システムソフトウェアと
オペレーティング・システム 82−11
（1999. 8. 2）

ギガビットネットワークを用いた並列プロセスマイグレーションの性能評価

西岡 利博　　堀 敦史　手塚 宏史
エム・アール・アイ・システムズ　新情報処理開発機構
{nishioka,hori,tezuka}@trc.rwcp.or.jp

本論文は、ギガビットネットワークを用いたクラスタ上での、並列プロセスマイグレーションの実現方式と、PC クラスタ上での予備評価について述べる。実現には、著者らが既に開発したチェックポイント機構を応用する。チェックポイントでは、プロセスの状態をファイルに保存するのに対して、並列プロセスマイグレーションではネットワークを通じて移送するので、その性能はネットワークの転送性能に大きく依存する。並列プロセスマイグレーションでは、複数のノードが一斉に、それぞれ異なるノードにデータを転送する。これまでそのような通信パターンでの性能評価が得られていなかったので、実施した。その結果、ネットワークがボトルネックとなり、16 以上のノードが同時にデータを転送する場合は、ノードあたり 16 MByte/sec 程度であり、各ノードでローカルディスクにチェックポイントする場合の 4 – 10 倍の性能であることが判明した。

Performance Evaluations of Parallel Process Migration with a Giga-bit Network

Toshihiro Nishioka　Atsushi Hori　Hiroshi Tezuka
M.R.I. Systems　Real World Computing Partnership
{nishioka, hori, tezuka}@trc.rwcp.or.jp

This paper describes a parallel process migration technique for clusters with a giga-bit network and its preliminary evaluation on a PC cluster. The checkpointing technique proposed by the authors will be applied to the implementation. The performance of the parallel process migration substantially depends on the network bandwidth because the status of processes are transferred through the network in case of migration as opposed to they are stored in file(s) in case of checkpointing. In parallel process migration, more than one node transfer data at same time, and the destination node differs each other. Since no evaluation report was available on the network performance of clusters for such a communication pattern, authors have made it. The result shows that due to the network bottleneck, the bandwidth per node is about 16 MByte/sec when more than or equal to 16 nodes transfers data at same time. This performance is 4 – 10 times faster than the checkpointing in which the status of the processes are stored to the local disk at each node.

1　はじめに

並列計算機やクラスタでは、ユーザ並列プロセスが投入されると、それを実行するノードを、全体の負荷が均衡するように割り当てるが、並列プロセスの終了により負荷の均衡が崩れる可能性がある。このとき、一部の並列プロセスを、負荷の高いノードから負荷の低いノードに移送 (マイグレーション) することができれば、この偏りを解消できる。

本論文では、並列プロセスを異なるノードへ移送することを「並列プロセスマイグレーション」(以下、PPM と略す) と呼ぶ。PPM は、並列プロセスの無矛盾な大域的状態を確保し、それを異なるノードで再現し、実行を再開することで実現される。

無矛盾な大域的状態を確保するという意味では、並列プロセスのチェックポイント [6] も、共通の技術要素を持つ。むしろ、チェックポイントは、時間軸方向の PPM と考えることもできる。このため、既にチェックポイント機構がある環境で PPM を実現しようとするとき、そのためにチェックポイント機構を利用するのは自然な発想と言える。

我々は、既に、クラスタ上で並列プロセスのチェックポインティング機構を実現している [12]。このチェックポインティング機構は、特定の通信方式に依存せず、いつでもチェックポイント可能で、チェックポイントしている時間以外には、アプリケーションプログラムの実行にオーバヘッドを生じないという特長を持つ。このチェックポインティング機構を PPM の実現に利用すれば、同じ特長を持つ PPM を実現できることが期待される。

チェックポイントは、並列プロセスの状態をステイブルストレージに出力するので、チェックポイントするノード全体から、ステイブルストレージへのデータ転送のバンド幅が、性能を決める支配的な要素となる。これに対し、PPM は、並列プロセスの状態を、ネットワークを通じて他のノードに転送するので、転送元のノードから転送先のノードへのデータ転送のバンド幅が支配的な要素となるだろう。我々のチェックポイント機構 [12] は、ステイブルストレージとして各ノードのローカルディスクを用いるので、ほぼスケーラブルな性能が得られるが、ネットワークの特性によっては、PPM が同じ性質を持つとは限らない。

PPM の設計に際しては、プロセスの状態を転送する際の転送バンド幅を見積もる必要がある。PPM の通信パターンは、複数のノードが一斉に大量のデータを他のノードに送信するタイプの通信パターンである (図 1)。これは、ポイント対ポイントの通信の性能とは性質が異なる可能性があり、ポイント対ポイントの性能評価から、ただちに PPM のデータ転送性能を見積もることはできない。PPM の実装に先立って、この通信パターンでのデータ転送性能を評価しておく必要がある。

本論文は、最初に、我々のチェックポインティング機構を利用した PPM の実装方式を示し、続いて、この方式で PPM を実現するに先立ち、PC クラスタ上で複数のノード間のデータ転送の性能を評価した実験の結果を示す。

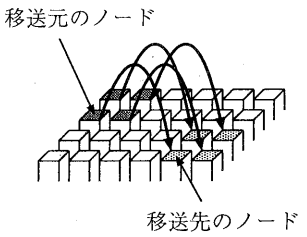

図 1: PPM の通信パターン

2 関連研究

プロセスマイグレーションは、分散 / 並列計算環境における負荷平衡に必要な技法であり、多くの研究がなされてきた。しかし、これまでの研究の多くは、LAN で接続された分散計算環境でノード間の負荷を調整するために、並列プロセスの一部のプロセスを他のノードに移すタイプのものであり (例えば [7, 1])、本研究のように、並列計算環境上で、並列プロセス全体を移送する報告例はない。

3 本研究の目的と範囲

本論文は、PC クラスタ上で、チェックポインティングの機構を利用した PPM が実用的な性能を得られる見通しを得ることを主題としている。

研究のプラットフォームとして、RWC PC Cluster II[10] (以下、PCC-II と略す) を用いている。

PCC-II のネットワークの性能評価は、これまでポイント対ポイントでなされてきた [8, 9] ため、図 1 のように、複数のノードがそれぞれ異なるノードに一斉に転送するパターンでの転送性能を計測した結果は得られていない。そこで、本論文では、本方式の PPM の実現に先立ち、複数のノードがそれぞれ異なるノードに一斉に転送するパターンでの転送性能を計測することで、PCC-II 上での PPM の性能を見積もり、実用的な性能を得られそうか否かを評価する。目安として、チェックポイントと同程度の性能が得られれば、実用的であると考えられる。

プラットフォームである PCC-II は 128 ノードで構成され、本計測も、最大 128 ノードで行われる。

4 SCore-D の概要

我々は、PC クラスタ上に SCore-D と呼ばれるクラスタ管理ソフトウェアを開発している [2, 11, 3]。SCore-D はクラスタを総体としてひとつのシステムとしてユーザに見せ、多重並列プログラミング環境を提

供することを目的とする。

効率の良い多重プログラミング環境を実現するために、SCore-D は、ギャングスケジューリングによる時分割スケジューリングを実現している。

SCore-D は、新たな並列プロセスを生成するとき、それに対して仮想ネットワークをひとつ割り当てる。並列プロセスの切替は、切替え前の並列プロセスのメッセージがネットワーク上からなくなるまで待ってからネットワークコンテクストを切替え後の並列プロセスのものに入れ換え、それから切替え後の並列プロセスを実行させることで行われる。この、ネットワークの切替の機構を、ネットワークプリエンプションと呼ぶ。並列プロセス切替に要する時間は、64 プロセッサの場合で 4 msec 以下である [11, 3]。

既に SCore-D 上にはチェックポイント機構が実現されている [12]。

SCore-D のチェックポイント機構では、ユーザの並列プロセスの大域的状態を、各ノードのユーザプロセスの状態と、SCore-D から提供を受けた仮想ネットワークの状態 (ネットワークコンテクスト) の組とみなす。

SCore-D のチェックポイント機構は、ネットワークプリエンプションを利用することで、ネットワーク上にメッセージのない状態を確保する。このための同期と、ネットワークコンテクストの退避、再起動時のネットワークコンテクストの復元は、SCore-D が担当し、ユーザプロセスの状態の退避と、再起動時の復元は、ユーザプロセス側にリンクされているチェックポイントライブラリが担当する。ネットワークコンテクストを SCore-D が取り扱うのは、ネットワークコンテクストに含まれる通信ハードウェアの状態を保護する必要があるからである。

SCore-D のチェックポイント機構は、こうして確保したユーザプロセスの状態とネットワークコンテクストを、それぞれのノードでローカルディスク上のファイルに書き込む。後でそれを読み込むことで、チェックポイントした時点から、実行を再開できる。

5 並列プロセスマイグレーションの設計

本研究の PPM は、PCC-II 上で、クラスタ管理ソフトウェア SCore-D の管理下で動作するよう設計されている。

本方式の PPM は、SCore-D のチェックポイント機構が提供する、無矛盾な大域的状態の確保と、その復元の機能を利用する。これにより、SCore-D チェックポイント機構の利点を受け継いだ PPM が実現されることが期待される。

SCore-D のチェックポイント機構の機能を利用して得られたプロセスの状態およびネットワークコンテクストは、ファイルに書き込むのではなく、ネットワークを通じて移送先のノードへと転送し、移送先のノードでそれを読み込んで実行を再開する。

チェックポイント機構では、ネットワークコンテクストの退避 / 保存 / 復元は、通信ハードウェアの保護のために SCore-D が担当したが、PPM でも、ネットワークコンテクストの退避 / 移送 / 復元は、SCore-D が担当することで、保護を実現する。

チェックポイントと異なり、PPM では、異なる並列プロセス間でデータの受渡しをしなければならないという問題がある。

多重計算環境を提供する並列計算機では、ある並列プロセスの通信が、別の並列プロセスの通信の正しさに影響を与えることがあってはならない。このため、SCore-D では、並列プロセスに仮想ネットワークを提供するときに、異なる仮想ネットワークの通信が干渉し合わないようにするための保護機構を提供している。異なる並列プロセス間でのデータの受渡しはこの保護機構によって阻まれるため、チェックポイントの場合のように、ユーザプロセスが自分で転送することができない。このため、ユーザプロセスは SCore-D に転送すべきデータを引き渡し、データの転送は SCore-D が担当する設計とした。

PPM におけるデータ転送には、次の性質がある。ある並列プロセスを実行するために割り当てられた一定の範囲のノードから、それと同じ大きさを持つ、移送先のノードへデータが転送される (図 1)。転送されるのは、そのプロセスの状態を表すデータであるが、それは主としてプロセスのメモリの内容であり、その量は、プロセスが確保したメモリの量に大きく依存する。転送するデータを準備するための時間は短く、評価にあたっては無視できる。移送元は、プロセスのメモリ内容を端から送っていき、移送先では、それを、移送元と同じアドレスのメモリ空間に端から埋めていく。典型的には、数 MByte ～ 数十 MByte の大きさのデータが、その並列プロセスを実行しているノードの数だけ、一斉に転送されることとなる。

以上から、PPM の性能に影響を与える要素としては、下記があると考えられる。

- ネットワーク上で並列プロセスの状態を転送する性能
- 移送元 / 移送先の各ノードで、ユーザプロセスと SCore-D との間でプロセスの状態を転送する性能 (SCore-D とユーザプロセスのプロセス切替のオーバヘッドを含む)

本論文では、前者の、並列プロセスの状態の転送の性能を評価した結果を報告する。

本方式の PPM は、下記の手順で実施される (図2)。

1. SCore-D に、 PPM を要求するシステムコールが発行されると、 SCore-D は、移送先にユーザプロセスを作成する。このユーザプロセスは、PPM の移送先処理のためのシグナルハンドラに入り、移送元のプロセスの状態を受けとれる状態になったことを SCore-D に通知してから、待機する。

2. 移送先の SCore-D が、移送先の全プロセスが待機状態に入ったことを確認してから、移送元の SCore-D に PPM 開始を指示する。移送元の SCore-D は、移送対象のプロセスの存在する全ノードで、対象プロセスに、 PPM を指示するシグナルを送出し、その後、対象プロセスを再開させる。

3. 対象プロセスでは、 PPM のためのシグナルハンドラに制御が移り、 SCore-D に対して、PPM 開始のためのバリア同期を要求する

4. SCore-D は、対象プロセスのネットワークコンテクストを移送先ノードの SCore-D に転送する。移送先ノードの SCore-D は、移送元から転送されてくるネットワークコンテクストを受け取り、待機中のプロセスが復帰時に使用すべきネットワークコンテクストとして設定する。

5. 全ノードが PPM の開始を要求したところで、SCore-D は、対象プロセスに PPM 処理を再開させる。対象プロセスは、 SCore-D に対して、チェックポイント機構が退避するのと同様の内容のプロセスの状態を出力し、SCore-D がそれを移送先ノードの SCore-D に転送する。

6. 移送先ノードの SCore-D は、移送元ノードから転送されてくるユーザプロセスの状態を受け取り、待機中のユーザプロセスに転送する。

7. 移送先ノードのユーザプロセスのシグナルハンドラは、 SCore-D から転送されてきたプロセスの状態を受けとり、復元する。

8. 移送先ノードのユーザプロセスのシグナルハンドラが、 SCore-D に、プロセス実行の再開のための同期を要求する。

9. SCore-D は、移送先のすべてのノードで同期が要求されるのを待ち合わせ、プロセスの実行を再開する。

10. ユーザプロセスがシグナルハンドラから抜け出し、移送開始前の状態で計算を再開する。

11. SCore-D が移送元の並列プロセスを破棄する。

6 評価

評価は、移送元のプロセスから、それぞれ 128 MByte のデータを移送先のプロセスへ転送し、その所要時間を測定することで実施した。移送先で実行を再開するわけではない。転送性能として、各移送元で記録した所要時間を平均した値を元に、ノードあたりのバンド幅を計算する。

プログラムは C で記述した。

データ転送には MPICH-PM[4] を用いた。この MPI の実装は、メッセージ通信のバンド幅を確保するため、バッファサイズが 8 KByte 以上になると、Myrinet の DMA 機構を活用したゼロコピー通信を使用する [9, 5].

移送元のプロセスと移送先のプロセスは、すべて、SCore-D 上の単一の並列プロセスの一部として実現した。この計測中 SCore-D の時分割スケジューリングは停止した (実際には、十分長い時分割間隔を指定した)。このため、計測結果には，ギャングスケジューリングの影響は含まれない.

表 1 に、評価に用いた PCC-II の仕様を示す[1]。

表 1: RWC PC Cluster II の仕様

#-of Compute Nodes	128
Network	Myrinet
Node Processor	PentiumPro
Chip Set	440FX
Clock [MHz]	200
Memory [MByte]	256
Disk [GByte]	4
I/O Bus	PCI (133 MByte/sec)
Local OS	Linux

PCC-II で使用されている Myrinet は、下記の性質を持つネットワークである。

[1]文献 [10] では 64 プロセッサ構成であったが、現在は 128 プロセッサ構成に拡張され、ローカル OS も Linux に変更されている。また SCore-D も Linux に移植された。

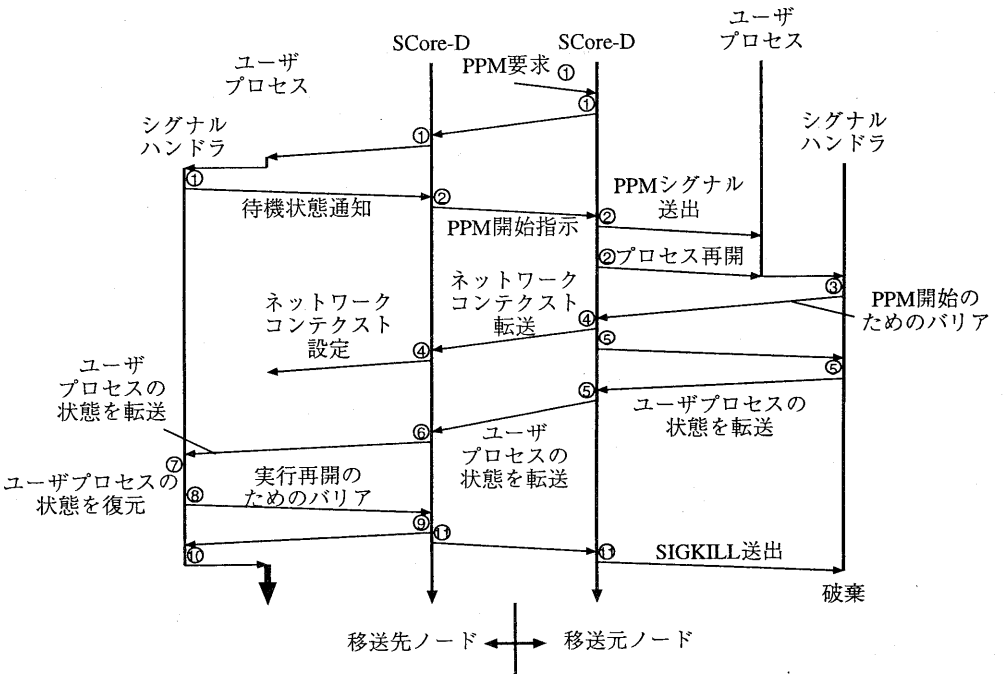

図 2: PPM の手順

PCC-II は、8 ポートのスイッチを 4 個ずつ 2 段に組み合わせ 16 ノードを接続したスイッチボックス (SB, 図 3) を単位とし、これを 8 個接続した形態となっている (図 4)。ノード番号が $16n \sim 16n+15$ (n は $0 \leq n \leq 8$ である整数) であるノードは、同じ SB に接続されている。

SB 内のリンクは、それぞれ 160 MByte/sec のバンド幅を持っている。スイッチはすべてフルクロスバスイッチであり、複数の通信で同じスイッチを使用していても、リンクが重複しない限り、転送性能は低下しない。SB 内の通信では、なるべくリンクが重複しないように経路が選択される。

8 個の SB は、それぞれ他のすべての SB と 2 本のリンクで接続されており、このリンクは、それぞれ 160 MByte/sec のバンド幅を有している。ある SB から別の SB への通信は、宛先の SB が同じなら、同じ 2 本のリンクのどちらかが使われる。同時に行われる通信が 2 以下であれば、リンクが重複しないように経路が選択される。

6.1 バッファサイズの影響

この評価では、ノード番号 0 から始まる N 個のノードが、移送元となる。移送元では 128 MByte の

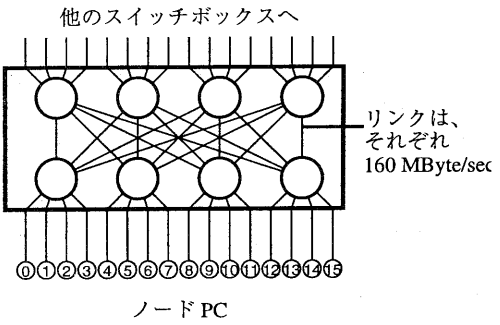

図 3: Myrinet のスイッチボックス内のリンク

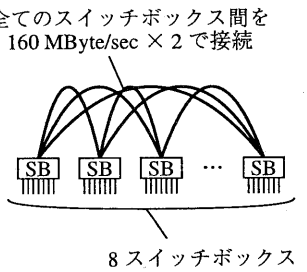

図 4: PCC-II のネットワークの概要

メモリ領域を確保し、その全体にわたってデータを書き込む。書き込みの終了を同期させてから、書き込んだデータを、移送元の各ノード n から、ノード番号 $64+n$ のノードに、あるバッファサイズ (M とする) ごとに分割して送信する。すなわち、すべてのデータ転送は、異なる SB に対して行われる。N と M を変化させることにより、転送するノード数とバッファサイズが転送性能にどう影響するかを測定し、評価した。この測定結果を図 5に示す。

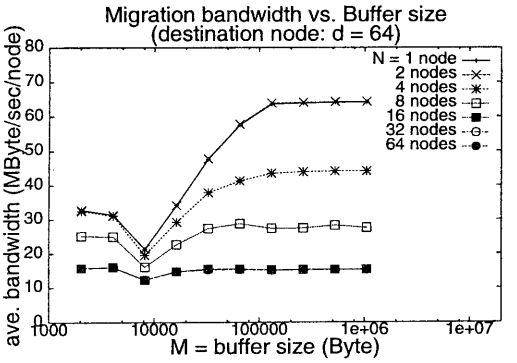

図 5: バッファサイズによる転送性能の変化

図 5から、$M = 8$ KByte 付近で転送性能が一旦遅くなっていることが分かる。これは、バッファサイズが 8 KByte 以上のとき、MPICH-PM の通信方式がゼロコピー通信に切り替わるためと考えられる [9]。

ゼロコピー通信では、転送するメモリ領域をピンダウンするが、一度ピンダウンしたメモリ領域は、解放せずにとっておき、再び同じメモリ領域を転送しようとしたときに、再度ピンダウンする手間を省く技法が用いられている (ピンダウン・キャッシュ)[9]。ゼロコピー通信の性能はキャッシュのヒット率に依存するが、本実験では同じメモリ領域は二度と転送されないため、ピンダウン・キャッシュの恩恵は受けられない。別の実験から、送信側のピンダウン・キャッシュのヒット率が 0% で、受信側のヒット率が 100% のとき、$M = 1$ MByte 付近での転送性能は、およそ 79 MByte/sec であることが分かっている [9]。本実験では、受信側でも、同じメモリには二度と読み込まないために、キャッシュヒット率が 0% なので、ノード数が 1 のときの転送性能は、最大でも、約 64 MByte/sec となっている (図 5)。

図 5では、$N = 1$ と $N = 2$ では、ほぼ同じ性能であるが、N が 4, 8 と増えるにしたがって性能が低下し、16 以上では、約 16 MByte/sec/node でほぼ頭打ちとなる。

この評価では、移送先ノードは異なる SB に接続されているので、転送には必ず SB 間リンクが使われる。$N = 1$ と $N = 2$ とで性能に差がないのは、$N = 1$ のときは、2 本ある SB 間リンクのうちの 1 本しか使われないものが、$N = 2$ では 2 本のリンクが使用されるためと考えられる。ひとつの SB には 16 ノードが接続されているので、$2 < N \leq 16$ では、ノードあたり $160 \times 2/N$ MByte/sec が理論上の上限となり、N が増えるにしたがって上限は低くなる。すなわち、SB 間リンクがボトルネックとなって性能が低下するものと考えられる。$N = 16$ での理論上の上限は 20 MByte/sec であるが、図 5では、このときの転送性能は約 16 MByte/sec となっている。

$N > 16$ では、移送元の SB が複数になる。この場合、これらは、それぞれ異なる SB 間リンクを経由し、異なる SB へとデータを転送する。これは、16 ノードの転送をそれぞれ独立に実行しているのと同じことである。図 6は 64 ノードの場合を図示したものであるが、16 ノードの転送が 4 つ独立に実行されているのに等しい。図 5において、$N \geq 16$ のとき、転

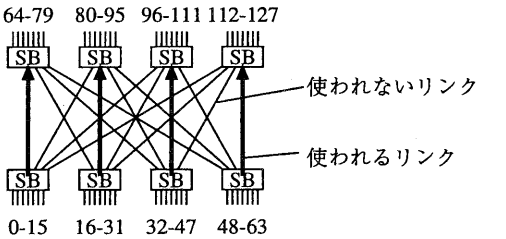

図 6: 64 ノードの転送

送性能が約 16 MByte/sec で変化しないのは、このためと考えられる。

以上から、PCC-II 上の PPM では、64 ノードの並列プロセスの移送でも、ノード間で 16 MByte/sec/node の転送性能が得られることが分かった。これは、PCC-II 上で、プロセスをローカルディスクにチェックポイントする場合 [12] の、4 – 10 倍の性能である。

6.2 移送先の位置による影響

この評価では、ノード番号 0 から始まる N 個のノードで、移送元のプロセスを実行する。これらは、128 MByte のメモリ領域を確保し、その全体にわたって、あるデータを書き込む。書き込みの終了を同期させてから、書き込んだデータを、移送先のノードに 1 MByte ごとに分割して送信する。移送先の位置

を表すパラメータとして d を導入し、移送元のノード n のデータを、ノード $d+n$ に転送する。d が大きくなるほど、番号的に離れたノードへ移送することとなる。これは、必ずしもネットワーク上の距離が遠くなることを意味しないが、少なくとも近くはならないので、近似的に移送距離を表していると考える。N と d を変化させることにより、転送するノード数と移送距離が、転送性能にどう影響するかを評価した。ここで、d は $d = 2^0, 2^1, 2^2, \ldots$ のように変化させ、移送先と移送元の範囲が重なるような場合は省いた (すなわち、$d \geq N$)。この測定結果を図7に示す。

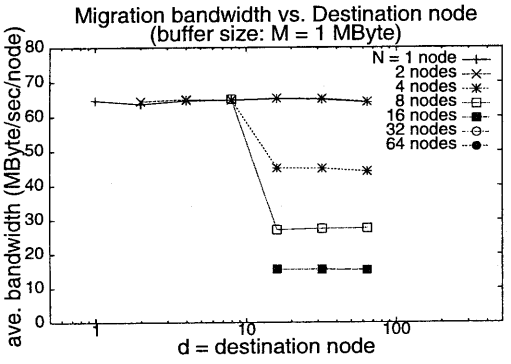

図 7: 移送先のノードによる転送性能の変化

図7のグラフは、$d < 16$ のときは、N に関わらず同じ転送性能となっているが、$d \geq 16$ のときは、$2 \leq N \leq 16$ では、N が増えるにしたがって転送性能が下がっている。また、$N = 1$ および $N = 2$ のときは、d の値による転送性能の変化はない。

移送元のノード番号は $0 \sim N-1$ であり、ノード $0 \sim 15$ は同じSBに接続されているので、$d < 16$ のときは、同じSB内での転送である。$d < 16$ かつ $1 \leq N \leq 8$ のときに転送性能が変わらないのは、同一SBでの、経路の重複しない転送だからと考えられる。

$d \geq 16$ のときは、異なるSBへの転送である。このときの転送性能は、図5に示された転送性能のグラフで、バッファサイズが1 MByteのときの値に一致している。$N = 1$ および $N = 2$ の場合に、同じSB内での転送と、異なるSBへの転送とで、性能に差がない。Myrinetのスイッチのレイテンシは1 μsec未満であり、多くのスイッチを経由する場合でも転送性能への影響は無視できることが判明した。

7 考察と今後の課題

7.1 提案方式における制約

チェックポイント機構がユーザレベルで実現されているため、プロセスの状態のうち、パイプ、ソケット、プロセス間共有メモリ、ビンダウンされたメモリの物理アドレスなど、カーネル内部に状態を持つものは移送できない。

また、アプリケーションは静的にリンクされている必要がある。これは、そうしない場合、共有ライブラリのテキスト領域もプロセスのメモリ内容として移送されるので、プロセスの再開時に使用するいくつかの関数 (例えば `mmap()`) のコード部分が、その実行中に上書きされてしまうという問題が生じるためである。

7.2 ネットワークコンテクストの移送

PCC-IIでは、ノード間の通信にはPM[8, 9]が用いられる。ネットワークコンテクストの実体は、PMの状態である。したがって、プロセスの移送のためには、PMの状態が移送可能であることが必要であるが、現在これはまだ実現されておらず、今後の課題である。PMの状態を移送可能なものにすることによって、プロセスの通常の実行にオーバヘッドが生じる可能性もあり、その実証的な評価も必要である。

7.3 SCore-D を介した転送性能

本論文の評価により、PCC-II上の並列プロセスをネットワーク上で移送する転送性能の下限は16 MByte/sec/nodeであると分かった。

実際のPPMでは、データの転送は、仮想ネットワーク間の保護を保つために、ユーザプロセスではなくSCore-Dが行う。このため、SCore-Dとユーザプロセスとの間の転送オーバヘッドが生じる。

別の実験から、PCC-IIの二つのノード上のユーザプロセスがSCore-Dを介してデータを転送するときの転送性能は、現在のSCore-Dの実装では、およそ10 MByte/secであることが分かっている。本論文の評価の下限値はこれを上回っているので、より高速化するためには、まずSCore-Dを介した転送性能を向上させる必要がある。これは今後の課題である。

8 まとめ

本論文では、クラスタにおける並列プロセスマイグレーションの実現方式を提案した。並列プロセスマイグレーションとは、並列プロセスの終了によって崩れたノード間の負荷平衡を取り戻すために、ある並列プ

ロセス全体を、負荷の高いノードから負荷の低いノードに移送することである。

提案方式は、無矛盾な大域的状態を確保する手段として、既に実現されているチェックポイント機構を利用する。このチェックポイント機構の性質から、アプリケーションの通信方式に依存せず、実行時オーバヘッドのない並列プロセスマイグレーションが実現できると期待される。

並列プロセスマイグレーションのように、多くのノードが一斉に大量のデータを転送する通信パターンでの転送性能の評価はこれまでなされていなかったので、提案方式の性能を見積もるために、128 台構成のクラスタ (RWC PC Cluster II) 上で、そのパターンでの転送性能を評価した。その結果、16 以上のノードで実行されている並列プロセスを移送する場合は、ネットワークがボトルネックとなり、ノードあたり 16 MByte/sec 程度の性能であることが分かった。これは、チェックポイント機構がローカルディスクにプロセスをチェックポイントする場合の 4 – 10 倍の性能であることが判明した。

参考文献

[1] A. Barak, O. La'adan, and A. Shiloh. Scalable Cluster Computing with MOSIX for LINUX. In *Linux Expo '99*, pp. 95–100, Raleigh, N.C., May 1999.

[2] A. Hori, H. Tezuka, and Y. Ishikawa. Global State Detection using Network Preemption. In D. G. Feitelson and L. Rudolph eds., *IPPS '97 Workshop on Job Scheduling Strategies for Parallel Processing*, Vol. 949 of *Lecture Notes in Computer Science*, pp. 262–276, Apr. 1997.

[3] A. Hori, H. Tezuka, and Y. Ishikawa. Highly Efficient Gang Scheduling Implementation. In *Supercomputing '98*, Nov. 1998.

[4] F. B. O'Carroll, H. Tezuka , A. Hori , and Y. Ishikawa. MPICH-PM: Design and Implementation of Zero Copy MPI for PM. Technical Report TR–97011, RWCP, Mar. 1998.

[5] F. B. O'Carroll, H. Tezuka , A. Hori , and Y. Ishikawa. The Design and Implementation of Zero Copy MPI Using Commodity Hardware with a High Performance Network. In *International Conference on Supercomputing '98*, July 1998.

[6] J. S. Plank. *Efficient Checkpointing on MIMD Architectures.* PhD thesis, Princeton University, 1993.

[7] G. Stellner. CoCheck: Checkpointing and Process Migration for MPI. In *Proceedings of the 10th International Parallel Processing Symposium (IPPS '96)*, pp. 526–531, Honolulu, Hawaii, Apr. 1996.

[8] H. Tezuka, A. Hori, Y. Ishikawa, and M. Sato. PM: An Operating System Coordinated High Performance Communication Library. In *High-Performance Computing and Networking (HPCN '97)*, Vol. 1225, pp. 708–717, Vienna, Austria, Apr. 1997. Springer-Verlag.

[9] H. Tezuka, F. B. O'Carroll , A. Hori , and Y. Ishikawa. Pin-down Cache: A Virtual Memory Management Technique for Zero-copy Communication. In *IPPS/SPDP '98*, pp. 308–314, Apr. 1998.

[10] 手塚, 堀, F. B. O'Carroll, 石川. RWC PC Cluster II の構築と性能評価. In *HOKKE '98*, Mar. 1998.

[11] 堀, 手塚, 石川. ギャングスケジューリングの高速化技法の提案. 並列処理シンポジウム JSPP '98, pp. 207–214, June 1998.

[12] 西岡, 堀, 手塚, 石川. クラスタにおけるコンシステントチェックポイントの実現. 並列処理シンポジウム JSPP '99, pp. 299–236, June 1999.