Implementation and Evaluation of GridFTP Automatic Parallelism Tuning Mechanism for Long-Fat Networks

Fumito Inoue, Takeshi Ito, Hiroyuki Ohsaki and Makoto Imase

Graduate School of Information Science and Technology, Osaka University

Yamadaoka, Suita, Osaka, 560-8531, Japan

E-mail: {f-inoue, t-itou, oosaki, imase}@ist.osaka-u.ac.jp

Abstract—In this paper, we present our implementation of *GridFTP-APT (GridFTP with Automatic Parallelism Tuning)*, an extension to GridFTP for optimizing its performance in a long-fat network. GridFTP has been used as a data transfer protocol for effectively transferring 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, to achieve high GridFTP throughput, the number of TCP connections should be optimized according to network condition. We have proposed an automatic parallelism tuning mechanism called GridFTP-APT that adjusts the number of parallel TCP connections only using information measurable at Grid middleware. In this paper, we first explain our implementation of GridFTP-APT based on the GridFTP client included in the Globus Toolkit (globus-url-copy program). We then investigate the effectiveness of GridFTP-APT in realistic network environments through several experiments using our GridFTP-APT implementation and a network emulator. Consequently, we demonstrate the effectiveness of GridFTP-APT in various network environments.

I. Introduction

GridFTP has been used as a protocol to effectively transfer large volume data in Grid computing [1, 2]. GridFTP is designed to solve the existing TCP problems and has various additional features to the existing FTP 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 [3-5]. However, it is difficult to configure GridFTP control parameters appropriately according to the condition of a network. Although a command set is defined in the GridFTP protocol for specifying the number of parallel TCP connections between GridFTP server and client, its configuration method is not specified in the GridFTP protocol specification [1, 2].

We have proposed GridFTP-APT (GridFTP with Automatic Parallelism Tuning) mechanism that automatically adjusts the number of parallel TCP connections of GridFTP [6]. GridFTP-APT operates on a GridFTP client, and automatically adjusts the number of parallel TCP connections so that the GridFTP goodput is maximized. GridFTP-APT utilizes the fact that GridFTP goodput is a convex function for the number of parallel TCP connections [3]. GridFTP-APT searches for the optimal number of parallel TCP connections using a numerical computation algorithm for a maximization problem.

In this paper, we explain our implementation of GridFTP-APT based on the GridFTP client included in the GlobusToolkit, which is a de facto standard middleware for Grid computing. Moreover, we evaluate the performance of GridFTP-APT in realistic network environments through several experiments using a network emulator. We conduct experiments while changing the bandwidth and propagation delay of the bottleneck link using the network emulator. In such network environments, we quantitatively evaluate the performance of GridFTP-APT. We also investigate the performance of GridFTP-APT in a network with background traffic or multiple GridFTP sessions. Consequently, we show that GridFTP-APT operates effectively in various network environments.

The structure of this paper is as follows. Section II explains the overview of GridFTP-APT, the automatic parallelism tuning mechanism for GridFTP. Section III explains how we implement GridFTP-APT based on the GridFTP client included in the Globus Toolkit. Section IV quantitatively evaluates the performance of GridFTP-APT under various network conditions through several experiments using the network emulator. Finally, Section V summarizes the paper and discusses future works.

II. Automatic Parallelism Tuning Mechanism for GridFTP

This section explains the overview of GridFTP-APT. Refer to [6] for the details of GridFTP-APT.

GridFTP-APT is a mechanism executed on the GridFTP client (Fig. 1). The basic idea of GridFTP-APT is that a GridFTP client splits a file to transfer into blocks called *chunk*, and adjusts the number of parallel TCP connections at the end of every chunk transfer.

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 [7].

A. Adjusting the Number of Parallel TCP Connections

First, for applying the GSS algorithm, GridFTP-APT searches the range of the number of parallel TCP connections, in which the GridFTP goodput takes a convex form.
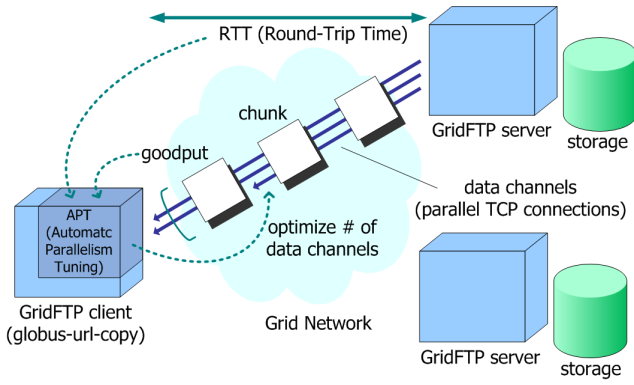
Fig. 1: GridFTP-APT (GridFTP with Automatic Parallelism Tuning) overview; the basic idea of GridFTP-APT is that a GridFTP client splits a file to transfer into blocks called *chunk*, and adjusts the number of parallel TCP connections at the end of every chunk transfer.

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 previous chunk transfer.

GridFTP-APT searches the bracket as follows.

1) Initialize the number of parallel TCP connections:

$$N \leftarrow N_0 \qquad (1)$$

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.

$$G(N) < G(N_{-1}) \qquad (2)$$

Otherwise, proceed to the step 4.

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

$$N \leftarrow \alpha \times N \qquad (3)$$

where $\alpha (> 1)$ is a control parameter.

Using the GSS algorithm [7], GridFTP-APT searches the number of parallel TCP connections that maximizes the GridFTP goodput within the bracket (l, m, r) during succeeding chunk transfers.

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

1) Update the number N of parallel TCP connections:

$$N \leftarrow \begin{cases} l + (m - l)\nu & \text{if } m - l > r - m \\ m + (r - m)\nu & \text{otherwise} \end{cases} \qquad (4)$$

where ν is the golden ratio $(= (3 - \sqrt{5})/2)$ [7].

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

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

$$G(N) > G(m) \qquad (5)$$

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

$$(l, m, r) \leftarrow \begin{cases} (l, m, N) & \text{if } m < N \\ (N, m, r) & \text{otherwise} \end{cases} \qquad (6)$$

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

$$(l, m, r) \leftarrow \begin{cases} (m, N, r) & \text{if } m < N \\ (l, N, m) & \text{otherwise} \end{cases} \qquad (7)$$

B. Determining Chunk Size

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 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 past chunk transfers, and determine the chunk size as

$$X \leftarrow G(N_{-1}) \frac{G(N_{-1})}{G(N_{-2})} \Delta, \qquad (8)$$

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

$$X \leftarrow \frac{N_0 W}{R} \Delta, \qquad (9)$$

where W is the TCP socket buffer size and R is the round-trip time.

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

$$X \leftarrow \alpha G(N_{-1}) \Delta. \qquad (10)$$

When GridFTP-APT searches 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.

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

$$\xi \leftarrow \begin{cases} \frac{N - l}{m - l} & \text{if } N < m \\ \frac{N - m}{r - m} & \text{otherwise} \end{cases} \qquad (11)$$
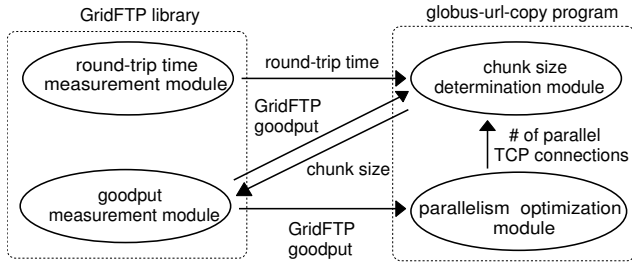
Fig. 2: Modular structure of GridFTP-APT implementation; we added four modules to the GridFTP library and the globus-url-copy program.
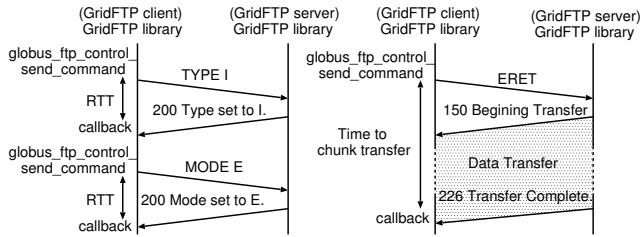

Fig. 3: Measurement method of round-trip time and GridFTP goodput; the round-trip time is measured by measuring the duration between when `globus_ftp_control_send_command` is invoked, and when the callback function is invoked using `gettimeofday(2)`.

III. GRIDFTP-APT IMPLEMENTATION

A. Overview

We implemented GridFTP-APT based on the GridFTP client included in the Globus Toolkit (globus-url-copy program). The globus-url-copy program is a GridFTP client that operates on the command line, and is invoked as `globus-url-copy [option] srcURL dstURL` where `srcURL` is a source URL and `dstURL` is a destination URL. For example, to copy a file `/foo/bar` from a GridFTP server `host` to a temporary directory, globus-url-copy is invoked as `globus-url-copy gsiftp://host/foo/bar file:///tmp/bar`

With our GridFTP-APT implementation, a user can specify control parameters (i.e., the initial number of parallel TCP connections N_0, the multiplicative increase factor of parallel TCP connections α, and the target value of chunk transfer time Δ) of GridFTP-APT with command-line options. GridFTP-APT is enabled by default, but it can be disabled by specifying `-noapt` option.

B. Modular Structure

To implement GridFTP-APT, we added four modules (i.e., round-trip time measurement module, goodput measurement module, chunk size determination module, and parallelism optimization module) to the globus-url-copy program and the GridFTP library (Fig. 2).

The round-trip time measurement module measures the average round-trip time between a GridFTP server

and client. This module measures the response time of GridFTP commands transfered on a control channel, and calculates the average round-trip time using an exponential weighted moving average.

By modifying the GridFTP library included in the Globus Toolkit, we implemented the round-trip time measurement module. `globus_ftp_control_send_command` is a function for sending the GridFTP command. After calling this function, a callback function is invoked when the response of the GridFTP command is received. This module measures the round-trip time by measuring the duration between when `globus_ftp_control_send_command` is invoked, and when the callback function is invoked using `gettimeofday(2)` (Fig. 3).

The goodput measurement module measures the goodput at every chunk transfer. This module calculates GridFTP goodput from the chunk size and time required to transfer the chunk.

By modifying the GridFTP library included in the Globus Toolkit, we implemented the goodput measurement module. A GridFTP client measures the duration between when the GridFTP client sends `ERET` command or `ESTO` command (i.e, when `globus_ftp_control_send_command` function is invoked) , and when the chunk transfer is completed (i.e., when the callback function is invoked) using `gettimeofday(2)` (Fig. 3).

The parallelism optimization module updates the number of parallel TCP connections according to the GridFTP-APT algorithm so that the GridFTP goodput is maximized. Specifically, this module determines the number of parallel TCP connections used for the next chunk transfer using Eqs. (1), (3) and (4).

By modifying the globus-url-copy program included in Globus Toolkit, we implemented the parallelism optimization module. The parallelism optimization module is automatically invoked from `globus_l_guc_transfer_files` function in the globus-url-copy program at the beginning of every chunk transfer.

The chunk size determination module calculates the chunk size for the next chunk transfer so that the chunk transfer time becomes as fixed as possible. Specifically, this module calculates the next chunk size using Eqs. (8)–(11).

By modifying the globus-url-copy program included in the Globus Toolkit, we implemented the chunk size determination module. The chunk size determination module is automatically invoked from `globus_l_guc_transfer_files` function in the globus-url-copy program at the beginning of every chunk transfer.

IV. PERFORMANCE EVALUATION

A. Experiment Setup

Figure 4 shows the network configuration used for all experiments. In this network, a GridFTP server and client are connected via two Ethernet switches and a PC router. The PC router is a network emulator [8] for imitating change in the bandwidth and delay of the bottleneck link. Since the bandwidth of all links is 1 [Gbit/s], a network emulator is the bottleneck. A computer for generating the
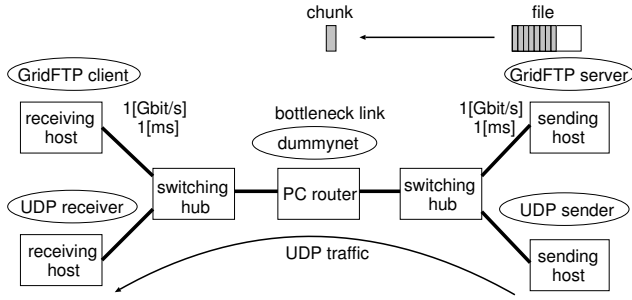
Fig. 4: Network configuration used in experiments; the PC router is a network emulator for imitating change in the bandwidth and delay of the bottleneck link.

TABLE I
PARAMETER CONFIGURATION USED IN EXPERIMENTS

Bandwidth of network emulator	1 [Gbit/s]
Delay of network emulator	10 [ms]
Buffer size of router	500 [packet]
Queue discipline of router	DropTail
TCP socket buffer size	64 [Kbyte]
TCP packet size	1,500 [byte]
Initial number of parallel TCP connections N_0	4
Multiplicative increase factor α	2
Target value of chunk transfer time Δ	3 [s]
File size transferred	50 [Gbyte]

UDP traffic at a fixed rate as background traffic is also used.

We used the same computers with an Intel Xeon 3.06 [GHz] processor and 2 [Gbyte] memory for the GridFTP server, the GridFTP client and the network emulator. For the GridFTP server and the GridFTP client, we used Debian GNU/Linux (kernel version 2.6.8) and Globus Toolkit 4.0.3. We used FreeBSD 6.1 and dummynet 1.3.14.1 for the network emulator. Table I shows the parameter configuration used in experiments. Unless explicitly stated, parameters shown in Tab. I are used in the following experiments.

B. Evolution of GridFTP Goodput

First, we investigate whether GridFTP operates effectively in realistic network configurations. The evolution of GridFTP goodput with GridFTP-APT is shown in Fig. 5. For comparison purposes, GridFTP goodput in steady state when fixing the number of parallel TCP connections at 1, 4, 8, 16, 32 and 64 are also plotted in the figure. One can find that the optimal number of parallel TCP connections seems to exist between 16–64 from the GridFTP goodput with the fixed number of parallel TCP connections. Moreover, this figure shows that the number of parallel TCP connections is optimized at approximately 40 [s], and the GridFTP goodput converges to 900 [Mbit/s]. The evolution of the number of parallel TCP connections in this scenario is shown in Fig. 6. This figure shows that the number of parallel TCP connections of GridFTP-APT converges to 32 at approximately 40 [s]. This agrees with the result in Fig. 5 where the optimal number of parallel TCP connections exists between 16–64. From
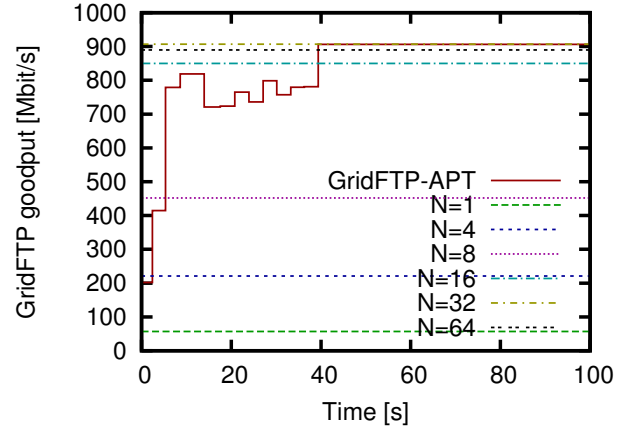

Fig. 5: Evolution of GridFTP goodput; the number of parallel TCP connections is optimized at approximately 40 [s], and the GridFTP goodput converges to 900 [Mbit/s].
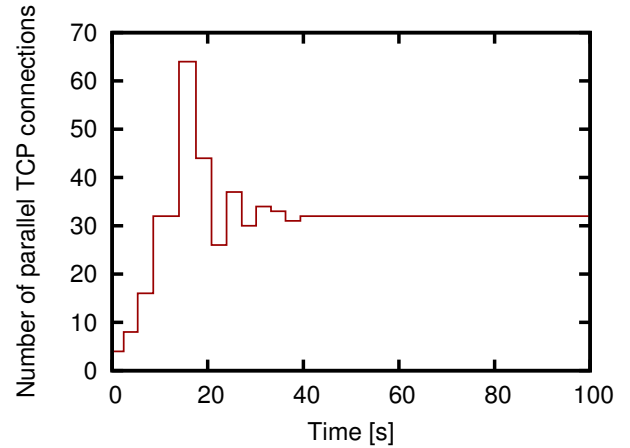

Fig. 6: Evolution of the number of parallel TCP connections; the number of parallel TCP connections of GridFTP-APT converges to 32 at approximately 40 [s].

these observations, we find that GridFTP-APT optimizes the number of parallel TCP connections at approximately 40 [s], and utilizes the network resource quite effectively.

C. Effect of Bottleneck Link Bandwidth

To investigate the effect of the bottleneck link bandwidth on GridFTP goodput, we performed experiments by changing the bandwidth of the network emulator. Figures 7 and 8 show the GridFTP goodput and the number of parallel TCP connections when the bandwidth of the network emulator is changed as 100–1000 [Mbit/s]. We conducted five experiments and measured the average and 95% confidence interval of GridFTP goodput and the number of TCP connections. For comparison purposes, GridFTP goodput in steady state when fixing the number of parallel TCP connections at 4,8 and 16 are also plotted in the Fig. 7.

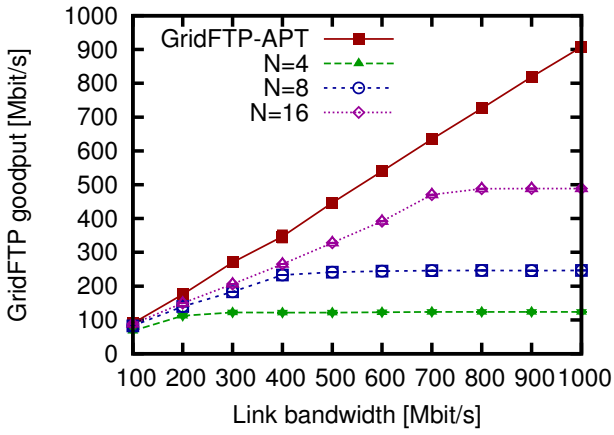Figure 7 shows that GridFTP-APT utilizes the network

Fig. 7: Bottleneck link bandwidth vs. GridFTP goodput; GridFTP-APT utilizes the network resource quite effectively regardless of the bottleneck link bandwidth.
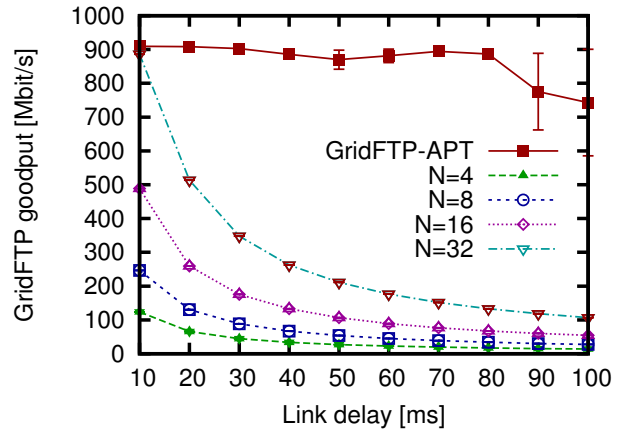

Fig. 9: Bottleneck link delay vs. GridFTP goodput; GridFTP-APT utilizes the network resource quite effectively when the delay of the network emulator is less than 80 [ms].
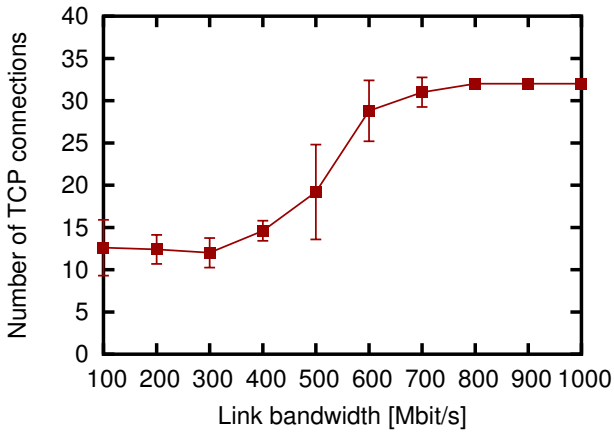

Fig. 8: Bottleneck link bandwidth vs. the optimal number of parallel TCP connections; the optimal number of parallel TCP connections increases as the network bandwidth becomes large.
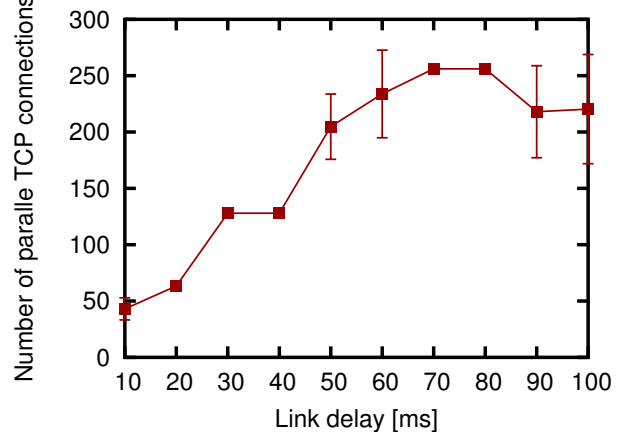

Fig. 10: Bottleneck link delay vs. the optimal number of parallel TCP connections; GridFTP-APT can optimize the number of parallel TCP connections regardless of the delay of the network emulator.

resource quite effectively regardless of the bottleneck link bandwidth. Moreover, Fig. 8 shows that the optimal number of parallel TCP connections increases as the network bandwidth becomes large. This can be explained as follows. The bandwidth delay product increases as the network bandwidth becomes large. Therefore, the number of parallel TCP connections required for fully utilizing the network resources increases.

D. Effect of Propagation Delay

To investigate the effect of the propagation delay of the bottleneck link on GridFTP goodput, we performed experiments by changing the delay of the network emulator. Figures 9 and 10 show the GridFTP goodput and the number of parallel TCP connections when the delay of the network emulator is changed as 10–100 [ms]. For comparison purposes, GridFTP goodput in steady state when fixing the number of parallel TCP connections at

4, 8, 16 and 32 are also plotted in Fig. 9.

Figure 9 shows that GridFTP-APT utilizes the network resource quite effectively when the delay of the network emulator is less than 80 [ms]. When the delay of the network emulator exceeds 80 [ms], the GridFTP goodput is slightly degraded. This is because the default value of the control parameter Δ is small. If the control parameter is configured according to the delay of a network, the GridFTP goodput can be improved. Figure 10 shows that GridFTP-APT can optimize the number of parallel TCP connections regardless of the delay of the network emulator.

E. Effect of Background Traffic

In realistic network configurations, the bottleneck link might be shared by many users. We therefore investigate the effect of background traffic on the performance of GridFTP-APT. Figure 11 show the GridFTP goodput when
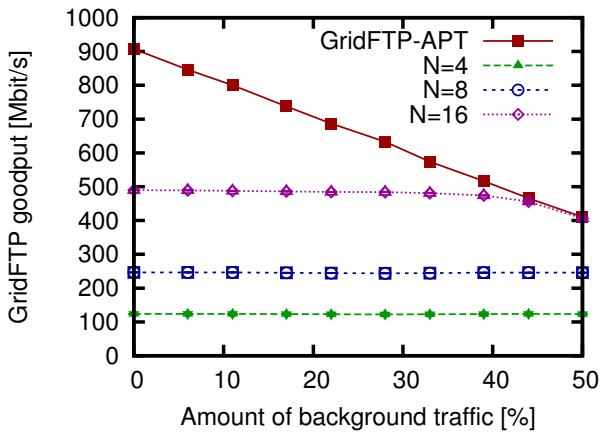
Fig. 11: Amount of background traffic vs. GridFTP goodput; GridFTP-APT utilizes the network resource quite effectively regardless of the amount of background traffic.
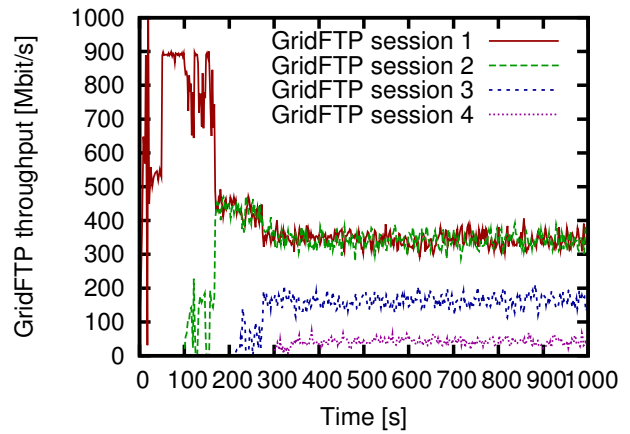

Fig. 12: Evolution of GridFTP goodput with four GridFTP sessions; the fairness among GridFTP sessions is not realized when there exist more than two GridFTP sessions.

the transmission rate of background traffic is changed as 10–50 % of the bottleneck link bandwidth. For comparison purposes, GridFTP goodput in steady state when fixing the number of parallel TCP connections at 4, 8 and 16 are also plotted in the figure. This figure shows that GridFTP-APT utilizes the network resource quite effectively regardless of the amount of background traffic.

F. Fairness among Multiple GridFTP Sessions

The bottleneck link might be shared by multiple GridFTP sessions. In this case, it is important that GridFTP sessions can fairly share the bottleneck link bandwidth. For investigating fairness among multiple GridFTP sessions, we measured the evolutions of GridFTP throughputs of multiple GridFTP sessions (Fig. 12). Four GridFTP sessions are activated every 100 [s]. Figure 12 shows that the fairness among GridFTP sessions is not realized when there exist more than two GridFTP sessions. This is an intrinsic problem of parallel TCP connections; i.e., the more TCP connections are established, the more bandwidth is gained. Namely, this result suggests that when parallel data transfer is widely used by many users, GridFTP requires some arbitration mechanism for fair bandwidth allocation to multiple GridFTP sessions.

V. CONCLUSION

In this paper, we have explained our implementation of GridFTP-APT based on the GridFTP client included in the GlobusToolkit, which was a de facto standard middleware of Grid computing. We have also investigated the effectiveness of GridFTP-APT in realistic network environments through several experiments using our GridFTP-APT implementation and a network emulator. Consequently, we have demonstrated the effectiveness of GridFTP-APT in various network environments.

As future work, it is of great value to further enhance the GridFTP-APT algorithm. For instance, we are currently working on shortening the convergence time of the number of parallel TCP connections. As shown in Section VI, our GridFTP-APT implementation takes approximately 40 [s] for optimizing the number of parallel TCP connections. We believe such convergence time is acceptable for practical purposes since GridFTP is generally used for large data transfers. However, if the convergence time is further shortened, GridFTP-APT can also be applied to small data transfers, which must be beneficial to various Grid applications.

REFERENCES

[1] 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/document/GFD.20.pdf.
[2] 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/document/GFD.47.pdf.
[3] 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)*, Oct. 2005, pp. 415–421.
[4] 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)*, Apr. 2002, pp. 434–443.
[5] 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," in *Proceedings of the 2000 ACM/IEEE Conference on Supercomputing*, Nov. 2000, pp. 4–10.
[6] T. Ito, H. Ohsaki, and M. Imase, "GridFTP-APT: Automatic parallelism tuning mechanism for data transfer protocol GridFTP," in *Proceedings of 6th IEEE International Symposium on Cluster Computing and the Grid (CCGrid2006)*, May 2006, pp. 454–461.
[7] W. H.Press, W. H.Vetterling, S. A.Teukolsky, and B. P.Flannery, *Numerical Recipes in C second edition*. Cambridge University Press Cambridge, May 1993.
[8] L. Rizzo, "Dummynet: a simple approach to the evaluation of network protocols," *ACM Computer Communications Review*, vol. 27, no. 1, pp. 31–41, Jan. 1997. [Online]. Available: http://citeseer.ist.psu.edu/rizzo97dummynet.html