* vol.1, no.3, pp.325-328, 2010

[9] XJ.Tang et al, “check the file systems” in *Science and Technology Information: Science Teaching and Research*, vol.2 no.18 pp.72-72(in Chinese)