ユーザーレベル通信ライブラリにおける packet ベース通信 API

渕田良彦[†] 河野真治[††]

我々は並列計算機，特に PC クラスタ上で密結合型通信を用いる計算モデルを解くために，ユーザーレベル通信ライブラリを用いた Packet ベース通信 API を提案する．本稿では，本研究室で開発されたトランスポート層プロトコルに UDP を採用したユーザーレベル通信ライブラリ Suci(Simple User level UDP Communication Interface) に Packet ベース通信 API を実装し，通信速度の比較，評価を行う．

Packet based communication API in the user level communication library

Fuchita Yoshihiko[†] and Shinji Kono[††]

We propose the Packet base communication API which used the user level communication library in order to solve the calculation model using tightly-coupled communication by parallel computer and PC cluster. In this paper, Packet base communication API is mounted in the user level communication library Suci (Simple User level UDP Communication Interface) which adopted UDP as the transport layer protocol developed at this laboratory, and comparison of transmission speed and evaluation are performed.

1. はじめに

近年，科学技術計算などの問題領域を目的とした並列分散システムはその性能価格比の向上や利用技術の進歩によって急速に普及した．これらの並列分散システムではプロセス間の通信，すなわちデータ転送や同期を行うためのメッセージ通信ライブラリが提供される．プログラム作成者はこのライブラリを利用して C 言語や FORTRAN 言語などで並列処理を行うプログラムを作成する．

ここで，並列分散システム用に提供されているメッセージ通信ライブラリにおいて，もっとも普及したものは MPIF (MPI Forum) において策定された，メッセージ通信ライブラリの標準仕様 MPI (Message Passing Interface) Standard [Wal94, For94, GLS94] であるが，MPI による並列計算は，バリア同期や配列の配布・収集を主としている為，コンテクストやデータ型の扱いを実現するために 1 回のメッセージ通信あたりのソフトウェアオーバーヘッドが非常に大きくなってしまう．そのため，現在の MPI の実装では粒度が小さく密結合型通信を必要とする計算モデルを扱うことが難しい．

今日，並列分散システムを現状以上に有用なものにするには，細粒度かつ非定型な数値計算を十分に支援する必要があると考えられるが，これらの密結合型並列計算では，リモートデータへのアクセスが小さいオーバーヘッドで実現できることが重要になる．例えば，順序機械などの非定型な状態を保持する構造データを用いた並列計算では通信遅延が多大なボトルネックとなる．

以上のことより，現状の MPI の実装を用いることは並列分散システムの応用分野を疎粒度あるいは定型的なものに限定してしまうという結果になる．

よって本研究では，ポーリングベースの非同期型通信を行い，ユーザーレベルにおけるフロー制御の記述を可能とした UDP ベースの通信ライブラリ，Suci を用いて細粒度な並列計算に耐え得る並列計算機向け通信ライブラリを開発することを目標として，Suci ライブラリにおけるソフトウェア的オーバーヘッドとなっている仕組みを解消し，ユーザープログラムレベルでトランスポート層を意識したデータ型を持つ通信 API，Packet ベース通信 API を実装し，その評価を行った．

† 琉球大学理工学研究科情報工学専攻
Interdisciplinary Infomation Engineering, Graduate School of Engineering and Science, University of the Ryukyus.
†† 琉球大学工学部情報工学科
Information Engineering, University of the Ryukyus.

2. Suci ライブラリの特徴

当研究室で開発された Suci ライブラリは並列分散環境における通信ライブラリである．その特徴として以下の点が挙げられる．

- トランスポート層プロトコルに UDP を採用し，信頼性のない UDP での通信に信頼性を付加している．
- UDP ベースなので TCP と比べてカーネルの資源消費を最小限におさえられる．
- ユーザーレベルでフロー制御，輻輳制御が可能．
- ポーリングベースの通信ライブラリなので無駄な CPU 時間を消費しない．

表 1 に Suci が提供している主な API を示す。

Suci API	
機能	API 名
アドレス DB ファイルを読む	addressdb_lex (fp, addressdb)
通信ソケットを開く	datagram (addrdb, myaddr, myport, myid)
通信宛先を選択する	datagram_destination (distid, sock)
データを送信する	datagram_write (sock, buf, len)
データを読み込む	datagram_read (sock, buf, len)
再送キューの長さを得る	datagram_queue_length (sock, id)
パケットをチェックする	datagram_ready (sock, sec, msec)
指定したノードへの再送処理	datagram_retransmission (sock, destid)

表 1 Suci の主な API

表 1 に示すような API を用いた典型的な並列分散プログラムは以下に示すようになる．

```
while(1){
   if(datagram_read(sock,buf,size)){
      {
      /* read したデータの処理 */
      }
   } else {

      /* ブロッキングしなかったときの処理 */
      /* 送信先の指定 */
      datagram_destination(sock,id1);
      /* SIZE の分だけ送信 */
      datagram_write(sock,buf,size);
      {
         /* write 後の処理 */
      }
   }

   /* 再送 queue を check */
   if(datagram_queue_length(sock,id2)){
      /*必要なら再送を行う*/
      datagram_retransmission(sock,id2);
   }
}
```

ここで，Suci ライブラリを用いた通信の流れについて説明する．Suci ライブラリを用いた通信アプリケーションは，addressdb というノード情報を記述したファイルを addressdb lex() によって解読し，datagram destination() で各ノード情報の確保とソケットを開く処理を行う．それらの処理が完了したら，datagram destination() によって送信先を指定し，datagram write によって出力を行うが，この時，必要ならば再送の処理を行う．再送処理は、データを送信した後に一時的に確保される再送 queue の状況をチェックし，datagram ready() によって確認応答の処理を行いつつ datagram retransmission() を呼び出すことにより提供される．このような仕組みを持つため，再送 queue の状態によって輻輳やバッファオーバフローをアプリケーションが検知し，再送の量を調節したり，再送タイミングを送らせることによってフロー制御を行うことが可能である．また，データ受信時には，まず datagram read() から datagram ready() が呼ばれることによって read ブロックを組み立て，ブロックが完成していれば，datagram read() がアプリケーションにデータを渡す．

2.1 Suci の問題点

これまで，Suci ライブラリの並列分散環境での通信ライブラリとしての利点となる部分を述べたが，現在の Suci ライブラリには，細粒度な密結合通信を用いる問題を解くのに，ボトルネックとなる部分が既存の通信 API である datagram write(),datagram read() に存在する．それは，既存の通信 API において，送受信バッファに対してメモリコピーオペレーションが用いられているということである．このような仕組みではメッセージ通信のバッファの確保・解放，コピーなどのオーバーヘッドが生じる．また，これらの通信 API では単一のバッファを用いてデータの送受信を行っているが，このようなデータ構造では，データを一つ送受信するごとにブロッキング・デブロッキング

やパケットの再送処理が必要となる．これらはデータ通信におけるオーバーヘッドとなる．その様子を図 1 に示す．

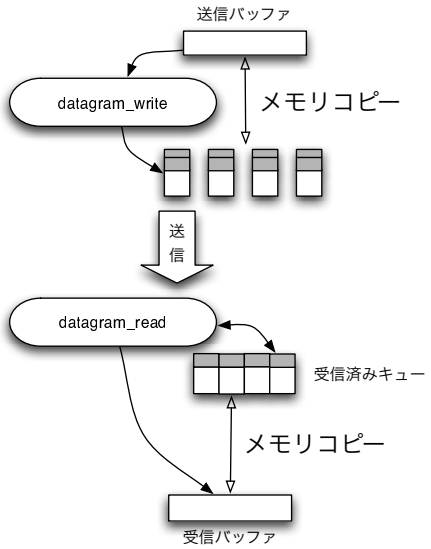

図 1 Suci ライブラリによるデータ送受信

3. 通信オーバーヘッドに考慮した通信 API の実装

並列分散システム，特に分散メモリ型並列計算機では，ノード数 (プロセッサ数) の拡張が比較的容易であるが，今回目標とするような比較的通信量の多い並列計算において，拡張したノード数に応じた高い性能を得ることは難しい．それは，ノード数が増えれば，ブロッキング・デブロッキング処理やデータ及び，確認応答の通信回数が増加し，実行時間に占めるそれらの処理時間が増大して性能を低下させる傾向があるからである．

これらの問題を解決するには，ネットワークのスループットを上げ，レイテンシを下げることが重要であり．そのため，プロセッサ性能，バス速度，ネットワーク機器などのハードウェアの性能は飛躍的に向上してきた．しかし，このようなハードウェアの性能向上によってネットワークのスループットが向上し，レイテンシが縮小されると，並列分散システムにおけるデータ通信の処理のうち，ソフトウェアで行われている通信データのハンドリング部分，即ちソフトウェアによる通信オーバーヘッドが並列計算機の性能低下の主たる要因になってくる．

3.1 packet ベース API

上記のような問題は Suci ライブラリにおいても存在し，前節で挙げたメモリコピーによるオーバーヘッド以外にも，通信データ構造によるデータハンドリング処理のオーバーヘッドがある．その為，本研究では Suci ライブラリで細粒度な問題を解くために，送受信バッファにダブルリンクドリストを用いるデータ構造を持つことによりデータの送信回数，送信確認応答数の削減，及びデータ構造によるデータハンドリング処理の低減を実現し，また，送受信の際にデータのコピーを行わない．つまり，データ通信のバッファの確保・解放，コピーなどのソフトウェア的なオーバーヘッドを減らすといった 2 つの特徴を持つ通信 API を Suci ライブラリに実装した．これらの通信 API を packet ベース API と呼ぶ．

Suci ライブラリに実装した packet ベース API の詳細は以下のようになっている．

void datagram_packet_put(datagram_socket *sock, SP_datagram_dlist dlist, int dest)

表 2 データ送信 API datagram_packet_put

datagram_packet_put() はデータ送信の際にメモリコピーを行わず，送信データを SP_datagarm_dlist という型で宣言されたダブルリンクドリスト（dlist）に格納し，宛先 ID(dest) で示されたホストへ送信する．通信のバッファにダブルリンクドリストを用いることで，一度に複数のデータをセットして受信ホストに送信する．

SP_datagram_queue *datagram_packet_get (datagram_socket *sock)

表 3 データ受信 API datagram_packet_get

datagram_packet_get() は Suci ライブラリによって与えられる受信済みブロックキューより datagram_packet_put API の送ったダブルリンクドリストを切り離し、戻り値としてそのダブルリンクドリストへのポインタをユーザープログラムへ返す。

4. packet ベース API のプログラミングスタイル

packet ベース API を用いた Suci ライブラリでのプログラミングの例を以下に示す.

```
/*----------- データ送信処理部分 ------------*/
//ダブルリンクドリストを size 分だけ確保
send_dlist = datagram_new_packet(sock, size);

/*  ここでリストへのデータ格納処理 */

//データを宛先 ID(2) へ送信する
datagram_packet_put(sock, send_dlist, 2);
//Ack が取れるまで，データを再送信
while(datagram_queue_length(sock, 2)){
   datagram_ready(sock, 0, 10);
   datagram_retransmission(sock, 2);
}
/*----------- データ受信処理部分 ------------*/
for(;;) {
   //パケットのチェック&ポーリングルーチン
   while(datagram_ready(sock, 0, 10)) {
   //受信ブロックキューが完成したら以下を行う
      recv_queue = datagram_packet_get(sock);
      //ACK 送信
      ack_sync(sock, 2);
      //受信したのでループを抜ける
      goto next;
   }
   /* ここでデータ読み込み時以外の処理 */

}
next:
//データ受信後の処理
```

packet ベース API を用いたプログラミングは，データ送信処理においては，送信バッファのダブルリンクドリストを datagram_new_packet() という API を用いて，引数 size 分の領域のバッファを得たあと，ユーザーがバッファに対してデータを格納する処理を行う．データをダブルリンクドリストのバッファに格納したら，datagram_packet_put() を用いて受信プロセスへデータを送信する．また，データ受信処理において，datagram_ready() というポーリングして待ち，パケットを受信したら，そのパケットから受信済みブロックキューをつくる API によってループ構造を持つ．datagram_ready() は受信済みブロックキューにダブルリンクドリストが一つ完成したらを 1 を返し，それ以外の場合は 0 を返す．1 が帰ってきた場合はプログラムの制御は datagram_packet_get() へと移り，ack_sync() で受信パケット分の ACKnowledge を返す．以上が packet ベース API を用いた Suci ライブラリでの典型的なプログラミングスタイルである．

4.1 packet ベース API による通信

次に本研究で Suci ライブラリに実装した packet_put /get/ready 方式による通信の詳細について説明する．本方式は並列計算環境において送受信するデータ構造を重視した通信方式であり，そのデータ送受信の流れは図 2 及び, 下記 (1)〜(9) のようになる．

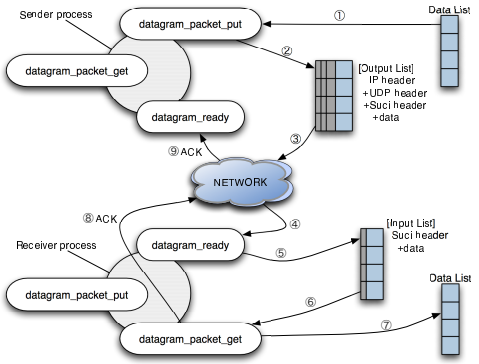

図 2 packet ベース API による通信

(1) 送信プロセスにおいてユーザーはデータの送信用にダブルリンクドリストを持ち，そのリストに送信するデータを格納して，宛先 ID とそのリストを datagram_packet_put に与える．

(2) データを受け取った datagram_packet_put はそのリストに必要なヘッダ情報を付加する．

(3) 受信プロセスへとデータを送信する．送信したリストは送信済みリストとして一時保存される．

(4) ポーリングして待っていた datagram_ready がパケットを受信する．

(5) datagram_ready は受信パケットのヘッダ情報を解析し，それからリストが完成するまで受信済みキューにデータを格納する．

(6) 受信済みキューにリストが一つ完成したら，datagram_packet_get へ制御を移す．

(7) databram_packet_get は受信済みキューからリスト一つを切り離し，受信データとして切り離したダブルリンクドリストを返す．

(8) 受信したリスト分のパケットに関して ACKnowledge を送信プロセスへ送信する．

(9) 送信プロセスの `datagram_ready` は ACKnowledge を受信すると，その分のリストを送信済みリストから削除する．以上でデータ送信の流れは終了する．

5. 評　　価

実装した packet ベース API での通信と Suci ライブラリ既存の通信 API である datagram_read(), datagram_write() を用いる通信のスループットを本研究室の PC クラスタシステムを用いて比較した．

行ったのは (1).2 ノード間での ping pong 通信 (図 3),(2).30 ノードでの broadcast 通信 (図 4), (3).10 ノードでの N 個の数値データの broadcast 通信 (図 5) である．

(1),(2) において，このベンチマークでは 128〜524288byte の送信バッファ量における転送速度を測定し，また (3) においては 1 100 個までの数値データを datagram_read(), datagram_write() を用いて送信する場合において, データを N 個送ってからそれらの ACK を待つ実装 (手法 1) と, データを一つ送るごとに ACK を待つ実装 (手法 2) を用いて packet ベース API を用いた通信との比較を行った．

5.1 評 価 環 境

ベンチマーク測定環境には本研究室の PC クラスタシステムを用いた．表 4 にその仕様を示す．

マシンスペック/ノード	
CPU	Pentium3 800Mhz
メモリ	512MB
バスクロック	133Mhz
NIC	100 BASE-TX
OS	Linux 2.4.27
スイッチ	Catalyst C2980-GA

表 4 ベンチマーク測定環境

5.2 ping pong 通信，broadcast 通信によるスループット

(1).2 ノード間での ping pong 通信,(2).30 ノードでの broadcast 通信のスループット測定結果をそれぞれ図 3, 図 4 に示す．

これらのベンチマークにおいて，2 ノード間の ping pong 通信及び，5 ノード,30 ノードでの broadcast 通信では転送容量の大きい場合においてスループットの向上がみられた．これはメモリコピーを無くした事によるオーバーヘッドの差による影響と考えられる．しかし，128〜65538byte の間には両者に大きな違いが見られなかった．これはメモリコピーの試行回数が今回のような単純なスループットを計測するベンチマークでは足りなかったためと考察する．

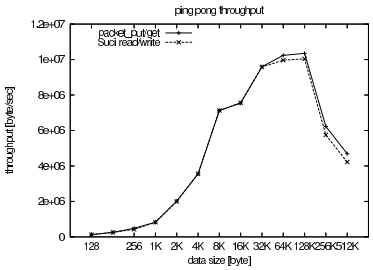

図 3 pingpon 通信のスループット

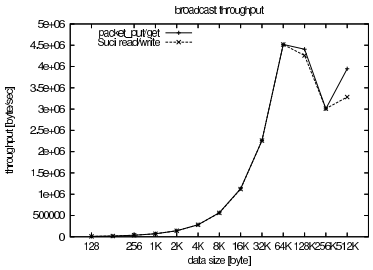

図 4 broadcast 通信のスループット

5.3 N 個の数値データの broadcast 通信によるスループット

(3).10 ノードでの N 個の数値データの broadcast 通信の測定結果を図 5 に示す．こののベンチマーク測定では,

- データを N 個送ってからそれらの ACK を待つ実装 (手法 1).
- データを一つ送るごとに ACK を待つ実装 (手法 2).
- packet ベース API による実装.

を用いて比較を行った．

N 個の数値データの broadcast 通信においては手法 1 は転送するデータ数が多いと ACK を受け取れなくなった、これは一気にパケットを送信したためネットワークの輻輳が起こった為と考えられる。また、手法 2 においてはブロッキング／デブロッキングのオーバーヘッドにより他の 2 手法ほどのスループットは出なかった．packet ベース API の通信においては安定して高いスループットを得る事ができた。

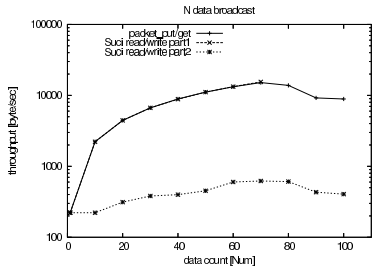

図 5 N 個の数値データの broadcast 通信のスループット

6. まとめと今後の課題

本稿では，UDP に信頼性を付加した並列分散環境向け通信ライブラリである Suci において，細粒度かつ密結合型通信を必要とする問題を PC クラスタを用いて解く事を目指した，通信データのハンドリング処理におけるソフトウェア的オーバーヘッド，及び通信バッファのデータ構造に考慮した通信 API を実装し，その実装方法，及び実装 API の性能評価について述べた．

また，今後の課題として，実装した API を用いて順序機械を用いた標準的関数の問題を並列計算量のクラスであるクラス NC スケールで解けることを，Suci ライブラリでの実装を用いて示す事，及び Suci ライブラリの再送制御やフロー制御といった現在通信のボトルネックとなっている部分の再実装などが挙げられる．

参 考 文 献

1) Ladner, R.E. ,Parallel prefix computation, J.ACM 27,pp831-838,1975
2) Postel, J. (ed.), TransmissionControl Protocol, DARPAInternetProgramProtocol Specification,RFC793,1981.
3) 並列分散ライブラリ Suci の実装と評価, 屋比久友秀, 河野真治, システムソフトウェアとオペレーティング・システム研究報告,2001
4) トランスポート層を考慮したスナップショット・アルゴリズムの考察, 屋比久友秀, 河野真治, 山城潤, 日本ソフトウェア科学会第 19 回大会 (2002 年度) 論文集,2002
5) PC クラスター向けの通信ライブラリー Suci for Java の開発, 山城潤, 琉球大学工学部情報工学科平成 14 年度卒業論文,2003
6) 並列アルゴリズム 理論と設計, 宮野悟, 近代科学社, 1993