Vol.2013-HPC-139 No.8
2013/5/29

UDTを用いた並列ファイル転送技術

渡邉 英伸[1]　黒澤 隆[2]　一岡 翔太郎[3,1]　木村 映善[4]　村田 健史[1]　建部 修見[5]

概要：ハイパフォーマンスコンピューティングなどで利用される広域分散ストレージ環境では，LFN(Long Fat Network) のエンドツーエンド通信における高速なデータ移行の課題が残っている．この問題を解決するために，アプリケーションレベルで高速データ転送を実現する手法が提案されてきた．しかしながら，既存の手法は，一対一のクライアント・サーバ型のデータ転送を前提として開発されたものが多く，一対多のクライアント・サーバ型のデータ転送においては，利用者が並列データ転送を制御する仕組みを開発する必要があるため，まだ一般利用への敷居が高い技術となっている．本研究は，一対多のクライアント・サーバ型のデータ転送も対象に，UDT を用いた並列ファイル転送システムを提案する．国内および日米間の評価実験において，約 7Gbps のデータ I/O(ダウンロード) 性能を達成できることを確認した．

キーワード：分散ストレージシステム, ファイル転送, UDT, LFN

Parallel File Transfer using UDT

Abstract: High-speed data migration remains one of problem in the high-performance distributed computing. There are researches to improve end-to-end data I/O performance in a LFN(Long Fat Network), but most of them have been developed for a one-to-one data transfer. In fact, they are not the easy technology to use in the case of a one-to-many data transfer because users such as science researchers are forced to develop a function to control parallel data transfer. We propose the parallel data transfer system using UDT, and confirmed about 7Gbps data I/O performance (download) of the proposed system in the both local evaluation experiment and global evaluation experiment.

Keywords: distributed storage system, file transfer, UDT, LFN

1. はじめに

ネットワーク上に流通・蓄積されるデジタル情報の爆発的な増大によって，デジタルデータは飛躍的な成長を遂げており，科学研究分野においても，いわゆるビックデータの波は押し寄せている．科学研究の分野では，これまで理論を用いた研究手法，観測データによる研究手法，計算機シミュレーションによる研究手法が進められてきており，科学研究で扱うデータのほとんどがデジタル化され，データのサイズや種類が大規模化・多様化するまでに至っている．今後，ますます科学データは量・種類とも増え続け，解析できない程の量と種類の科学データが埋もれる懸念がある中，これからの研究手法として大規模データや複雑で多種多様なデータを解析するデータ指向型研究手法が提唱されている [1].

情報通信研究機構 (NICT) は，国内外の観測拠点で生成される観測データやスパコンで生成されるシミュレーションデータなどあらゆる科学データを蓄積し解析できる科学研究向けのクラウドシステム (NICT サイエンスクラウド) を構築している．NICT サイエンスクラウド [2] は，国内 4 地区にあるデータセンターに分散配置した計算機，広域分散ファイルシステムの Gfarm[3]，10Gbps の L2 高速バッ

[1] 情報通信研究機構
NICT, 4-2-1, Nukui-Kitamachi, Koganei, Tokyo 184–8795, Japan
[2] 株式会社日立ソリューションズ東日本
Hitachi Solutions East Japan, Ltd.
[3] 電気通信大学 大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
[4] 愛媛大学大学院医学系研究科
Medical School of Ehime University
[5] 筑波大学計算科学研究センター
Center for Computational Sciences University of Tsukuba

クボーンネットワーク網である JGN-X[*1] から成る大規模分散システムであり，2013 年 4 月現在で約 2 億ファイル (冗長化込み) を蓄積している．また，約 500 コア，約 2PB ディスク容量の分散並列処理環境によって観測データやシミュレーションデータをリアルタイムに可視化処理する仕組みも整備されており，既に一部の科学研究プロジェクトで利用されている [4][5].

一方，一般的なクラウドストレージサービスと同様に広域分散ストレージ環境のエンドツーエンド通信における高速なデータ移行 (ダウンロード/アップロード) が課題の一つとして残っている．TCP は，LFN(Long Fat Network) において高いスループットを維持し難いことが知られており [6]，この問題を解決するため TCP ストリームの並列化など TCP を改善する方式 [7][8]，GridFTP[9] や UDT[10] などアプリケーションレベルで高速データ転送を実現する手法が提案されてきた.

アプリケーションレベルの高速データ転送技術は，ユーザにネットワーク運用・管理を意識させないことを前提に開発されている．また，ユーザが定義可能な転送制御フレームワークも提供することから，科学研究者が利用する場合には，導入の容易性や制御の柔軟性の観点から有望なアプローチ手法と考えられる．しかしながら，既存の手法は，一対一のクライアント・サーバ型のデータ転送を前提として開発されたものが多く，NICT サイエンスクラウドが構築した広域分散ストレージ環境のような一対多のクライアント・サーバ型のデータ転送においては，利用者が並列データ転送を制御する仕組みを開発する必要があり，科学研究者が容易に利用するためには，まだ敷居が高い技術となっている.

本研究は，一対多のクライアント・サーバ型のデータ転送も対象に，簡易な並列データ転送制御機構を有する UDT を用いた並列ファイル転送システムを提案する．UDT (UDP-based Data Transfer) プロトコルは，高帯域ネットワークの長距離データ転送のために開発されたアプリケーションレベルのデータ転送プロトコルである．UDP によるデータのバルク転送と RTT(Round Trip Time) に依存しない独自のフロー制御や輻輳制御を提供し，LFN において TCP よりも高スループットで大容量データ転送が可能である [11]．提案システムは，利用者が要求したファイル群を所有するクライアント端末またはサーバ群に対して動的にデータ転送を制御するジョブスケジューラ機能と複数の UDT コネクションを確立しデータ転送する並列データ転送機能を提供する．これらの機能によって，一対多のクライアント・サーバ型のファイル転送における高速性と導入の容易性を担保する.

以下の論文構成について説明する．2 章では，提案する並列転送システムについて述べ，3 章で，基本性能の評価結果を考察する．最後に，4 章で本稿のまとめについて述べる.

[*1] http://www.jgn.nict.go.jp/

2. UDT を用いた並列ファイル転送

本研究は，大規模・多量のファイルを Gfarm で構築した広域分散ストレージシステムの複数のストレージサーバから一台のクライアント端末へ LFN を介して高速かつ効率的な並列ダウンロード，または一台のクライアント端末から複数のストレージサーバへ並列アップロードが可能なシステムの実現を目的とする．提案システムは，以下の機能を提供する.

ユーザインタフェース　保存先計算機，保存先ディレクトリ，保存ファイルパスの指定が可能で，大量のファイル転送要求を一括で処理できるインターフェース．リストで対象となるストレージサーバ数を指定することも可能．UDT で設定可能な UDT 送受信バッファ・UDP 送受信バッファ，通信時のウインドウサイズ，転送レートの上限を指定することも可能.

情報収集　保存先を指定するために参考となる保存先計算機のリソース状態や保存するデータの情報を収集する機能.

転送制御機能　ユーザインタフェースから入力されたパラメータをもとに転送を制御する機能．プライマリ・スレーブ型のデータ転送制御体制を提供.

ディスク間並列ファイル転送機能　転送元のディスクに保存されたファイル群を並列に転送し，転送先のディスクに保存する機能．通信は生成した子スレッドにより並列に行い，確立した SSH のコネクションを維持したまま連続的にファイルを転送し続けることが可能．次転送までの sleep 時間，各転送タイミングを制御するための時間を設定することも可能.

ファイル保存機能　指定したディスク領域に受信ファイルを保存する機能．ローカルディスク領域，Gfarm ディスク領域，RAM ディスク領域など選択することが可能.

ログ出力機能・表示機能　並列ファイル転送処理のディスク使用率，送受信 I/O 開始終了時間ならびに定期的に計測した CPU 使用率や送信レート等をログファイルに出力する機能．ログファイルの出力先ディレクトリの指定も可能.

図 1 にダウンロードの概要を，図 2 にアップロードの概要を示す．提案システムは，クライアント API と I/O モジュールで構成する．クライアント API は，クライアント端末上のユーザアプリケーションから手続き呼び出しの形式で呼ばれ，ストレージサーバに保存されるファイルのダウンロード，アップロードを行う．I/O モジュールは，全

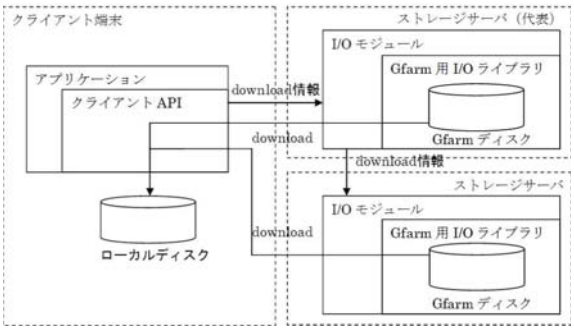

図 1 ダウンロード概要
Fig. 1 Download Image.

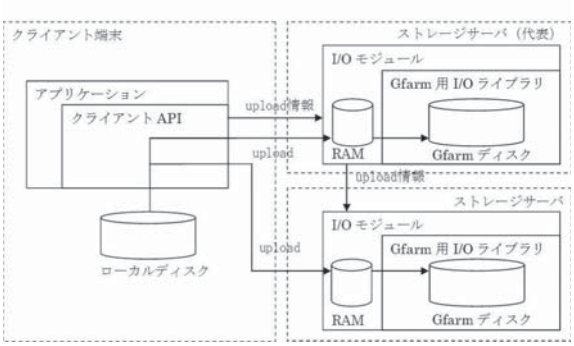

図 2 アップロード概要
Fig. 2 Upload Image.

てのストレージサーバに導入され，Gfarm 用の I/O ライブラリを利用して，Gfarm ディスク領域にファイルの読み書きを行う．提案システムは，一対多のクライアント・サーバ型に対応するため，クライアントが要求した情報が代表となるストレージサーバを介して全ストレージサーバに伝搬するプライマリ・スレーブ型の構造を採用している．クライアント API は，下記の処理を行う．

(1) 各ストレージサーバ上の I/O モジュールの起動
(2) UDT 送受信パラメータの設定
(3) 代表となるストレージサーバの I/O モジュールとの情報共有
(4) ストレージサーバの I/O モジュールからのデータ受信とローカルファイルへの書き込み (ダウンロード) またはローカルファイルのデータ読み込みとストレージサーバの I/O モジュールへのデータ送信 (アップロード)
(5) クライアント API 処理性能のログ出力
(6) 各ストレージサーバのステータスログ収集
(7) 各ストレージサーバ上の I/O モジュールの終了

また，ストレージサーバの I/O モジュールは下記の処理を行う．

(1) UDT 送受信パラメータの設定
(2) クライアント API との情報交換

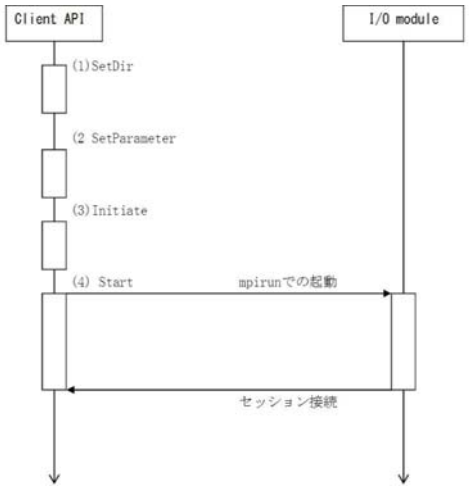

図 3 起動処理のフロー
Fig. 3 Flow of the boot process.

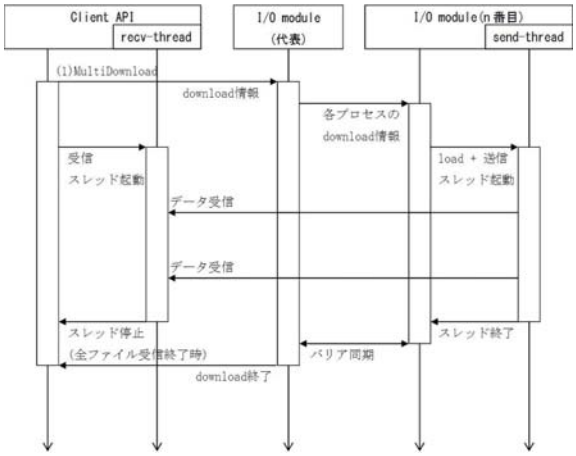

図 4 ダウンロード処理のフロー
Fig. 4 Flow of the download process.

(3) ストレージサーバのステータスログ提供
(4) 他のストレージサーバの I/O モジュールとの情報共有
(5) 出力先ディスク領域が指定されない場合の出力先ディスクの決定
(6) ディスクからのファイルの読み込みとクライアント API への送信 (ダウンロード) またはクライアント API からのデータ受信とディスクへのファイル書き込み (アップロード)
(7) I/O モジュール処理性能のログ提供

2.1 動作手順

2.1.1 起動処理

図 3 に起動処理のフロー図を示す．クライアント API は，SetDir 関数で指定されたパスをログ出力先として設定する．このパスは全てのストレージサーバの I/O モジュールに渡される．次に，SetParameter 関数で指定された送受信パラメータを UDT 通信のパラメータ値として保存する．この値も全てのストレージサーバの I/O モジュールに渡さ

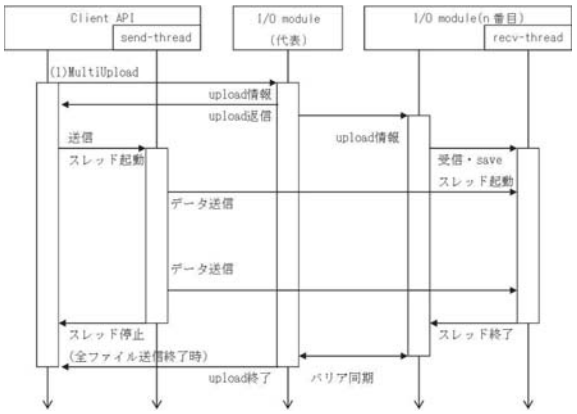

図 5 アップロード処理のフロー
Fig. 5 Flow of the upload process.

図 6 ステータス収集処理のフロー
Fig. 6 Flow of the status collection process.

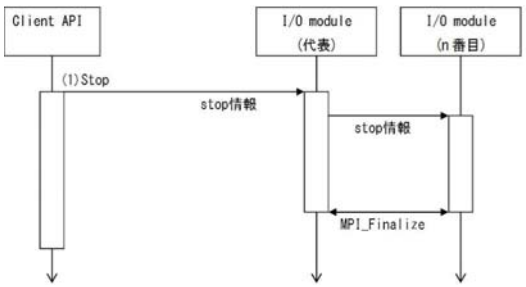

図 7 停止処理のフロー
Fig. 7 Flow of the stop process.

れ，クライアント API と I/O モジュールの両者に適用される．その後，Initiate 関数によって指定されたストレージサーバのホスト名 (または IP アドレス) を host ファイルに出力し，代表のストレージサーバに scp コマンドで host ファイルをコピーする．これにより，クライアント API と I/O モジュールの両者が情報を共有すべきストレージサーバ群を把握することができる．なお，一番目に指定されたホスト名がストレージサーバの代表となる．そして，Start 関数で各ストレージサーバと ssh で接続し，mpirun コマンドを使用してそれぞれの I/O モジュールの起動とセッション接続処理を実行する．この時，I/O モジュール起動時の引数としてクライアント端末の IP アドレス，ポート番号，送受信パラメータ，ログディレクトリパスの情報を与える．最後に，各 I/O モジュールからの通信接続要求に応じるため，UDT 通信のソケット作成，送受信パラメータ設定，bind，listen，accept を実行する．ストレージサーバの I/O モジュールは，起動後に引数を解析し，UDT 通信のソケット作成，送受信パラメータ設定，connect を実行し，クライアント API とのセッションを張る．

2.1.2 ダウンロード処理

図 4 にダウンロード処理のフロー図を示す．クライアント API は，MultiDownload 関数によって代表のストレージサーバの I/O モジュールへ”download”フラグ，ダウンロード対象のファイル名，ファイル数，ディレクトリパス名を渡す．その後，受信スレッドを起動しストレージサーバの I/O モジュールから送信されるデータを受信し，ローカルディスクへファイルを保存する．データ受信は，生成した子スレッドにより並列で行う．受信したファイル数が事前に把握していたファイル数と一致した場合，受信スレッドを停止させる．代表のストレージサーバの I/O モジュールは，gfwhere コマンドでダウンロード対象のファイルを送信するストレージサーバを決定する．そして，全 I/O モジュールに対し，gfwhere コマンドで決定したホスト名とファイル名を送信する．該当する I/O モジュールは，この内容に従ってファイルの読み込みとクライアント API へのデータ送信を行う．次転送までの sleep 時間，並列転送タイミング時間が指定されている場合は，この値に従って通信間隔を制御する．ファイルの送信終了後は待ち状態に遷移する．代表のストレージサーバの I/O モジュールは，全ての I/O モジュールとバリア同期を行い，ダウンロード処理の状況を管理する．全てのダウンロード処理が終了すれば，代表のストレージサーバの I/O モジュールからクライアント API へダウンロード処理の終了が通知される．

2.1.3 アップロード処理

図 5 にアップロード処理のフロー図を示す．クライアント API は，MultiUpload 関数によって代表のストレージサーバの I/O モジュールへ”upload”フラグ，アップロード対象のファイル名，ファイル数の情報を渡す．その後，代表ストレージサーバの I/O モジュールから提示される転送先ストレージサーバのステータス情報をもとにデータの実体を送信する．データ転送は，生成した子スレッドにより並列で行う．次転送までの sleep 時間，並列転送タイミング時間が指定されている場合は，この値に従って通信間隔を制御する．代表のストレージサーバの I/O モジュールは，クライアント API から取得した情報を該当のストレージサーバの I/O モジュールへ送信するとともに該当ストレージサーバのステータスログをクライアント API に提供する．該当する I/O モジュールは，代表のストレージサーバを介して取得したファイル数とファイル名の情報を参考に，起動処理で確立したコネクションを使用して自身に割り当てられたファイル数分のデータを受信する．その後，gfs_pio_write 関数で Gfarm ディスク領域にファイルを保存する．ファイルの保存終了後は待ち状態に遷移する．後は，ダウンロード処理と同様に代表のストレージサーバの

表 1　マシン仕様
Table 1　Machine specification.

	ストレージサーバ A(5 台)	ストレージサーバ B(8 台)	クライアント端末
メモリ	144GB(DDR3)	16GB(DDR2)	144GB(DDR3)
CPU	Intel Xeon(R) CPU X5550 @ 2.67GHz	AMD Opteron 2350 @1GHz	Intel Xeon(R) CPU X5550 @ 2.67GHz
コア数	16core	4core	16core
NIC	10GbE	1GbE	10GbE
HDD 規格	SATA3 6TB(RAID5)	SATA3 2TB	SATA3 6TB(RAID5)
read 速度	685MB/sec	105MB/sec	685MB/sec
write 速度	103MB/sec	84MB/sec	103MB/sec
OS	OpenSUSE11.1(x86_64)	OpenSUSE11.1(x86_64)	OpenSUSE11.1(x86_64)

I/O モジュールが全てのダウンロード処理の終了を確認した後，クライアント API へダウンロード処理の終了を通知する．なお，今回はラウンドロビン方式で次の保存先ストレージサーバを選択することとした．動的なスケジューラは今後の課題である．

2.1.4　ステータス収集処理

図 6 にステータス収集処理のフロー図を示す．クライアント API は，Status 関数によって代表のストレージサーバの I/O モジュールに”status”フラグを渡し，代表のストレージサーバから各ストレージサーバのステータス情報を取得する．代表ストレージサーバの I/O モジュールは，”status”フラグを受け取った場合，”gfhost -l”や”gfdf”を実行し，出力結果を文字列でクライアント API に渡す．その後，クライアント API の次の処理に応じるため要求待ち状態に遷移する．代表ストレージサーバ以外の I/O モジュールにおいては，状態遷移や処理は発生しない．

2.1.5　停止処理

図 7 に停止処理のフロー図を示す．クライアント API は，Stop 関数によって代表のストレージサーバの I/O モジュールに”stop”フラグを渡す．代表ストレージサーバの I/O モジュールは，”stop”フラグを全 I/O モジュールに対して送信する．”stop”フラグを受け取ったストレージサーバの I/O モジュールは，繰り返し処理を行うメインループを抜け，MPI 終了処理後にプロセスを終了する．

3.　基本性能評価

提案システムの基本性能として，国内評価実験環境で UDT を用いた並列転送のデータ I/O を測定した．また，2012 年 11 月 10 日-16 日に米国のソルトレイクシティで開催された SuperComputer2012(SC12)[*2]において，日米間の実ネットワーク環境を使用して転送実験を実施した．なお，本稿では紙面の都合上，並列ダウンロードのみの基本性能を報告する．1 台のクライアント端末と 2 種類のストレージサーバを計 13 台用意し評価を行った．表 1 に利用したマシン仕様を示す．国内評価実験環境では，クライアント端末とストレージサーバ群の間には，遅延 Box を介入させ 150msec の RTT を設定した．日米間評価実験環境は，ソルトレイクの SC 会場にクライアント端末を設置し，NICT 小金井にある 13 台のストレージサーバを JGN-X，Pacific Wave[*3], Internet2[*4]経由の 10GbpsVLAN ネットワークで接続した．また，UDP による iperf 通信にて，10Gbps に近い通信帯域とパケットロスがないことも確認済みである．東京・ソルトレイクの SC 会場までの実際の RTT は 143msec であった．なお，日米間のネットワーク実験環境はタイムスケジュールで利用することになっており，利用時間内においては，基本的に他の通信フローはほとんど無い状態である．今回，ダウンロード性能はダウンロードしたデータサイズの合計をクライアント API のダウンロード開始時刻と終了時刻の時間差で割った値とした．また，全ストレージサーバには 125MB のダミーファイルを 512 個配置し，全ファイルが転送されるまで繰り返し転送を行った．最後に，送受信パラメータを表 2 に示す．

表 2　送受信パラメータ
Table 2　Parameter of sender/receiver.

項目		設定値
ストレージサーバ	UDT_MSS	1,500
	UDT_SNDBUF	159,999,040
	UDP_SNDBUF	10,000,000
	UDT_MAXBW	100,000,000
	UDT_RCVBUF	37,683,200
	UDP_RCVBUF	160,000,000
クライアント端末	UDT_MSS	9,000
	UDT_SNDBUF	975,211,540
	UDP_SNDBUF	1,000,000
	UDT_MAXBW	100,000,000
	UDT_RCVBUF	229,683,200
	UDP_RCVBUF	160,000,000

[*2] http://sc12.supercomputing.org/

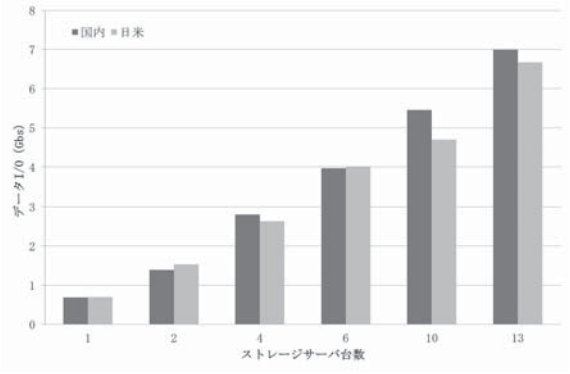

図 8　並列ダウンロード性能
Fig. 8　The performance of parallel download.

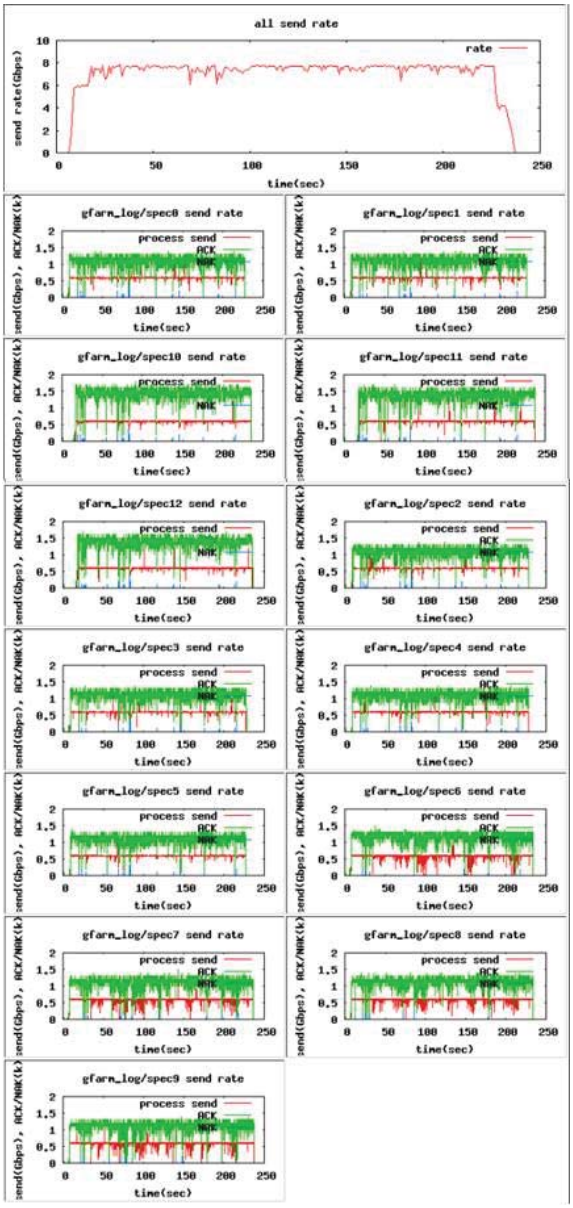

図 9　国内評価実験における 13 並列ダウンロードの合計データ I/O(上段) と内訳結果
Fig. 9　The result of 13 parallel download in a local evaluation experiment.

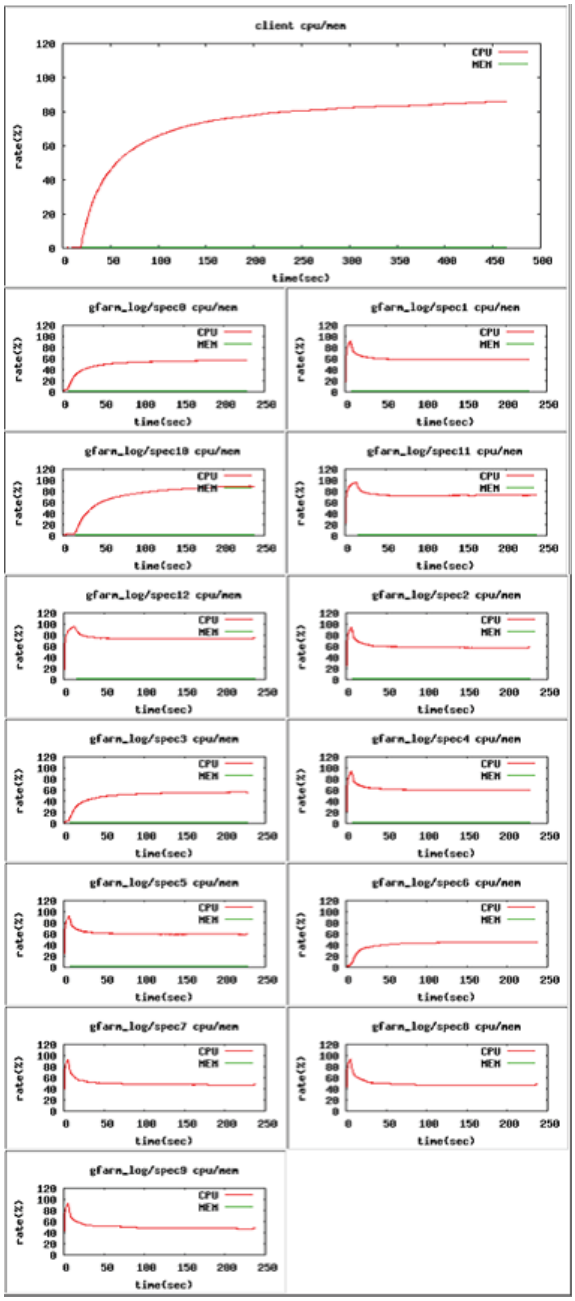

図 10　国内評価実験における 13 並列ダウンロード時のクライアント端末のステータス (上段) と各ストレージサーバのステータス結果
Fig. 10　The resource status result of 13 parallel download in a local evaluation experiment.

3.1　結果と考察

並列数に応じたダウンロード性能結果を図 8 に示し，国内評価実験における最大のデータ I/O 性能を達成した 13 並列ダウンロード時の結果を図 9 に，各計算機の CPU・メモリ使用率の結果を図 10 に示す．また，日米間評価実験における 13 並列ダウンロード時の結果を図 11 に，各計算機の CPU・メモリ使用率の結果を図 12 に示す．各数値

*3　http://www.pnwgp.net/services/pacific-wave-peering-exchange/
*4　http://www.internet2.edu/ion/

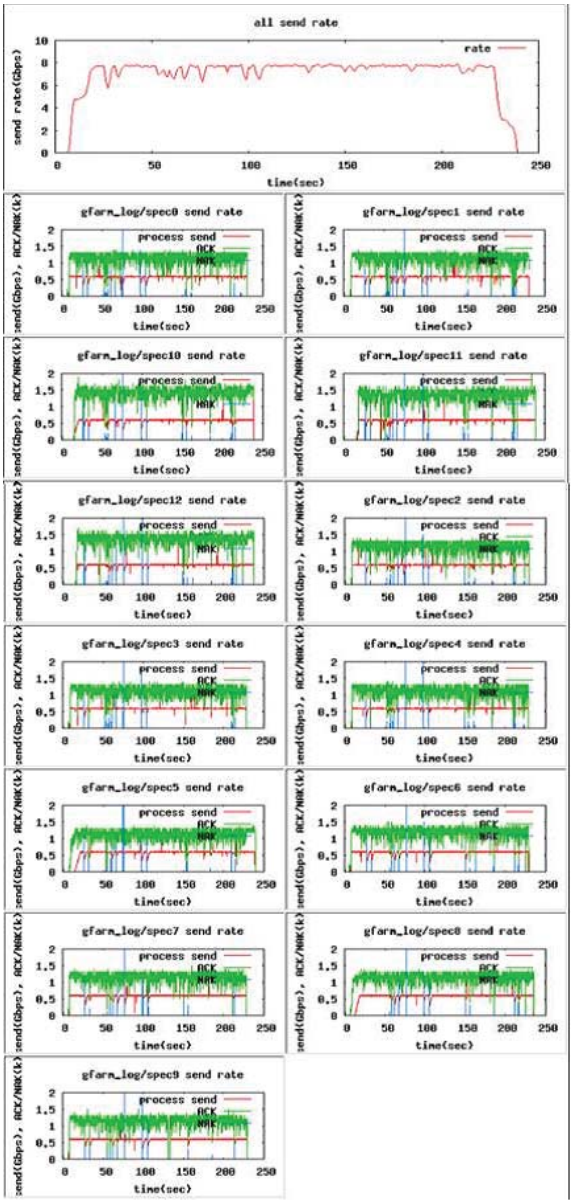

図 11 日米間評価実験における 13 並列ダウンロード合計データ I/O(上段) と内訳結果

Fig. 11 The result of 13 parallel download in a global evaluation experiment.

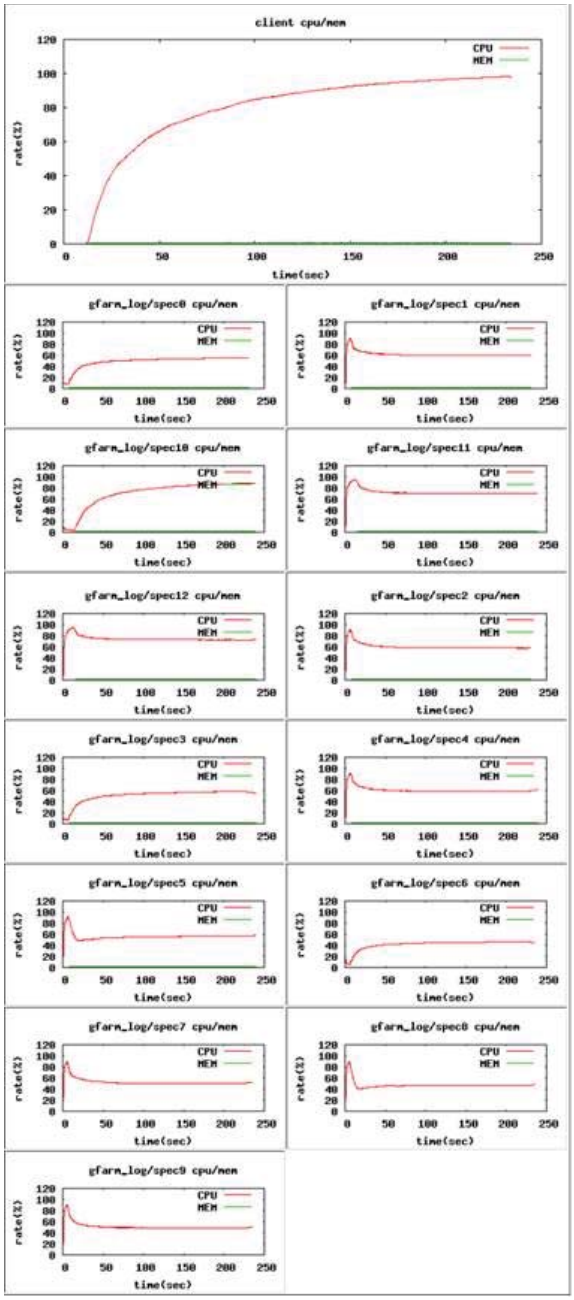

図 12 日米評価実験における 13 並列ダウンロード時のクライアント端末のステータス (上段) と各ストレージサーバのステータス結果

Fig. 12 The resource status result of 13 parallel download in a global evaluation experiment.

は，3 回測定した平均値である．

図 8 より，国内評価実験ならびに日米間評価実験両方とも並列数に応じてデータ I/O 性能は高くなっていることがわかり，ほぼ同様な傾向が見られる．図 9 と図 11 に示す通り，最大のデータ I/O は 13 並列データ転送の時であり，国内評価実験では 6.992Gbps，日米間評価実験では 6.672Gbps であった．各ストレージサーバで得られた転送レートは約 0.5Gbps であり，台数分のデータ I/O を加算した結果がトータルのデータ I/O 性能になっていることがわかる．また，図 10 と図 12 においても，クライアント端末およびストレージサーバ群の処理能力が限界でないことが読み取れる．

今回の評価実験では，マルチプロセス・マルチスレッドを組み合わせた並列ファイル転送を実施した．マルチスレッドの並列転送では，1 セッションの転送レートの低下が他のセッションにも影響を与え全体の転送レートを下げてしまう．今回使用したクライアント端末では，1 プロセスあたり 4 スレッド以上の並列転送をした場合，2.8Gbps の性能より上がらなくなる現象が発生した．すなわち，今回使用したクライアント端末の仕様では 4 スレッドが限界だったということが考えられる．そこで，13 並列のダウンロードの評価実験において，13 の通信セッションを 4 つ

のプロセス (1 プロセスあたり 3 または 4 スレッド) で処理した．マルチプロセス化の併用は，加算的なデータ I/O に対して各スレッドが与える性能低下の影響範囲を最小限にできる．したがって，提案システムで実現した約 7Gbps のファイルダウンロード性能はマルチプロセス・マルチスレッドの併用によって達成できたと考えられる．一方，計算機毎に立ち上がり時の CPU 使用率が低いものと高いものがはっきりした．立ち上がり時の CPU 使用率が低い計算機は，送信レートの立ち上がりも悪い結果となっており，今後改善すべきポイントと考える．また，アプリケーションへの導入を考慮した場合の課題も顕著化した．スレッド並列は逐次アプリまたはマルチプロセスアプリのどちらに対しても導入の容易性があるが，3Gbps 以下の転送性能となる．プロセス並列は 10Gbps に近い高速転送が可能になるが，専用のプログラムが必要のため，特に逐次アプリでの利用は複雑になる．今後，性能と汎用性をいかに共存できるかが重要な課題と考える．

4. おわりに

本稿では，一対多のクライアント・サーバ型のデータ転送モデルを対象に，UDT を用いた並列ファイル転送システムを提案した．提案システムは，マルチプロセス・マルチスレッドによる並列データ転送が可能で，プライマリ・スレーブ構成の簡易並列データ転送制御機構も備える．加えて，逐次アプリおよびマルチプロセスアプリのいずれにも対応可能である．国内および日米間の評価実験では，1 台のクライアント端末から 13 台のストレージサーバに保存されたファイル群をマルチプロセス・マルチスレッドで並列ダウンロードすることで，約 7Gbps のデータ I/O を達成できることを示した．今後の課題としては，立ち上がりの CPU 使用率の改善と動的なデータ転送制御機構の整備を実施し，汎用性に関する改良も行う．また，詳細な性能評価を行い，実用化を目指す．

謝辞

本研究は，NICT サイエンスクラウド上の計算機リソースを用いて実施しています．ここに記して謝意を表します．

参考文献

[1] Tony Hey, Stewart Tansley, and Kristin Tolle: *The Fourth Paradigm: Data-Intensive Scientific Discovery*, 入手先 ⟨http://research.microsoft.com/en-us/collaboration/fourthparadigm/⟩, (2009).

[2] Ken T. Murata, S. Watari, T. Nagatsuma, M. Kunitake, H. Watanabe, K. Yamamoto, Y. Kubota, H.Kato, T. Tsugawa, K. Ukawa, K. Muranaga, E. Kimura, O. Tatebe, K. Fukazawa and Y. Murayama: *A Science Cloud for Data Intensive Sciences*, Data Science Journal, Vol. 12, pp.139-146, (2013).

[3] O. Tatebe, K. Hiraga, N. Soda: *Gfarm Grid File System*, New Generation Computing,Ohmsha,Ltd.and Springer,Vol.28,No.3,pp.257-275, (2010).

[4] T. Tsugawa, A. Saito, Y. Otsuka, M. Nishioka, T. Maruyama, H. Kato, T. Nagatsuma, and K. T. Murata: *Ionospheric disturbances detected by GPS total electron content observation after the 2011 off the Pacific coast of Tohoku Earthquake*, Earth Planets Space, Vol.63, pp.875-879, (2011).

[5] Y. Kubota, K. Yamamoto, K. Fukazawa, and Ken T Murata: *Visualization of the Flux Rope Generation Process Using Large Quantities of MHD Simulation Data*, Data Science Journal, Vol. 12, pp.134-138, (2013).

[6] S. Floyd: *HighSpeed TCP for Large Congestion Windows*, RFC 3649, (2003).

[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*, the 16th International Parallel and Distributed Processing Symposium, pp. 314-323, (2002).

[8] Z. Zhang, G. Hasegawa, and M. Murata: *Analysis Evaluation of Parallel TCP : Is it really effective for Long Fat Networks?*, IEICE Transactions on Communications Vol.E90-B, No.3. 559-568, (2007).

[9] W. Allcock: *GridFTP: Protocol extensions to FTP for the Grid*, GGF Document Series GFD.20, 入手先 ⟨http://www.gridforum.org/GFD.20.pdf⟩ (2003).

[10] Y. Gu, and R L. Grossman: *UDT: UDP-based Data Transfer for High-Speed Wide Area Network*, The International Journal of Computer and Telecommunications Networking, Vol.51, Issue 7, pp.1777-1799, (2007).

[11] K. Kumazoe, K. Kouyama, Y. Hori,M. Tsuru, and Y. Oie: *Can high-speed transport protocols be deployed on the Internet?:Evaluation through experiments on JGNII*, PFLDnet 06, (2006).