ZCM-DE: A New Approach to Enhance Local Data Transfer Rate of Small Files in Windows 10's OS

Magid Abdullah Basmael[#1] and Dr. Saeed Mohammed Baneamoon[#2]
[#1]*Department of Information Technology,*
[#1]*Faculty of Engineering and Information Technology,*
[#1] *Al-Rayan University, Yemen*
[#2]*Department of Computer Engineering,*
[#2] *College of Engineering & Petroleum,*
[#2] *College of Computers & Information Technology*
[#2]*Hadhramout University, Yemen*

***Abstract*— Windows is the most popular computer operating system. It is considered as a multitask operating system. One of the main and frequent tasks is the data transfer locally from a source to a destination. The main problem is the effect of the operating system data transfer rate due to size change and number of files that would be transferred, specifically when many small files are transferred. Consequently, this problem has led to negative effects on the performance of transferring many small files locally. There are solutions that enhance small files' issues, but unfortunately, many of them are for distributed operating systems such as CW-FFS, LHF and HAR, and if no one, a very few are for local computers "data transfer rate". In this paper, a new approach is proposed, based on Zero-Compression Merge and Direct Extraction (ZCM-DE), to enhance local data transfer rate of many small files, from a source to a destination in Windows 10's, without any modifications to the operating system architecture or file system structure, and also to reduce very many operations that the operating system performs during the file transfer process. The result show that the new proposed approach enhances Windows 10's operating system data transfer rate of many small files more efficient compared with other related approaches.**

***Keywords*—Data transfer rate, Extraction, Local, Many small files, Merging**

I. INTRODUCTION

In the early digital computing, people were penetrating with a few amounts of data, and there were less problems affecting data transfer. Later, requesting data drastically increased.

Technology moved on, so speeds and volumes spread vertically and horizontally. In modern technology, speeds and volumes are beyond imagination. The average annual personal computer data traffic worldwide from 2015 to 2022 would raise up from 0.4 to 1.3 exabytes [1], and based on Statcounter, only desktop Windows 10's version market share worldwide is 70.98% [2]. Even the operating systems contains internally many small files to work properly. A quick search on my Windows 10's c:\ volume, there are 414142 files up to 256KB.

Nowadays, working with big data and transferring small files is a natural routine on personal computers. The computer technology timeline suffered from many obstacles, and one of those obstacles is slow data transfer rate. There are contributions that have improved big data transfer rates on both, personal computer and distributed architecture.

In fact, the real problem is not in big data transfer rate; it is about small data transfer, especially when transferring many small data. On the PC, when transferring big-size files, the data transfer rate is fine. In contrast, and in a case of transferring many small-size files, the operating system data transfer rate is so negatively affected [3].

Hence, this paper will focus on the challenge of Windows 10's OS data transfer rate for small files in order to enhancement of operating system.

The rest of this paper is organized as follows. Section II discusses related work about processing small files problem. Section III analyses the data transfer rate of an operating system, specifically Windows. Section IV presents the design details of ZCM-DE, including fundamental idea, data management and the

implementation. Section V analyses the experiments, discuses the results, and evaluates the performance of the developed approach. Finally, conclusions are presented in Section VI.

II. Related Work

Weakness of operating system data transfer rate of small files is a challenge. There are many contributions state-of-the-art to enhance the problem in the literature. Those contributions vary upon perspective view of the problem reasons and environments. Some focused on local environment or distributed environment. Some contributions focused on physical structure or logical structure and some others focused on both. On the other hand, there are contributions enhanced the hard disk drive, others enhanced the main memory (RAM) and some others enhanced the internal build of storing. Also, there are contributions enhanced the problem by adding extra hardware.

The following review summarizes several related works.

CW-FFS is proposed as a multiple-file write approach for improving write performance of small files in Fast File System FFS. The approach basic idea is to collect large numbers of modified small files in a buffer cache, then write the data to disk in large disk I/O's to improve the performance of small file writes in server-based environment. The approach has some limitations. It is based on modifying a file system hierarchy on a server to deal with only the files that their contents are 768B [4].

hashFS approach applied hash function to hash pathname to increase small file read performance for a workload typical in Web 2.0 scenarios and does not rely on a name-based or temporal locality or large in-memory lookup tables. A single small file read is performed with a single seek nearly independent of the organization and size of the file set or the available cache. The approach enhances only file read process, modifies the file system and directed to the web environment [5].

FMP-SSF is an approach to optimize storing and accessing small files on cloud storage. The approach improved the memory available on NameNode limitation of HDFS scalability. Also, it classified files into three types: structurally-related files, logically-related files, and independent files. File merging strategy and file grouping strategy are adopted. Also, it adds other adaptations. The approach is for a distributed system and has a set of many techniques other than merging that are sophisticated and might add time delay [6].

SHDFS approach is proposed for HDFS system. The approach applied two modules: merging module and caching module. In merging module, a set of correlated files is combined, as identified by the client, into a single large file. In caching module, design of a special memory subsystem in which some data are duplicated for quick access. This approach is for a distributed system, uses extra hardware and merge a set of files identified by user to improve access [7].

LHF is an approach proposed to accelerate many small files access performance in HDFS using linear hashing file. In this approach, small files problem of HDFS is improved and the files to be merged do not need to be sorted, which makes appending additional files to existing merged file easier. This approach is for file access of a distributed system [8].

The previous related works have been reviewed and have led to the identification of certain strengths, contrasts and limitations of the existing approaches.

It can be seen that all of the previous approaches are for distributed systems, none of them is for local systems and data transfer. Also, some of them are sophisticated implemented by many steps and may produce more resources consume.

Therefore, in this paper, the proposed approach improves the "data transfer rate" of many small files locally in relation to the personal computer operating system such as Windows 10's OS with less cumbersome techniques and resource consume. It includes all file formats, does not add more hardware, and independent of a file system.

III. BACKGROUND

Local data transfer rate of Windows 10's OS is affected by a number of files and changes of file sizes. So, if there are many small files to be transferred, Windows 10's OS is affected negatively in terms of data transfer rate and resources consumption.

This section presents a background through which the causes of the problem are explained in detail.

A. Same Data Rate but Different File Rate

Because Windows 10' OS data transfer rate is affected by many transferred small files, it implies that the "file transfer rate" changes according to the change of their sizes and number, even if the data transfer rate of a PC has the same value. So, the time of transferring 20 files their total size is 2000 KB, differs from the time of transferring 40 files their total size is 2000 KB [9]. So, file transfer rate term will be used if needed.

B. Windows OS's Interfaces

Windows OS is a software which acts as an interface between the end user and computer hardware (resources). So, it carries out tasks from any resource to any other resource based on the user's instructions [10]. So, for the operating system to work perfectly, Windows OS is divided into interfaces, each of which provides operations from an interface to another. The interfaces are: Application interface, Call interface and Service interface. So, for any task user executes by a computer, it must pass through all of the OS's interfaces. They isolate the user from complicated details of resources [11].

C. Windows OS's I/O's Operations

Windows OS interfaces may be considered as regulating gates to use resources with a correct way [11]. The interfaces execute many Input/Output (I/O's) operations. In Windows OS, they are processed by "IRP" requests, which are the responsibility of I/O Manager [12].

D. Layered Operating System

From above, it can be seen that any task would be executed using Windows 10's OS, it must pass through many interface I/O's operations up and down. This is called "layered OS" approach. So, any task even a small task, passes through many operations. So, a file transfer task I/O's operations are divided into read operations and write operations [11] [12]. We have found that the total number of read/write operations of one complete file transfer task is 106 operations. So, in case of many small files, it is educed that the number of read/write operations increases bigger and bigger leading to the problem of software weaknesses of the operating system data transfer rate, increasing a time delay and consuming more resources.

So, since a layered approach software has some success, it is generally not ideal for designing operating systems due to performance problems [11].

IV. PROPOSED METHOD

Many I/O's operations of the operating system produce the software weaknesses of data transfer rate and more resource consume.

So, to enhance this problem in case of transferring many small files, a new approach is proposed. Its main idea is minimizing I/O's operations of Windows 10's OS. This is by Zero Compression Merge and Direct extract (ZCM-DE).

A. Design

Files are divided into 12 groups or folders based on their sizes as follows:

512B, 1KB, 2kB, 4kB, 8kB, 16kB, 32kB, 64kB, 128kB, 256kB, 512kB and 1MB, and each folder contains 2000 files.

So, the first folder contains 2000 files, each of which is 512B, the second folder contains 2000 files, each of which is 1KB, and so forth.

Using the data transfer approach of Windows 10's OS, the files of size 512B are transferred to a clean formatted destination volume (F:\ partition).

The complete transfer time of the traditional technique of the operating system is approximately 27.22s, and the data transfer rate is 73.48 file/s.

B. Proposed Approach

The new proposed approach will be as follows:

Firstly, the files are merged together in the source to create an enough big file to minimize many I/O's operations of Windows 10's OS. Secondly, as soon as the merge file is created, all the files which it contains are extracted directly to the destination. Thirdly, when the all files have been already extracted, the merge file is deleted. The new proposed approach is shown in Fig. 1.

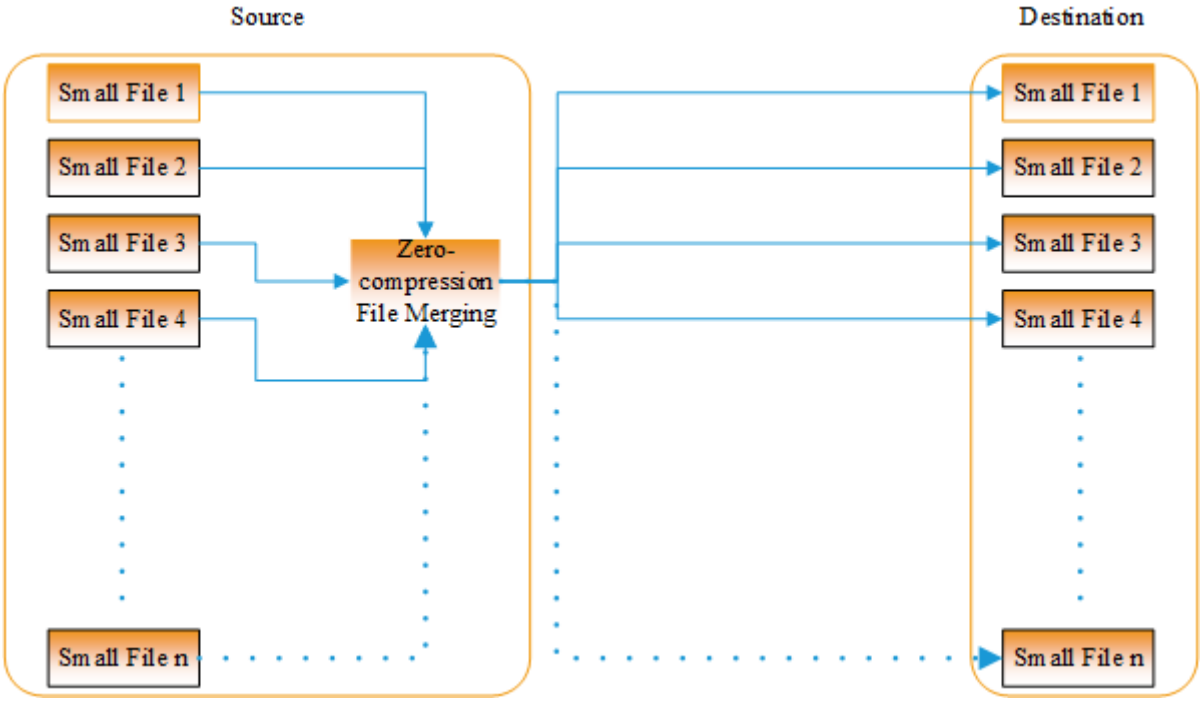

Fig. 1 Principle of ZCM-DE approach

The complete transfer time of the proposed approach for this task is approximately 11.85s and the data transfer rate is 168.78 file/s.

V. Experimental Results

A. Proposed Approach

The new proposed approach is evaluated from two enhancement perspectives; time and resources consume efficiencies.

The above scenario is applied repeatedly to all folders related to their sizes. Table 1 summarizes all experiments of the data transfer of Windows 10's OS approach and ZCM-DE approach. It is shown that using the new proposed data transfer approach, the data transfer rate of the 512B files is 168.78 file/second, whereas using the traditional operating system data transfer approach, the data transfer rate is 73.48 file/second.

TABLE I
CALCULATIONS OF EXPERIMENTS

Folder files	Proposed DTR	Folder files
512B	168.78	73.48
1KB	135.87	63.94
2KB	152.09	68.14
4KB	134.59	63.63
8KB	132.71	69.47
16KB	108.46	63.63
32KB	101.83	63.80
64KB	92.12	53.94
128KB	84.60	41.55
256KB	52.25	27.94
512KB	12.33	18.48
1MB	6.01	6.92
Mix of above	62.23	28.05

Fig. 2 shows the efficiency of the proposed approach in term of the data transfer rate compared with Windows 10's OS. It is shown that the data transfer rate of the new proposed approach is efficient until 512KB file size.

On the other hand, resource consumption is evaluated as well. The tests are observed from perspective view of disk read/write operations of the file transfer using DiskCountersView application. The average read and write count per second of the proposed approach from the source to the destination is approximately 10,837 read/second and 13,928 write/second respectively, whereas for Windows 10's OS, they are 8341 read/second and 9605 write/second respectively. So, the new approach improves the performance of the operating system resources consume efficiently.

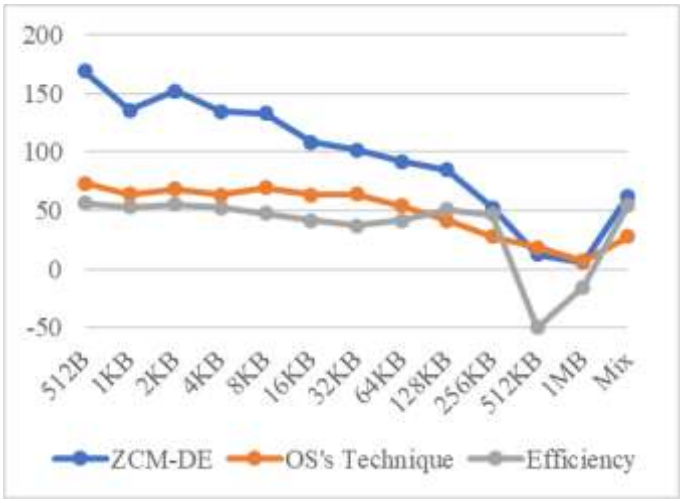

Fig. 2 Efficiency of data transfer rate compared with the operating system

B. Algorithm of The Proposed Approach

It is shown from the experiments, that the new approach is efficient up to an average of 341KB/file as shown in Fig. 3. So, the average is called THRESHOLD.

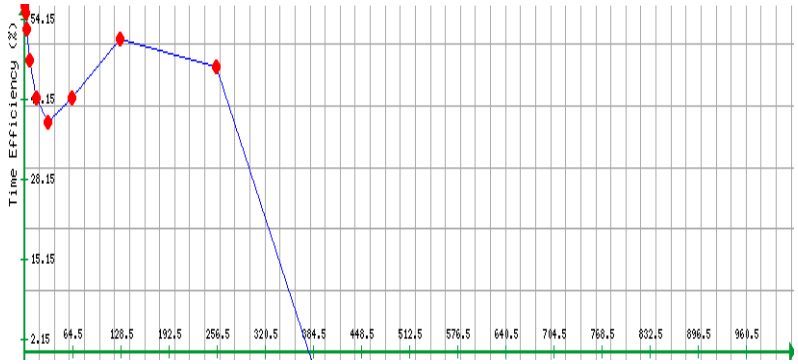

Fig. 3 Efficiency change according to file size

So, to perform the new proposed approach, supposing pure merge without compression, the algorithm is used as shown in Fig. 4.

For example, if there are 4000 files to be transferred, and their total size is 25MB, then:

RESULT = 25MB ÷ 4000 files = 6.4KB/file

1. **Begin**
2. Count a number of files in a source.
3. Calculate the total size of files.
4. Divide the total size by the number of files.
5. **if** (division RESULT ≤ THRESHOLD) **then**
6. Merge the files into one file in the source.
7. Extract the merge file directly to the destination.
8. Delete the merge file.
9. **else** apply Window 10's OS approach.
10. **end if**
11. **end**

Fig. 4 ZCM-DE algorithm

Therefore, RESULT is less than THRESHOLD, so ZCM-DE approach is applied. If there are files to be transferred locally from a source to a destination, and RESULT is more than THRESHOLD, Windows 10's OS approach is applied.

C. Coding The Algorithm

To make the new proposed approach to be as a program, the algorithm is coded as shown in Fig. 5.

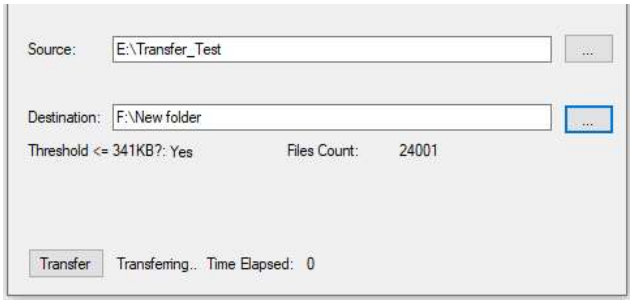

Fig. 5 Screenshot of ZCM-DE program

The algorithm code is evaluated as well. Three small size file folders are created, containing 8,000 files, 16,000 files and 24,000 files respectively. All three folders are tested, and the results are efficient compared with Windows 10's OS.

D. Comparison of Results With Previous Works

Previous approaches have presented solutions for distributed environment systems. They have introduced improvements to challenge of the problem of poor software of an operating system when transferring many small files. However, current approaches have no improvements to this problem "locally" in term of "data transfer rate". Nevertheless, our approach is compared with some previous closer approaches.

Using merging approach in buffer, Ahn et al. [6] proposed an approach called CW-FFS. It enhanced the small file write of a server-based environment by modifying Fast File System FFS. Its aim is a file size of 768B or less, and the average efficiency is 33%.

Our new proposed approach is introduced for enhancing Windows 10's OS small data transfer rate locally from a source to a destination without a file system modification. Also, the new proposed approach enhances the data transfer rate of more file size up to 341KB by average efficiency of 48.25%.

Tao et al. [8] proposed a new approach called LHF. It is for improvement of small files for distribution-based Hadoop File System HDFS. The approach proposed a merging technique with two steps, the first step is merging small files into some part files and the second step is that meta-information is built into the index files to improve file access only. Also, the approach supports appending additional files to a current archive. Again, our new approach is not for a distributed system and not only for improving file access, it is specifically for data transfer rate improvement of Windows 10's OS locally from a source to a destination.

Fig. 6 shows a comparison of the three approaches in term of the data transfer time. Using the new proposed approach, the data transfer time is less than CW-FFS and LHF approaches.

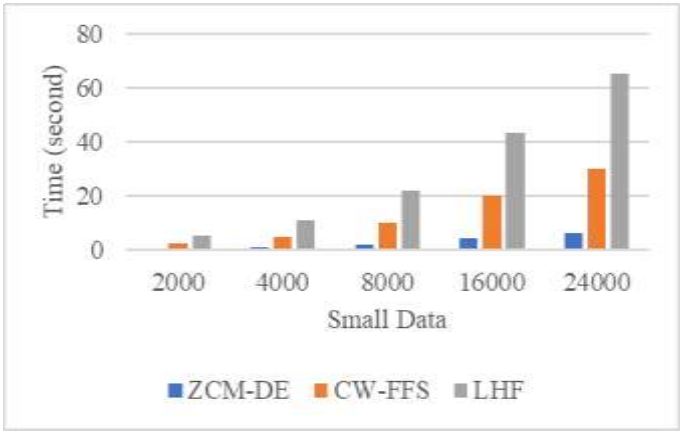

Fig. 6 Comparison of data transfer time

VI. CONCLUSIONS

When transferring a file locally from a source to a destination, the operating systems such as Windows 10's OS executes many I/O's operations in a very short time. So, big files data transfer rate is fine. When many small files are transferred, this leads to two negative effects on the operating system; first, very low data transfer rate and second, high resources consumption.

To enhanced this problem, a new approach is proposed by using two principles; zero-compression merge in the source and direct extract to the destination. There are solutions, but all of them are for

distributed or server-based systems and complex. Our approach is to enhance the performance of "local data transfer rate" from a source to a destination of an operating system such as Windows 10's OS. The new approach has been tested, evaluated and compared with other approaches. Simplicity is an advantage of our approach. It is efficient in terms of time and resources consume.

REFERENCES

[1] "Monthly mobile PC data traffic worldwide 2015-2022," *Statista*. https://www.statista.com/statistics/739014/worldwide-monthly-traffic-mobile-pc/ (accessed Apr. 05, 2020).

[2] "Desktop Windows Version Market Share Worldwide," *StatCounter Global Stats*. https://gs.statcounter.com/os-version-market-share/windows/desktop/worldwide (accessed Apr. 05, 2020).

[3] Z. Wang, K. Chen, Y. Wu, and W. Zheng, "SepStore: Data Storage Accelerator for Distributed File Systems by Separating Small Files from Large Files," in *Internet of Vehicles – Technologies and Services*, vol. 8662, R. C.-H. Hsu and S. Wang, Eds. Cham: Springer International Publishing, 2014, pp. 272–281.

[4] W. H. Ahn, K. Lee, J. Oh, K. Min, and J. S. Hong, "A multiple-file write scheme for improving write performance of small files in Fast File System," *Information Processing Letters*, vol. 109, no. 18, pp. 1021–1026, Aug. 2009, doi: 10.1016/j.ipl.2009.05.010.

[5] P. Lensing, D. Meister, and A. Brinkmann, "hashFS: Applying Hashing to Optimize File Systems for Small File Reads," in *2010 International Workshop on Storage Network Architecture and Parallel I/Os*, Incline Village, NV, USA, May 2010, pp. 33–42, doi: 10.1109/SNAPI.2010.12.

[6] B. Dong, Q. Zheng, F. Tian, K.-M. Chao, R. Ma, and R. Anane, "An optimized approach for storing and accessing small files on cloud storage," *Journal of Network and Computer Applications*, vol. 35, no. 6, pp. 1847–1862, Nov. 2012, doi: 10.1016/j.jnca.2012.07.009.

[7] P. Matri, M. S. Perez, A. Costan, and G. Antoniu, "TýrFS: Increasing Small Files Access Performance with Dynamic Metadata Replication," in *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, Washington, DC, USA, May 2018, pp. 452–461, doi: 10.1109/CCGRID.2018.00072.

[8] L. Xiong, Y. Zhong, X. Liu, and L. Yang, "A Small File Merging Strategy for Spatiotemporal Data in Smart Health," *IEEE Access*, vol. 7, pp. 14799–14806, 2019, doi: 10.1109/ACCESS.2019.2893882.

[9] "Why does copying multiple files take longer time than copying a single file of same size? - Quora." https://www.quora.com/Why-does-copying-multiple-files-take-longer-time-than-copying-a-single-file-of-same-size# (accessed Jun. 21, 2020).

[10] "Operating System Tutorial: What is, Introduction, Features & Types." https://www.guru99.com/operating-system-tutorial.html (accessed Jun. 08, 2020).

[11] A. Silberschatz, P. B. Galvin, and G. Gagne, "Operating System Concepts," p. 1278.

[12] tedhudek, "Example I/O Request - An Overview - Windows drivers." https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/example-i-o-request---an-overview (accessed Jun. 08, 2020).

[13] W. Tao, Y. Zhai, and J. Tchaye-Kondi, "LHF: A New Archive Based Approach to Accelerate Massive Small Files Access Performance in HDFS," in *2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, Newark, CA, USA, Apr. 2019, pp. 40–48, doi: 10.1109/BigDataService.2019.00012.