Efficient Resource Management Scheme of TCP Buffer Tuned Parallel Stream to Optimize System Performance*

Kun Myon Choi[1], Eui-Nam Huh[2], and Hyunseung Choo[1,**]

[1] School of Information and Communication Engineering,
Sungkyunkwan University, Korea
{dilly97, choo}@ece.skku.ac.kr
[2] College of Electronics and Information, Kyung Hee University, Korea
johnhuh@khu.ac.kr

Abstract. GridFTP is a high-performance, secure and reliable parallel data transfer protocol, used for transferring widely distributed data. Currently it allows users to configure the number of parallel streams and socket buffer size. However, the tuning procedure for its optimal combinations is a time consuming task. The socket handlers and buffers are important system resources and must therefore be carefully managed. In this paper, an efficient resource management scheme which predicts optimal combinations based on a simple regression equation is proposed. In addition, the equation is verified by comparing measured and predicted values and we apply the equation to an actual experiment on the KOREN. The result demonstrates that the equation predicts excellently with only 8% error boundary. This approach eliminates the time wasted tuning procedure. These results can be utilized directly and widely for the fast decision in typical applications such as GridFTP.

1 Introduction

According to the recent research and development on global climates, high energy physics, and computational genomics, Grid technology has advanced as a promising method for widely distributed high speed data transfer. The Grid Alliance Team developed GridFTP especially to support safe and efficient transfer of distributed data. In addition, this tool uses a parallel data transfer mechanism to enable application level high-speed data transfer. Currently, GridFTP allows users to configure the number of parallel sockets and socket buffer size. In other words, the decision, for selecting the number of parallel sockets and buffer size for maximizing the network availability, should be made by a user [1, 2].

So far, studies regarding high performance data transfer. TCP buffer tuning [3, 4, 5] and parallel data transferring [6, 7] have been executed separately. In a Gigabit scale network, the separate use of these transferring mechanisms can cause each

* This work was supported in parts by Brain Korea 21 and the Ministry of Information and Communication in Republic of Korea.

** Corresponding author.

T. Enokido et al. (Eds.): EUC Workshops 2005, LNCS 3823, pp. 683–692, 2005.

end system overhead, due to its large buffer size or overwhelming parallel streams. To overcome this problem, using parallel streams simultaneously with TCP buffer tuning, can be a good solution. This combination balances the number of parallel streams and buffer size, reducing these to a reasonably low level.

The socket buffer size and number of socket handlers are directly related to the memory limits and maximum number of concurrent users, respectively. Therefore, such system resources need to be carefully and efficiently managed. However, balancing procedures require background knowledge in networks, and is obviously a time-consuming task. Actually, it often takes more than 20 minutes. In addition, only a few studies have delved into the characteristics of parallel data transfer. In particular, the subject of buffer tuned parallel sockets has never been examined in detail.

Therefore, in this paper, the characteristics of buffer tuning and parallel data transfer are discussed by analyzing the results of various experiments. After, a simple relational equation, representing the relationship between the optimal number of parallel streams and buffer size, is developed. The system validity is also verified on the KOrea advanded REsearch Network (KOREN) testbed that supports Giga-bit bandwidth, by applying the equation to actual data transfer.

The remaining sections of this paper are organized as follows. Section 2 explains the concept of manual tuning, automatic tuning, parallel data transfer mechanism, and multiple regression. Section 3 develops a simple relational equation based on the analysis of various experiments. Next we verify the accuracy by applying the equation to an actual data transfer demonstration on the KOREN. In Section 5, this paper is concluded, and future work is discussed.

2 Related Works

Current main streams high performance data transfer mechanisms are divided into two categories; the first is a parallel data transfer mechanism [6, 7] to avoid TCP buffer limit by using multiple sockets, and the second is TCP buffer tuning, which adjusts the socket buffer to an appropriate size for each connection. These TCP buffer tuning can also be categorized into manual turning [3, 4] and automatic tuning [5]. The former configures the optimized buffer size referred to

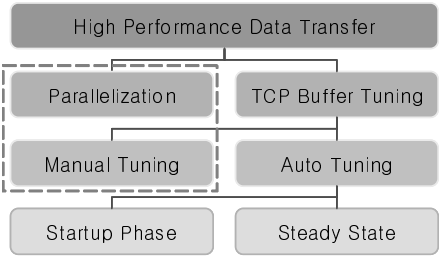

Fig. 1. Hierarchy of high performance transfer mechanisms

the network estimation tool by a network administrator, and the latter tunes up without the network administrator, through tuning daemon tool or the tuning algorithm in TCP stack. Automatic tuning techniques are also classified into startup phase tuning [8] which configures the buffer size at connection setup time, and the steady state tuning [5, 9, 10] which continuously adjusts buffer size during the life time. This paper focuses on both the parallel transfer mechanism and TCP buffer tuning represented by the dotted line in Figure 1.

Manual Socket Buffer Tuning: The throughput of FTP on links having both High Bandwidth and High Delay (HBHD) can be improved by "tuning" operating system parameters, using the TCP networking stack on the file transfer endpoints. This tuning usually involves increasing memory buffer sizes to fulfill the large data window's request in high bandwidth. TCP uses the concept of "windows" to throttle the amount of data that the sender injects into network, depending on the receiver's available capacity and the other link traffic. A window is the amount of new unacknowledged data that the TCP allows to be in flight between sender and receiver. In order to continuously full the link capacity, the TCP must maintain the window size into the Round Trip Latency (RTT) multiplied by the bandwidth, therefore, called the Bandwidth-Delay Product (BDP). Generally, in order to achieve maximum throughput, TCP requires a receive buffer equal to or greater than the BDP, in addition, TCP often requires a sender side socket buffer of 2*BDP to recover from errors [3, 4].

Automatic Socket Buffer Tuning: Linux autotuning refers to a memory management technique used in the Linux kernel, (stable version 2.4). The central difference between Linux autotuning and other automatic tuning techniques is that autotuning does not measure the BDP, therefore it does not consider the condition of the network. Linux autotuning increases window size for TCP connections when the socket buffer is full. Therefore the performance enhancements represent a side effect of increased socket buffer size. The range of memory, which can be allocated to the socket buffer, is limited by the proc files tcp_wmem, tcp_rmem.

Parallel Data Transmission: Parallel data transmission establishes multiple connections; data is split and transferred simultaneously through each connection. This technique operates at application level. GridFTP uses this technique, because it can complement the shortcomings of TCP buffer tuning, and simultaneously can achieve maximum throughput on high performance-long distance network environments [2, 6, 7]. Parallel data transferring, unlike buffer tuning, can get maximum achievable throughput even if RTT is short by using a sufficient number of streams. For this reason, it is a good complementary scheme for the buffer tuning. To identify the strength and weakness of parallel data transferring, well-known facts regarding parallel streams are summarized below.

- Maximum throughput can be achieved without the administrator's help.
- Parallel streams are stronger than the buffer tuned stream against packet loss.
- Parallel data transferring is effective regardless of RTT.
- Increasing the number of parallel streams reduces traffic interference.

Multiple Regression: The purpose of multiple regression is to identify the relationship between several independent variables and a dependent variable, and to predict dependent variable by using two or more independent variables. The multiple regression equation takes the form $y = b_1x_1 + b_2x_2 + ... + b_nx_n + c$. The b_i's are the regression coefficients, representing the amount the dependent variable y changes when the independent changes 1 unit. The c is the constant, where the regression line intercepts the y axis, representing the amount the dependent y will be when all the independent variables are 0. Power terms can be added as independent variables to explore curvilinear effects. Cross-product terms also can be added as independent variables to explore interaction effects. Interaction effects are sometimes called moderator effects because the interacting third variable which changes the relation between two original variables is a moderator variable which moderates the original relationship.

3 The Proposed Scheme

In this section, we propose an efficient resource management scheme that predicts optimal combinations of the number of parallel streams and buffer size with considering delay variation. The main idea of this scheme is that the relationship of the number of parallel streams, buffer size, and network delay can be represented by a single regression equation. In order to derive this equation, multiple regression approach is used. The important contribution on using this equation is that it can eliminate the time wasted for whole tuning procedure. In addition, this equation gives important information which is used for balancing the number of sockets and buffer size to tolerable level.

3.1 Modeling of Regression Equation

In this subsection we first set up some notations which will be used throughout this paper. A certain buffer size exists, at which, increasing the buffer size has no effect on throughput. In addition, a certain number of parallel streams exist, at which, increasing the number of sockets does not increase throughput. In addition, there is tradeoff between the optimal buffer size and optimal number of parallel sockets. Therefore, It can be said that an optimal relationship exists for achieving maximum throughput without waste of memory or socket handler. For convenience, the buffer size and the number of parallel streams in its optimal relationship are denoted as B_o (Optimal Buffer size) and P_o (Optimal number of Parallel sockets), respectively.

In the case of buffer tuning, throughput increases proportional to the buffer size. However, the throughput is may not be exactly proportional to the buffer size. Therefore, a power term C_2 is added as independent variable to Eequation 1 below for representing curvature. In the case of parallel transfer, the i^{th} individual socket throughput is denoted by T_i and the aggregate throughput is denoted by T. The T is exactly proportional to the T_i when T is less than the maximum available throughput, and there are P_o number of buffer tuned

sockets. Therefore, the aggregate throughput is represented as a cross product of B_o and P_o terms. In addition, throughput increases when BDP increase. This proportional relationship can be represented by introducing a constant C_1. The equation for the collective characteristics of P_o based on B_o and RTT can be derived as follows:

$$BW \times RTT = C_1 \times P_o \times B_o^{C_2} \tag{1}$$

$$P_o = \frac{RTT \times BW}{C_1 \times B_o^{C_2}} = C_1^{'} \times \frac{RTT}{B_o^{C_2}} \tag{2}$$

Here, the bandwidth BW is fixed to $1Gbps$, because we assume that the network environment is Giga-bit Ethernet. For this reason, $C_1^{'}$ is introduced which is representing $\frac{BW}{C_1}$. The decision of correlation coefficients differs between experiments. Therefore, average coefficients of each experiments are used, and these can be determined using heuristic multiple regression analysis of many experiments results.

3.2 Determining Correlation Coefficients Using Regression Analysis

The experimental environment is constructed using a NISTNet WAN emulator [13], with a varying RTT and packet loss rate to determine the coefficients $C_1^{'}$ and C_2 in Equation 2.

- Two 800MHz Pentium-III hosts A and B are used, and are directly connected by a 1GBps Ethernet link.
- Enable high performance options: Window Scaling, Selective Acknowledgement, Time stamps [15].
- Install the NISTNet WAN emulator on host A and impose an artificial delay.

In addition, iperf [14] is used to measure throughput. The socket buffer size can be set using the -w option. Note that if the buffer size is set to 64KB then the actual buffer size is set to 128KB. For accuracy, each experiment duration is set to 30 seconds using the -t option, and an average of 20 experiment results is used. The iperf using setsockopt(), application level system call, to set the buffer size. In this case, the buffer size is limited by the maximum system buffer size. Therefore, the maximum buffer size needs to be changed for providing enough space and buffer setting commands are summarized in Table 1.

First, the RTT is configured to 5ms by using NISTNet. In order to examine the optimal number of parallel streams, the number of parallel streams is increased until throughput is saturated. The experiment is repeated for each 32KB, 64KB, 128KB, 192KB, 256KB, 384KB, 512KB, 768KB and 1024KB socket buffer size.

Table 1. Maximum buffer setting for application level buffer tuning in Linux system

Sender buffer size setting	Receiver buffer size setting
sysctl -w net.core.wmem_max=128388607	sysctl -w net.core.rmem_max=128388607

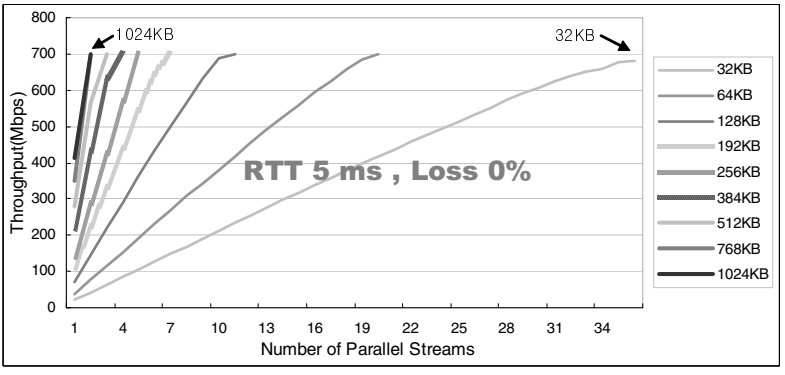

Fig. 2. Throughput for different number of parallel streams

Figure 2 demonstrates that throughput increases whenever the number of parallel sockets or buffer size increases.

Then the optimal number of parallel streams for each buffer size is determined by following procedure. As throughput reaches its maximum, a throughput gain for increasing parallel streams is reduced. In the case of a 256KB buffer size, as the number of parallel streams is increased from 1 to 4, a steady throughput gain of approximately 148Mbps, 141Mbps, 142Mbps and 131Mbps is shown. From the point of an application using 4 parallel streams, the throughput gain decreases and the throughput does not change when using more than 4 parallel streams. To verify the result, 50 more parallel streams are provided with no additional throughput gain. Finally, it can be concluded that P_o is 5 and B_o is 256KB.

The optimal number of parallel streams for each buffer size is presented in Table 2. The result shows that P_o is in inversely proportional to B_o.

In order to examine the coefficient changes when RTT changes, the same experiments are repeated on 10ms, 20ms, 30ms and 50ms RTT, and the results obtained demonstrates a similar trend in the previous result of 5ms RTT. The maximum throughput is decreased by 10Mbps for each 10ms RTT increase. P_o for the same buffer size is increased proportional to RTT.

Among the components of equation, B_o can easily be affected by packet loss. However, research networks such as the KOREN have no packet loss. Therefore, studies considering packet loss are not referred to in this paper, and are classified as future work. Figure 3 shows the relationship between B_o and P_o for each RTT from the previous experiment result.

Table 2. Optimal number of parallel streams for each buffer size

	B_o (KB)								
	32	64	128	192	256	384	512	768	1024
P_o	36	20	11	7	5	4	3	2	2

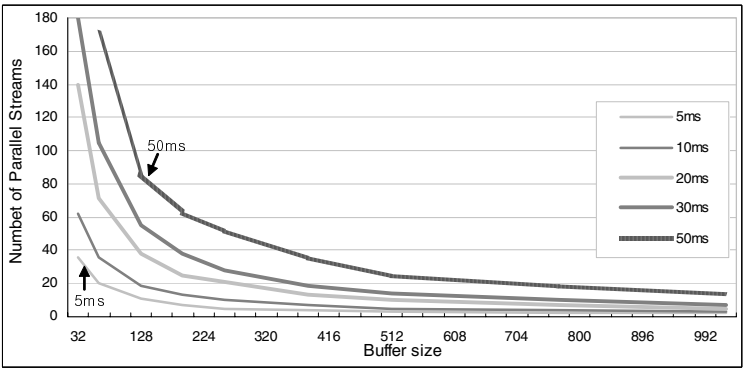

Fig. 3. Relationship between B_o and P_o on various RTT

From the previous result, five regression equations can be easily obtained with its own coefficients for each RTT of 5ms, 10ms, 20ms, 30ms and 50ms. Because there are many applications which provide single regression analysis functionality such as Excel and MATLAB. The regression equation for the i^{th} RTT is presented as the following form.

$$P_o = \frac{C_1^{''}(i)}{B_o^{C_2(i)}} = C_1^{'}(i) \times \frac{RTT}{B_o^{C_2(i)}} \tag{3}$$

Reviewing the five equations, $C_1^{''}(i)$ is proportional to RTT and $C_2(i)$ is almost stationary to RTT, with minimal irregular variance. In order to present five equations as a single equation, $C_2(i)$ must be averaged and $C_1^{''}(i)$ must be replaced by a constant multiplied by RTT. The constant can be represented as the average slope of $C_1^{''}(i)$ for varying RTT. With the additional calibration process, the representative value of $C_1^{'}$ and C_2 is obtained as 155 and 0.86 respectively.

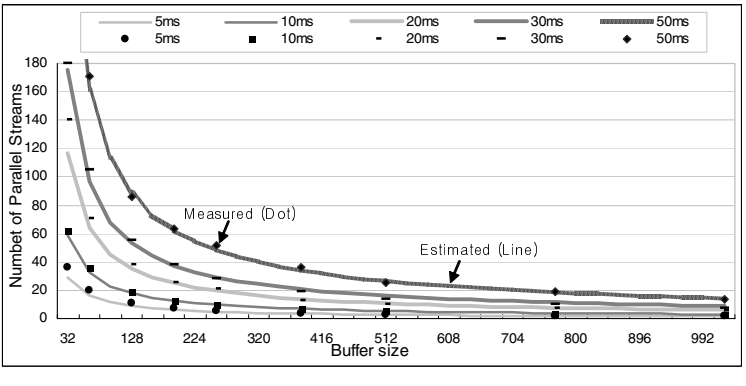

Fig. 4. Comparison between estimated and measured results

This process is called as multiple regression analysis. Finally, the equation can be written in the form of Equation 4 by replacing $C_1^{''}(i)$ with a constant and RTT. For convenience, RTT is used in ms and B_o is used in KB.

$$P_o \approx \frac{155 \times RTT}{B_o^{0.86}} \tag{4}$$

Figure 4 demonstrates the differences between measured and estimated results. The dotted graph represents the measured result and the line graph represents the estimated result. The equation measures the relationship between the optimal buffer size and the number of parallel streams with minimal 8% difference. Thus, the results of the experiment strongly support the accuracy of the proposed equation.

4 Verification of Relational Equation

The regression equation is developed based on a WAN Emulator. Therefore, verification of the equation on a real environment is required. A verification experiment is made on the KOREN, between Suwon and Daegu, which providing 1 Gbps bandwidth, and verifying the equation. The average RTT between Suwon

Table 3. The experiment environment of two organizations at a distance of 250km

	KNU Server at Daegu	SKKU Server at Suwon
CPU	AMD Opteron(tm) Processor 244 1.8GHz	Intel(R) Xeon(TM) CPU 2.40GHz
NIC	Broadcom Corp. NetXtreme BCM5701 Gigabit Ethernet	Intel Corp. 82540EM Gigabit Ethernet
OS	Linux version 2.4.20-31.9smp	Linux version 2.4.20-8smp

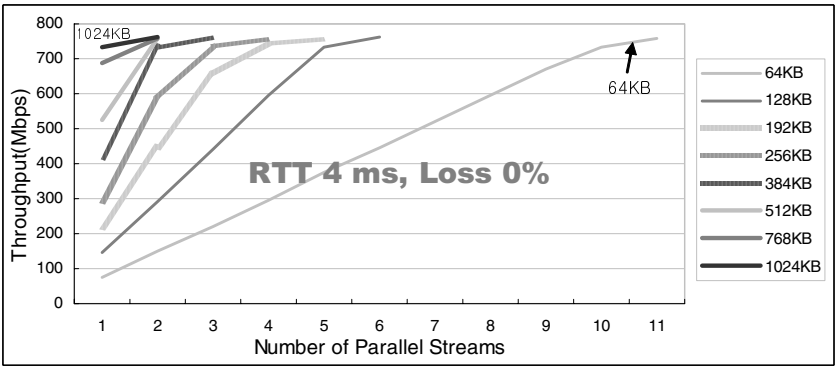

Fig. 5. Actual transfer from Suwon to Daegu

Table 4. The comparison of measured and estimated results

	B_o (KB)							
	64	128	192	256	384	512	768	1024
P_o(measured)	11	6	4	3	2	2	2	1
P_o(estimated)	11	7	5	4	3	2	2	1

and Daegu is 4ms. The server specifications used for the experiment are presented in Table 3 and the results are presented in Figure 5.

Table 4 summarizes the result of Figure 5 as a relationship between B_o and P_o, comparing the result with an estimation. Based on this comparison, it can be concluded that our equation estimates the relationship between the optimal socket buffer size and the number of parallel sockets, even if it uses different hosts with differing network performance.

This result of the optimal relationship can be utilized on a decision basis for all applications using a parallel stream such as GridFTP. When these applications establish connections, the buffer size and number of parallel connection streams can be automatically configured to the most appropriate combination without wasting time based on the equation.

5 Conclusion

A relationship based on statistical data can be identified using multiple regression analysis. Our regression based approach starts from it. Various experiments on TCP buffer tuned parallel streams with varying RTT and packet loss rate were conducted using the NISTNet WAN Emulator to obtain single regression equation which represents a relationship of the number of parallel streams, socket buffer size, and RTT. In order to verify our approach, the equation was applied using actual data transfer between Suwon and Daegu. The verification experiment results demonstrate that the proposed scheme can predicts extremely well. The results can be utilized on a decision basis for all applications that require balancing network resources such as the socket handlers and socket buffers.

In order to improve the equation, packet loss and traffic interference must be carefully considered. Moreover, a parallel transfer management mechanism which can be applied to GridFTP has to be studied in detail using the research results.

References

1. A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets," Journal of Network and Computer Application, vol.23, pp. 187-200, 2001.
2. B. Allcock, I. Mandrichenko, and T. Perelmutov, "GridFTP v2 Protocol Description," Germi National Accelerator Laboratory, 2004.

3. D. Kakadia, "Understanding Tuning TCP," Sun BluePrints, Mar. 2004.
4. B. Tierney, "TCP Tuning Guide for Distributed Applications on WAN," In USENIX&SAGE Login, http://www-didc.lbl.gov/tcp-wan.html, Feb. 2001.
5. J. Semke, J. Mahdavi, and M. Mathis, "Automatic TCP Buffer Tuning," ACM SIGCOMM 1998, vol. 28, no. 4, 1998.
6. H. Sivakumar, S. Bailey, and R. L. Grossman, "PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks," IEEE/ACM SC2000, Nov. 2000.
7. R. L. Grossman, H. Sivakumar, and S. Bailey, "PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks," Supercomputing, IEEE and ACM, 2000.
8. B. L. Tierney, D. Gunter, J. Lee, and M. Stoufer, "Enabling Network-Aware Applications," IEEE-HPDC, 2001.
9. M. K. Gardner, W. Feng, and M. Fisk, "Dynamic Right-Sizing in FTP (drs-FTP): Enhancing Grid Performance in User-Space," IEEE Symposium on High-Performance Distributed Computing (HPDC-11/2002), Edinburgh, Scotland, July 2002. LA-UR 02-2799.
10. E. Weigle and W. Feng, "Dynamic Right-Sizing:A Simulation Study," IEEE IC-CCN, 2001.
11. E. Weigle and W. Feng, "A Comparison of TCP Automatic Tuning Techniques for Distributed Computing," HPDC-11, 2002.
12. M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the Congestion Avoidance Algorithm," Computer Communications Review, vol. 27, number 3, July 1997.
13. M. Carson and D. Santay, "NIST Net: A Linux-based Network Emulation Tool," Computer Communication Review, June 2003.
14. M. Gates, A. Tirumala, J. Dugan, and K. Gibbs, Iperf, http://dast.nlanr.net/Projects/Iperf/, NLANR, 2003
15. TCP Extensions for High Performance, RFC1323.
16. Linux man page, http://www.die.net/doc/linux/man/man7/tcp.7.html