受信メッセージ予測によるユーザプログラムの実行性能

1N－4

岩本 善行　澤田 康雄　大津 金光　吉永 努　馬場 敬信

宇都宮大学工学部

1 はじめに

本研究では、並列計算機アーキテクチャとその上に実装されている独自の並列化ライブラリを分析し、特定の通信アーキテクチャに依存せずに高速なメッセージ転送を実現する方法を提案することを目的としている。このための第一段階として、これまでに商用並列計算機富士通AP1000およびNEC Cenju-3を取り上げ、メッセージピンポン処理やリダクション処理などのベンチマークプログラムを作成し、そのメッセージ送受信性能を評価した[1]。同時に、各マシンの並列ライブラリ、OS、カーネルのソースプログラムの提供を受け、メッセージ送受信処理部分の分析を進めてきた。この過程で、メッセージ転送処理においては受信処理に時間が多くかかることや、アイドル時間の有効利用が必要であることが明らかになってきた。

このことから、並列計算機におけるメッセージ処理を高速化するために、これから到着するメッセージをアイドル時間を利用してあらかじめ予測し、その結果に基づいて投機的な実行を行う受信メッセージ予測を提案する。

本稿では、受信メッセージ予測の具体的な方法と、A-NETマルチコンピュータおよび富士通AP1000上に実装して得た評価について述べる。

2 受信メッセージ予測

2.1 基本方針

並列計算機上で応用プログラムを作成する場合、多数のノードを使ったプログラム中では多くのノードで似たような処理を行うことが多いと考えられる。このため、各ノード間でやり取りされるメッセージは、到着するメッセージのサイズやそれによって発生する受信処理に強い規則性を見い出すことが可能である。

このことから、ノードがアイドル状態時に、これから到着するメッセージをあらかじめ予測し、その受信処理やユーザプログラムを投機的に先行実行する受信メッセージ予測法によるメッセージ処理の高速化が期待できる。この方法によって、予測が的中した場合には、従来のアイドル時間をプログラムの実行に充てることができるため、大幅な高速化が期待できる。また、予測が外れた場合においても、OSやカーネルによる受信処理は、到着したメッセージの内容に依存しない部分も多く存在するため、結果的にこれらの受信処理部をアイドル時にキャッシュにプリフェッチすることとなり、高速化が期待できる。この受信予測に関連の深い研究として、現在、計算機アーキテクチャの分野では、条件分岐での分岐予測、あるいは投機的実行といった研究が盛んになされている[2]。

2.2 予測方式の検討

主にアイドル時間の長さによって、実装できる予測アルゴリズムの複雑さが決定される。予測項目が複数あることや、限られたアイドル時間を利用することを考慮すると、

1) 前回のメッセージのみを参照
2) 過去のメッセージの平均や発生頻度を参照
3) メッセージ発生系列のマルコフ連鎖
4) 前回の実行が完結したときのトレース

などが考えられる。最も容易に実装可能であるのは1)の手法で、これは前回に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、受信後の処理を行う。2)では、毎回到着するメッセージサイズが均一であるアプリケーションで有効であり、実装もメッセージサイズの合計や到着したメッセージの種類のカウントなど比較的容易である。3)では、メッセージの到着順序に強い規則性がある場合や、関数やメソッドの起動順序に制限があるような場合には有効である。4)は、データやその規模を変えて何度も実行する場合に有効となる。

全体的に、1)、2)、3)、4)の順で実装が複雑となり、メモリ領域が必要となるが、的中率も上がることが期待される。この中でどの手法を用いるかは各アーキテクチャでのアイドル時間やメモリサイズとのトレードオフとなる。

2.3 メッセージ受信処理の流れ

通常実行時(図1の上段参照)には一般的に、(1)ユーザプログラム(ユーザオブジェクト)が終了した場合、(2)他ノードへメッセージを送信し、その返答を待つ場合などに、システム(OS)に制御が移りその後処理を行う。後処理の終了後、次に実行されるべき処理が決定されてない場合、アイドル状態(メッセージ待ち状態)となる。他ノードからメッセージが到着すると、メッセージに対する領域確保や認識が行われ、ユーザプログラムの実行が継続される。

予測受信時（図1下段参照）においては、ユーザプログラムが終了し、後処理が行われた後、次に実行されるべき処理が決定されてない場合に、前節で述べた手法によって次に到着するメッセージを予測し、その結果にもとづいて、直ちに受信処理に入り、ユーザプログラムを実行／継続する。この投機的実行時には、予測失敗時にもとの状態に復元できるように、値が更新される変数について、そのアドレスと元の値を逐次記録するか、そのプログラムの変数領域を投機実行前にすべて保存しておく。

予測実行時に実際のメッセージが到着したときは、到着したメッセージと予測したメッセージとの比較を行い、同時に、到着したメッセージを今後の予測のためにログとして保存しておく。予測が成功した場合は、ユーザ処理を継続し、失敗した場合はふたたび受信処理から実行する。

また、予測したメッセージによる投機実行時には、他ノードへの影響が発生する可能性のある処理(他ノードへのメッセージ送信など)に到達した場合、そこで処理を停止しメッセージ待ちのアイドル状態とする。これは、予測が失敗であることが確定したとき

Performance improvement of user programs by message prediction.
Yoshiyuki IWAMOTO, Yasuo SAWADA, Kanemitsu OOTSU, Tsutomu YOSHINAGA and Takanobu BABA
Utsunomiya University

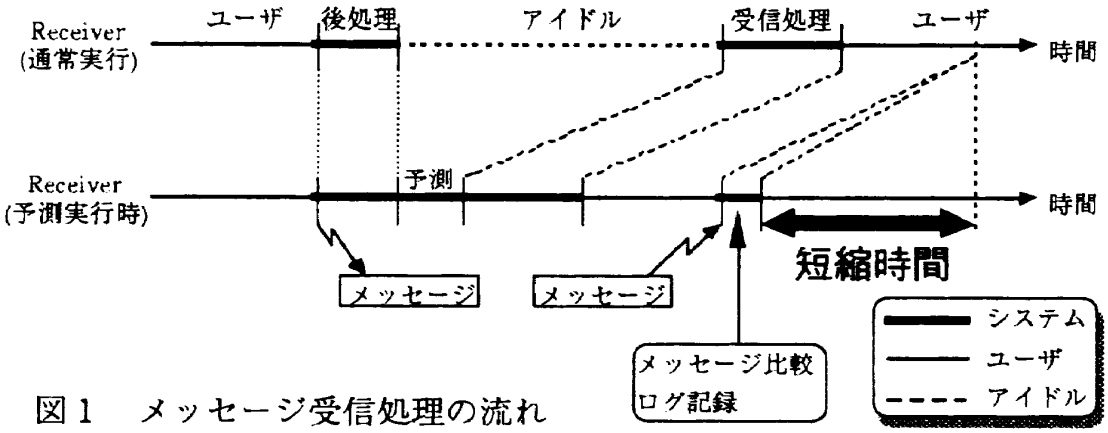

図1　メッセージ受信処理の流れ

に、影響を与えたすべてのノードの状態を回復するためのロールバックが必要となるからである。ロールバックの実現手法としては、ハードウェアによるサポートや、無効化のメッセージを送信することなどが考えられるが、実装が複雑になったり、処理や通信トラフィックの増加となる可能性が高い。

2.4 A-NETおよびAP1000に対する実装

受信メッセージ予測を評価するため、これまでに、A-NETマルチコンピュータおよび富士通AP1000に実装した。

A-NETマルチコンピュータではすでにOSのファームウェア化が行われているため、ファームウェアレベルのOSに対して実装する。A-NETL言語ではユーザプログラムの終了がtrm命令で行われ、その後処理後にメッセージ割り込み待ちに入ることでアイドル状態となり、このときに予測および投機的実行を開始する。

AP1000では、通常のメッセージ受信において頻繁に使われるl_arecv、readmsgの各命令に対して、並列ライブラリレベルで実装する。l_arecv命令が実行されたときに、引数として指定されたメッセージが到着してない場合にはメッセージ受信バッファ(リングバッファ)が書きかわるまで確認しつづけるため、このときに予測を行うこととする。

また、2.3節の最後に述べたように、予測実行時に他ノードへ影響を与える処理が実行されるときにはその前で処理を停止する必要がある。このために、A-NETおよびAP1000の両者において、メッセージ送信命令に対しても変更を加えたが、非常に簡単な変更で実現可能であった。

3　基本性能の評価

受信メッセージ予測法の効果を確認するために、基本的なメッセージ転送機能として、メッセージピンポンプログラムを作成し、予測導入前と予測導入後での実行時間を比較した。一般的に、受信メッセージ予測を行うためには、ある程度のアイドル時間が必要となるため、A-NET上ではメッセージピンポンの間に加算命令を1つ付け加え、この命令のみを投機的実行し停止する。これは、最低限の粒度であり、より粗粒度の一般的なプログラムではさらに高い効果が期待できる。一方のAP1000では、より一般的なアプリケーションに近いものとして、収束演算(ここでは単純にeのx乗を求める)をメッセージピンポンの間に(投機的に)実行させるプログラムを作成して実験を行った。

その結果、A-NETでは、予測実行時にはアイドル時間がなくなり、1回のメッセージピンポンあたり、従来の763マシンサイクル(MC、1MC=167ns)が711MCとなり、約6.8%の高速化を実現した。また、AP1000でも、ピンポンにおけるメッセージサイズが16バイトの場合で従来の1,015μ秒が716μ秒となり、約29.4%もの時間を削減した。また、メッセージサイズ1,024バイトの場合でも約17.7%の時間を削減できた。

4　おわりに

本稿ではメッセージ転送処理の高速化のために受信メッセージ予測法を提案し、その基本方針や予測の方法などを検討した。また、この方式をA-NETマルチコンピュータおよび富士通AP1000という異なったアーキテクチャ上に実装して基本的なメッセージ転送機能の評価をし、その効果を確認した。この手法はここで実装したA-NETやAP1000のアーキテクチャの他に、並列処理のライブラリとして一般に普及しているMPIにも、AP1000のl_arecvと同様な処理が行われていると考えられるMPI_Recv命令があるため、応用可能であると考えられる。

今後は、一般的なアプリケーションにおけるその効果の確認と、さらに複雑なメッセージ送受信パターンを持つアプリケーションにも対応できるように、複数の予測方法の実装とその評価を行う予定である。また、アイドル時間の長さをも予測することによって、複数の予測方法の中から動的に選択する方法の検討・実装・評価を行う。さらに、MPIなどの一般的に普及しているライブラリや他のアーキテクチャに対する導入も検討していく。

謝辞

本研究は、一部文部省科学研究費、基盤(C)09680324、並列・分散処理研究推進機構の援助による。また、OSのソースプログラムを提供いただいた富士通並列処理研究センターおよび、NEC C&C研究所 並列処理センターに感謝する。

参考文献

[1] Y.Iwamoto, K.Ooguri, T.Yoshinaga and T.Baba: A Comparison of Communication Performance in the NEC Cenju-3 and FUJITSU AP1000, Proc. of the FIRST CENJU WORKSHOP, pp60-64(1997).

[2] 児島彰, 弘中哲夫, 高山毅, 藤野清次: 複数分岐での投機的実行の有効性, 情報処理学会研究報告, 97-ARC-123-10, pp55-60(1997).