Vol.2011-HPC-129 No.25
2011/3/16

インタークラウド間での大規模データ転送の高速化

千葉立寛[†1,†2] 松岡聡[†1,†3]

大規模データを用いるデータインテンシブアプリケーションの実行環境としてクラウドが近年注目されている．クラウドを用いることでデータ処理を行うノードを問題サイズに応じて必要な分だけ動的にスケールアウト/インすることが可能となる一方で，処理対象となるデータがクラウドストレージ上へ効率よく転送されない場合，現実的には転送スループットによって性能が律速されてしまう．また，現状多数のクラウドが存在する中，異なるクラウドでの実行やデータプリケーションのためにインタークラウド間でのデータマイグレーションを高速に行うことはデータ量が多くなるにつれてますます重要となる．本稿では，現在利用の盛んな 2 つのクラウド環境，Amazon EC2/S3，および，Windows Azure を対象として，2 つのクラウド間でのデータ転送を高速化する手法について述べる．複数の中継ノードを用意し，それらが協調的に対象となる大規模データを分割して転送を行う．特に，Windows Azure Blob に保存されているデータを Amazon S3 へマイグレーションする場合について実験を行い，複数の中継ノードを用いることで転送スループットが向上することを確認した．

High Performance Large Data Transfer for Inter-Clouds

Tatsuhiro Chiba[†1,†2] and Satoshi Matsuoka[†1,†3]

Recently cloud environments are expected to execute data-intensive applications they use huge data sets. It is possible to dynamically add and remove computing nodes depending on the size of problem, however, application performance will be limited by data transfer throughput, in case the data does not transfer efficiently. Moreover there are many available clouds, high-throughput data staging for migration between inter-clouds is becoming more important as the data size grow. In this paper we address a technique to boost data transfer throughput between inter-couds, especially for Amazon EC2/S3 and Windows Azure Platform they are generally used. We prepare multiple relay nodes and utilize them cooperatively by splitting data and using multiple transfer connections. Especially we focus on from Azure Blob to Amazon S3, and we confirm multiple connections scheme acheive higher data migration throughput.

1. はじめに

近年，Amazon Web Services(AWS)[1] や Windows Azure[2] などのクラウドが数多く登場し，様々な分野でその利用が広がっており，従来スーパコンピュータや大規模クラスタで行われてきた HPC アプリケーションにおいてもクラウドで実行することへのニーズが高まってきている．特に，BLAST[3] のような大規模データに対してデータ並列的に計算ノードをスケールアウトさせて処理を行うデータインテンシブアプリケーションの分野でクラウドの利用が期待されており，実際に Windows Azure 環境上での BLAST サービスを展開する例も登場している．[4]

データインテンシブアプリケーションでの処理対象となるデータサイズは莫大で，ATLAS プロジェクト[5] では一年間に数ペタバイトの実験データが生成され，それらを階層的に構成されたデータグリッド上にレプリケーションしている．このような大量に生成されるデータをクラウドストレージ上へ転送する場合，データ転送に要するコストが非常に大きくなってしまう．例えば，10 Mbps 程度の転送スループットで 1 TB のデータを転送するには約 9.25 日もかかり，データサイズの増加に対してできるだけ高速かつ効率的な転送手法を開発することは今後ますます重要となる．論文[6] では，実際に Amazon EC2 や Azure SQL 上に 100 GB 程度の天文データベース (SDSS) を展開した実例を紹介している．また，単純にクラウドへデータを移行をするだけではなく，クラウド上に保存されているデータを取り出して異なるクラウドにレプリケーションして計算を実行させるような場合においても，インタークラウド間でのデータ転送の高速化は非常に重要である．論文[7] でも述べられている通り，クラウドに実行環境を移行すべきかどうかの判断材料の一つとしてデータ移動のコストが挙げられている．ネットワークに掛かるコストは，CPU やストレージにかかるコストに比べてはるかに大きく，データ移動に対する金銭的時間的コストを出来るだけ最小化することがクラウド利用に向けての重要な課題の一つである．

本稿では，現在利用の盛んな 2 つのクラウド環境である Amazon EC2/S3 と Windows Azure を対象として，2 つのクラウド間でのデータ転送を高速化する手法について述べる．

†1 東京工業大学
Tokyo Institute of Technology
†2 日本学術振興会特別研究員 DC
Research Fellow of the Japan Society for the Promotion of Science
†3 国立情報学研究所
National Institute of Infomatics

特徴としては，複数の中継ノードを用意しそれが協調的に転送対象となる大規模データを分割して複数コネクションを用いて転送を行う．特に，Azure Blob に保存されたデータを Amazon S3 に転送する場合についての実験を行った．理論的には中継ノード数を増やせば転送スループットは向上すると考えられるが，実際に転送する際にはクラウドストレージ，クラウド間を結ぶインターネット，ノードのバンド幅，スイッチ，さらにはクラウドシステムでのバンド幅利用制限など，様々なレイヤにおいてボトルネックリンクが存在する可能性がある．本稿では，Windows Azure 環境での Blob ダウンロード性能を計測し，これらの状況を踏まえて実際に転送を行った際にどのレイヤにボトルネックが存在し，どの程度の転送スループットが出すことが可能で，現状，どの程度のデータサイズまでの転送であれば許容範囲でなのかということに関して言及していく．

2. 関連研究

並列分散環境におけるバルクデータ転送に関する研究はこれまでにも数多くなされてきているが，グリッド環境においてこのような巨大なデータを異なるドメイン間で効率よく転送するためのツールとして GridFTP[8] がデファクトスタンダードなデータ転送プロトコルとしてよく用いられている．GridFTP では，複数のデータソース，または，同一のデータソースから複数の TCP コネクションを束ね，さらに個々の TCP コネクションでの最適なウィンドウサイズを調整することで WAN やインターネットに対して高スループットかつ高信頼な転送を実現している．また，送受信ノード間に用意した複数の中間ノードを経由し，かつ．ネットワークバンド幅を考慮して複数の通信パスをアグリゲートして，GridFTP のクライアント・サーバ間でのスループットを最大化する手法も提案されている．[9]

UDT のような高速なプロトコルもある一方で，一般的な TCP を用いて高帯域かつ高遅延な WAN でバルクデータ転送を行う場合，TCP のウィンドウサイズや輻輳制御アルゴリズムの影響によりエンド・ツー・エンドでの性能を向上させることは難しい．そのため，出来るだけ輻輳せずに最大限 WAN を使えるよう QoS が確保されたリンクと，そのリンク上で利用可能なバンド幅をもとに，転送をスケジューリングや最短経路問題に帰着させて高効率な転送を行うための手法も数多くある．[10][11]

一方，このような WAN に対するデータ転送最適化のみならず，一般的なインターネット上でのストリーミングやデータ共有のような分野でのアプリケーションレイヤでのオーバーレイマルチキャスト技術においても，ツリーやメッシュなどのトポロジを通じて動的に転送を最適化する手法も数多く研究されている．[12][13][14]

今回我々が行った Azure Blob と Amazon S3 間でのデータ転送においても，本節で述べられたような中継ノードを用いて，かつ，データを細かいチャンクに分割して転送を行う手法を用いているが，クラウドストレージにおいては一般的な POSIX でデータを直接読み書きすることは出来ず REST API を用いる必要がある点が異なっている．

3. クラウドシステム

3.1 Amazon Web Services

Amazon Web Services[1](以下，AWS) は，Amazon が展開する IaaS(Infrastructure as a Services) 型のクラウドサービスであり，現在様々なクラウドサービスを提供しているが，本稿では Amazon EC2/S3 と呼ばれるクラウドサービスについてのみ扱う．EC2 はユーザに仮想化された計算資源を提供するサービスで，xen を用いて仮想化されたノード (インスタンス) を CPU 性能やメモリサイズを選択して起動して利用する．S3 は高い信頼性や可用性，を持ったストレージで，Bucket と呼ばれるフォルダに似たスペース上に最大 5TB のファイルを保存可能である．

3.2 Windows Azure Platform

Windows Azure Platform(以下，Azure)[2] は，Platform as a Services(PaaS) 型のクラウドサービスである．Azure における計算ノードは，仮想化された Windows 環境上に用意されるが，Amazon EC2 のようにユーザが VM インスタンスを立ち上げて OS やアプリケーションをセットアップするのではなく，計算処理を行う部分を Web Role, Worker Role, VM Role に分けてそれぞれについて必要な処理をユーザがプログラムして実行する方式となっている．

また，ユーザが利用できるストレージとして Azure は，Blob，Table，Queue という 3 つの異なる抽象化されたストレージを提供している．Blob はファイルのように利用可能なストレージで通常の Blob で 50 GB，Block Blob で 200 GB，Page Blob で最大 1 TB のファイルを保存可能である．Table は Key-Value 型のストレージで，にエンティティと最大 256 個の任意の型のプロパティを持つデータを保存出来る．また，Queue は，Role 間での簡単な通信をキュー構造に従って容易に行うことが可能となるストレージである．

3.3 クラウドストレージの特徴

本稿では，特にクラウドストレージ間でのデータ移動に着目しているが，Amazon S3 および Azure Blob では，Restful なアクセスによってのみデータの追加・取得・削除などの操作が行える．また，どちらのストレージにおいても分割アップロード操作が可能となって

いる．これは，例えば，1GB のデータをアップロードする際に，一度の PUT トランザクションで転送せずに，データを 4MB などで区切った多数のチャンクでの PUT を行い，最終的にファイナライズ要求をすることで転送を完了する方法である．また，どちらのストレージにおいても，データの冗長性を確保するため，S3 においては複数のデバイスにまたがってオブジェクトが保存され，Blob においては 3 つのレプリカファイルが作成される．

4. データ転送アプリケーションの設計

4.1 設　　計

現状のクラウドストレージは，REST や SOAP の API を通してファイルにアクセスするように設計されている．これは，ユーザが通常のファイルシステムとして POSIX 経由で直接ファイルを read/write することはできないことを意味している．そのため，データを実際に転送する際には，何らかの転送ノードを用意する必要がある．転送ノードをどちら側のクラウドに用意するかによって，取りうる経路は異なる．Azure にあるデータを S3 に転送する場合，現状以下の 3 通りの経路が考えられる．

(1) Azure のインスタンスを起動し，データをダウンロードした後，そのインスタンスが直接 S3 REST API 経由で転送

(2) Azure と EC2 のインスタンス間でデータを転送した後，EC2 インスタンスが S3 へアップロード

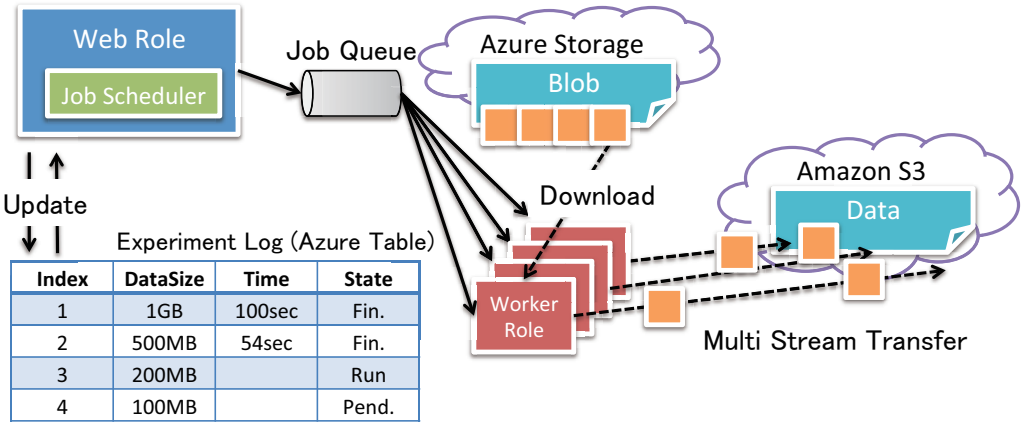
(3) Azure のデータを EC2 インスタンスがダウンロードした後，EC2 インスタンスが S3 へアップロード

(1) と (3) に関しては，ホップ数 2 で転送が完了するが，(2) はホップ数 3 を必要とする．本稿では，対象となる転送データが Azure 側にある点を考慮し，(1) の経路で転送行っている．

4.2 転送アルゴリズム

実際にクラウド間でデータ転送を行うためには，上述したようにいくつかのフェーズに分けて転送が行われる．あるデータを転送する際，1 ノードがデータを転送するのではなく，複数ノードが協調的に転送をすることによって転送スループットを向上させる手法が数多く提案されている．[15] 本稿でもこれにならい，複数ノードによるマルチコネクション転送を行う．対象ファイルを F，転送を行うノード数を N，各ノードを $n_0, n_1, \cdots, n_{N-1}$ として，以下のように転送を行う．

- **ダウンロードフェーズ:** 対象となるデータ F に対して N ノードで転送を行うので，それぞれのノード $n_x (1 < x < N-1)$ は F/N ずつのデータを取得する．効率よくデータをダウンロードするため，ダウンロード対象となっているデータをさらに細かいチャンクに分割して各ノードが多数のスレッドを用いてデータを取得する．
- **転送フェーズ:** 取得したデータをまた細かい論理的なチャンクに分割して各ノードがクラウドストレージに対して転送を行う．ノードの転送速度を最大化するため，転送フェーズにおいても各ノードが多数のスレッドを用いてマルチコネクションで転送する．



Index	DataSize	Time	State
1	1GB	100sec	Fin.
2	500MB	54sec	Fin.
3	200MB		Run
4	100MB		Pend.

図 1 実装した転送アプリケーションの概要
Fig. 1 Architecture of Our Transfer Application from Azure to S3

5. 実　　装

前節で示した転送アルゴリズムを実現するため，Windows Azure 上に図 1 に示すような Azure Blob から Amazon S3 へのデータ転送を行うアプリケーションを実装した．Web Role ではユーザからの転送リクエスト (データサイズ，チャンクサイズ，ノード数，etc) を処理する役割を持つ．また，どのような転送を行って結果がどうであったかをを Azure Table に保存し表示させている．また Web Role の中にスケジューラスレッドをを用意し，適切なタイミングで Azure Queue に対して転送リクエストを発行する．

Worker Role では Queue からリクエストを読み出してダウンロードと転送を行う．同時に転送を行うノード数に応じてそのノードが担当すべき範囲 (ブロック) が一意に決定される．決定された範囲のデータをさらに 4MB の論理的なチャンクに分割して複数のスレッド

で効率的にダウンロードを行う．これらのデータは，メモリ上またはストレージ上に保存される．その後，各ノードは任意の転送用のチャンク (e.g. 256KB) サイズに区切り複数スレッドで S3 間に対して PUT リクエストを発行して転送する．

下のプログラム 1 は，ダウンロード及び S3 への転送に使った Worker Role のコード断片である．分割したチャンク (key,value) として Queue に登録し，これを多数のスレッドが読みだして転送やダウンロードを開始する．処理が終わったスレッドから順に次の新しい転送タスクを開始できるため，非常に効率よく転送を行うことが可能である．

プログラム 1　WorkerRole.c

```
1  bufferLength = 4 * 1024 * 1024
2
3  Queue<KeyValuePair<long, long>> queue = new Queue<KeyValuePair<long, long>>();
4       long offset = myOffsetAndLength[0];
5       while (myLength > 0)
6       {
7             long chunkLength = Math.Min(bufferLength, myLength);
8             queue.Enqueue(new KeyValuePair<long, long>(offset, chunkLength));
9             offset += chunkLength;
10            myLength -= chunkLength;
11      }
```

転送を終えた Worker Role は，どの程度の時間要したかをマネージメントノードである Web Role に伝える．Web Role 側ではその転送タスクに対して適切な処理を行い，その結果を Azure Table へ反映させる．

6. 性能評価

実装した転送アプリケーションを用いて，Azure Blob に保存されているデータの Amazon S3 へ転送する実験を行った．転送元データや計算ノードはどちらも East Asia リージョンに用意し，転送先の Amazon S3 は SouthEast Asia (singapore) リージョンのものを使用した．Windows Azure で用いる計算ノードは，表 1 に示されているようなインスタンスをユーザが選択して起動することが可能である．本稿では，これらのうち small から extra-large までのインスタンスを使用した．また，1 アカウントにつき 20small インスタンス相当までしか起動出来ない制約がある．これは，例えば，medium は small2 個分に相当することを意味する．そのため，管理用の Web Role を除き，19small インスタンス相当まで使用できるので，extra-large な Work Role は最大 2 個までしか起動できない．

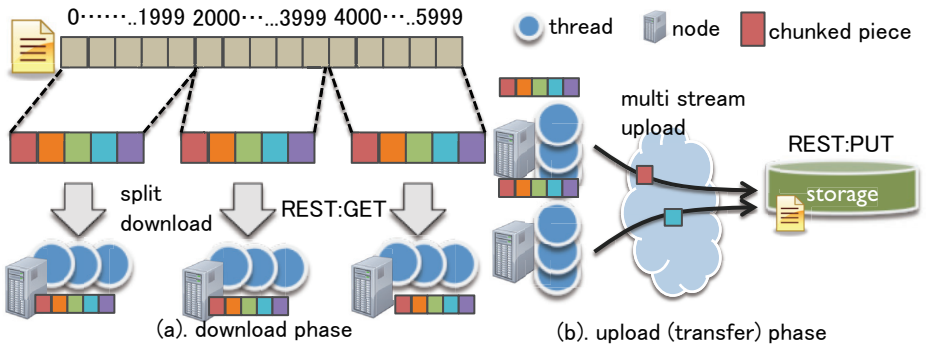


図 2　転送アルゴリズムの概要
Fig. 2　abstract of inter cloud data transfer algorithm from azure blob to amazon S3

表 1　Azure での利用可能な VM インスタンス
Table 1　selectable VM instance type of Windows Azure

Type	CPU	Cores	Mem.	Disk	I/O perf.	Price
Extra-small	1.0 GHz	1	1.7 GB	20 GB	low	$0.05/hr
Small	1.6 GHz	1	1.75 GB	225 GB	middle	$0.12/hr
Medium	1.6 GHz	2	3.5 GB	490 GB	high	$0.24/hr
Large	1.6 GHz	4	7 GB	1000 GB	high	$0.48/hr
Extra-large	1.6 GHz	8	14 GB	2040 GB	high	$0.96/hr

6.1 ダウンロード性能

まず始めに，転送を行う Worker Role からの Blob ダウンロード性能を計測した．図 3 は 1 ノードで使用するダウンロードスレッド数を変化させ，1GB の Blob を取得するときに達成されるダウンロードスループットを示している．small インスタンスでは，多数のスレッドを用いても高々60MB/sec 程度であった．一方，medium インスタンス以上ではほぼ同じで 110MB/sec 程度のダウンロードスループットが得られている．これは，インスタンスに割り当てられている I/O 性能の差が現れているものと考えられる．また，コア数が増えてもダウンロードスループットが増えないことから，1 ノードが達成できる同一リージョン内の Blob を取得するダウンロードスループットは 110MB/sec 程度が限界であることが分かった．

次に，図 4 は，転送手法で示すように 1GB のファイルを複数のノードで協調的にダウンロードしたときの合計スループットを示している．各ノードは 20 スレッド程度を用いてダ

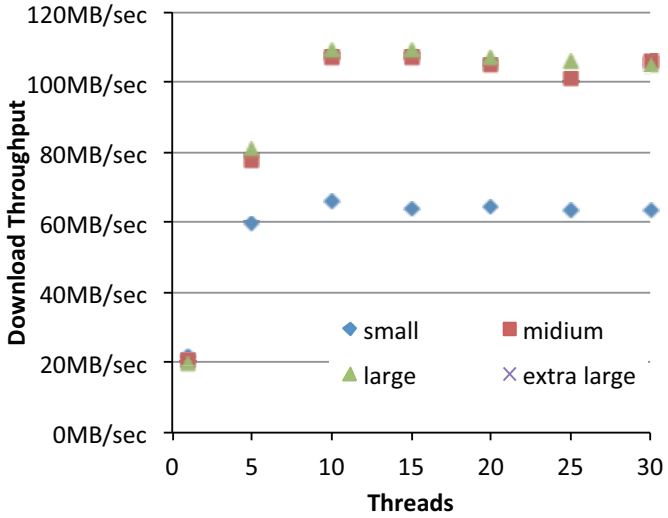


図 3 ダウンロードスレッドの数を変化させたときの 1 ノードあたりのダウンロードスループット
Fig. 3 achievable Blob download throughput with 1 node and multiple threads

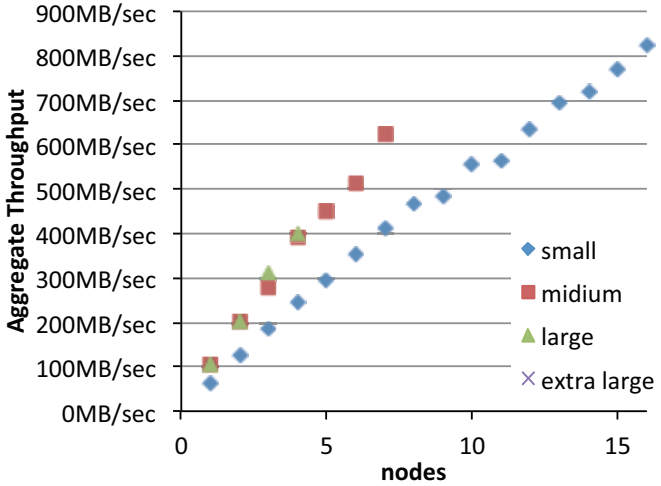


図 4 複数ノードでの合計ダウンロードスループット
Fig. 4 achievable aggregated Blob donwload throughput with multiple nodes

ウンロードを行った．どのインスタンスタイプにおいてもノード数の増加に伴って合計のスループットが線形に向上している．これは，多数のノードを用いたときでも 1 ノードあたりのダウンロードスループットがあまり変わらないことを意味している．Blob ストレージのシステム側の限界を知ることは出来なかったが，Azure で提供されている Blob ストレージは多数のノードを用いれば用いるほどある程度までは合計のダウンロードスループットが稼げることが予測される．

6.2 Azure から S3 への転送性能

この結果を踏まえて実際に Azure 側から Amazon S3 側への転送を行った．図 5 は，1 ノードの Worker Role を用いて転送するチャンク数とスレッド数を変化させたときの 1GB のファイルの転送スループットを示している．Worker Role は middle インスタンスを用いた．この図から，転送するスレッド数では，20-30 程度のコネクションで転送スループットの限界に達することが分かる．また，転送チャンクサイズの大きさは，転送スループットの性能にあまり影響を与えないことも分かった．

次に複数のノードを用いて Azure 側から Amazon S3 側へ転送を行う実験の結果を図 6 に示す．個々のノードは，前の実験を踏まえて，30 スレッド程度かつ 2MB のチャンクサイズで S3 へ転送を行っている．small インスタンスは外側へ向かうネットワークに対する I/O 性能が著しく低く抑えられており，多数のノードを用いても 16MB/sec 程度でしか転送を行うことが出来なかった．一方，middle インスタンス以上ではノード数の増加に対して比較的スケールして転送性能が向上する．middle インスタンスでは，6 Worker Role 程度で性能が限界に達し，1GB のファイルを 100MB/sec 程度の速度で転送することが出来た．large 以上のインスタンスに関しては，1 ノードあたりの転送性能が向上する結果となっている．これは，複数コアを効率よく用いて S3 側に転送できている結果であると考えられる．さらに多数の worker role を用意して転送を行った場合にどうなるかを調べることは，今後の課題の一つである．

7. 結論と今後の課題

本稿では，Azure Blob から Amazon S3 へ複数の worker role を協調的に用いて分割ダウンロードと分割転送を行う手法について，実際に Windows Azure 上に転送アプリケーションを実装して実験，検証を行った．同一リージョン内の Worker Role からの Blob のダウン

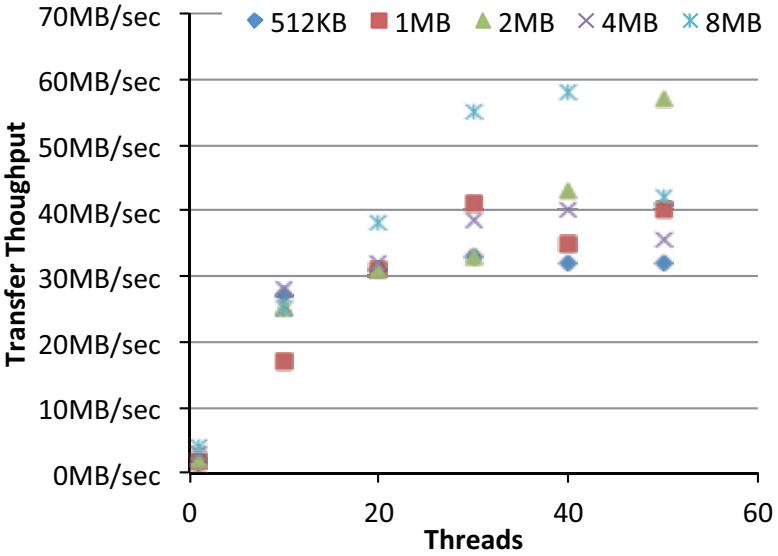


図 5 チャンクサイズを変えたときの 1 ノード当たりの転送スループット
Fig. 5 achievable Blob transfer throughput to S3 from 1 node with varying transfer chunk size

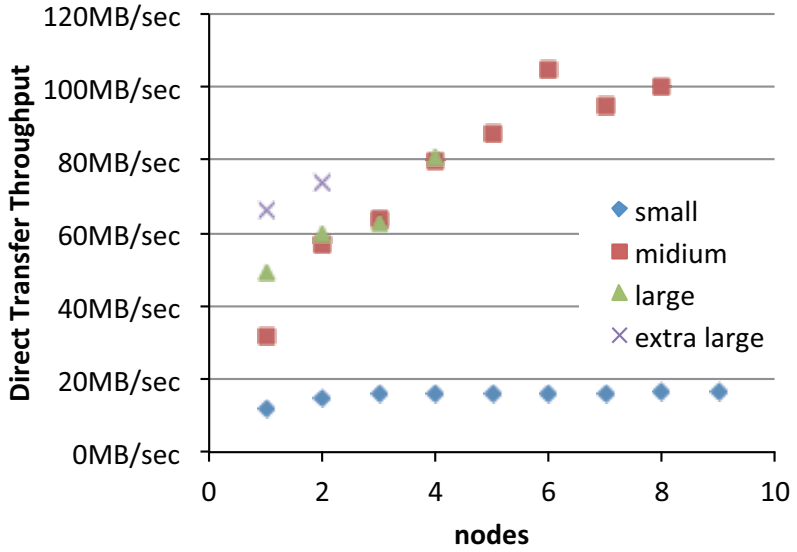


図 6 複数ノードを用いたときの合計転送スループット
Fig. 6 achievable Blob transfer throughput to S3 from multiple nodes

ロードスループットは多数のスレッドを並列に使うことで 1 ノードあたり 110MB/sec 程度のダウンロード性能が達成できた．多数のノードを用いた場合，各ノードはこの限界のダウンロード性能で blob を分割ダウンロードすることができ，ノード数の増加に応じて線形に性能が向上した．Blob から S3 への転送では，多数のノードで複数スレッドを用いてマルチコネクションでバースト的に S3 に REST で転送した結果，Azure(East Asia)-S3(SouthEast Asia) 間で，約 100MB/sec 程度の転送スループットが達成された．今後の課題としては，より多数のノードを協調的に用いた転送を行うことや，EC2 側でもノードを立ち上げて多段で転送の中継を行う手法，さらには今回とは逆に S3 側から Azure 側への転送を行う際の性能について検証する予定である．

謝辞 本研究の一部は，文部科学省科学研究費補助金 (特別研究員奨励費 20・8911，特定領域研究 18049028) の支援によって行われた．

参 考 文 献

1) Amazon Web Services: `http://aws.amazon.com/`.
2) Windows Azure: `http://www.microsoft.com/windowsazure/`.
3) Altschul, S.F., Gish, W., Miller, W., Myers, E.W. and Lipman, D.J.: Basic Local Alignment Search Tool, *Journal of Molecular Biology*, Vol.215, No.3, pp.403–410 (1990).
4) Lu, W., Jackson, J. and Barga, R.: AzureBlast: A Case Study of Cloud Computing for Science Applications, *1st Workshop on Scientific Cloud Computing* (2010).
5) A Toroidal LHC ApparatuS Project (ATLAS): `http://atlas.web.cern.ch/`.
6) Thakar, A. and Szalay, A.: Migrating a (Large) Science Database to the Cloud, *1st Workshop on Scientific Cloud Computing*, pp.430–434 (2010).
7) Armbrust, M., Fox, A., Grif?th, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I. and Zaharia, M.: Above the Clouds: A Berkeley View of Cloud Computing, Technical Report EECS-2009-28, EECS De-

Vol.2011-HPC-129 No.25
2011/3/16

partment, University of California, Berkeley (2009).
8) Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Liming, L., Tuecke, S., Memo, S. O.T., Liming, L. and Tuecke, S.: GridFTP: Protocol extensions to FTP for the Grid, *Grid Working Draft (FGFD-R-P.020)* (2003).
9) Khanna, G., Catalyurek, U., Kurc, T., Kettimuthu, R., Sadayappan, P., Foster, I. and Saltz, J.: Using Overlays For Efficient Data Transfer Over Shared Wide-Area Networks, *In Proceedings of the ACM/IEEE conference on Supercomputing (SC '08)* (2008).
10) Bouabache, F., Herault, T., Pevronnet, S. and Cappello, F.: Planning Large Data Transfer in Institutional Grids, *10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010)*, pp.547–552 (2010).
11) Ramakrishnan, L., Guok, C., Jackson, K., Kissel, E., Swany, D. M. and Agarwal, D.: On-demand Overlay Networks for Large Scientific Data Transfers, *In Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010)* (2010).
12) Jannotti, J., Gifford, D.K., Johnson, K.L., Kaashoek, M.F., J.JamesW.O ' Toole: Overcast: Reliable Multicasting with an Overlay Network, In Proceedings of the 4th conference on Symposium on Operating System Design & Implementation (OSDI ' 00), pp.197–212 (2000).
13) Kostic, D., Rodriguez, A., Albrecht, J. and Vahdat, A.: Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh, *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP '03)* (2003).
14) Cohen, B.: Incentives build robustness in BitTorrent, *Workshop on Economics of Peer-to-Peer Systems* (2003).
15) Chiba, T., den Burger, M., Kielmann, T. and Matsuoka, S.: Dynamic Load-Balanced Multicast for Data-Intensive Applications on Clouds, *In Proceedings of 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid 2010)*, pp.5–14 (2010).