[解説記事]

大規模科学計算システムにおける高速ファイル転送

後藤英昭[1]　大泉健治[2]　高橋洋一[2]　花岡勝太郎[2]
吉田 智[3]　岡崎昌夫[3]　山形正明[3]　菅雄一郎[3]　金野浩伸[3]
[1] 東北大学サイバーサイエンスセンター スーパーコンピューティング研究部
[2] 東北大学情報部 情報基盤課 共同研究支援係
[3] 日本電気株式会社 文教・科学ソリューション事業部

1　はじめに

サイバーサイエンスセンター (旧称・情報シナジーセンター) 大規模科学計算システムのスーパーコンピュータシステムは，5 年ごとに更新されてきました。平成 20 年 3 月に稼働を開始した現行の SX-9 システム [1] の総合演算性能は 26.2TFLOPS で，前システムの SX-7 (2.1TFLOPS) と比較すると 12 倍以上，前々代からは 100 倍以上になります。機種更新における計算機利用の動向で興味深いことは，演算性能が上がると，シミュレーションの精度が上げられたり，投入されるジョブが増加したりすることにより，ほどなくして演算資源の限界まで使われるようになることです。

演算量の増加は，扱うデータのサイズにも反映されます。ファイルサーバの使用量は，平成 18 年 3 月に 1.5TB 前後であったものが，平成 20 年 3 月には 3TB，本年 3 月には 17TB 前後まで増加しました。利用者数の増減はほとんどないため，利用者あたりのファイルサイズが増加していることになります。大規模科学計算システムにおいて，小さなデータを入力して巨大なデータが得られたとして，そのデータを同じシステムの上で解析して，グラフ化できる程度のコンパクトなデータに変換できるならば，最終的には小さなファイルの転送だけで済むでしょう。しかし，計算結果によっては巨大なデータを研究室に持ち帰ったり，他大学のシステムにコピーしたりといった利用形態も生じます。

現在，100Mbps 以下のネットワークで接続されている研究室が多いでしょうから，仮に 90%のデータ転送能力が得られたとしても，100GB のファイルの転送に 2.5 時間以上かかることになります。これを十分速いと見るか，遅いと見るかですが，近年の PC の著しい高性能化や，今後の計算機の性能向上，Gigabit Ethernet (GbE) の普及，グリッドの台頭などを考えると，ずいぶんと遅い物に見えてきます。高速なデータ転送ができるようになれば，今までより自由に，すばやいデータ処理が可能になり，計算機の利用形態の自由度も増すことでしょう。

本稿では，GbE の利用を前提として，特殊なネットワーク機器を用意することなく，高速なファイル転送を実現する方法を探ってみます。現在の大規模科学計算システムは GbE で学内ネッ

トワーク TAINS に接続されています。学内では建物まで GbE が通っている所が多いため，館内ルータから研究室まで GbE を引くことができれば，それだけでもある程度の高速化は達成できるでしょう。

一方，ネットワークの帯域幅の改善ばかりではなく，ファイル転送に使うソフトウェアも適切に選ばなければ，高速なデータ転送は望めません。現在，当センターでは Secure Shell (scp, sftp コマンド等) によるファイル転送のみがサポートされています。しかしながら，Secure Shell にはデータの暗号化によるオーバーヘッドがあるので，そのデータ転送能力は一般に FTP に劣ると言われています。本稿では，様々な条件とツールでファイル転送速度を測定し，実際にどれぐらいの性能が得られるのかを見てみることにします。

2　使用機材と実験環境

大規模科学計算システムでは，スーパーコンピュータ SX-9 に直接ではなく，並列コンピュータ gen に対してファイル転送を行います。gen のハードウェアは NEC TX7/i9610 で，8 コア分の Intel Itanium2 (1.6GHz) が搭載されています。OS には SUSE LINUX Enterprise Server 9 が使われています。

センター内にあるスーパーコンピューティング研究部の研究室に，測定用の PC を二台用意しました。使用機は Lenovo ThinkCentre A61e Ultra Small で，いずれも AMD Athlon X2 BE-2350 (Dual-core, 2.1GHz) が搭載されています。ネットワークのコントローラは Broadcom BCM5786 です。openSUSE Linux 11.1 を導入し，ファイルサーバとして動かしている方を，以下では PC2 と呼びます。もう一台はクライアント PC として，以降 PC1 と呼びます。PC1 には openSUSE Linux 11.1 と，比較用に Windows XP (SP3) を導入しました。

研究室のネットワークは，センター建物のルータに GbE で接続されています。この研究室は大規模科学計算システムのファイアウォールより外側にあり，学内ネットワーク TAINS の幹線は通りませんが，TAINS 経由の接続に近くなっています。

測定に用いるファイルとして，手元にあった Tru64 UNIX (Alpha プロセッサ) 用の netscape の実行形式 (約 17.7MB) と，それを 100MB になるまで繰り返して継ぎ足したものの，二種類を用意しました。

3　ファイル転送速度の評価

3.1　使用ツールとサーバプログラム

FTP によるファイル転送は，ID とパスワード，データがすべて平文で (暗号化なしに) ネットワークを流れることから，セキュリティ上の問題があるとされています。しかしながら，FTP はデータ転送効率に優れたプロトコルなので，盗聴の危険性のないローカルなネットワークなどでは現在でもしばしば使われています。GbE によるファイル転送能力の限界を探るために，まずは PC

対 PC (GbE L2 スイッチ経由) と PC 対 gen で FTP による速度測定を行ってみることにします。

今回の測定では，Linux に標準で付属している ftp コマンドを使用しました。サーバ用のプログラムには，これも OS 付属の pure-ftpd を用いました。

SSL による暗号化をサポートし，従来の ftpd と上位互換を有する，vsftpd というサーバプログラムがあります [2]。これに lftp [3] などのクライアントプログラムを組み合わせると，ID とパスワードを暗号化し，必要に応じてデータも暗号化して送ることが可能です。ユーザ認証にかかる時間はわずかなので，データを暗号化しないようにすれば，FTP と同程度の効率になることが期待されます。暗号化がない分，セキュリティ面の不安が残りますが，大規模科学計算における巨大なデータが盗聴されたとして，意味のある情報として悪用される可能性を考えれば，それほどの驚異ではないかもしれません。問題があるとすれば，プログラムコードなど秘密にしたいものを，いつもの操作でうっかりとネットワークに流してしまうなどの操作ミスが考えられます。今回の性能評価ではサーバ側 (gen, PC2) に vsftpd を用い，クライアントの lftp コマンドを用いてファイルを get します。

インターネット越しのファイル転送では，旧来の FTP に代えて，Secure Shell の sftp や scp が現在よく使われています。Linux で動く Secure Shell には幾つかの異なる実装がありますが，SUSE Linux をはじめとするほとんどのディストリビューションで採用されている，OpenSSH を使いました。Windows 上では，これも有名な WinSCP を用いました。

Secure Shell ではデータの暗号化が常に行われるため，この処理による速度低下があります。しかし，最近の高速なプロセッサを搭載したコンピュータでは，Fast Ethernet (100Mbps) が簡単に飽和してしまいます。GbE では FTP に遠く及ばない速度になることが多いようです。また，scp では Round Trip Time (RTT) の大きな遠隔地の通信で極端に効率が落ちることが知られており，通信に用いるバッファのサイズを自動最適化することで速度低下を抑える，“High Performance SSH/SCP – HPN-SSH” と呼ばれる実装が開発されています [5]。HPN-SSH は米ピッツバーグスーパーコンピューティングセンターによって開発・公開されているもので，OpenSSH に対するパッチとして提供されています。HPN-SSH の論文 [6] によると，グリッドコンピューティングのミドルウェアの他，多くの研究機関で利用実績があり，HP-UX 等の OS にも組み込まれているそうです。

HPN-SSH には，データを暗号化せずに転送するオプションもあります。これを利用すれば，ユーザ認証の部分は保護しつつ，FTP に迫る効率が得られる可能性があります。

3.2 PC-PC 間の速度評価

はじめに，GbE と使用機材の限界を見るために，PC を一台の GbE スイッチで結んだ単純なネットワーク構成で，FTP によるファイル転送速度を測定しました。結果を表 1 に示します。

ftp については，コマンドで表示された速度を記入してあります。lftp コマンドには速度表示がないので，time コマンドを用いて経過時間を測りました。データ転送部分のみの時間を測りたいのですが，その手段がないので，仕方なく上のような方法をとっています。ユーザ認証の時間が

表 1: ファイル転送速度の比較 – PC 対 PC

プログラム			平均速度	
PC1	方向	PC2	サイズ 17.7MB	サイズ 100MB
ftp	←	pure-ftpd	112MB/s	110MB/s
lftp	←	pure-ftpd	73.7MB/s	104MB/s
lftp	←	vsftpd	95.6MB/s	109MB/s
scp	←	sshd	30.5MB/s	41.5MB/s
hpn-scp	←	hpn-sshd	30.6MB/s	41.5MB/s
hpn-scp-nocipher	←	hpn-sshd	50.5MB/s	83.3MB/s

オーバーヘッドとして含まれるので，サイズの小さいファイルでこの影響が相対的に大きくなり，平均速度が低めに出ています。ホームディレクトリの .netrc に ID とパスワードを記入することで，自動ログインさせています。

ftp, lftp ともに，一回目はファイルを読み出すのにディスクアクセスが発生するので，この影響を避けるために，OS のディスクキャッシュにファイルが入った二回目以降で，安定したと思われる時間を採用しました。書き込み側のディスクの影響を抑えるために，転送されたデータはデータシンク (/dev/null) に捨てています。実際にファイルに書き出した場合は，ハードディスクとキャッシュの速度に律速されます。

FTP 系のいずれの組み合わせでも 100MB/s を越える転送速度が出ていて, 換算すると 800Mbps にもなり，GbE としては相当に高速なものと思われます。数年前に，高級なネットワークカードでも 700Mbps 止まりだったり，300Mbps も出ないような廉価なネットワークコントローラをよく目にしていたので，今回の測定結果には少々驚きました。

続いて，Linux に標準装備の scp と sshd (いずれも OpenSSH-5.1p1 より)，HPN-SSH (最適化処理あり)，および HPN-SSH (データ暗号化無し) を用いてファイル転送速度を測りました。HPN-SSH は OpenSSH-5.1p1-hpn13 v5 を prefix 以外はデフォルトのオプションのままでコンパイルしたものを使用しました。scp が表示する速度がおかしいようなので，先の lftp と同様に time コマンドを使って，ユーザ認証も含む経過時間を測定しています。RSA 公開鍵暗号方式を用いて自動ログインしています。

標準の scp/sshd でも 41.5MB/s (332Mbps) の速度が得られており，Fast Ethernet ではボトルネックになることがわかります。RTT が小さいため，HPN-SSH の最適化だけでは改善効果がありません。

HPN-SSH の scp にオプション `-oNoneSwitch=yes -oNoneEnabled=yes` を付けて起動し，暗号化を無効にした状態でデータ転送時間を測ってみました。FTP 系には及びませんが，83.3MB/s (666Mbps) という高速度が得られました。

表 2: ファイル転送速度の比較 – PC 対 gen

プログラム			平均速度	
PC1	方向	gen	サイズ 17.7MB	サイズ 100MB
lftp	←	vsftpd	32.8MB/s	40.0MB/s
scp	←	sshd	23.0MB/s	31.4MB/s
hpn-scp-nocipher	←	hpn-sshd	44.2MB/s	73.0MB/s
vsftpd	→	ftp	91.7MB/s	93.5MB/s
sshd	→	scp	22.1MB/s	28.2MB/s

3.3 PC-gen 間の速度評価

大規模科学計算システムの利用を想定して，PC と並列コンピュータ (gen) の間でファイル転送速度を測定しました。結果を表 2 に示します。

セキュリティ上の理由により，gen では通常の FTP のサービスは停止しています。今回は評価のために vsftpd を試験的に動かしました。ID とパスワードは必ず暗号化するような設定にしました。

lftp でデータを平文で転送しているにも関わらず，40MB/s で頭打ちになりました。期待したよりも低い値ですが，これでも Fast Ethernet の帯域幅をはるかに越えます。

HPN-SSH の暗号化無しでは 73MB/s という速度が得られました。GbE の実効的な速度としては，これでもかなり速い方です。PC のローカルのディスクに 100MB のファイルを書き込んだ場合は，ディスクドライブに律速されて 67MB/s となりました。

反対方向のファイル転送についても詳しく調べたかったのですが，gen に lftp が用意できなかったので，通常の ftp の結果だけを示します。ファイルを /dev/null に捨てる方法で 93.5MB/s，gen のローカルディスク (/tmp) に書き出す状態で 87MB/s という高い速度が得られました。ただし，ID とパスワードが暗号化されないこと，ユーザ側にサーバを立ち上げなければならないという点で，この手法はお奨めできません。

3.4 WinSCP によるファイル転送

Windows 上で利用できる scp/sftp クライアントとして，WinSCP が有名です。グラフィカルな表示で大変便利なものですが，使っていて「遅いんじゃないか?」と思ったことはないでしょうか。今回，WinSCP 4.1.8 (ビルド 415) を使って PC-PC 間のファイル転送速度を測ってみたので，紹介します。

WinSCP は，デフォルトの状態では sftp モードでファイル転送を行います (Secure Shell には 2 つのモードがあります)。オプションで scp モードに切り替えることが可能です。平均の速度は

表示されないので，ファイル転送中に画面に表示される値がピークで安定したところを見計らって，その時の値を拾いました。二つのファイルでほぼ同じ値になるので，100MB の方だけ示すと，sftp モードで 1560KB/s，scp モードで 2850KB/s となりました。受信側の PC には書込速度が公称値 90MB/s(最大) の SSD (Solid State Drive) が搭載されており，ドライブに律速されているわけではありません。念のために受信側の PC を AthlonX2 5200+ (Dual-core, 2.6GHz), Windows Vista の組み合わせに変更してみましたが，scp モードで 3200KB/s 程度になりました。Linux の scp と比べて一桁も遅いことがわかります。実装上の問題かもしれません。

以上のように，大容量のファイルを転送するのに WinSCP は不向きなことがわかりました。

4　おわりに

大規模科学計算システムの並列コンピュータ gen を対象に，GbE の普及を見越して様々なファイル転送方式の速度調査を行いました。近年の高速な計算機では Secure Shell の scp でも Fast Ethernet が容易に飽和することから，研究室まで GbE を引く価値はあると言えるでしょう。

今回は RTT の大きい遠隔地に GbE の環境を用意できなかったので，特に HPN-SSH の評価はほとんどできませんでした。学内の別キャンパスや，国内遠隔地での評価は，今後の課題にしたいと思います。

vsftpd や HPN-SSH を使えば，ID とパスワードは保護しつつ，暗号化無しの高速なデータ転送が可能です。データ暗号化無しの高速ファイル転送の需要があれば，これらのサービスを提供する価値がありそうです。本稿では残念ながら，読者がすぐに試すことができる方法をほとんど示すことができませんでした。今後，利用者の意見を聞きながら，ネットワーク環境の改善につなげていければ良いと考えています。

参考文献

[1] 情報基盤課システム管理係, サイバーサイエンスセンタースーパーコンピューティング研究部, “スーパーコンピュータシステム SX-9 利用ガイド,” 東北大学サイバーサイエンスセンター大規模科学計算機システム広報 SENAC Vol.41, No.2, pp.19–33 (2008).

[2] vsftpd - Secure, fast FTP server for UNIX-like systems : `http://vsftpd.beasts.org/`

[3] LFTP - sophisticated file transfer program : `http://lftp.yar.ru/`

[4] WinSCP - Free SFTP Client, Secure File Transfer Protocol, Secure FTP : `http://winscp.net/eng/index.php`

[5] High Performance SSH/SCP - HPN-SSH : `http://www.psc.edu/networking/projects/hpn-ssh/`

[6] Chris Rapier, Benjamin Bennett, “High Speed Bulk Data Transfer Using the SSH Protocol,” Mardi Gras Conference 2008, ACM.