Optimization of Large Scale of Files Transfer in Meteorological Grid

Tinghuai Ma[1,2], Hao Cao[2], Jin Wang[2], Wei Tian[2] and Keddy Wornyo Dickson[2]

[1]*Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science and Technology, China*
[2]*School of Computer and Software, Nanjing University of Information Science and Technology, China*
thma@nuist.edu.cn

Abstract

This paper focuses on performance enhancement of large scale of small files transfer, which is critical to the performance of meteorological grid. GridFTP and compression techniques are used to optimize the efficiency. The transfer parameters are configured before transmission, such as extended block mode (Mode E), TCP buffer size. Compression is used to compress files into single file, in order to eliminate negotiating time before transmission. During the course of our research, we sacrifice the disk space for avoiding the vicious recurrence of compression-decompression-compression. In the end, the experiments show the advantages of our approach.

Keywords: *Small Files, transfer performance, GridFTP, Compression, Meteorological Grid*

1. Introduction

Recent developments in the area of grid computing, especially in the fields of meteorology, have focused on the need to efficiently transfer large scale of small files. In the area of meteorology, it generates about 1TB data every day, and there are large amounts of small files in it [9]. So, how to transfer these small files effectively comes to be an issue that needs to be solved.

In meteorological grid, the data should be transferred in real time. Thus, services that the grid offers can be made most of, such as data management, resource management and products transfer [16]. In a word, the transmission time should be as little as possible. So, we not only implement the large scale of small files transfer, but also optimize it to reduce the transmission time.

The most known protocol for transfer files in wide area networks is GridFTP, which has a good performance in large file transfer, but it is not for large amounts of small files transfer [1]. In order to address this issue, we can utilize compression technique and augment the transmission bandwidth.

Compression is one of techniques that improve the performance of large scale of small files transfer. Compression can be used offline or on-the-fly (as the data is generated) [7]. It reduces transmission time by eliminating much of the redundancy that is characteristic in most small files. But we should also take its disadvantage into account. Compression is based on CPU and memory usage. This method trades off between bandwidth and CPU, memory.

Mode E [15] is another way we used to increase the amount of bandwidth available to the situation that transfers plenty of compressed files. Mode E is a critical GridFTP component because it allows for out of order reception of data. It means we can send the data down

multiple paths and do not need to worry if one of the paths is slower than the others and the data arrives out of order [2]. To make most of GridFTP, TCP buffer size should be set to bandwidth-delay products (BDP) which is the optimal value [5].

In this paper, we utilize the techniques mentioned above to optimize the large scale of small files in meteorological grid. In the end, we conduct several experiments to show the advantage of our approach.

In the following section, we survey the related works. In Section 3, we present our optimizing ways of transfer: network transfer protocol in Section 3.1, compression technique in Section 3.2. The conclusion will be made in Section 4.

2. Related Works

The literatures [6, 9 and 14] implemented operation environment for meteorological grid. Within implementation of these architectures, they took transmission service as the core part, and tried best to handle large scale of files efficiently, and made the data management service more usability and stability.

In order to improve the performance of files transfer, especially large scale of small files transfer, many works have been done in the last few years [4]. Most of researchers optimize the transfer protocol and the file itself to augment the performance [15]. Even though GridFTP has been proposed as a protocol to effectively transfer large scale of data in grid computing, there are also some optimization can be done. And GridFTP provides the extended interface for developers [2]. There have been several studies which modify the GridFTP control parameters to improve the transmission performance. The literature [4] proposed an automatic parameter configuration mechanism for GridFTP, which optimized the number of parallel TCP connections by utilizing measurement results of network status, such as the goodput and the round-trip time of GridFTP data channels. In [5], they described the integration of dynamic right-sizing—an automatic and scalable buffer management technique for enhancing TCP performance—into GridFTP. Ohsaki, *et al.*, investigated the optimal parameter configuration of GridFTP in terms of the number of TCP connections and the TCP socket buffer size [3].

There are also other works to optimize the performance of GridFTP. Patrick McClory etc. described MNEMONIC, an end-to-end cyber infrastructure system designed to improve scientists' productivity by automatically and transparently optimizing and tuning data movement over the network [18]. And they showed that MNEMONIC+GridFTP outperformed the TCP+GridFTP implementation by over a factor of 10x, while remaining easy to install and operate. In [19], they optimized the Data Storage Interface (DSI), which specified how to read and write to the storage system. They designed a new DSI, based on MAPFS, a parallel file system.

To optimize the regular file itself, there are also many works have been done. Krintz and Sucu presented a system called the Adaptive Compression Environment (ACE) [7]. It applied compression to a communication stream automatically and transparently. In [8], they proposed AdOC (Adaptive Online Data Compression) Algorithm, which is a balance between compression time and propagation time. Its scheme is to design a chunk buffer to put files waiting for compression and a FIFO queue to place files waiting for transmission. If the number of files in FIFO queue is larger than threshold value, then enhance compression level to increase compression time, so that the number of files waiting for transmission can be reduced.

3. Our Approach

Without modification of the protocol itself, speeding up the transfer of small files can be done by adjusting factors in overall transfer process. We can compress these files into single file, and then transmit it. Then, we can use the advantage of GridFTP to improve the transmission performance of this single file.

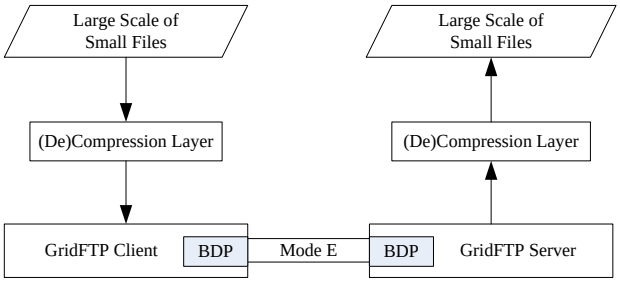

Figure 1. The Architecture of our Prioritization Scheme

As shown in Figure 1, before the large scale of small files transfer, in compression layer, we compress all files separately using LZO (Lempel-Ziv-Oberhumer) compression algorithm. In transport layer, we first configure the network parameters, such as TCP buffer size and mode E which is a critical GridFTP component because it allows for out of order reception of data.

3.1. Network Transfer Protocol

3.1.1. GridFTP. In our approach, we use the GridFTP as file transfer protocol. GridFTP is a high-performance, secure, reliable data transfer protocol optimized for high-bandwidth wide-area networks [2]. It is based on FTP, the highly-popular Internet file transfer protocol. And it offers several services that suit the grid environment extremely. We utilize some of them in our approach, such as channel caching, optimal TCP buffer size and mode E.

The network environment varies frequently. It means the bandwidth and round trip time (RTT) is very unstable. Unfortunately, both of them affect file transfer obviously. So, in order to get high performance, we have to optimize the network parameters. As shown in Figure 1, we set the TCP buffer size according to bandwidth-delay products (BDP). For any given connection, the optimal TCP buffer size is equal to the BDP of the connection [5]. GridFTP supports several tools to set TCP buffer size, so how to calculate BDP comes to be a matter.

A simple equation is:

$$\text{Buffer size (KB)} = \text{Bandwidth (Mbs)} * \text{RTT (ms)} / 8 \quad (1)$$

In this equation, KB and MB are based on 1000 rather than 1024. According to the expression, BDP is related to bandwidth and RTT, so the solution is used to assess the value of bandwidth and RTT. For determining these values, we can use diagnostic tools, such as iperf, nettimer and nettest [5]. Thus, we will manually tune the buffer sizes to keep the network pipe full [3].

3.1.2. Experimental Setup. Our experimental environment consists of three identical machines connected via 1000-Mbps Ethernet. Each machine contains dual 2.66-GHz

Intel(R) Core(TM) processors with 2-GB RAM and a 1000-Mbps on-board Broadcom NetLink(TM) Network Interface Card (NIC). We use Fedora 13 as the operating system on every machine, and establish several accounts for GridFTP users, include globus, sev53, and sev158. Globus Toolkit is installed on each of them, and GridFTP component is configured properly. We also set up Grid Security Infrastructure (GSI). And a central concept in GSI authentication is the certificate. Before data transfer, every user and service on the Grid is identified via a certificate, which contains information vital to identifying and authenticating the user or service. Thus, the commands of GridFTP can be used to test the transmission performance.

For testing how to tune TCP buffer size, we utilize "–tcp-bs" command. We transfer files between the client and server machines in mode E, and the value of TCP buffer size and the mode E are specified as an option to the globus-url-copy program [2]. During all of our experiments, we get the average results which conduct 10 times.

3.1.3. Deciding the TCP Buffer Size. In our grid environment, we conduct two kinds of tests from one host named sev53 to another named globus at each of the following TCP buffer sizes 1KB, 2KB, 4KB, 8KB, 16KB, 32KB, 64KB, 128KB, 256KB, and 1MB. One is single file transfer and another is many-file transfer. And the total size of the single file and many-file both are 50MB. In this experiment, we keep the other aspects to be same, such as the number of parallel data streams. Each time for transfer is shown as Figure 2.

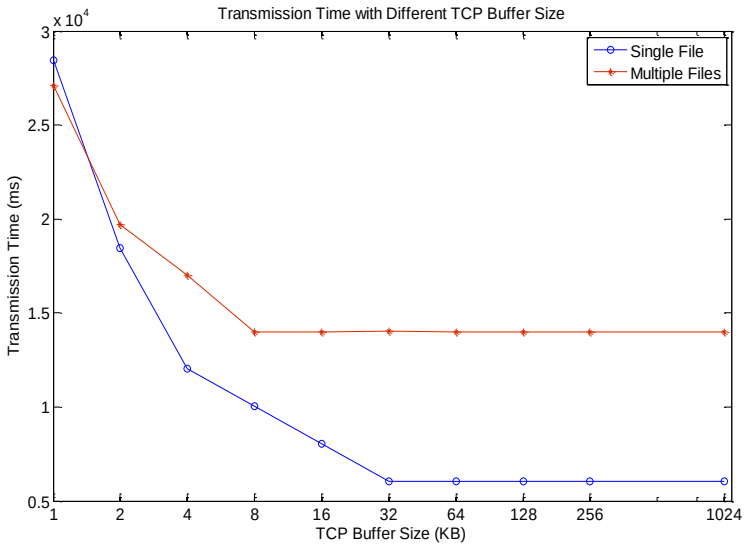

Figure 2. Transmission Time VS TCP Buffer Size

As shown in Figure 2, single file transfer and multiple-file transfer have a similar trend. It indicates that there is an optimal value of TCP buffer to the certain circumstance. So, it is critical that the TCP buffer should be set properly before data transfer.

TCP buffer consumes the CPU and memory. But the CPU and memory are the critical resource in our approach. So, there is no need to assign a high value for the TCP buffer. But, transmission performance will be good with large TCP buffer size. In our circumstance, it's enough to let TCP buffer size be 128KB.

3.1.4. MODE E. As presented above, we transfer files using GridFTP in mode E. In comparison with mode S, mode E is faster slightly and allows for out of order reception of data. So, it can greatly reduce the time for transmission when we transfer a certain amount of files. In this experiment, we test the performance of mode E. We prepare seven groups of files that each group contains 1000 files but the files of each group have different size (see Table 1).

Table 1. Test Number and the File Size of the Tests

Test No.	Each File(KB)	Total size
1	1	1000K
2	10	10M
3	50	50M
4	100	100M
5	250	250M
6	500	500M
7	1000	1000M

As shown in Figure 3, we can obtain the maximum benefit of mode E. In meteorological grid, the files belong to different fields that represent distinct meaning. So, we may compress them respectively, and then it generates lots of compressed files. To those files, we can reduce the time for transmission in mode E.

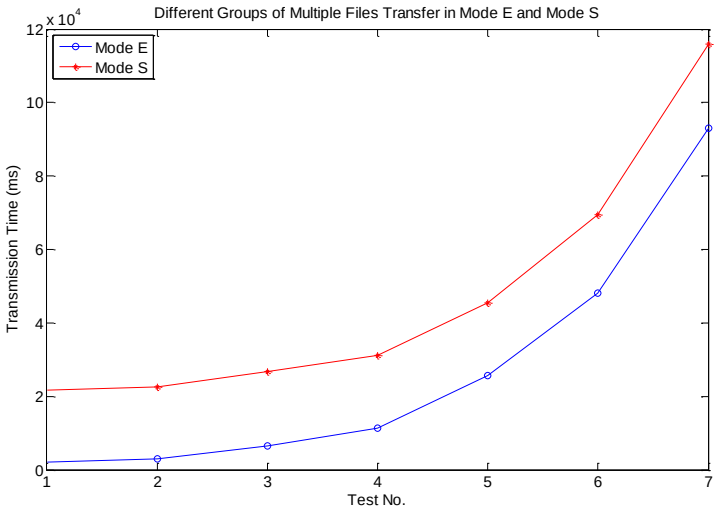

Figure 3. Multiple Files Transfer (1000 files)

3.2. Applying Compression Technique

As shown in Figure 1, we insert a compression layer before transmission and a decompression layer after received the single file.

Let T be the time that some process costs. Then, Let T(ot) be the time of original files transfer, T(c) be the time to compress original files, T(ct) be the time of compressed file transfer, and T(dec) be the time to decompress the file received. The primary goal of our approach is to design a scheme to save the transmission time by

compressing data before transfer. So, we let compression scheme satisfy the criterion. We decide which compression algorithm to use next.

$$T(ot) > T(c) + T(ct) + T(dec) \quad (2)$$

First of all, we discuss some properties and background on compression that will help improve the performance of large amounts of small files transfer. Compression is useful because it helps reduce the consumption of expensive resources, such as hard disk space or transmission bandwidth.

There are two kinds of compression algorithms called lossless and lossy algorithms [13]. Lossless compression algorithms usually utilize statistical redundancy to represent the sender's data more concisely without error. It is required for text and data files, such as binary files, text articles, and bank records. By contrast, lossy data compression compresses data by discarding some of it. It is most commonly used to compress multimedia data (audio, video, *etc*.), especially in applications such as streaming media and internet telephony.

In our situation, in order to keep the meteorological data precisely, we use lossless compression algorithms. Here we list three popular algorithms, and we will choose the most suitable one to apply.

- *GZip* [10] — A compression utility designed to be a replacement for compress. Its main advantages over compress are much better compression and freedom from patented algorithms. But, it cannot be divided into chunks and parallel processing.
- *BZip2* [11] — BZip2 compresses data in blocks and more effectively than the older LZW (.Z) and Deflate (.zip and .gz) compression algorithms, but is considerably slower. The program itself has no facilities for multiple files, encryption or archive-splitting.
- *LZO* [12] — A compression algorithm offers pretty fast compression and extremely fast decompression. It includes slower compression levels achieving a quite competitive compression ratio while still decompressing at this very high speed. And it supports blocking and parallel processing.

In our experiments, we will utilize the GridFTP to transfer files concurrently. And we must reduce the time for compression, transfer and decompression. So, we choose LZO to help improve the performance.

3.3. The Vicious Recurrence of Compression-Decompression-Compression

We found another issue during the course of our research. Just imagine, we compress files into single file before transfer, and then the terminal decompresses it. And now, if some node wants to download the same files from that terminal, it has to compress them again. That plunges the vicious recurrence of compression-decompression-compression.

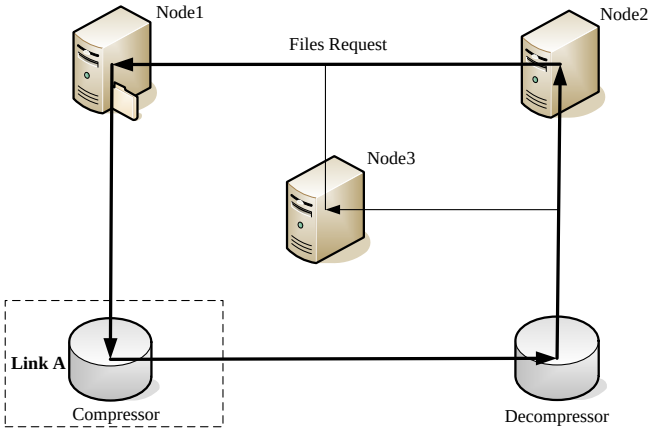

Figure 4. The Vicious Recurrence of Compression-Decompression-Compression

As shown in Figure 4, the bolder line is original circulation. The files node2 requests pass through compressor, network and decompressor. And now, node3 requests the same files. As usual, these files have to pass the same route. Then, problem has cropped up. It plunges the vicious recurrence of compression-decompression-compression.

We address this issue on Link A (see Figure 4). Nowadays, the price of hard disk is lower. We can deploy a certain amount of storage resources to save the compressed files. This in turn, means we compress files at the first time and do not delete the compressed files after successful transfer but save them on hard disk. For convenience to search, we put compressed files in parent directory of original files, and rename them meaningfully. As presented in Figure 4, when node3 requests files on node1, node1 firstly lookups corresponding compressed files. If exist, then transmit directly. This approach trades disk space for reducing transmission time.

3.4. Experiment Result

In this experiment, we apply all the methods mentioned above. In this case, to calculate total time, we add compression time, transmission time, and decompression time. The experimental data is shown in Table 2. As shown in Figure 5, test1 presents an obvious result. With our approach, transmission time is reduced greatly. And if the vicious recurrence of compression-decompression-compression is addressed, further promotion will be obtained. Of course, it will cost certain disk space.

Table 2. Data Source for Experiments

Test No.	Each File Size （K）	Number of Files
1	1	10000
2	10	1000
3	100	100

We also realize that our scheme doesn’t work well with lower amount but greater size of files. But to meteorological grid, this circumstance which is approximate to test3 almost won’t appear.

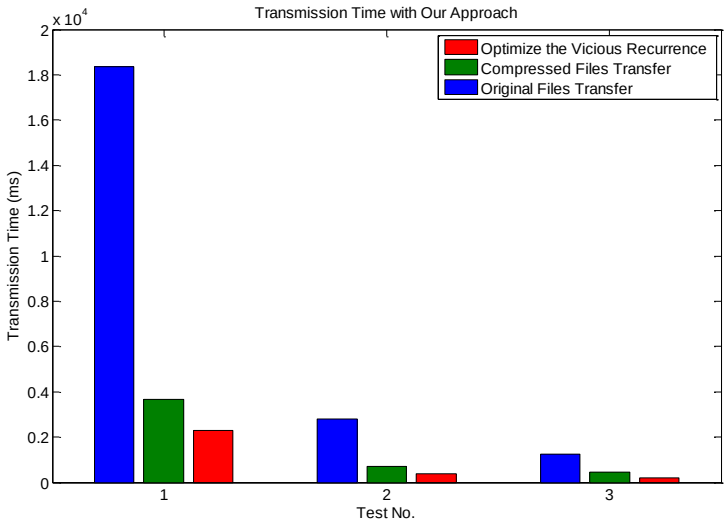

Figure 5. Compressed VS Uncompressed

4. Conclusion and Future Topics

In order to improve performance of very large amounts of small file in meteorological grid, we utilize compression techniques and GridFTP toolkit. According to the characteristic of meteorological data and GridFTP, we analyze several compression algorithms, and choose LZO at last. To make the most of GridFTP, we adjust TCP buffer size with BDP which is optimal value and transfer files in mode E. The experimental results show the advantage of our approach. It can reduce transmission time. We also overcome the vicious recurrence of compression-decompression-compression by trading disk space for lower transmission time. It improves the transmission performance further.

As mentioned in this paper, we will utilize multi-stream transfer in our next research. And further promotion will be obtained by the moment. Meanwhile, in this paper, we speed up the transfer of small files only by reducing the impact the factors have on the overall transfer execution. We can also optimize the protocol itself. GridFTP support UDP-based Data Transfer Protocol (UDT) by specifying as an option to the globus-url-copy program [17]. So, UDT is a possible change we will make in the future.

Acknowledgements

This work was supported in part by the National Science Foundation (61173143) and the Special Public Sector Research Program of China (GYHY201206030), and was also supported by A Project Funded by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

[1] W. Allcock, "GridFTP: Protocol Extensions to FTP for the Grid", In Global Grid Forum, **(2003)**.
[2] GridFTP, **(2011)**, http://www.globus.org/datagrid/gridftp.html.
[3] T. Ito, H. Ohsaki and M. Imase, "On parameter tuning of data transfer protocol GridFTP in wide-area Grid computing", in Proc. 2th International Workshop on Networks for Grid Applications (GridNets 2005), **(2005)**, pp. 415-421.

[4] I. Takeshi, O. Hiroyuki and I. Makoto, "Automatic Parameter Configuration Mechanism for Data Transfer Protocol GridFTP", in Proc. 2006 International Symposium on Applications and the Internet (SAINT'06), **(2006)**, pp. 32-38.
[5] S. Thulasidasan, W. Feng and M. K. Gardner, "Optimizing GridFTP through dynamic right-sizing", in Proc. IEEE International Symposium on High Performance Distributed Computing, **(2003)**, pp. 14-23.
[6] T. H. Ma, J. Ge, H. Cao and Y. L. Wang, "Design and Implementation of Virtual Resources Management in Meteorology Grid", in Proc. 9th International Conference on Grid and Cooperative Computing (GCC), (**2010**), pp. 58-63.
[7] C. Krintz and S. Sucu, "Adaptive on-the-fly compression", IEEE Trans. Parallel Distrib. Syst., vol. 17, no. 1, **(2006)**, pp. 15-24**.**
[8] E. Jeannot, B. Knutsson and M. Björkman, "Adaptive Online Data Compression", in Proc. 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11 2002), (**2002**), pp. 379-388.
[9] D. Q. Li, Y. Wang, A. Y. Xiong and T. H. Ma, "High Performance Computing Model for Processing Meteorological Data in Cluster System", Journal of Convergence Information Technology, vol. 4, no. 6, **(2011)**, pp. 92-98.
[10] Gzip homepage, **(2011)**, http://www.gzip.org.
[11] BZIP2 Compression, **(2011)**, http://en.wikipedia.org/wiki/Bzip2.
[12] Lempel-Ziv-Oberhumer (**LZO**) Compression, **(2011)**, http://www.oberhumer.com/opensource/lzo.
[13] Data compression, **(2011)**, http://en.wikipedia.org/wiki/Data_compression.
[14] I. Rao, E. Huh and S. Lim, "An Adaptive and Efficient Design and Implementation for Meteorology Data Grid Using Grid Technology", in Proc. 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06), **(2006)**, pp. 239-246.
[15] J. Bresnahan, M. Link, R. Kettimuthu, D. Fraser and I. Foster, "GridFTP Pipelining", Teragrid Conference, **(2007)**, pp. 1-6.
[16] W. Zhao, M. H. Li, J. X. Liu, Q. Zhang, Y. M. Tang, H. Y. Wang and T. H. Ma, "Construction of national meteorological computational Grid", in Proc. 2th International Conference on Information Science and Engineering (ICISE2010), **(2010)**, pp. 1-3.
[17] Y. Gu and R. Grossman, "UDT: UDP-based data transfer for high-speed wide area networks", Computer Networks, 7(51), **(2007)**, pp. 1777-1799.
[18] M. Patrick, K. Ezra, S. Martin and T. Michela, "MNEMONIC: A Network Environment for Automatic Optimization and Tuning of Data Movement over Advanced Networks", in Proc. 18th International Conference on Computer Communications and Networks, **(2009)**, pp. 1-7.
[19] A. Sánchez, M. S. Pérez, P. Gueant, M. Jesús and H. Pilar, "A parallel data storage interface to gridftp", LNCS, vol. 4276, **(2006)**, pp. 1203-1212.
[20] H. Wang, F. Wang, Z. Guo and K. Li, "A Flexible Bandwidth Allocation Scheme Using Specified Duration Based Data Acquisition in Heterogeneous Networks", International Journal of Advanced Science and Technology, vol. 6, SERSC, **(2009)**, pp. 25-42.
[21] M. M. A. Ghazala, M. F. Zaghloul and M. Zahra, "Performance Evaluation of Multimedia Streams Over Wireless Computer Networks (WLANs)", International Journal of Advanced Science and Technology, vol. 13, SERSC, **(2009)**, pp. 61-74.
[22] F. Y. Sattarova, T. A. Furkhat and D. Bhattacharyya, "Network Productivity and Quality Indicators of Broadband Connection Channels: a Review", International Journal of Hybrid Information Technology, vol. 2, no. 4, **(2009)** October, pp. 39-51.