社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－HPC－99　(41)
2004／8／1

接続を動的に制御するメッセージパッシングシステム

金田憲二[†,††] 田浦健次朗[†,††] 米澤明憲[†]

接続を動的に制御するメッセージパッシングシステムを設計・実装する．既存のメッセージパッシングシステムでは，ノード間で接続が全対全に維持される場合が多い．その結果，ノード数の増加につれて接続数が爆発し，計算性能に悪影響を与えるという問題がある．そこで，接続を効率よく制御するメッセージパッシングシステムを設計・実装する．このシステムにおいては，各ノードは少数の接続のみ維持し，それにも関わらず性能を低下させることなくお互いに通信し合うことができる．より具体的には，このシステムは以下の機能をもつ，まず，物理的なネットワークトポロジーを自動推定し，その情報を利用して動作中に維持する接続を計算する．次に，必要に応じて動的に接続を追加・削除することにより，通信性能を向上させる．128 プロセッサ上で整数ソートを走らせ性能を測定した．その結果，全対全に接続を維持した場合と同程度以上の性能を，小さい接続数で達成することができた．我々は，広域環境で並列計算を行う際もこの機構は利用可能であると考える．

A Message Passing System with Dynamic Connection Management

Kenji Kaneda,[†,††] Kenjiro Taura[†,††] and Akinori Yonezawa[†]

We have designed and developed a message passing system that controls connections dynamically. In many message passing systems, every node establishes connections to all the other nodes. As a result, the number of connections that each node keeps becomes very large as the number of nodes increases; this degrades the performance of the systems. For this puropsoe, we have designed and developed a message passing system that manages connections efficiently. In this system, each node keeps only a small number of connections and can communicate with one another without degrading the performance. More specifically, our system has the following features. First, it calculates connections that are maintained during computation by automatically discoverying network topology. Second, this system adds/deletes some connections dynamically to improve the communication performance. We measured the performance of our system by running Interger Sort on 128 processors. The experimental result shows that our system achieved the performance nearly equal to (or better than) the case where all possible connections are established. We believe that our mechanism can also be applied to parallel computing in wide-area networks.

1. はじめに

メッセージパッシングモデルは，最も一般的な並列計算のプログラミングモデルの一つであり，このモデルに基づいて数多くの科学技術計算が記述されている．そのため，クラスタなどの多数の計算資源を利用して高性能を実現するメッセージパッシングシステムに関する研究が，近年非常に盛んに行われている．(例えば MPICH-G[1)]，GridMPI[2)]，Phoenix[3)]) ．そうしたメッセージパッシングシステムはノード間の通信に TCP などのコネクション指向ストリーム通信を用いることが多い．特に，広域環境で動作する場合には，輻輳制御の難しさや実装の汎用性などを理由に TCP を用いる場合が多い．

しかし，TCP を用いてノード間で通信を行う場合，単純にノード間で全対全に接続を確立すると，ノード数に応じて確立する接続の数が増大し，既存の TCP や OS の実装が問題となる．例えば，`select` システムコールは接続数が増加するにつれて処理性能が低下する．また，接続の最大数が OS によって制限されている場合もある．より重要な問題としては，接続によって大量の資源が消費されることが挙げられる．特に，Long Fat Pipe Network (LFN) と呼ばれる帯域幅遅延積 (bandwidth-delay product) の高いネットワークにおいては，ネットワーク容量を埋めるために必要な通信バッファ量が大きいため，接続数が多いと全体として確保されるバッファサイズが非常に大きくなってしまう．

† 東京大学
University of Tokyo
†† 科学技術振興機構 さきがけ
Presto, JST

そこで，TCP や OS の実装を改変することなしに，確立する接続を制御することによって以上のような問題を解決する．具体的には，我々の設計・開発しているメッセージパッシングシステム Phoenix[3] を拡張し，以下を行う．

- ノード間の RTT とスループットを測定することにより，物理的なネットワークトポロジーを推定する．
- 推定された物理的なネットワークトポロジーを元に，各ノードは計算中に利用する接続を計算する．これにより，ノード間で全対全に接続を確立するのではなく，少数の接続のみ確立する．
- 必要に応じて計算中に動的に接続を追加・削除する．これにより，接続数を一定以下に保ちながらも，他ノードを中継してメッセージを配送するのを極力避ける．

これらの仕組みにより，通信性能を低下させることなく確立する接続数の削減を実現することができる．

このシステムの性能評価を、複数のスイッチにより階層的に構成されるクラスタ上の 128 プロセッサを用いて整数ソートを実行することにより行なった．その結果，全対全に接続を確立した場合と同程度以上の性能を，小さい接続数で達成することできた．また，今回の実験は一つのクラスタ内のノードで行なったが，我々の機構は広域環境にも同様に用いることができると考えている．

本稿は以下のように構成される．2 節でネットワークの自動推定について述べる．3 節で接続の動的な管理機構について述べる．4 節で性能評価実験について述べる．5 節で関連研究との比較を行い，最後に 6 節でまとめと今後の課題について述べる．

2. ネットワークトポロジーの自動推定

この節では，物理的なネットワークトポロジーを推定する機能について説明する．この機構は，SNMP や `traceroute` などから得られる情報を利用せずに，ノード間の RTT とスループットを測定することによって推定を行なう，詳しくは次節で述べるが，推定された情報は，メッセージパッシングシステムが動作中に維持する接続を選択する際に利用される．

具体的には，まず，各ノード間の RTT を測定することにより，どのノードがどのスイッチに属するかを特定する．次に，スイッチ間の通信スループットの干渉関係を測定することにより，スイッチ間の結合関係を推定する．この節の残りで，それぞれの手順の詳細を説明する．

2.1 各ノードが属するスイッチの推定

ノード間の RTT を測定することにより，各ノードが直接つながれているスイッチを特定する．具体的には，ノード u は以下の操作を行なう．

(1) 自分以外の全ノードに対して順番に，ICMP パケットを送信することによって RTT を測定する．数回 RTT の測定を行い，そのうちの最小の値を採用する．

(2) 取得したデータを昇順に並びかえる．この並びかえられた RTT の列を，

$$r_{v(0)}, r_{v(1)}, \ldots, r_{v(N-1)}$$

と表す.（ただし，$r_{v(i)}$ を u と $v(i)$ の間の RTT とし，ノード数を N とする.）

$$d(i) = r_{v(i+1)} - r_{v(i)}$$

とする時，$d(i)$ と $d(i-1)$ の差が著しいときの（例えば差が最大となるときの）i を選び，$j > i$ であるノード $v(j)$ は，u と異なるスイッチにつながっているとみなす．

以上の操作を全てのノードが行い，得られたデータを総合することによって，各ノードが直接つながるスイッチを特定する．

ただし，測定の最中にバックグランドで RTT に影響を与えるプロセスが走っていないなど，物理ネットワーク上でのホップ数の差が，RTT の有意な差として現れることを仮定している．

2.2 スイッチ間の結合関係の推定

各ノードが属するスイッチの特定後，スイッチ間の結合関係の推定を行う．この際，どのスイッチとどのスイッチがリンクを共有しているかは，スループットの変化で推定可能であると仮定している．例えば，スイッチ ab 間でのみ通信をしている時と，cd 間でのみ通信をしている時のスループットが共に 100Mbits/秒とする．そして，ab 間と cd 間で同時に通信している時のスループットは共に 50Mbits/秒となったとする．このとき，ab と cd の 2 つの経路はリンクを共有し，それが通信のボトルネックになったとみなす．

そこで，まず，スループットを測定することにより経路間の干渉関係を求める．具体的には，2 つの異なるスイッチに属するノード間で通信を行い，その時の各ノードのスループットの総計を測定する．また，2 組のスイッチで同時に送信を行い，その時のスループットも同様に測定する．こうして，同時に通信を行うことによってスループットの低下する，互いに干渉する組を記録する．

次に，こうして得た経路の干渉情報を元に，ネットワークトポロジーを推定する．推定を行なう上で，ネッ

トワークトポロジーを木構造に限定する．2.1 節で推定されたスイッチを葉とし，葉から頂点へと新しくスイッチを追加しながら木の推定を行なっていく．**図 1** はその推定アルゴリズムを示しており，アルゴリズム中では以下の変数が用いられる．

- V: 頂点集合．初期値は 2.1 節で求めたスイッチの集合．
- E: 辺集合．初期値は空集合．
- X: まだトポロジーの推定が終わっていないスイッチの集合．初期値は 2.1 節で求めたスイッチの集合で，X が空になった時にアルゴリズムは停止する．
- R: 互いに干渉しない経路の関係．初期値はスループットの測定時に得られたもの．ab，cd 間で同時に通信を行った際に，スループットが低下しなかった場合 $(ab, cd) \in R$ となる．

アルゴリズムが停止したときに，グラフ (V, E) が推定されたネットワークトポロジーを表す．

merge(X, R) は，X 中に含まれるスイッチで以下の条件を満たす集合 $S_1, S_2, \ldots, S_n$ を求める．

$$\forall u, v, w, x \in S_i : (uv, wx) \in R$$

ただし，u，v，w，x は互いに異なるとする．

update(R, T) は，新たに追加されたスイッチの干渉関係を返す．

$$\textbf{update}(R, T) = \{(f(a)f(b), f(c)f(d)) \\ : (ab, cd) \in R\}$$

ただし，

$$f(v) = \begin{cases} s & \text{if } (v, s) \in T \\ v & \text{otherwise} \end{cases}$$

とする．

```
1:  while X ≠ ∅ do begin
2:    if |X| = 2 then begin
3:      E := E ∪ {ab} where X = {a, b};
4:      X := ∅
5:    end else if |X| = 3 then begin
6:      if ∃a, b, c ∈ X : (ab, ac) ∈ R then begin
7:        Let a, b, c be elements of X s.t. (ab, ac) ∈ R;
8:        E := E ∪ {ab, ac}
9:      end else begin
10:       Let s be a fresh switch;
11:       V := V ∪ {s};
12:       E := E ∪ {as, bs, cs} where X = {a, b, c}
13:     end;
14:     X := ∅
15:   end else begin
16:     (S_1, S_2, ..., S_n) = merge(X, R);
17:     T := ∅;
18:     forall i ∈ {1, ..., n} do begin
19:       Let v_1, v_2, ..., v_m be elements of S_i;
20:       Let s be a fresh switch;
21:       V := V ∪ {s};
22:       E := E ∪ {v_1 s, v_2 s, ..., v_m s};
23:       X := X ∪ {s} \ S_i;
24:       T := T ∪ {(v_1, s), (v_2, s), ..., (v_m, s)}
25:     end;
26:     R := R ∪ update(R, T)
27:   end
28: end
```

図 1 ネットワークトポロジー推定するアルゴリズム
Fig. 1 Algorithm for estimating network topology

3. 接続の制御

この節では，我々のシステムがどのように接続を制御するかについて述べる．まず，前節で得られたトポロジー情報を利用して各ノードは接続を確立する．MPI などのように全対全で接続を確立せず，接続を少なく保つ．基本的には，この接続を通してメッセージを配送するが，必要に応じて接続を動的に生成することにより，通信遅延が大きくなったり一つのノードが多数のメッセージを中継したりするのを回避する．

3.1 物理的なネットワークトポロジー情報を利用した接続確立

前節で推定されたネットワーク情報を元に，各ノードは確立する接続を選択する．この際に，確立することによって通信性能を改善する可能性の高い接続を優先的に確立する．そこで，具体的には，各ノードは下の方針で接続を確立・維持する．

- 自分と同じスイッチに属するノードについては，直接そのノードへ接続を張る．
- 自分と異なるスイッチに属するノードについては，各スイッチにつき，それに属するノードのうちどれか一つに接続を張る．

こうして確立された接続を通して各ノードはメッセージの送受信を行う．必ずしも全ノード間で直接接続を確立しているわけではないので，必要に応じてメッセージは複数のノードを中継して宛先へと配送される．各ノードがメッセージを交換しあいながら，ルーティ

ング表を構築することによって，こうした配送経路は求められる☆.

3.2 コネクションキャッシング

上述のように確立される接続のみで通信を行なっている場合，以下のような問題が生じる可能性がある.

- メッセージを宛先へ配送するのに他ノードによる中継が必要な場合，通信遅延が大きくなる.
- 一つのノードが多数のメッセージ送信の中継点になり，CPU や通信性能などの限界によりボトルネックになる.

そこで，こうした問題を解決するために，必要に応じて動的に接続を確立するコネクションのキャッシング機構を導入する．各ノードは他のノードを中継してメッセージが配送された場合，その宛先へと接続を試み（この接続をショートカットと呼ぶ.），以降はその接続を用いて，メッセージを直接宛先へ送信しようとする.

例えば，ノード a がノード b にメッセージを送信する場合を考える.（**図 2** 参照.）この時にノード a は以下のように動作する.

(1) a は，最初は節で述べられた接続を通してメッセージを配送する．この場合，メッセージは他のノードを中継して b へと配送される.

(2) a は，メッセージが迂回して配送されたので，そのメッセージの宛先 b へショートカットを張る.

(3) これ以降，a は b へメッセージを送信する際に，ショートカット接続を通してメッセージを送信する．これにより，余計な第三者の中継無しにメッセージを直接宛先のノードに送信することができ，通信遅延の低下などを避けることができる.

ただし，図 2 中の直線は確立された接続（破線はショートカット）を，矢印はメッセージの配送経路を表すとする.

一定数以上こうしたショートカットが生成された場合には，新たにショートカットを追加する際に，すでに確立済みのショートカットのうちのどれかを代わりに削除する．LRU アルゴリズムによって，最も長い間送信・受信の行なわれていない接続を削除する.

4. 実　　験

これまでに述べてきたネットワークトポロジーを自動推定する機構と接続を制御する機構とを実装し，性能評価を行なった.

☆ これは Phoenix に既に備わってい機能を利用したものである.

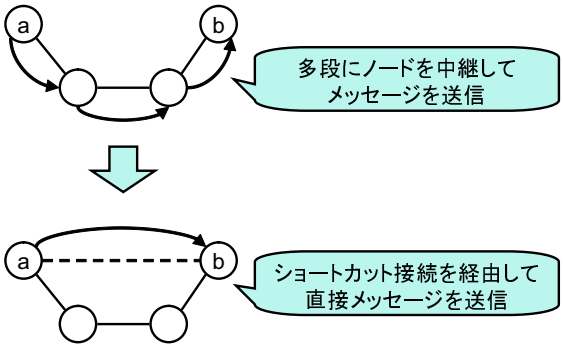

図 2 ショートカット接続の確立
Fig. 2 Establishment of a shortcut connection

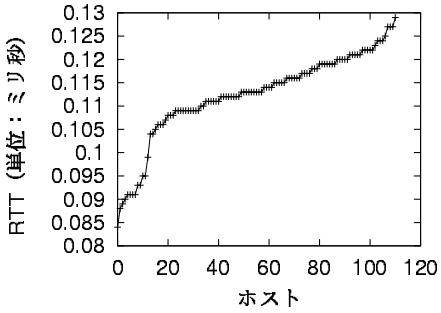

図 4 計測された RTT
Fig. 4 Measured RTT

4.1 ネットワークトポロジーの自動推定

図 3 に示される 112 ノードからなるクラスタ上でネットワークトポロジーの自動推定を試みた．このネットワークでは，各スイッチに 14 ノードづつ分かれてつながれており，スイッチ間も階層的に結ばれている.

このネットワーク環境においては，正しくネットワークトポロジーを推定することができた．例えば，**図 4** は，あるノードにおいて計測された RTT を昇順に並べたものを示している．この図では，13 ノード目と 14 ノード目の間で RTT が大幅に変化しており，この情報を利用して個々のノードが属するスイッチを特定することができる.

4.2 性 能 評 価

NAS Parallel Benchmark[4] の整数ソートを用いて，システムの性能評価を行なった．図 3 中の 128 プロセッサ（64 ノード）を実験に用いた．各ノードの CPU は Intel Xeon 2.40GHz，メモリは 2GB．OS は Linux 2.4 である．これらのノードは 6 つのスイッチにまたがり結合されている.

以下に示すいくつかの接続方式で，整数ソートの実行時間を測定した.（**図 5** 参照.）

- 動的に接続可能な接続の最大数（キャッシュサイ

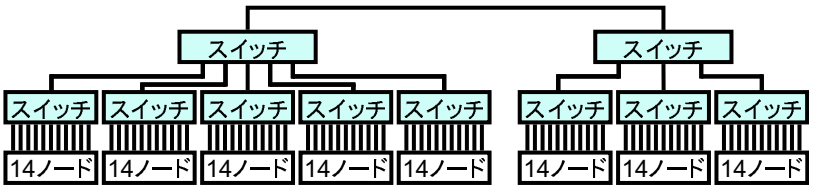

図 3 実験環境
Fig. 3 Experimental environment

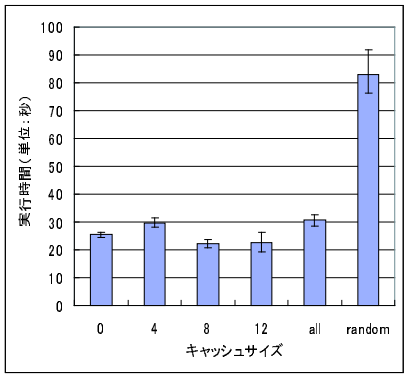

図 5 整数ソートの実行時間
Fig. 5 Execution time of interger sort

ズ）を 0 から 16 にまで変更した場合.（静的に確立される接続数は最大 31 本.）
- 全対全で接続を確立した場合.（図 5 中の all.）
- 各ノードが，確立可能な接続中の半分（64 本）をランダムに選択して確立した場合.（図 5 中の random.）

ただし，整数ソートが使用している MPI の集合通信は，Phoenix においては point-to-point 通信で代替される[3].

図 5 に示された実験結果より，以下の二点が分かる．まず，全ての接続を確立したりランダムに接続を確立するよりも，物理的なネットワークトポロジーを考慮して接続を確立した方が，接続数は少ないにも関わらず，良い性能が達成できているということが分かる．そして，動的に接続の確立を行うことによって性能が向上していることも分かる．

まとめると，以上の実験により，我々の機構が有用であることが示せた．

5. 関 連 研 究

5.1 Grid-enabled MPI

近年，MPICH-G[1]，GridMPI[2], PACX-MPI[5] など，グリッド環境で高性能な並列計算を実現するための MPI の実装が数多く提案・実装されている．しかし，そのどれも，物理的なネットワークトポロジーを考慮に入れた接続の確立や，その動的な制御などを十分に行なっているとはいえない．

PACX-MPI[5]，Stampi[6]，MPICH/MADIII[7] は，複数のクラスタにまたがって多数の接続が確立されるのを回避する．具体的には，ユーザがプロキシを設定し，クラスタ間の通信をこのプロキシで中継することにより実現する．この手法は接続数を削減することができるが，以下に述べるような欠点をもつ．まず，利用者が手動で経路を設定する必要がある．そして，少数のプロキシに通信が集中するため，プロキシがボトルネックになり通信性能が低下しやすい．それに対して我々のシステムは，物理的なネットワークトポロジーを自動推定し，かつ，動的に接続を追加・削除するため，そうした欠点をもたない．

GridMPI は，文献 2) で広域環境で MPI を動作させるにあたって障害となる OS やソケット等の実装の問題点（例えば，接続数の増大によって `select` システムコールが性能低下するなど）を整理している．そして，文献 8) で，多くの接続がある中で，トラップを少なくして効率的に処理を行なうデバイスドライバを設計・実装している．これに対して我々は，OS などの実装を改変することなく，接続を柔軟に制御することによって問題を解決することを目標としている．

MPICH-G では，LAN と WAN の通信性能の差を考慮に入れることにより全体通信を効率的に行う．しかし，接続の制御といった機構は特にもたない．

5.2 ネットワークトポロジーの自動発見

ネットワークトポロジーの発見に関する既存研究の多くは，SNMP や `traceroute` などによって情報が得られることを前提としている[9),10]．それに対して我々は，SNMP などから得られる情報を利用せずともネットワークトポロジーに関する情報を（並列計算を高性能に行なう上で十分な精度において）取得することを目指している．

6. お わ り に

本稿では，動的に接続を制御するメッセージパッシ

ングシステムについて述べた．今後の課題として以下が挙げられる．

より大規模な環境での実験　今現在は小規模な環境でのみ実験を行なった．今後は，より大規模かつ広域な環境（例えば複数のサブネットにまたがる数千ノード程度）でシステムを実行し，性能評価を行う．

スケーラブルなネットワークトポロジーの自動推定　今現在の推定方法のままでは，ネットワークの規模が大きくなるにつれて，RTT やスループットの測定回数が増大する．そのため，並列に測定可能なところは並列に測定を行なったり不必要な測定を省くなどによって，測定時間を短縮することが必須となる．その際に，SNMP や `traceroute` 等から得られる情報も必要ならば補助的に利用する．

動的なネットワーク構成の変化への対応　今現在は，ネットワークトポロジーの推定は静的に行なわれている．そして，その情報を元にして接続の確立を行なっており，動的なノードの参加・脱退などに耐えることができない．メッセージの配送経路を多重化するなどによって，動的なネットワークトポロジーの変化に適応可能にする．

並列ストリーム　LFN などの帯域幅遅延積の高いネットワークにおいて高性能通信を達成する手段として，複数の TCP ストリームを用いてノード間で通信を行う手法が提案されている[11)~13)]．そうした手法を我々の動的な接続の制御と組み合わせることを目指す．

参 考 文 献

1) Karonis, N., Toonen, B. and Foster, I.: MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, *In Journal of Parallel and Distributed Computing (JPDC)*, Vol. 63, No. 5, pp. 551–563 (2003).
2) 石川裕, 松田元彦, 工藤知宏, 手塚宏史, 関口智嗣: GridMPI – 通信遅延を考慮した MPI 通信ライブラリの設計, *In Proceedings of 16th Summer United Workshops on Parallel, Distributed and Cooperative Processing (SWoPP'03)*, 情報処理学会, pp. 95–100 (2003).
3) Taura, K., Endo, T., Kaneda, K. and Yonezawa, A.: Phoenix: a Parallel Programming Model for Accommodating Dynamically Joining/Leaving Resources, *In Proceedings of 9th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP'03)*, pp. 216–229 (2003).
4) NAS Parallel Benchmark. http://www.nas.nasa.gov/Software/NPB/.
5) Gabriel, E., Resch, M., Beisel, T. and Keller, R.: Distributed Computing in a Heterogeneous Computing Environment, *In Proceedings of 5th European PVM/MPI Users' Group Meeting (EuroPVM/MPI'98)*, pp. 180–187 (1998).
6) Imamura, T., Tsujita, Y., Koide, K. and Takemiya, H.: An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers, *In Proceedings of 7th European PVM/MPI Users' Group Meeting (EuroPVM/MPI'00)*, pp. 200–207 (2000).
7) Aumage, O. and Mercier, G.: MPICH/MADIII: a Cluster of Clusters Enabled MPI Implementation, *In Proceedings of 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'03)*, pp. 26–33 (2003).
8) 松田元彦, 石川裕, 工藤知宏, 手塚宏史: MPI 通信モデルに適した非同期通信機構の設計と実装, *In Proceedings of 2nd Annual Symposium on Advanced Computing Systems and Infrastructures (SACSIS'04)*, pp. 365–372 (2004).
9) Bejerano, Y., Breitbart, Y., Garofalakis, M. and Rastogi, R.: Physical Topology Discovery for Large Multi-Subnet Networks, *In Proceedings of 4th IEEE INFOCOM'2003*, pp. 342–352.
10) Yao, B., Viswanathan, R., Chang, F. and Waddington, D.: Topology Inference in the Presence of Anonymous Routers, *In Proceedings of 4th IEEE INFOCOM'2003*, pp. 353–363.
11) Denis, A., Aumage, O., Hofman, R., Verstoep, K., Kielmann, T. and Bal, H. E.: Wide-Area Communication for Grids: An Integrated Solution to Connectivity, Performance and Security, *In Proceedings of 13th IEEE International Symposium on High-Performance Distributed Computing (HPDC'04)* (2004).
12) Kamezawa, H., Nakamura, M., Inaba, M. and Hiraki, K.: Coordination between parallel TCP streams on Long Fat Pipe Network, *In Proceedings of 1st International Workshop on Data Processing and Storage Networking: Towards Grid Computing (DPSN'04)* (2004).
13) GridFTP Protocol Specification (2003). Global Grid Forum Recommendation GFD.20.