計算機アーキテクチャ 139－13
（２０００．８．４）

異なるプラットフォームにおける受信メッセージ予測法の性能評価

足立涼子† 岩本善行‡ 大津金光†
吉永努† 馬場敬信†

†宇都宮大学 工学部 情報工学科

‡宇都宮大学 サテライト・ベンチャー・ビジネス・ラボラトリ

本稿では、WSクラスタとSun Enterprise 3500の２つの異なるプラットフォームに対し、メッセージ通信を行うために利用されているMessage Passing Interface(MPI)に受信メッセージ予測法を実装し、その評価結果を比較した。受信メッセージ予測法とは、これから受信するメッセージを予測し、その結果に基づいてメッセージ受信処理を先行実行することでメッセージ受信の処理を高速化する手法である。
受信するメッセージの予測方式として、前回のメッセージを残しておき、それを今回受信するメッセージとしてその後の処理を行う直前予測方式と、マルコフモデルを使った方式を実装した。NAS Parallel Benchmark Programによる評価では、総実行時間において最大約6.5%の速度向上を達成した。

Perfomance Evaluation of the Receiving Message Prediction Method on Different Platforms

RYOKO ADACHI†, YOSHIYUKI IWAMOTO‡,
KANEMITSU OOTSU†, TSUTOMU YOSHINAGA†
and TAKANOBU BABA†

†Department of Information Science, Faculty of Engineering,
Utsunomiya University

‡Satellite Venture Business Labolatory, Utsunomiya University

We have implemented receiving message prediction method in the Message Passing Interface(MPI) library on the WS cluster and Sun Enterprise 3500 and evaluated the perfomance.
Receiving message prediction pre-executes the part of the message reception code using the predeicted valus. The first method is to predict the next message as the last received message and the second method is to predict the next message by using the message reception history.
The evalution using the NAS parallel benchmark programs, shows taht the maximum 6.5 % speed up can be achieved.

1. はじめに

現在、並列分散計算機が広く使用されているなか、並列計算機の高速化について、RISCアーキテクチャや命令のプリフェッチ[1]などをキーワードとしてさまざま研究がされており、その有効性が確認されている。

我々は、並列計算機のメッセージ受信処理の高速化手法として、**受信メッセージ予測法**を提案している。これは、受信処理におけるアイドル時間を利用して到着するメッセージを予測し、それに基づいて受信後の処理を投機的に実行するものである[2]。

並列分散計算機上でメッセージ通信を行うためにMessage Passing Interface(MPI)[3]が広く使用されている。このMPIは実用的、効率的、かつ適応性のあるメッセージ通信を目的としたもので世界的に広く使用されている。このMPIを使用したメッセージ通信におけるブロッキング受信命令では、受信メッセージが到着するまで次の処理へ進むことができない。このため受信メッセージ待ちでアイドル時間が生じ、プロセッサ資源を無駄にすることがある。

本研究では、受信待ちのアイドル時間を利用し、受信メッセージ予測法をMPI_Recv関数に実装した。MPIで同期的にメッセージを受信する関数の多くは内部でMPI_Recv関数を呼び出し実行している。このことにより、MPI_Wait、MPI_Alltoall、MPI_Allgather、MPI_Allreduce、MPI_Gather、MPI_Reduce関数等でも受信メッセージ予測が可能となった。

今回はワークステーション(WS)クラスタとSun Enterprise 3500[4]の２つの異なるプラットフォーム

link	次の受信メッセージへのポインタ
orig_len	受信データのサイズ
type	受信データの型
to	送信先のランク
from	送信元のランク
datatype	送信データの型
len	送信データのサイズ
ack_req	受信確認の要求
msg	受信バッファの先頭アドレス

図1　構造体の内容

上で受信メッセージ予測法を実装、評価を行い性能を比較している。

本稿では、2章で受信メッセージ予測法の実装法を、3章では実装後の性能評価を、4章で考察を、最後に5章で本稿のまとめと今後の課題について述べる。

2. メッセージ受信処理の高速化

2.1 MPIの概要

MPIとは世界中で広く使用されている並列分散計算機用のメッセージ送受信機構のことである。MPIの各命令はFortran言語、あるいはC言語で記述されたプログラムから、サブルーチンまたは関数として呼び出すことで利用する。

2.2 MPI_Recvの流れ

メッセージ受信処理を行なう時、先に述べたように、メッセージ検索でアイドル時間が生じる。そのアイドル時間を利用して受信メッセージ予測を行なうため、MPI_Recv命令におけるアイドル時間がどこで作られるか説明する。

MPI_Recv命令の流れは、大きく分けると以下の3つに分けられる。

(1)　引数チェック
(2)　メッセージ到着の確認
(3)　後処理

(1)MPI_Recv命令では、まず、メッセージのサイズ、タグ、送信元のチェックを行う。このとき、メッセージサイズが負の場合や、タグが指定されたものと違っている場合、送信元が全ノード数以上の場合など引数に間違った値が設定されていたらエラーを返す。

次に、受信バッファの先頭アドレス、受信データのサイズ、受信データの型などを1つの構造体(図1)にまとめる。

(2)メッセージ到着の確認は、ユーザが要求したメッセージの情報(要素数、要素の型、送信元等)と一致するメッセージを既に到着しているメッセージの中から探し出す処理によって行われる。その流れは、次の通りである。

1. すでに到着しているメッセージを先頭から順に調べる。
2. 一致するメッセージが見つかったらそのメッセージを返す。
3. 最後までメッセージを調べても要求と一致するメッセージがなかった場合、要求したメッセージが到着するまで待つ。

この3.の状態の時にアイドル時間が生じる。(3)の後処理では、到着したメッセージに対して

1.　バイトオーダの変換
2.　メッセージ種別の認識
3.　メッセージ到着
4.　待ちキュー内の検索
5.　メッセージサイズの認識
6.　受信バッファからユーザバッファへのコピー
7.　受信ステータスの設定

などを行っている。受信メッセージを予測した場合は、これらの処理を先行実行することが可能となり、受信処理の高速化が期待できる。

2.3 受信メッセージ予測法

2.3.1 受信メッセージの規則性

予測法について検討するため、NAS Parallel Benchmark Program[4] のSP(非優位対角なスカラ5重対方程式を解くプログラム)、CG(大規模で正値対称な疎行列の最小固有値を求めるプログラム)などの通信パターンやその内容を詳細に分析した。

その結果、ノード数8、CLASS=Aで実行したSPにおいて、出現するMPI関数を調べたところ、MPI_Isend、MPI_Irecv、MPI_Waitall、の3種類の命令が全体の99.6%を占めた。メッセージの内容は、200台、0～6、10^5台の3種類の実数となっていた。命令の発生順序を調べると、同じ順序で繰り返される命令の集合が大量にあった。

また、ノード数8、CLASS=Aで実行したCGでは、メッセージの内容と要素数を比べた所、共通性を持つ5つのグループに分けることができた。5つグループにA～Eの記号をつけてメッセージの内容と出現間隔を分析し、以下の結果を得た。

グループAは2秒間隔で出現し、内容は不規則に変化している。グループBは0.02～0.05秒間隔で出現し、内容は規則に変化している。グループCは0.02～0.05秒間隔で出現し、内容は1/16 ～1/35の変化で減少し、25回目に新しい値に変わる。グループDは2秒間隔で出現し、内容は小数点第3位以降が変化している。さらに、一回ごとに正負が反転している。また、各グループの順序関係は

ＡＢＣＢＣＢＣ …… ＢＣＤＥ

のようにＢＣは２５回繰り返し、さらにA～Eを１６回繰り返した。

以上のような分析結果より、メッセージの発生系列には規則性があることが分かる。よって、このように、何らかの規則性を持ってメッセージが発生するとすれば、そのメッセージを予測することが可能であると考えられる。

2.3.2 予測方式

受信メッセージを予測する時、アイドル時間の長さなどによって、予測アルゴリズムの複雑さが決定される。予測アルゴリズムとしては

(1) 直前に到着したメッセージのみを参照する
(2) これまでに到着したメッセージの平均値
(3) 最も回数が多かったもの
(4) メッセージ発生系列のマルコフ連鎖

などが考えられる。この中で、最も容易に実装できるのが(1)の方法である。この方法は、前回到着したメッセージを保存しておき、そのメッセージを用いて受信後の処理を先行実行しようというものである。この方式は、同じメッセージが続けて送られるようなプログラムに最適であると考えられる。

(2)は過去に到着したメッセージを用いて平均値を取り、それを次に到着するメッセージであると予測する方法である。この方式は、メッセージサイズの予測には適しているが、文字列などには使用できない。

(3)は過去に到着したメッセージの中で最も到着した回数が多いものを次に到着するメッセージであると仮定する方法である。この方法は、ある1つのメッセージが頻繁に出てくるようなプログラムで有効である。

(4)はメッセージの発生系列をマルコフ連鎖により予測する方法である。これは、メッセージの到着順序に強い規則性のある場合や、関数やメソッドの起動順序に制限があるような場合に有効である。しかし、予測に時間がかかり、また、連鎖の記録に多くのメモリ領域を必要とする。

このように、予測アルゴリズムが複雑になるほど、予測が的中する確率が上がると思われるが、その一方、予測に必要となる時間が増大し、アイドル時間が多く必要となる。そのため、メッセージ受信を高速化するには、予測アルゴリズムの複雑さに対応したアイドル時間が必要となる。

2.3.3 アイドル時間と高速化

受信メッセージ予測を行なうことによりアイドル時間が十分に長い場合は、図2に示す通り高速化が期待できる。この場合は、アイドル時間内に、その後の処理を先行実行し終えることが出来る。受信メッセージ予測に失敗した場合は、受信メッセージ予測の成功の確認作業によるオーバーヘッドが存在するが、そのコストは極めて小さい。

一方、受信メッセージ予測を行なうときに、アイドル時間が十分にない場合を、図3に示す。この場合は、通常実行時でのメッセージ待ちのアイドル時間内に、その後の処理を先行実行し終えることが出来ない。受信メッセージ予測に成功した場合でも、高速化した時間が小さいことを意味し、さらに、受信メッセージ予測が失敗した場合は、受信メッセージ予測の成功の確認作業にかかる時間に加え、メッセージ到着から確認作業に入るまで

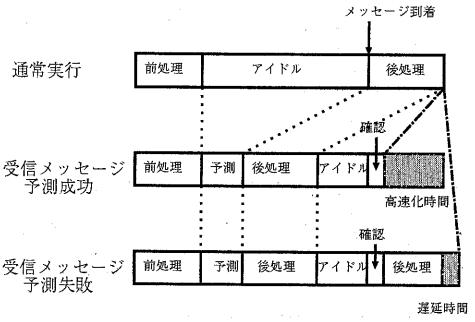

図2 アイドル時間が長い場合の高速化

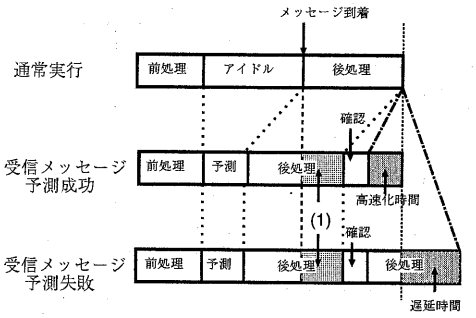

図3 アイドル時間が短い場合の高速化

の時間(図3(1))がオーバーヘッドとなる。

このことより、受信メッセージ予測を行なって高速化するためには、十分なアイドル時間が必要であることがわかる。

2.3.4 実　　装

受信メッセージ予測を行なうために、今回、2.3.2節に説明した(1)直前に到着したメッセージのみを参照する方法と、(4)マルコフモデルを用いた実装法を用いた。(1)の予測法は予測のコストが小さいため、高速化の時間が大きい可能性が大きいという理由から、(4)の予測法は予測のコストは大きいが、予測の成功率が高いので結果として高速化する可能性が大きいという理由から、この2つの実装法を選択した。受信メッセージ予測法導入時のMPI_Recvの処理の流れを図4に示す。

はじめに、直前受信メッセージ予測法導入時のMPI_Recvの流れとして、引数のチェックなどの前処理を行なう〈1〉。その後、通常どおりメッセージが到着しているかをチェックする〈2〉。その時、既にユーザが要求したメッセージが到着している場合〈YES〉は、そのメッセージをそのまま残しておく〈3〉。その後はそのまま通常の受信処理を行なう。

メッセージが到着しているかをチェックした時〈2〉に、まだユーザの要求するメッセージが到着していなかった場合は受信メッセージ予測を行う〈4〉。予測メッセージは、前回受信したメッセージ

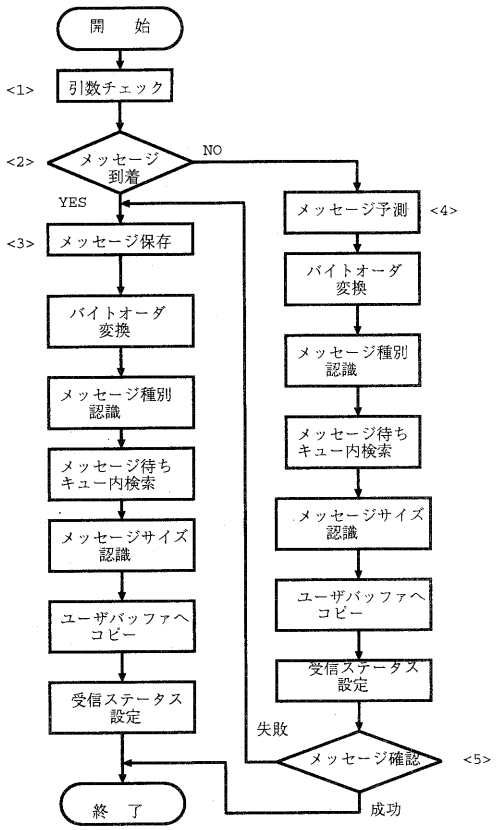

図4　受信メッセージ予測時の MPI_Recv の流れ

を、これから受信するメッセージとしてその後の処理を続ける。受信処理が終った後は、予測したメッセージが、ユーザが要求したメッセージと一致しているか(予測が成功したか失敗したか)の確認を行なう〈5〉。予測が成功していた場合は、そのまま受信処理を終了することが可能である。予測が失敗していた場合は、メッセージ受信処理をもう一度実行し直す。

マルコフ受信メッセージ予測の場合は、メッセージ保存〈3〉で直前予測と異なる処理を行う。具体的には、到着しているメッセージが以前使用されたメッセージの場合、マルコフテーブルに出現回数を加算し、その後、次に到着する確率が最も高いメッセージを準備する。到着しているメッセージがはじめて出現したメッセージの場合、マルコフテーブルにメッセージを記録する。

今回実装および実験を行ったWSクラスタの仕様を表1に示す。WSクラスタの各ワークステーションは、100Base-T Ethenet で接続されている。また、Enterprise 3500の仕様を表2に示す。

3. 評　　価

表1 WSクラスタの構成

CPU	Memory	OS
UltraSPARC-II(300MHz)×2	512MB	Solaris2.5.1
UltraSPARC(200MHz)×2	320MB	Solaris2.5.1
UltraSPARC-IIi(270MHz)	128MB	Solaris2.5.1
UltraSPARC(167MHz)	128MB	Solaris2.5.

表2 Enterprise 3500の構成

CPU	Memory	OS
UltraSPARC-II(400MHz)×8	3GB	日本語Solaris7

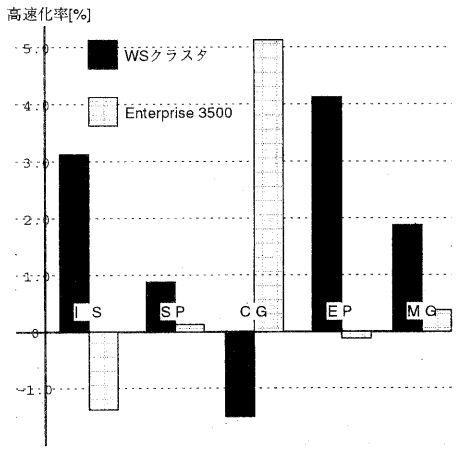

図5　直前受信メッセージ予測による高速化率

3.1　評価プログラム

受信メッセージ予測法の効果を確認するため、ここでは、NAS Parallel Benchmark Program を用いて評価した。本稿ではその中で特徴的なIS、SP、CG、EP、MGの5つの結果を示す。CLASSはW(orkstation) を使用した。

3.2　プラットフォームの違いによる比較

3.2.1　高速化率の比較

WSクラスタ上で受信メッセージ予測法を実装した場合と、Enterprise 3500上で受信メッセージ予測法を実装した場合のメッセージ受信処理の高速化率を図5に示す。

WSクラスタでは5つのプログラム中、CGを除く4つのプログラムで高速化に成功した。同様に、Enterprise 3500では、5つのプログラム中SP、CG、MGの3つのプログラムで高速化に成功し、両プラットフォームの中で、最も高速化率の良かったものは、Enterprise 3500でのCGという結果になった。

プラットフォーム別に見ると、WSクラスタにおいて、最も高速化率が良いプログラムはEP、最も高速化率が悪いプログラムはCGとなり、Enterprise 3500において最も高速化率が良いプログラムはCG、最も高速化率が悪いプログラムはISとなった。

表3 WSクラスタ(直前)

プログラム	予測率(%)	ヘッダのみ成功率(%)
IS	3.55	1.38
SP	0.05	6.67
CG	0.008	0.0
EP	30.13	8.51
MG	3.19	2.34

表4 Enterprise 3500(直前)

プログラム	予測率(%)	ヘッダのみ成功率(%)
IS	4.46	1.46
SP	0.08	8.00
CG	0.004	25.00
EP	40.37	10.33
MG	3.48	1.43

表5 WSクラスタ(マルコフ)

プログラム	予測率(%)	ヘッダのみ成功率(%)
IS	3.87	6.64
SP	0.04	0.0
EP	28.95	1.33

3.2.2 予測率と予測成功率の比較

プログラム実行時に発行された受信命令中、どれだけの受信命令に対して受信メッセージ予測を行ったかという予測率、受信メッセージ予測を行ったうち、どれだけ予測に成功したかという予測成功率を調べた。

WSクラスタ上で受信メッセージ予測を行った結果を表3に、Enterprise 3500で受信メッセージ予測を行った結果を表4に示す。

その結果、WSクラスタとEnterprise 3500において、予測率に大きな違いは表れなかった。ヘッダのみの成功率では、プログラムCGにおいて、WSクラスタでの成功率0.0%に対して、Enterprise 3500では25.0%となり違いが見られたが、その他のプログラムでは、大きな違いは見られなかった。

3.3 受信メッセージ予測法の違いによる比較

3.3.1 高速化率の比較

WSクラスタ上に、直前予測による受信メッセージ予測法(以下、直前受信メッセージ予測法)とマルコフモデルを使用した受信メッセージ予測法(以下、マルコフ受信メッセージ予測法)、2つの異なる予測法を実装し評価を行った。その結果を図6に示す。

プログラムISとEPでは直前予測法、マルコフ予測法ともに速度が向上した。プログラムISではマルコフ予測法の方が高速化率が高くなり、プログラムEPでは高直前予測法の方が高速化率が高くなっている。プログラムSPでは直前予測法では速度が向上したが、マルコフ予測法では速度が低下した。

3.3.2 予測率と予測成功率の比較

プログラム実行時に出された受信命令中、どれだけの受信命令に対して受信メッセージ予測を行ったかという予測率、受信メッセージ予測を行ったうち、どれだけ予測に成功したかという予測成功率を調べた。

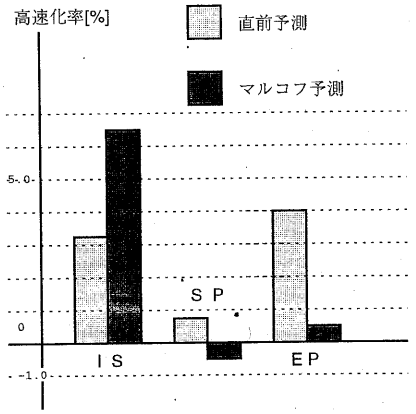

図6 2つの受信メッセージ予測による高速化率

マルコフ受信メッセージ予測法を実装した結果を表5に示す。

その結果、直前受信メッセージ予測法(表3)とマルコフ受信メッセージ予測法において、予測率では大きな違いは見られなかった。

ヘッダの成功率を比較すると、プログラムISではマルコフ受信メッセージ予測法の方がヘッダの成功率が高く、プログラムSP、EPでは直前受信メッセージ予測法がヘッダの成功率が高くなっている。

4. 考　　察

受信メッセージ予測を行いNAS Parallel Benchmark Programを実行したところ、通常実行と比べて、受信メッセージ予測法を実装し実行した方が速度が向上するものと、速度が低下するものに別れた。以下にこれらの評価結果に対する考察を示す。

4.1 プラットフォームの違いによる性能評価

まず初めに、異なるプラットフォーム上での高速化率を比較した結果について考察する。

WSクラスタ、Enterprise 3500で速度向上が見られたものは、メッセージのヘッダ部分が予測に成功していた。このことから、メッセージ本体ではなく、ヘッダ部分の予測に成功するだけでも受信処理を高速化することが可能であると言える。

また、WSクラスタ上で速度が低下したCGでは一度も予測に成功していないため、受信メッセージ予測の成功確認の時間がオーバーヘッドとなり、プログラムの実行時間が遅くなったと考えられる。

また、Enterprise 3500では、2つの矛盾する結果が出た。1つは、プログラムIS、EPにおいて、ISでは1.46%、EPでは10.33%メッセージのヘッダ部分の予測に成功しているにも関わらず高速化率がISで−1.38%、EPで−0.39%と、プログラムの実

行時間が遅くなったという結果である。もう1つは、プログラムMG、IS、EPにおいて、WSクラスタとEnterprise 3500で予測率、成功率ともに大きな違いが認められないが、WSクラスタの方が高速化率が良いという結果である。

このように、一見矛盾しているような結果が出た理由として、アイドル時間の長さの違いなどが挙げられる。

Gigaplaneを使用したEnterprise 3500はEthenetを使用したWSクラスタに比べてメッセージ通信が高速である。そのため、1度目にメッセージの確認を行った時にまだメッセージが到着しておらずに受信メッセージ予測を行い、その後の処理を先行実行したが、受信後の処理の先行実行が終わる前に、要求したメッセージが到着していたため、メッセージ予測が成功していても、短縮時間が短くなるためである。このことから、受信処理を高速化するためには、予測成功率も高くしなければならない。また、受信メッセージ予測が失敗した場合のオーバーヘッドも大きくなる(図3)。

一方WSクラスタは、先に述べたように、Enterprise 3500と比べるとメッセージ通信に時間がかかる。このため、1度目の確認で要求したメッセージが到着しておらず、受信メッセージ予測を行い、その後の処理を先行実行し終わっても、まだ要求したメッセージが到着していない。このため、予測に失敗した場合でも、オーバーヘッドが小さくなり、予測に成功した場合は、短縮時間も長くなる(図2)。

このようなことから、同じような予測率、成功率にもかかわらず、WSクラスタの方が高速化率が良くなると考えられる。

4.2 予測法の違いによる性能評価

次に、実装アルゴリズムの違いによる高速化率を比較した結果について考察する。

直前受信メッセージ予測法とマルコフ受信メッセージ予測法の2つの予測アルゴリズムを比べると、最も高速化された予測法はプログラムISにおける高速化率6.84%のマルコフ受信メッセージ予測法であった。しかし、プログラムSPにおける高速化率が−0.6%となっており、最も高速化されなかった予測法もマルコフ受信メッセージ予測法であった。

マルコフ受信メッセージ予測法が直前受信メッセージ予測法と比べてより高速化された理由として、直前受信メッセージ予測法でのヘッダの成功率1.36%に対し、マルコフ受信メッセージ予測法でのヘッダの成功率は6.64%となっており、より予測成功率が高いことが挙げられる。

また、マルコフ受信メッセージ予測法が直前受信メッセージ予測法と比べ、遅くなった理由として、次の理由が考えられる。マルコフ受信メッセージ予測法は、受信メッセージをマルコフテーブルに格納し、メッセージが受信されるごとに出現回数をカウントしている。このように、直前受信メッセージ予測法に比べて、メッセージの予測に時間がかかるため、後処理の先行実行中にメッセージが到着し、オーバーヘッドとなる可能性が高い(図3)。

予測が成功した場合は通常実行時より時間が短縮されるが、予測が失敗した場合は、直前受信メッセージ予測法で予測が失敗した時よりオーバーヘッドが大きくなってしまう。

以上の結果より、マルコフ受信メッセージ予測法を実装する場合は、十分なアイドル時間が必要であると結論づけられる。

5. おわりに

本稿ではWSクラスタ、Sun Enterprise 3500上のMPIをとりあげ、メッセージ受信処理の高速化を「受信メッセージ予測法」にて行ない、NAS Parallel Benchmark Programの実行時間を用いて評価を行った。今回は、前回のメッセージを保存しておき、それを今回受信したメッセージとして受信後の処理を進める「直前受信メッセージ予測法」と、マルコフを用いて次に受信するメッセージを予測する「マルコフ受信メッセージ予測法」を使用した。

直前予測方式では、受信メッセージを予測するとき、前回のメッセージをそのまま受信メッセージとして受け渡すだけで済むため、予測処理に時間がかからず、アイドル時間が短い場合でも十分に実装可能である。その結果WSクラスタでは最大4.13%、Enterprise 3500では最大5.01%の高速化に成功した。

同様に、マルコフ予測方式では、予測法が複雑になるため予測成功率が高くなる。その結果、直前予測方式に比べ最大6.84%の高速化に成功した。

今後の課題として、アイドル時間に合った予測法を実装していく必要がある。

謝辞

本研究は、一部文部省科学研究費(基盤(B)課題番号10558039, 基盤(C)課題番号12680328, 奨励(A)課題番号11780190),並列・分散処理研究推進機構の援助による。

参考文献

1) Vasta Santhanam,Edward H.Gornish,Wei-Chung Hsu:Data Prefetching on HP PA 8000,ISCA'97,June(1997)
2) Y.Iwamoto, K.Ootsu, T.Yoshinaga, and T.Baba: Message Prediction and Speculative Execution of the Reception Process, Proceedings of the Eleventh IASTED International Conference, Parallel and Distributed Computing and Systems(PDCS '99), pp.329-334(1999).
3) MPI日本語訳プロジェクト,"MPI:メッセージ通信インターフェース標準(日本語ドラフト)",(1996)
4) Sun Microsystems:Inc.901 SAN ANTONIC ROAD,PALO ALTO CA 94303-4900USA,(1999)
5) "http://www.nas.nasa.gov/NAS/NPB"