GridFTP-APT: Automatic Parallelism Tuning Mechanism for Data Transfer Protocol GridFTP

Takeshi Ito, Hiroyuki Ohsaki and Makoto Imase
Graduate School of Information Science and Technology
Osaka University
1-5, Yamadaoka, Suita, Osaka, 565-0871, Japan
{t-itou, oosaki, imase}@ist.osaka-u.ac.jp

Abstract

GridFTP has been used as a data transfer protocol to effectively transfer a large volume of data in Grid computing. GridFTP supports a feature called parallel data transfer *that improves throughput by establishing multiple TCP connections in parallel. However, for achieving high GridFTP throughput, the number of TCP connections should be optimized based on the network status. In this paper, we propose an automatic parallelism tuning mechanism called* GridFTP-APT (GridFTP with Automatic Parallelism Tuning) *that adjusts the number of parallel TCP connections only using information measurable in the Grid middleware. Through simulation experiments, we demonstrate that GridFTP-APT significantly improves the performance of GridFTP in various network environments.*

1. Introduction

GridFTP has been proposed as a protocol to effectively transfer a large volume of data in Grid computing [16, 4, 15]. GridFTP is designed to solve the existing TCP problems and has various additional features for this purpose.

These features include, for instance, parallel data transfer using multiple TCP connections and automatic negotiation of TCP socket buffer size. It is known that the effectiveness of GridFTP depends largely on control parameter configuration such as the number of parallel TCP connections [7, 14]. However, examination has not been conducted sufficiently so far regarding how to configure GridFTP control parameters. For example, although a command is defined in the GridFTP protocol to specify the number of parallel TCP connections between GridFTP server and client, it is not specified at all in the GridFTP protocol how to determine the number of parallel TCP connections.

There have been several studies concerning the configuration method of the GridFTP control parameters. In [17], the authors propose a method to determine the required TCP socket buffer size for GridFTP. By measuring the round-trip time and the bandwidth between GridFTP server and client, this method calculates the bandwidth-delay product of the network, and determines the TCP socket buffer size. However, since the GridFTP protocol needs to be modified when using the method proposed in [17], interoperability with existing GridFTP servers is unrealizable. Moreover, in [17], the authors focus only on the TCP socket buffer size, and do not consider the number of parallel TCP connections.

In [7], the throughput of parallel TCP connections is derived using a simple analytic model. However, the analytic approach in [7] does not model degradation of throughput when the number of parallel TCP connections is too large.

Moreover, in [9], the optimal control parameters of GridFTP is derived using mathematical analysis. In [9], the authors clarify the configuration method of GridFTP control parameters (i.e., the number of parallel TCP connections and the TCP socket buffer size) by deriving the GridFTP goodput in steady state. However, to apply the result in [9] to a real network, the round-trip time and the bottleneck link bandwidth of a network must be known in advance. However, it is not discussed in [9] how the round-trip time and the bottleneck link bandwidth are measured by GridFTP.

Moreover, in [10], the authors propose an automatic parameter configuration mechanism for GridFTP. The proposed method transfers a file as a series of blocks called *chunk*, and measures network status (i.e., the round-trip time and GridFTP goodput) for every chunk transfer. Based on the analytic result in [9] and measurement results, the number of parallel TCP connections is adjusted. In [10], three operation modes are proposed for heuristically adjusting the number of parallel TCP connections. However, there is a problem that GridFTP goodput does not improve so much in some conditions — for instance, in a network with a large bandwidth-delay product [10].

In this paper, we therefore propose a GridFTP-APT (GridFTP with Automatic Parallelism Tuning) mechanism that automatically adjusts the number of parallel TCP connections of GridFTP. GridFTP-APT operates on GridFTP clients by utilizing only information measurable in the Grid middleware. GridFTP-APT utilizes the fact that GridFTP goodput is a convex function for the number of parallel TCP connections. GridFTP-APT searches for the optimal number of parallel TCP connections using a numerical computation algorithm for a maximization problem. We evaluate the performance of GridFTP-APT through simulation experiments. Consequently, we show that GridFTP with GridFTP-APT can realize high throughput in various network environments.

The structure of this paper is as follows. First, Section 2 provides a general description of GridFTP as well as the explanation on parallel data transfer, one of the notable features of GridFTP. Section 3 discusses design principles of an automatic parallelism tuning mechanism for GridFTP. We then explain the basic ideas and algorithm of our proposed GridFTP-APT. Section 4 demonstrates the effectiveness of GridFTP-APT through simulation experiments. Finally, Section 5 summarizes the paper and mentions future tasks.

2. GridFTP

GridFTP is a data transfer protocol, which is designed to effectively transfer a large volume of data in Grid computing. GridFTP is an extension of FTP (File Transfer Protocol) [12, 6, 8] that has been widely used, and is currently under standardization in GGF (The Global Grid Forum) [1]. GridFTP, which uses TCP as its transport layer communication protocol, is designed to solve several problems of TCP. For example, besides the features of the existing FTP, it has additional features such as automatic negotiation of TCP socket buffer size, parallel data transfer, third-party control of file transfer, partial file transfer, security, and reliable data transfer [4].

Most of these specific features of GridFTP are realized by a new transfer mode called extended block mode [4]. Currently, GridFTP server and client software conforming to GridFTP version 1 (GridFTP v1) is included in the Globus Toolkit [2], which is the de facto standard middleware for Grid computing. However, in this specific GridFTP implementation, the feature of the automatic negotiation of TCP socket buffer size is not implemented, so a user must manually specify the number of parallel TCP connections for parallel data transfer. In addition, GGF has been discussing the problems with GridFTP v1 and also undertaking a study on GridFTP v2 (version 2) as a solution to these problems [11]. Additional features have been incorporated in GridFTP v2. These features relax several limitations of the extended block mode in GridFTP v1, such that data transfer is restricted to a single direction and unable to open/close data channels in the midst of the data transfer. However, how to configure the GridFTP control parameters regarding parallel TCP connections has not been addressed even in the discussion on GridFTP v2.

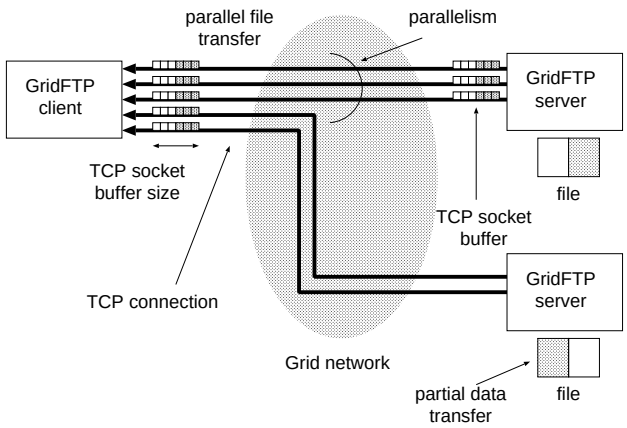

Figure 1. Parallel data transfer in GridFTP

Multiple TCP connections can be established in parallel in GridFTP using `OPTS RETR` command (Fig. 1). With this feature, a single file can be transferred from a single server through multiple TCP connections. Higher throughput can be expected by aggregating multiple TCP connections in comparison to using a single TCP connection [14].

This can be explained by the following reasons: (1) By aggregating multiple TCP connections, larger bandwidth can be gained in the TCP congestion avoidance phase than those gained by other competing TCP connections. This situation results from the fact that, in the TCP congestion avoidance phase, the AIMD window flow control is being executed, and therefore aggregated multiple TCP connections can transfer data more advantageously through the network with a smaller packet loss probability. (2) By aggregating multiple TCP connections, the total TCP socket buffer size available to the file transfer increases. Aggregation of N TCP connections can utilize as N times TCP socket buffer size as that of a single TCP connection. (3) By aggregating multiple TCP connections, ramp-up time of the transfer rate in the TCP slow-start phase is shortened. In the slow-start phase, the congestion window doubles every round-trip time. For this reason, through the aggregation of N TCP connections the speed of the transfer rate increase becomes N times as fast as that of a TCP connection.

However, the throughput drops if the number of aggregate TCP connections, N, becomes too large since this may cause the following situations: (1) The window size per TCP connection becomes smaller, so TCP timeout would

frequently occur. (2) The overhead required for the server to process TCP protocol stack would increase. Therefore, the optimal number of TCP connections, N, should be determined based on the network status. Nevertheless, how to optimize the number of parallel TCP connections, N, in GridFTP has not sufficiently been studied and still remains as an open issue.

3. Automatic Parallelism Tuning Mechanism for GridFTP

3.1. Design Principles

First, we explain the basic principles for designing an automatic parallelism tuning mechanism for GridFTP.

It is extremely important to provide compatibility with existing GridFTP servers in designing an automatic parallelism tuning mechanism for GridFTP. GridFTP is implemented in the Globus Toolkit and has been spreading rapidly in recent years. Since a number of GridFTP servers have already been in operation, it is preferable to realize the automatic parallelism tuning mechanism at the side of GridFTP client. Additionally, it is also favorable to realize the automatic parallelism tuning mechanism without changing the existing GridFTP protocol so as to enable interconnection with existing GridFTP servers. If the automatic parallelism tuning is executed on the side of GridFTP client, it will be difficult to realize the automatic parallelism tuning during the third party transfer. Nonetheless, we believe that this would not be so problematic because most of the transfers are executed between GridFTP servers and clients.

Next, it is desirable that the automatic parallelism tuning mechanism for GridFTP can easily be installed in Grid computing environment. Generally, Grid computing is characterized by the heterogeneity of computers and networks constituting Grid. Therefore, the automatic parallelism tuning mechanism needs to operate in various computer environments as well as various network environments. For this reason, it is preferable for the automatic parallelism tuning mechanism to be realized in the Grid middleware layer. In other words, it is important for the automatic parallelism tuning mechanism to avoid using any function specific to certain operating systems or network devices on the computers.

3.2. Basic Ideas of GridFTP-APT

In [9], the GridFTP goodput G in steady state is approximately derived as

$$G \simeq \min\left(\frac{N\,W}{R}\,,\ \frac{N\,(1-p^*)}{2\,R}\left(-3+\frac{\sqrt{6+21\,p^*}}{\sqrt{p^*}}\right)\right), \quad (1)$$

$$p^* \simeq \left(-2+\frac{2\,B\,R}{N}+\frac{2}{3}\left(\frac{B\,R}{N}\right)^2\right)^{-1}, \quad (2)$$

where N is the number of parallel TCP connections, W is the TCP socket buffer size for each TCP connection, B is the bottleneck link bandwidth, and R is the round-trip time of the TCP connections.

Equation (1) indicates that the GridFTP goodput G is a convex function for the number N of parallel TCP connections. Thus, the number of parallel TCP connections, N, should be selected so as to maximize the GridFTP goodput G.

In this paper, we propose a GridFTP-APT mechanism that automatically adjusts the number of parallel TCP connections. GridFTP-APT utilizes the fact that the GridFTP goodput is a convex function for the number of parallel TCP connections. The basic idea of GridFTP-APT is that a GridFTP client divides a file to transfer into blocks called *chunk*, and adjusts the number of parallel TCP connections at the end of every chunk transfer. Chunk-based transfer can be realized using the extended block mode of GridFTP. More specifically, GridFTP-APT measures the goodput at every chunk transfer. According to measurement results, GridFTP-APT adjusts the number of parallel TCP connections so that the GridFTP goodput is maximized using a numerical computation algorithm for a maximization problem.

GridFTP-APT uses the GSS (Golden Section Search) algorithm, one of numerical computation algorithms for a maximization problem [13]. GSS algorithm is a technique of numerically searching for x that maximizes $f(x)$, when $f(x)$ is a convex function in the range of $[x_l : x_r]$ and derivatives of $f(x)$ are unknown. GridFTP-APT numerically searches for the optimal number of parallel TCP connections that maximizes the GridFTP goodput using GSS algorithm.

The GridFTP goodput of each chunk transfer can be calculated from the chunk size and its transfer time, which can be measured by transferring the chunk in the extended block mode [4]. Specifically, a chunk is transferred by `ERET` or `ESTO` command in the extended block mode of GridFTP while measuring its response time, T. Since the response to `ERET` or `ESTO` command is returned when the chunk transfer is completed [4], the GridFTP goodput can be calculated as $G = X/T$ from the chunk size, X, and the response time, T.

3.3. Adjusting the Number of Parallel TCP Connections

First, for applying the GSS algorithm, GridFTP-APT searches for a range of the number of parallel TCP connections called *bracket*, in which the GridFTP goodput takes a convex form.

GridFTP-APT starts from a small number of parallel TCP connections, and multiplicatively increases the number of parallel TCP connections at every chunk transfer until GridFTP goodput decreases. GridFTP-APT determines the bracket — the range of the number of parallel TCP connections covering the optimal value that maximizes the GridFTP goodput.

In what follows, N is the number of parallel TCP connections used for a chunk transfer, $G(N)$ the GridFTP goodput measured at the chunk transfer, and N_{-k} the number of parallel TCP connections used for the k-th last chunk transfer.

GridFTP-APT searches for the bracket as follows.

1. Initialize the number of parallel TCP connections:

$$N \leftarrow N_0,$$

where N_0 is the initial number of parallel TCP connections.

2. Transfer a chunk while measuring the GridFTP goodput $G(N)$.

3. If the following inequality is satisfied, determine the bracket as (N_{-2}, N_{-1}, N) and terminate the algorithm.

$$G(N) < G(N_{-1})$$

Otherwise, proceed to the step 4.

4. Increase the number of parallel TCP connections as follows, and return to the step 2.

$$N \leftarrow \alpha \times N,$$

where $\alpha(> 1)$ is a control parameter. Note that, since the number of parallel TCP connections is a positive integer, the integer closest to N in the above equation is used as the number of parallel TCP connections.

Using the GSS algorithm, GridFTP-APT searches for the number of parallel TCP connections that maximizes the GridFTP goodput within the bracket (l, m, r) during succeeding chunk transfers.

GridFTP-APT searches for the optimal number N of parallel TCP connections as follows.

1. Update the number N of parallel TCP connections:

$$N \leftarrow \begin{cases} l + (m - l)\nu & \text{if } m - l > r - m \\ m + (r - m)\nu & \text{otherwise} \end{cases} \tag{3}$$

where ν is the golden ratio ($= (3 - \sqrt{5})/2$) [13]. Note that, since the number of parallel TCP connections is a positive integer, the integer closest to N in the above equation is used as the number of parallel TCP connections.

2. Transfer a chunk while measuring the GridFTP goodput $G(N)$.

3. If the following inequality is satisfied, proceed to the step 4.

$$G(N) > G(m)$$

If the above inequality is not satisfied, change the bracket as follows and return to the step 1.

$$(l, m, r) \leftarrow \begin{cases} (l, m, N) & \text{if } m < N \\ (N, m, r) & \text{otherwise} \end{cases}$$

4. Change the bracket as follows, and return to the step 1.

$$(l, m, r) \leftarrow \begin{cases} (m, N, r) & \text{if } m < N \\ (l, N, m) & \text{otherwise} \end{cases} \tag{4}$$

3.4. Determining Chunk Size

It is an important problem to appropriately determine the size of a chunk in every chunk transfer.

For accelerating search of the optimal number of parallel TCP connections, it is desirable to keep the chunk size as small as possible. With a small chunk size, since each step in the algorithm described above is completed more shortly, it is expected that the number of parallel TCP connections converges faster to the optimal value.

However, if the chunk size is too small, the goodput of each TCP connection cannot be measured accurately because of TCP's characteristics [5]. Consequently, since the number of parallel TCP connections does not converge to the optimal value, the GridFTP goodput cannot be maximized. For this reason, it is necessary to increase the chunk size so that the TCP goodput can be measured accurately.

The chunk size required for accurately measuring the TCP goodput is affected by the network bandwidth. Hence, it is necessary to determine the chunk size according to the network bandwidth. However, of course, we cannot know the network bandwidth *before* the chunk transfer. Since the GridFTP-APT uses the extended block mode of GridFTP for chunk transfers, there is a limitation that the block size must be specified before starting the block transfer.

For solving this problem, GridFTP-APT predicts the GridFTP goodput of the next chunk transfer, and dynamically configures the chunk size so that the chunk transfer time becomes as fixed as possible. Specifically, GridFTP-APT determines the chunk size X as follows.

When searching for the bracket, GridFTP-APT predicts the GridFTP goodput of the next chunk transfer as $G(N_{-1}) \times G(N_{-1})/G(N_{-2})$ from the ratio of the last two chunk transfers, and determine the chunk size as

$$X = G(N_{-1})\frac{G(N_{-1})}{G(N_{-2})}\Delta,$$

where Δ is a control parameter, which is the target value of the chunk transfer time.

Note that, at the time of the first chunk transfer, since the GridFTP goodput $G(N_{-1})$ and $G(N_{-2})$ are unknown, the chunk size X is determined as follows:

$$X = \frac{N_0\, W}{R}\Delta,$$

where W is the TCP socket buffer size and R is the round-trip time. The round-trip time is measured from response times of commands on the GridFTP control channel. Note that, at the time of the second chunk transfer, the GridFTP goodput $G(N_{-2})$ is unknown. So the chunk size X is determined as follows.

$$X = \alpha\, G(N_{-1})\Delta$$

When GridFTP-APT searches for the optimal number of parallel TCP connections with the GSS algorithm, the GridFTP goodput of the next chunk transfer is predicted by the interpolation of two samples of the GridFTP goodput in the bracket (l, m, r). Namely, the chunk size is determined as follows.

$$X = \begin{cases} ((1-\xi)\, G(l) + \xi\, G(m))\, \Delta & \text{if } N < m \\ ((1-\xi)\, G(m) + \xi\, G(r))\, \Delta & \text{otherwise} \end{cases}$$

where

$$\xi = \begin{cases} \frac{N-l}{m-l} & \text{if } N < m \\ \frac{N-m}{r-m} & \text{otherwise} \end{cases}$$

3.5. Example of GridFTP-APT Operation

In what follows, we illustrate an example operation of GridFTP-APT. Figure 2 shows an example operation of GridFTP-APT when searching for a bracket with $N_0 = 1$ and $\alpha = 2$. The number k shown in a circle indicates the k-th chunk transfer.

GridFTP-APT searches for a bracket (i.e., the range of the number of parallel TCP connections covering the optimal value). First, GridFTP-APT initialize the number N of parallel TCP connections to N_0 (= 1). GridFTP-APT multiplicatively increases the number of parallel TCP connections as 1 → 2 → 4 → 8 at every chunk transfer until the GridFTP goodput starts to decrease. Since the GridFTP goodput decreases when the number N of parallel TCP connections changes as 4 → 8, the bracket is determined as (2, 4, 8).

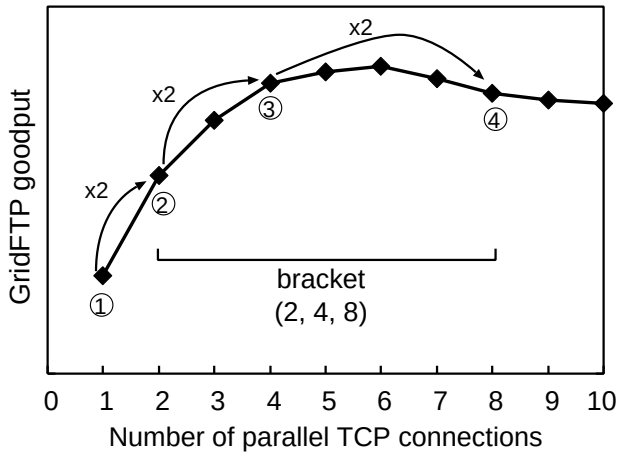

Figure 2. Example of GridFTP-APT operation when searching for a bracket

Next, Fig. 3 shows an example operation of GridFTP-APT when searching for the optimal number of parallel TCP connections. In Fig. 3, the number k on an arrow indicates that the bracket is changed at k-th chunk transfer.

The GSS algorithm is applied to the bracket (2, 4, 8) during chunk transfers, and the number of parallel TCP connections is adjusted for maximizing the GridFTP goodput. Since the bracket is (2, 4, 8), the number of parallel TCP connections at the 5-th chunk transfer N is determined as $N = 6$ from Eq. (3). The GridFTP goodput in the 5-th chunk transfer is $G(6)$, and since $G(4) < G(6)$ is satisfied, the bracket is updated as (4, 6, 8) from Eq. (4). Hereafter, in a similar way, GridFTP-APT changes the number of parallel TCP connections N as 6 → 7 → 5, and updates the bracket as (4, 6, 8) → (4, 6, 7) → (5, 6, 7). Finally, when the bracket is (5, 6, 7), the number of parallel TCP connections is fixed at $N = 6$, which maximizes the GridFTP goodput.

4. Simulation

In this section, we quantitatively evaluate the effectiveness of GridFTP-APT through simulation experiments.

Figure 4 shows the network model used in simulation. We performed simulation by changing the propagation de-

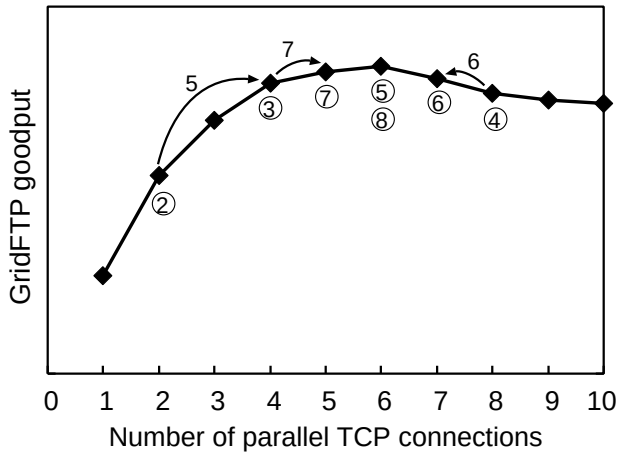

Figure 3. Example of GridFTP-APT operation when searching for the optimal number of TCP connections

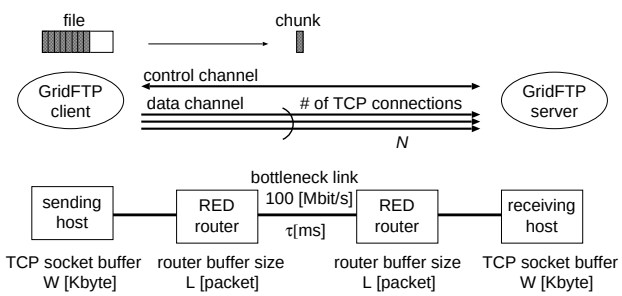

Figure 4. Network model used in simulation

lay of the bottleneck link in Fig. 4. In this network model, a GridFTP server and client are connected via two RED routers. Data transfer was continuously performed from the GridFTP client to the GridFTP server. Moreover, for clarifying the fundamental characteristics of GridFTP-APT, we performed simulation in an environment where background traffic does not exist. Note that ns-2 simulator [3] with our GridFTP-APT implementation was used for simulation.

Table 1 shows the parameter configuration used in simulation. Unless explicitly stated, parameters shown in Tab. 1 are used in the following simulations.

The evolution of the number of parallel TCP connections of GridFTP-APT when the propagation delay of the bottleneck link is 10 [ms] is shown in Fig. 5. This figure shows that the number of parallel TCP connections of GridFTP-APT converges to $N = 13$ at approximately $t = 15$ [s]. In this case, the bracket was determined at approximately $t = 10$ [s]. After approximately 5 [s], the number of parallel TCP connections was optimized.

Table 1. Parameter configuration used in simulation

Bottleneck link bandwidth	100	[Mbit/s]
Propagation delay of the bottleneck link	10 or 20	[ms]
Buffer size of RED router	100	[packet]
Control parameter of RED router max_{th}	75	
Control parameter of RED router min_{th}	25	
Control parameter of RED router max_p	0.1	
Control parameter of RED router w_q	0.002	
TCP socket buffer size	64	[Kbyte]
TCP packet size	1000	[byte]
Initial number of parallel TCP connections N_0	4	
GridFTP-APT control parameter α	2	
Target value of chunk transfer time Δ	1.0	[s]

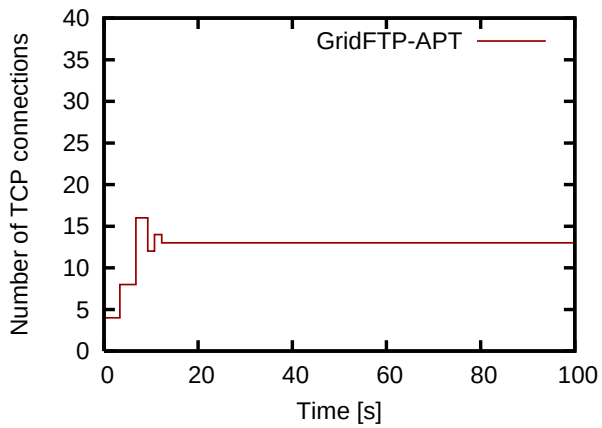

Figure 5. The number of TCP connections in GridFTP-APT for $\tau = 10$ [ms]

The GridFTP-APT goodput in this scenario is shown in Fig. 6. For comparison purposes, GridFTP goodput in steady state when fixing the number of parallel TCP connections at 1, 4, 8, 16, and 32 are also plotted in the figure. First, one can find that the optimal number of parallel TCP connections seems to exist between 8–16 from the GridFTP goodput with the fixed number of parallel TCP connections. GridFTP-APT reaches goodput of approximately 80 [Mbit/s] at approximately $t = 10$ [s]. After approximately 5 [s], the GridFTP goodput converges to 85.3 [Mbit/s]. Note that, since the buffer size of RED routers is small (i.e., 100 [packet]) in our simulations, the maximum goodput was approximately 85 [Mbit/s] even with the optimal number of parallel TCP connections. Namely, this indicates that GridFTP-APT can utilize the network resource quite effectively after approximately

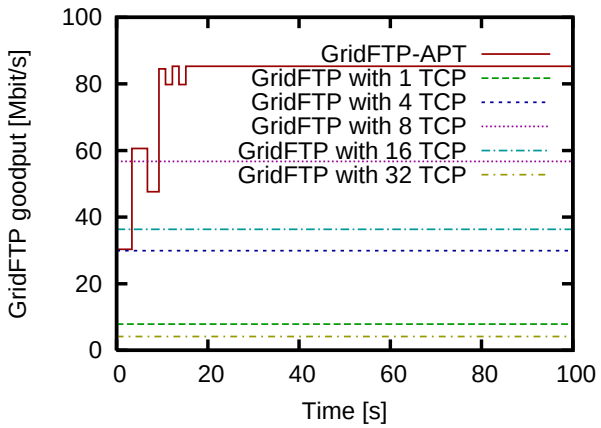

Figure 6. Evolution of goodput in GridFTP-APT for $\tau = 10$ [ms]

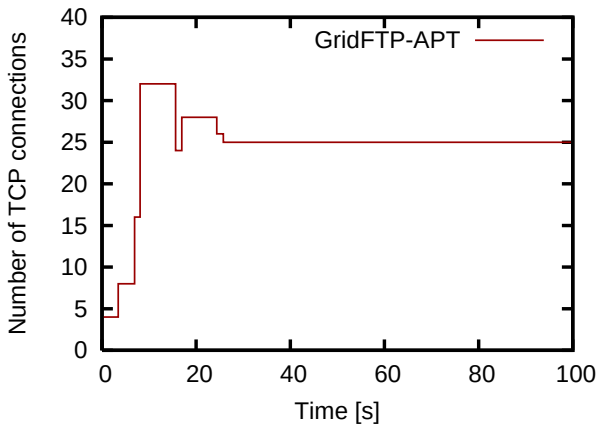

Figure 7. The number of TCP connections in GridFTP-APT for $\tau = 20$ [ms]

15 [s] from starting the transfer.

Evolutions of the number of parallel TCP connections and the GridFTP goodput when the propagation delay of the bottleneck link is 20 [ms] are shown in Figs. 7 and 8, respectively.

Figure 7 shows that the number of parallel TCP connections converges at approximately $t = 25$ [s]. Since the propagation delay of the bottleneck link is larger than the previous case (Fig. 7), the convergence time of the number of parallel TCP connections becomes larger. One can find that the optimal number of parallel TCP connections is $N = 25$, which is significantly different from that in the previous case ($N = 13$).

Figure 8 shows that the GridFTP-APT can realize a high goodput by appropriately adjusting the number of parallel TCP connections. Figure 8 shows that the GridFTP goodput degrades temporarily until GridFTP-APT determines the bracket. However, Fig. 8 also shows that GridFTP-APT finally achieves high goodput by adjusting the number of parallel TCP connections during chunk transfers. Note that, in this simulation, due to small buffer of RED routers and a large propagation delay, the maximum GridFTP goodput was approximately 67.5 [Mbit/s] even with the optimal number of parallel TCP connections. Namely, this indicates that GridFTP-APT can utilize the network resource quite effectively regardless of the propagation delay of the bottleneck link.

5. Conclusion

In this paper, we have proposed GridFTP-APT, an automatic parallelism tuning mechanism for GridFTP, mainly focusing on the parallel data transfer feature for GridFTP. GridFTP-APT utilizes the fact that GridFTP goodput is a convex function for the number of parallel TCP connections. GridFTP-APT searches for the optimal number of parallel TCP connections using the GSS algorithm, one of numerical computation algorithms for a maximization problem. In this paper, we have shown that GridFTP with GridFTP-APT realizes high throughput in several network environments.

As future work, we are planning to do performance evaluation of GridFTP-APT in more general network environments. In particular, we need performance evaluation of GridFTP-APT in a network with background traffic and/or multiple GridFTP-APT sessions. Moreover, it would be valuable to implement our proposed GridFTP-APT and to demonstrate its effectiveness in a real network.

Acknowledgments

We would like to express our appreciation to Prof. Masayuki Murata for joining meaningful discussions. This work is supported by the NAREGI (National Research Grid Initiative) Project from the Ministry of Education, Culture, Sports, Science and Technology, Japan. This work is also supported by a Grant-in-Aid for Scientific Research on Priority Areas (No. 16016261) from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References

[1] Global Grid Forum. `http://www.ggf.org/`.

[2] Globus Toolkit. available at `http://www.globus.org/`.

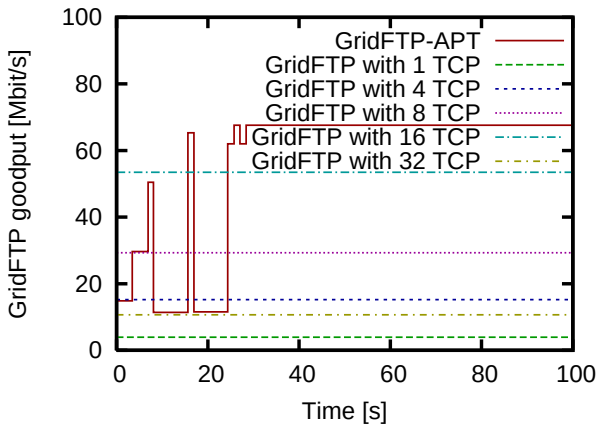

Figure 8. Evolution of GridFTP goodput for $\tau = 20$ [ms]

[3] The network simulator – ns2. available at `http://www.isi.edu/nsnam/ns/`.

[4] W. Allcock et al. GridFTP: Protocol extensions to FTP for the Grid. *GGF Document Series GFD.20*, Apr. 2003. Also available as `http://www.ggf.org/documents/GFD.20.pdf`.

[5] N. Ehsan and M. Liu. Analysis of TCP transient behavior and its effect on file transfer latency. In *Proceedings of IEEE International Conference on Communications (ICC2003)*, volume 26, pages 1806–1811, May 2003.

[6] R. Elz and P. Hethmon. FTP security extensions. *Request for Comments (RFC) 2228*, Oct. 1997.

[7] T. J. Hacker, B. D. Athey, and B. Noble. The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network. In *Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS)*, pages 434–443, Apr. 2002.

[8] P. Hethmon and R. Elz. Feature negotiation mechanism for the file transfer protocol. *Request for Comments (RFC) 2389*, Aug. 1998.

[9] T. Ito, H. Ohsaki, and M. Imase. On parameter tuning of data transfer protocol GridFTP in wide-area Grid computing. In *Proceedings of Second International Workshop on Networks for Grid Applications (GridNets 2005)*, pages 415–421, Oct. 2005.

[10] T. Ito, H. Ohsaki, and M. Imase. Automatic parameter configuration mechanism for data transfer protocol GridFTP. In *Proceedings of the 2006 International Symposium on Applications and the Internet (SAINT 2006)*, pages 32–38, Jan. 2006.

[11] I. Mandrichenko, W. Allcock, and T. Perelmutov. GridFTP v2 protocol description. *GGF Document Series GFD.47*, May 2005. Also available as `http://www.ggf.org/documents/GFD.47.pdf`.

[12] J. Postel and J. Reynolds. File transfer protocol (FTP). *Request for Comments (RFC) 959*, Oct. 1985.

[13] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, 1992.

[14] L. Qiu, Y. Zhang, and S. Keshav. On individual and aggregate TCP performance. In *Proceedings of Internetl Conference on Network Protocols*, pages 203–212, Oct. 1999.

[15] The Globus Project. GridFTP update January 2002, 2002. available at `http://www.globus.org/datagrid/deliverables/GridFTP-Overview-200201.pdf`.

[16] The Globus Project. GridFTP: universal data transfer for the Grid. *White Paper*, Sept. 2003. available at `http://www.globus.org/datagrid/deliverables/C2WPdraft3.pdf`.

[17] S. Thulasidasan, W. Feng, and M. K. Gardner. Optimizing GridFTP through dynamic right-sizing. In *Proceedings of IEEE International Symposium on High Performance Distributed Computing*, pages 14–23, June 2003.