Vol.2009-HPC-121 No.1
2009/8/4

メッセージ衝突を防止する適応的な集合通信

吉富翔太[†1]　田浦健次朗[†1]

ネットワークのコンテンションは集合通信の通信性能を大きく悪化させる要因の一つであり，通信の性能を向上するにはネットワーク経路上のスイッチ・ルータでの通信の衝突を引き起こさないように通信を制御する必要がある．

本稿ではコンテンションを引き起こさないよう通信を制御すると同時に，ノード間の同期操作の頻度を少なくすることで，大規模なネットワークに対しても適応可能な，効率的かつスケーラブルな gather を行うアルゴリズムを提案した．そして実環境を用いた性能評価により，本手法が既存の複数のアルゴリズムに比べて高いスループットが得られることに加え，大規模な環境においてもスケーラブルであることを確認した．

An Adaptive Collective Communication Suppressing Contention

Shota Yoshitomi[†1] and Kenjiro Taura[†1]

Network contention can significantly affect the performance of collective communication. Many gather algorithms ignore communication collision at routers and switches. Hence the network contention causes degradation of the communication performance.

In this paper, as the first step toward writing a collective communication algorithm suppressing contention, we focus on the gather operation that is one of Many-to-One operations. We propose an adaptive gather algorithm for distributed envirionments. The heart of our proposal is communication scheduling methods avoiding contention and minimizing frequency of network synchronization. We also evaluate it for some clusteres. The result of examination shows that gather operation scheduled by our algorithm is more efficient and scalable than other algorithms.

†1 東京大学
University of Tokyo

1. はじめに

1.1 背　景

計算機やネットワークの性能が多種多様なグリッド環境で，集合的な通信の最適化を図る際に考慮しなければならないことは非常に多い．例えば，ネットワークの遅延やバンド幅を元に最適な通信経路を設定する必要があったり，あるいはネットワークのトポロジーを考慮することでボトルネックとなりうる通信路にできるだけ大量の通信が同時に集中しないように通信のタイミングを変える，といったようなことが挙げられる．他にも，特に今回取り上げる多対一型の集合通信で大きな問題となるものがコンテンションの影響である．コンテンションが発生する主要因はネットワークやノードの容量を超えるデータが流入しようとすることであり，後述するように通信性能を非常に悪化させる原因となる．例えば，1 スイッチにわずか 3 台のノードが接続されたような環境であっても，個々のノードが協調しないで gather を行うとコンテンションにより激しく通信性能が悪くなることが分かっている．

集合通信に関する最適化に関する研究は数多く存在するが，多くは計算機間の End-to-End の遅延やバンド幅，送受信のメッセージ長といったパラメータのみを用いたり，あるいは LAN と WAN といった大雑把な区分でトポロジーの違いを考慮した上で集合通信をモデル化するものである．そしてこのモデルに基づき，全体の実行時間が最小となるようにノード間の通信パターン決定している．これらは厳密なネットワークトポロジーを考慮せず通信パターンが作られるので，途中の一部のルータやスイッチに通信が集中してしまい局所的にコンテンションが発生しうる問題点がある．

そのため，ネットワーク上のあらゆるスイッチやルータにおいて，通信のコンテンションが発生しないようにノード間の通信を制御することが通信性能を悪化させない上で必要不可欠である．このような通信を実現する方法として一般的なものは，複数の通信が意図せず重複するのを防ぐために，ノード間で逐一同期しながら協調して通信を行う方法[4] が挙げられる．ところが，ノード間で同期を行うのにかかる時間はノード間の遅延に依存するため，高遅延なネットワークでは同期に要する時間が莫大なものとなり全体の通信性能がかえって悪化してしまうという問題がある．

1.2 本研究の目的

そこで，コンテンションを抑制しつつ，なおかつネットワークの遅延の大小を問わず様々な環境に適応するスケーラブルな多対一型・多対多型の集合通信を実現することを目的とし，その第一歩としてコンテンションの抑制を主眼に置いた gather の手法を提案する．

さて，問題とするコンテンションが発生する条件は，あるルータ・スイッチにおいて複数ノードからの同一時刻に同一宛先ノードへのメッセージが重複することである (以下これを「メッセージの衝突」と呼ぶ)．これを考慮し，ノード間で逐一同期を取らずともメッセージの衝突を回避するということを，各ノードがメッセージをルートノードに到達するまで次々とリレー方式で中継するようなパイプライン転送を行わせることによって実現した．この方法では，各ノードはそれぞれある別のノードに対して独占的にメッセージを中継するため，同期をせずともメッセージの衝突を起こすことがない．さらに，実際には単独のパイプライン構造を作るだけでは通信性能が向上しないケースがあるため，複数ノードからのメッセージの受信を許可する代わりに，メッセージが衝突しないよう逐次に同期を取りながら通信するアルゴリズムを補助的な手法として利用する．

以降，まず 2 章で既存の gather でのコンテンションの抑制に関して考察し，3 章で本研究の問題設定としてコンテンションの影響について述べる．その元で 4 章で提案する適応的な gather のアルゴリズムを提案し，5 章にて分散環境での本手法の性能について評価を行う．

2. 関 連 研 究

2.1 組み合わせ最適化問題としての gather

不均質なネットワークでの効率的な gather を行うアプローチとして，エンドホスト間のネットワークのパラメータを利用するものが挙げられる[3),5)]．これらの手法はネットワークの遅延・バンド幅・通信メッセージサイズといったパラメータから，個々のエンドホスト間の通信コストを，全てのノード間でそれぞれ算出し，通信コストの組み合わせ最適化問題と見なし，全体の総通信コストが最小となるように各ノードについて通信相手のノードを決定するというものである．これらは一つの通信が他のあらゆる通信に影響を与えないならば性能の良い解を与えるが，複数の通信が同時に同一のリンクを利用するような望ましくないスケジューリングがなされてしまうと，ネットワークの至る所でコンテンションが発生して通信性能が悪化してしまう可能性がある．そのため現実の複数のルータ・スイッチで接続されたネットワーク上での gather を考える際にはこの手法だけでは不十分である．

2.2 既存の MPI ライブラリでの gather の実装

また，並列分散環境で良く利用される MPI ライブラリには標準で MPI_Gather という gather を行うための集合通信の API が用意されている．分散環境で利用可能な MPI のライブラリに関する研究や実装は数多く，gather に関しても様々な実装がなされている．

例えば MPI の有名なライブラリの一つである OpenMPI[2)] では，ルートノード以外のノードがルートノードに対してメッセージを協調せず一斉に送信するという最も単純な実装が行われている．この手法はコンテンションにより通信性能が悪化する可能性があることは前に述べた通りである．もう一つの有名なライブラリである MPICH,MPICH2[1)] では，binomial tree を用いた通信パターンを採用している．これはルートノードを根とする $O\log(n)$ の深さのツリーを作り，そのツリーに沿ってデータを送信するというものである．ところがこの手法でも複数ノードから同時にデータを受信しようとすることに変わりはなく，コンテンションを完全に防止することはできない．またこれらの実装は均質なネットワークを対象としたものであり，本研究が対象としているネットワークが不均質な環境ではそもそも通信性能が極端に悪化してしまうという問題がある．

一方，グリッド環境に適応した MPI ライブラリの研究も数多く行われている．Thilo Kielmann らによる MagPIe[7)] は LAN の内外の区別をし，各 LAN の中では MPICH と同様のアルゴリズムを用い， LAN 内の起点となる 1 ノードのみがルートノードと通信を行うというものである．また松田らによる GridMPI[9)] は，この考え方を発展させ複数ノードが WAN のリンクでコンテンションを起こさない程度にクラスタ間の通信を同時に行うことで高いスループットを得ている．しかしこれらは LAN 内での通信パターンについては考慮されておらず，ネットワークの構成や通信パターンによっては複数の通信が同時に集中してしまうことから，依然として LAN 内の通信コンテンションにより通信性能が劣化してしまう可能性があることが問題点である．

3. コンテンションの影響に関する検討

3.1 概　　略

以下に本稿で gather を考える上で問題とする，コンテンションの影響についての概略を示す．Lastovetskym らによっても関連する現象が報告されている[8)] が，一つのスイッチに複数の計算ノード群が接続されていて，

(1) 同一の宛先に対して同じノード群内の他の複数のノードからの通信が重なる
(2) 受信ノードが処理できないくらい，スイッチに流れ込む合計メッセージサイズが大きい．

という条件のとき，その流れ込むメッセージサイズ，受信ノードの処理速度やスイッチの性能によってはスイッチのバッファ溢れにともなうパケットロスが生じる．これと後述する TCP の性質とが合わさり非常に集合通信の性能が低下することがあるというものである．

Vol.2009-HPC-121 No.1
2009/8/4

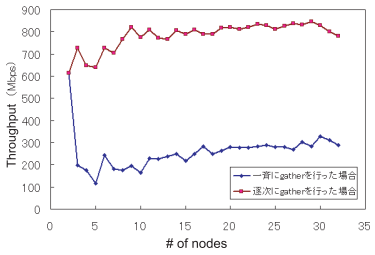


図 1　ノード間の協調の有無によるノード数 vs.gather のスループット比較

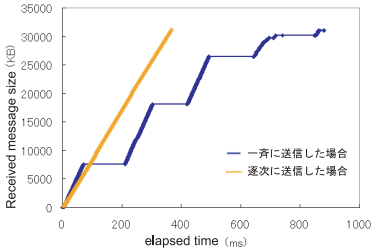


図 2　32 ノードでのデータの受信状況の比較

3.2　ノード間の協調の有無による gather のスループットの比較

具体的な通信に与える影響を見るため，次のような簡単な実験を行った．実験の主目的は各送信ノードが協調せず同時にルートノードに向けてデータを送り出し，前項の条件が満たされてしまうようなとき，どれほど通信性能が悪化してしまうかを検証することである．

実験環境は 48 ポートの一つのスイッチに 32 台のノード (Linux Kernel 2.6.18) が 1Gbps のリンクで直接接続されたものである．まずこの環境で，各々のノードが 1MB のデータをルートノードに向かって

(1)　ノード間が協調せずデータを一斉に送信する場合
(2)　1 ノードずつ逐次にルートノードに向かってデータを送信する場合

とでスループットにどれだけの差が生じるかを比較した．前者と後者の違いは同時にネットワークを流れるメッセージの流量であり，前者は前項の条件を満たしコンテンションが発生する可能性がある．

図 1 の結果を見ると，逐次に通信を行った場合は総じて 700〜800 Mbps のスループットが得られているのに対し，協調せず一斉に送信した場合は参加ノード数が 3 以上でスループットがおよそ 200〜300 Mbps と半分以下に激減していることが見て取れる．

3.3　受信ノードのデータ受信状況の比較

続いて，今度は参加ノードが 32 台の場合で，受信ノードが時系列でどのようにデータを受信しているかを調査した．**図 2** は，受信ノードを除く 31 ノードがそれぞれ 1MB のデータを送信し，受信ノードが時々刻々と合計 31MB のデータを受信していく様子をプロットしたものである．

これを見ると，逐次に通信を行った場合は時間軸に対してほぼ線形に受信したデータ量が増えて行っており，受信ノードは間断なくデータを受けとることができている．ところが，一斉に通信を行った場合は図中で 4 箇所ほど，約 200 ミリ秒の間データを全く受信できていない時間帯が存在している．この全くデータを受信していない時間帯が存在することが図 1 で一斉に通信を行うとスループットが激減する原因であると判断できる．

3.4　スループットの低下を引き起こす要因

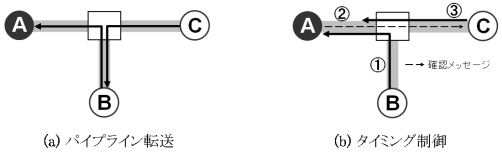
以上の調査により，データを一斉にある 1 ノードに向けて送出すると，そうでないときと比較してスループットが明確に低下することが分かった．また図 2 でデータを受信できている時間帯での両者のグラフの傾きを比較するとほとんど差がないことから，受信ノード側では到達したデータを処理することができていると考えられる．すなわちデータを受信していない 200 ミリ秒の間は通信自体が行われていないと推測できる．

一方，別途行った調査でノード間のパケットをダンプすることにより，送信されたパケットが衝突により途中のスイッチでロスしてしまっていることも判明した．TCP では，パケットロスが発生した場合は送信ノードが受信ノードから重複した ACK を 3 回受信するとそのパケットが相手に届いていないと判断しパケットの再送を行う高速再送アルゴリズムが働いている．しかし，データストリームの末尾のなど後続するパケットが十分な数存在しない部分のパケットが何らかの理由により落ちてしまうと，受信ノード側はそれ以上のデータを受け取らないため， ACK を送ることができなくなってしまう．その場合，送信ノード側はある一定の待ち時間 (RTO) だけ待ってからパケットの再送を行う．実験に使用したノードのカーネル 2.6.18 では RTO の値は過去の RTT の重み付き平均と分散から動的に調整されるが、その最小値が 200 ミリ秒に設定されており，この待ち時間が図 2 の通信が行われない 200 ミリ秒の時間帯に相当すると考えられる．

さらに図 1 からも分かる通り，高々2 ノードが同時に 1 ノードにデータを同時に送信する場合でもコンテンションにより通信性能が悪化してしまう．従って，gather に限らず多対一型の通信で高いスループットを得るためには，このコンテンションを抑制するような通信を行うことが必要不可欠であると言える．

4.　提 案 手 法

さて前章では gather を各ノードが協調せず無計画に行うと，通信性能が劇的に悪化してしまうことがあるという事例を紹介した．大元の原因として考えられることは，複数のノードが同時に同一宛先に対して同じスイッチ・リンクを使用して通信を行おうとしたことや，ノードの受信能力が限界に達してメッセージを受信し切れないことによるものである．これ



(a) パイプライン転送　(b) タイミング制御

図 3 基本スケジューリングアルゴリズム

らの現象を避けるためにはどのルータ・スイッチにも複数のノードから「同一時刻」に「同一宛先ノード」へのメッセージが流れないようにすることが必要である．

我々はこの条件に適応する手法として，以降 4.2 に示すような基本となる 2 種類の通信のスケジューリング方法を提案する．そして，具体的に 2 種類のアルゴリズム用いて個々のノードの通信をどのように制御するかについて，その全体的な流れを 4.4 で述べる．

4.1 問題設定

アルゴリズムを構築する前提として，本手法を適用するネットワークは通信に参加するエンドホストと，ルータやスイッチのような中継に参加する機器の 2 種類のみで構成されていることを想定している．利用するネットワークのトポロジーや遅延・バンド幅といった構成情報は，例えばネットワークの情報を取得する手法[10),11)] によって入手可能である．これらの手法によるネットワーク情報の推定時間は規模にもよるが，数秒から数十秒程度である．これは gather の一回当たりの実行時間と比べると非常に大きいが，ネットワークの状態は頻繁に変わるものではないと考えられるため，本手法では予め取得しておいた静的なネットワーク構成情報を用いるものとし，前準備としてネットワーク情報の取得に要するコストかからないものとする．

また，提案手法のアルゴリズムを考えるにあたって，同様の gather が同時多発的にノードを重複しては起こらないという仮定をおく．さらに，使用するネットワークは全二重であり，送信と受信を同時に行えるものとする．

4.2 基本スケジューリング手法

複数のノードが通信に参加し，かつメッセージの衝突が起こり得るような場合の，最も簡単なネットワークは**図 3** のような 1 スイッチに 3 ノードが接続されているものである．この形態のネットワークを基本ネットワークと呼ぶことにする．図中 3 ノードのうち B，C の 2 ノードが送信ノードとなり残りの A に対してメッセージを送信するものとする．

4.2.1 同期を取りながらの逐次送信

一つ目は図 3(b) に示すようにノードがそれぞれ同期を取りながら通信を行う方法である．

(1) B がまず A に対してメッセージを送信する．
(2) A は B からのメッセージを受信したら，C に対して 1 バイトのパケットを送出する（同期操作）．
(3) C は A からの確認メッセージを受信するまでブロックし，確認メッセージを受信し次第 A に対してメッセージを直接送信する．

この方法の特徴は，(2) で実現されるような，間に同期操作を挟むことで 2 ノードからのメッセージが同一時刻に流れ込むことがないということを保証する点である．

一見して自明に思えるこの手法であるが，欠点が存在する．(2) における確認メッセージのやりとりによって，単純に A,C 間のリンクの遅延分の時間が総実行時間に加算される．一般のクラスタ内のようにリンクの遅延が非常に小さい場合はこの時間はほぼ無視できるが，クラスタ間をまたがった遅延が大きいノード間では無視できないものとなる．

4.2.2 パイプライン転送

二つめは図 3(a) に示すようなパイプライン転送と呼ぶ手法である．C は A に対して直接メッセージを送るのではなく，B に対して A に送るべきメッセージを送信する．一方 B は，自身が本来保持しているメッセージを A に送りつつも，C からのメッセージを待ち，メッセージを受け取り次第即座に A に対して再び送出する．このように，C → B → A とあたかも一本のパイプラインでメッセージを次から次へと隣の別ノードに対してリレーしていく．

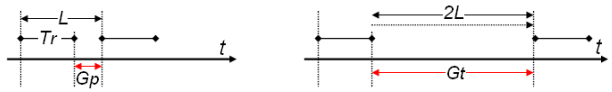
この手法では，パイプライン上のあらゆるノードは別のあるノードに対してのみ独占的にメッセージを送り，また別のノードから独占的にメッセージを受け取るという性質を持つ．従って，複数のノードから同一の宛先ノードへと向かうメッセージは存在しないため，コンテンションが起きることがない．

4.3 アルゴリズムの理論上の通信コストの比較

4.3.1 前提条件

続けて，以上の 2 手法および，単純な手法との理論的な通信コストの比較を行い，手法のメリット・デメリットについて論じる．

簡単の為，1 クラスタ内でリンクの遅延 L やバンド幅 B が全て等しい環境を考える．このとき，ルートノードが 1 つのメッセージを受信し始めてから受信し終わるまでの時間 T_r は，通信パターンによらず一定で，バンド幅，メッセージサイズ M を用いて $T_r = \frac{M}{B}$ と表せる．ここで，ルートノードが一つのメッセージを受信してから次のメッセージを受信するまでの受信待ちをしている間隔を到着間隔 G とする．すると，メッセージ自体を受信している時間 T_r は前述の通り一定であるので，G の値が小さければ小さいほど効率の良い通信



(a) パイプライン転送のコスト　(b) 同期しながら逐次に送るコスト

図 4 メッセージの受信コストと受信間隔

パターンであると言える．

4.3.2 コストの比較

以上の定義を行った上で，

(1) N ノードが全て 4.2.1 の要領で逐次にデータを送信

(2) MPICH のように Binomial tree を使い，かつ同世代内では 4.2.1 の要領で逐次に同期を取りつつ通信

(3) N ノードが全て 4.2.2 の要領でパイプライン転送

の 3 通りのコンテンションを抑制できる手法の総通信時間 (コスト) の比較を行う．メッセージ自体を受信している時間 T_r は前述の通り一定であるので，G の値のみを比較すれば良い．

まず 1 のケースでは，同期に要する時間 $2L$ がそのまま受信間隔となる (**図 4**b)．同期回数は N 回なので，総受信間隔 W_s は $W_s = 2LN$ となる．同様に 2 のケースは，同期に要する時間がそのまま受信間隔となるが，Binomial tree を組むことにより同期回数を $\log N$ 回に抑えることができて，送受信間隔は $W_t = 2L \log N$ である．一方，3 のケースでは，そのコストは図示すると図 4(a) のようになり総受信間隔 W_p は

$$W_p = \begin{cases} 0 & L \leq T_r \text{ のとき} \\ (L - T_r)N & L > T_r \text{ のとき} \end{cases} \quad (1)$$

となる．

一見コンテンションを防止する自明な方法と考えられる 1 や 2 の手法は W_s や W_t が遅延の 2 倍 $2L$ に比例しているため，WAN をまたいだ通信のように遅延が大きくなる場合は明らかに $W_p, W_t > W_p$ となる．よって，1 や 2 の手法を単にどのような環境でも機械的に当てはめるだけでは，たとえコンテンションが起きないといっても効率のよいアルゴリズムになっているとは言えない．また，そうでなくとも，T_r の値が遅延 L に近いかそれ以上の時は，$W_p, W_t > W_p$ となる．これは，1,2 の場合同期のコストがかかるが，3 の場合はそれが無いためである．

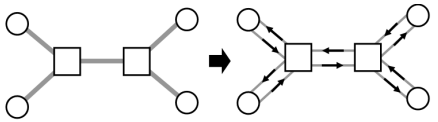


図 5 2 本の片方向リンクに分解されたネットワーク

一方で 3 の手法では，パイプライン構造が何段にも重なった場合 (N が大きくなったとき)，余分な経路を辿ることによって通信時間が増大し，結果的に通信性能が悪くなることが有り得る．例えば，経路の途中でバンド幅が狭いリンクが存在していたり，リンクの遅延が非常に大きくなっていた場合，そのリンクがボトルネックとなってしまい逐次的に通信するよりも通信性能が悪化してしまう．従って全てのメッセージをパイプライン状で転送することもまた最適な方法になるとは限らない．メッセージがパイプライン状の通信でボトルネックの経路を通過するよりも，一部だけ同期を取って逐次的に通信することでその経路を回避して通信することで，同期を取るコストを差し引いても通信性能は悪くならないことがあるからである．

以上のことから，本手法では，1 と 3 の手法，すなわち 4.2.2 と 4.2.1 の手法を，通信時間がより短くなるように両手法を組み合わせて通信パターンを作成する．

4.4 ネットワーク全体のスケジューリング

以降で通信に参加するノード全体がどのように通信を行うか，その経路を決定するための手法について述べる．なお「どのノードがどのノードに対してメッセージを送信するか or 受信するか」という関係を表す構造を，以降は「通信ツリー」と総称する．

また，本手法のアルゴリズムはネットワーク全体をコンテンションが起きない条件を満たしつつ一筆書きの要領で辿ることができることが必要である．まずはじめに，この仮定が一般のネットワークで成り立つことを証明する．

4.4.1 前提条件の証明

さて，そもそもネットワーク全体でコンテンションが起きないような条件を満たしつつ一筆書きで辿れるということは，ネットワークにおいて

(1) どのリンクも同方向には高々1 度しか通らない

(2) 全てのエンドノードを必ず 1 度だけ経由する

の両者の条件を満たす一つの経路が存在する必要があるということと同値である．そしてこのような経路が存在することは，以下の定理から導くことができる．

Lemma 1. ツリー構造を持つネットワークには，任意のノード X から *(1),(2)* を満たし

た状態で再び X へと戻ってくる経路が少なくとも一つは存在する．

証明． どのリンクについても片方向には 1 度ずつ通れるという条件より，(1) はネットワーク上の全てのリンクを**図 5** のように並行する上りと下りの 2 本に分割して考えたときにネットワーク全体を一筆書きの要領で一巡できる…(1') ということと同値である．一方 (2) は，片方の端点がノードであるリンクを 2 本に分割した上でノード側の端点を連結することにより，(1') を満たせば自動的に満たされることは明らかである．

(1') に関しては，全てのリンクを 1 本ずつのそれぞれ片側通行のリンクとみなし，2 本に分割して考えれば，全ての頂点 (＝スイッチ) に接続されているリンクの本数は必ず偶数本である．従って，リンク分割後のネットワークはオイラーグラフとなる．オイラーグラフであれば，一筆書きをしてグラフ中を一巡する経路が少なくとも一つは存在するので，(1') もまた成立する．以上より，題意を満たす経路が存在することが示された． □

Lemma 2. ツリー構造を持つネットワークでは，任意のノード X，Y を結ぶような条件 *(1),(2)* を満たす経路が少なくとも一つ存在する．

証明． 定理 1 より (1),(2) を満たす X を始点とする閉路が存在するから X,Y を直接結ぶパスをその閉路より取り除くことで，X，Y を端点とする (1),(2) を満たすような経路を作成することができる． □

4.4.2 通信ツリーの構築順序

以上より，端点をルートノードとしたときに，残りのノードをコンテンションを起こさない条件を満たしつつ一筆書きすること順路が存在する．この順路にしたがって，ルートノードから順にノードを辿り，一つずつ通信ツリーに加えていく．

また，端点を決定したとしても途中のノードの辿り方の組み合わせは複数存在する．そこで，複数の選択肢が存在する場合，ルートノードとの間のバンド幅が大きいノードを優先的に選択する．これは，パイプライン構造に従ってメッセージを送った場合はルートノードに近いノードほど多くのメッセージを送受信することになるので，それらをなるべくバンド幅が大きいリンクを通すようにするためである．

4.4.3 3 ノード間の通信ツリーの構築

まず最も簡単な場合として基本ネットワークにおいて，送信ノード (B)・受信ノード (A) にもう一つ送信ノード (C) を通信ツリーに付け足すことを考える．ノード C のデータの末尾がノード A に到着する時間を以って通信完了とすると，パイプライン通信の通信時間 T_p

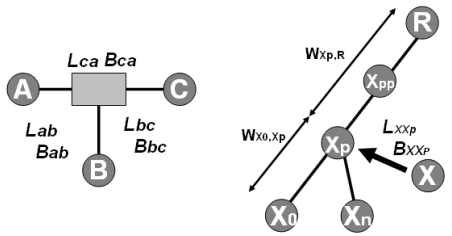


図 6 通信ツリー構築 (左：3 ノード間の場合　右：4 ノード以上の場合)

及び同期付き逐次通信の通信時間 T_s はそれぞれバンド幅 B と遅延 M，メッセージサイズ M を用いて下式で表される．

$$T_p = L_{ab} + \frac{M}{B_{ab}} + \max\left[\frac{M}{B_a b}, L_{bc} + \frac{M}{B_{bc}}\right] \tag{2}$$

$$T_s = L_{ab} + \frac{M}{B_{ab}} + 2L_{ca} + \frac{M}{B_{ca}} \tag{3}$$

本手法では仮定より B, L, M は既知であるから，両者の大小は一意に定まる．ゆえにこの 2 式の比較し，ノード C は値がより小さくなる手法でデータを送信すれば効率が良いということになる．

4.4.4 4 ノード以上での通信ツリーの構築

これを一般のネットワークに拡張する．すなわち，ルートノード R を頂点とする通信ツリー C にノード X を追加することを考える．このときノード X が次にデータを送るノードを $X_p \in C$，X_p が次にデータを送るノードを X_{pp} とし， X_p に対し，X よりも先にデータが到着するようにスケジューリングされているノード群 $X_{b0}, \cdots, X_{bn} \in C$ (ただし X_{b0} の方が先にスケジューリングされている) とする．また T_{X_{b0},X_p} を X_{b0} のデータの末尾が X_p に届くまでの時間，$T_{X_p,R}$ を X_p のデータがルートノードに到着する時間とすると，T_{X_{b0},X_p}，$T_{X_p,R}$ は前項の 3 ノードの場合の計算の繰り返しで求めることができる．

すると，ノード X のデータがルートノードに到着する時間 $T_{X,R}$ は以上のパラメータを用いて

$$T_{X,R} = \begin{cases} T_{X_{b0},X_p} + 2L_{XX_p} + \frac{M}{B_{XX_p}} + T_{X_p,R} & (X_p \text{ がツリーの途中にある場合}) \\ T_{X_p,R} + \max\left[L_{XX_p} + \frac{M}{B_{XX_p}}, \frac{M}{B_{X_pX_{pp}}}\right] & (X_p \text{ がツリーの末尾にある場合}) \end{cases} \tag{4}$$

のように求めることができる．これを全ての X の送信相手の候補 $X_p inC$ について求め，

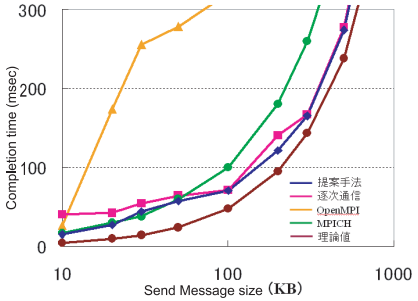


図 7 1 クラスタ (56 ノード) での性能

それが最小となるノードを真の X の送信相手として決定する．

以上の手順をネットワーク中の全てのノードについて適用したものが全体の通信ツリーとなる．次章では，この提案手法の性能の検討を行う．

5. 実験と評価

前章までに述べたアルゴリズムを実際に C 言語で実装し，既存の MPI の API の実装などとの比較を行った．なお今回の実験では全てのホストがグローバル IP アドレスを持ち，全対全で任意の方向に通信が行える環境を用いている．また，ノードに直接接続しているリンクのバンド幅は全て 1Gbps，OS は Linux (kernel 2.6.18) である．また，通信ツリーの構築に関わる処理時間は多くとも数 10 ミリ秒で，通信自体に比べると殆ど無視できるものであった．

5.1 1 クラスタ内でのスループット

まず，単一のクラスタ内での評価を行うため，56 ノードが 3 つのスイッチに接続されている環境で実験を行った．比較対象として，各ノードが一斉に通信を行った場合 (concurrent)・各ノードが逐一同期しながら逐次に通信を行った場合 (sequential)，および既存の MPICH を用いた 3 手法を用いている．**図 7** がその結果である．

図中の右側，メッセージサイズが大きい領域で OpenMPI，MPICH の性能が他と比して悪いのは，前述したコンテンションの影響によるものである．また図中左側，メッセージサイズが小さい領域で sequential の性能が悪化しているのは，ノード間の同期に要するコストが無視できなくなるためである．本手法による gather の実行時間と理論値との比は，1.1〜3.0 程度であった．従って提案手法は 1 クラスタ内では他の 3 手法と比較してもメッセージサイズによらず同等以上の性能を得られていることが分かる．

5.2 複数のクラスタを利用した場合の台数効果

続いて，複数クラスタでの性能比較を行った．まず，通信に参加するノード数を増減させたときの台数効果を**図 8** に示す．1 クラスタの場合と同様に一斉送信した場合と逐次送信した場合の 2 つと，加えて LAN の内外で異なる通信パターンを持つ，既存の MagPIe のアルゴリズムを利用し性能比較を行った．

図 8 の結果の特徴的な点は，一斉に通信した場合や MagPIe のアルゴリズムでは，ノード数によらずほぼ実行時間が横ばいなことである．これはコンテンションによって通信性能が悪化していることが原因である．一方提案手法のアルゴリズムではクラスタ内でもコンテンションによる性能悪化を引き起こすことなく，ほぼ台数に応じて実行時間が増加していく．逐次に送る場合はルートノードとそれ以外のノードとの遅延がそのまま実行時間に加算されていくため，クラスタ間の大きな遅延の影響で性能は極端に劣化している．

5.3 WAN のバンド幅の利用効率の評価

近年では WAN のバンド幅は 10Gbps を超える広帯域になり，パイプライン転送や逐次通信のように WAN のリンクを高々1 本のデータストリームしか流れない場合は WAN の帯域を有効活用できているとは言い難い．加えて，WAN でコネクション 1 本あたりの帯域が制限されていたり，他の不特定多数のユーザと回線を共用している場合などでは，できるだけ多くのデータを同時に流すほど帯域を大きく占有することができ結果的にデータを高速に転送することができる利点がある．言い換えると，一斉に複数ノードが通信し WAN のリンクに同時に大量のデータを流し込むことで，コンテンションによる性能悪化が起こりつつも，それを相殺するできるほどのスループットを得られることができることが有り得る．

これを検証するため，9 クラスタ 160 ノードを利用して gather を行った場合の性能を**図 9** に示す．提案手法 (pipeline + sequential)，パイプライン転送のみを行った場合 (pipeline)，ルートノードと同一クラスタ内では提案手法を用い，別クラスタからのデータは一斉に受け取る (pipeline+concurrent)，一斉にデータを送信する (concurrent) の 4 つの比較を行った． 図 9 を見ると，グラフ左側のデータサイズが比較的小さい領域では提案手法やパイプライン転送のみを行った場合でのスループットが一斉送信した場合に比べて勝っているものの，データサイズが大きい領域では逆に大きく劣っている．

これは上述したことを裏付けるものだと言える．データサイズが小さい領域ではコンテンションを抑制することにより提案手法はスループットで勝るが，データサイズが大きくなると WAN の帯域を一度に多く利用できる一斉送信の方が良くなるのである．

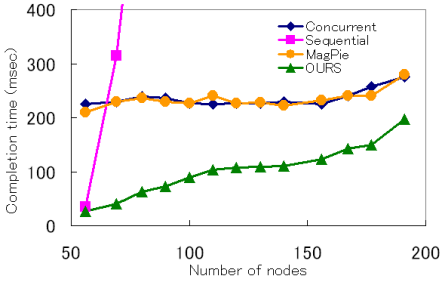


図 8 複数クラスタで 20KB のメッセージの gather した場合のスループット

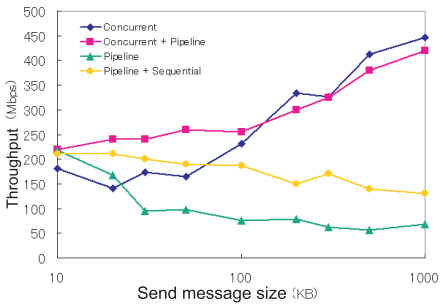


図 9 9 クラスタ (160 ノード) での性能

このように「一度に流すデータ量を制限しコンテンションを抑制すること」と「一度に大量のデータを流し WAN の帯域を有効利用すること」は相反する命題であるが，どちらも複数クラスタをまたいだ集合通信を最適化する上で重要な要素である．従って，この兼ね合いを如何にして決定するかが今後の重要な課題となっていると言える．

6. おわりに

6.1 まとめ

本論文では，既存の gather 操作のコンテンションによる通信性能の悪化に着目し，ネットワークトポロジーを利用することでルータ・スイッチでのコンテンションを防ぎ，かつ様々な環境に適応するスケーラブルで高性能な gather を実現するアルゴリズムを提案した．そして 1 クラスタ，あるいは複数クラスタにまたがった環境でのアルゴリズムの性能評価を行い，1 クラスタ 56 ノードでは最大で理論値の 0.85 倍のスループットを，9 クラスタ 193 ノードでは既存の他手法と比べ最大で 1.2 倍のスループットを得ることができた．また我々の手法がノード数の増加について高いスケーラビリティを持つということを示した．

6.2 今後の展望

今後の課題点として挙げられるのは，以下の 3 つである．

- 広帯域な WAN のバンド幅の考慮
- より適応的なアルゴリズムへの改良
- 既存のオーバーレイフレームワークの組み込み

まず初めの点として，先に挙げた WAN の帯域を有効に利用することが挙げられる．

また，2 つめとして本手法は静的なネットワークの情報に大部分を依存していた．しかしネットワークや計算機の状態は時々刻々と変化するものであり，予め取得していた情報が正確でなくなる可能性がある．そのため実行中に動的に情報を収集し，経路決定アルゴリズムへフィードバックすることで，より正確なスケジューリングが行えるようにする必要がある．

三点目としては例えば，デッドロックフリールーティングを実現するためにグリッド環境上で接続数やルーティングを制限した TCP のオーバレイを構築する研究[6] もなされている．このようなオーバレイ上でもコンテンションフリーな通信が行えるような手法を目指していく．

謝辞 本研究の一部は文部科学省科学研究費補助金特定領域研究「情報爆発に対応する新 IT 基盤研究プラットフォームの構築」の助成を得て行われた．

参考文献

1) mpich. http://www.mcs.anl.gov/mpi/mpich.
2) Openmpi. http://www.open-mpi.org/.
3) PrashanthB. Bhat, C.S. Raghavendra, and ViktorK. Prasanna. Efficient collective communication in distributed heterogeneous systems. *JPDC*, pp. 15–24, 1999.
4) Ahmad Faraj and Xin Yuan. Message scheduling for alltoall personalized communication on ethernet switched clusters. *IEEE IPDPS*, 2005.
5) Junichi Hatta and Susumu Shibusawa. Scheduling algorithms for gather operations in distributed heterogeneous systems. *Technical report of IEICE*, April 2000.
6) Ken Hironaka. A high-performance deadlock-free overlay for wide-area parallel and distributed computing. *Master's Thesis, the University of Tokyo*, 2008.
7) Thilo Kielmann, Rutger F.H.Hofman, Henri E.Bal, Aske Plaat, and Raoul A.F.Bhoedjang. Magpie: Mpi's collective communication operations for clusterd wide area systems. *PPoPP'99*, May 1999.
8) Alexey Lastovetsky and Maureen O'Flynn. A performance model of many-to-one collective communications for parallel computing. *Proc. of IPDPS*, 2007.
9) Motohiko Matsuda, Yutaka Ishikawa, Tomohiro Kudoh, Yuetsu Kodama, and Ryousei Takano. Efficient collective algorithms for grid environment. *IPSJ SIG Notes Vol.2006*, No.87, pp. 257–262, Jul 2006.
10) Sho Naganuma, Kei Takahashi, Hideo Saito, Takeshi Shibata, Kenjiro Taura, and Takashi Chikayama. Improving efficiency of network bandwidth estimation using topology information. *SACCIS 2008*, pp. 359–366, Jun 2008.
11) Tatsuya Shirai, Hideo Saito, and Kenjiro Taura. A fast topology inference-a building block for network-aware parallel computing. *In Proceedings of the 16th IEEE International Symposium on HPDC*, pp. 11–21, Jun 2007.