卒業研究報告書

題 目

PVMの仮想並列計算実験

指 導 教 員

石 水　隆 助手

報 告 者

02–1–47–087

仲 村　尚 記

近畿大学理工学部情報学科

平成18年2月10日提出

概要

複雑な問題を高速に解くためには並列処理が有効な手段である。並列処理を行うためには並列計算機が必要となる。しかし、並列計算機は非常に高価であり、また計算機自体の性能拡張も容易ではないため、実際に並列計算機を使用するには様々な障害がある。このため、ネットワーク接続された計算機の集合を仮想的な分散メモリ型並列計算機として用いるクラスタ (Cluster) 処理が注目されている。クラスタ処理は、単一の並列計算機に比べ、費用、機能拡張性といった点で優れており、最小限の追加コストで大規模な問題を解くことができるため、現在並列処理の主流となっている。

クラスタ処理を行うためのソフトとして、PVM (Parallel Virtual Machine)[4]、MPI(Message-Passing Interface)[3] 等がある。PVM はクラスタ処理を行うための実装パッケージであり、計算機の集合を単一の大規模並列計算機として用いることができる。また、MPI は、計算機間のインタフェースを規定しており、この規定に従うことにより計算機間の協調が可能となる。

本研究では、PVM を Windows 上に実装するに当たり、その方法と実装中および使用中に起こり得る問題点について調査および考察を行う。

目 次

1 序論

1.1 並列アルゴリズム

地球規模の気象シミュレーションや天体の軌道計算など、計算量の大きな問題を短時間で解く必要のある分野は多岐に渡っている。これらの問題に対して、従来の 1 台のプロセッサから成る逐次計算機を用いた逐次処理では非常に大きな時間が掛かる。このため、これらの問題を解く手法として、複数のプロセッサを持つ並列計算機 (Parallel Computer) による並列処理 (Parallel Processing) が現在注目されている。複数のプロセッサが協調してデータを処理することにより、問題を短時間で解け、またより複雑な問題を解くことができるようになる。しかし、並列処理を行うためには、プロセッサ間のデータのやり取りやメモリへのアクセス、プロセッサ間の同期等、並列特有の問題を解決せねばならない。このため、従来の逐次処理で用いられてきた逐次アルゴリズムをそのまま並列処理に用いることはできず、並列処理専用のアルゴリズム、すなわち並列アルゴリズム (Parallel Algorithm) が必要となる。そのため、現在様々な分野で、高速に処理を行う並列アルゴリズムが求められている。

1.2 並列計算機

複数のプロセッサを持ち、並列アルゴリズムを実行できる計算機が並列計算機である。並列計算機は、全てのプロセッサが共通したメモリに対して読み書きを行い、プロセッサ間の通信は共有メモリ (Shared Memory) を通して行う共有メモリ型並列計算機 (Shared Memory Parallel Computer) と、、それぞれのプロセッサが局所メモリ (Local Memory) を持ち、プロセッサ間の通信はネットワークを通じて行う分散メモリ型並列計算機 (Distributed Memory Parallel Computer) に大別される。並列計算機には上記の 2 種類あるため、それぞれの種類ごとにアルゴリズムが工夫されている。

また、並列計算機には、専用アーキテクチャを備えた専用マシンと汎用アーキテクチャの組み合わせによって構成された 2 種類のシステムが存在する。

前者は専用のバスを備えた SMP マシンが一般的である。例えば Sum エンタープライズサーバの一部はこれら専用ハードウェアによる共有メモリ型の並列計算機である。他にも、NUMA3 を用いたシステムもある。これらは、分散メモリマシンを共有メモリとしてハードウェアでエミュレートしたものである。これらのシステムは専用の部品で組まれているため、価格が非常に高いことが大きなネックとなっている。

後者のシステムは、クラスタ (Cluster) によって並列計算機を構成しようというものである。クラスタは複数台の計算機を高速のネットワークで繋いだものである。クラスタはかつてはシステムの信頼向上のための方法として主

に用いられていたが、現在は並列処理を実現するための技術として利用されることが主になり、急速に広まりつつある。このシステムは、分散メモリ型並列計算機に分類される。これらのシステムが重宝される理由として以下の3点が存在する。

1. 各マシンは PC(Parsonal Computer) レベルの性能の計算機で充分である。
2. 昨今の PC は非常に高性能である。
3. PC 自体の価格が低下してきている。

1. に挙げた通り、コストを優先させつならば各マシンは PC で充分である。このことにより、2.3. の理由と相まって高速な計算環境を比較的安価に構築することができるため、ネットワーク接続された計算機の集合を仮想的な分散メモリ型並列計算機として用いるクラスタ処理が注目されている。

1.3 PVM(Parallel Virtual Machine)

クラスタ処理を行うためのソフトとして、PVM (Parallel Virtual Machine)[4]、MPI(Message-Passing Interface)[3] 等がある。PVM はクラスタ処理を行うための実装パッケージであり、計算機の集合を単一の大規模並列計算機として用いることができる。また、MPI は、計算機間のインタフェースを規定しており、この規定に従うことにより計算機間の協調が可能となる。

PVM は異機種の計算機間の通信を前提としているため、PVM を実装する計算機の機種や OS を統一する必要は無い。一方、MPI は異機種間での通信は仮定されていないため、場合によっては機種、OS を統一する必要がある。また、PVM が動的プロセス管理により実行中の計算機中を変更できるのに対し、MPI は変更を行うことはできない。しかし、MPI には世界標準規定であるため移植性に富むこと、通信処理速度が速いといった利点があるため、PVM と MPI のどちらが優れているかは一概には言えない。

現在、PVM あるいは MPI により、重要な科学、産業、医学上の問題が解かれており、クラスタ処理における事実上の標準となっている。

本研究では、PVM を Windows 上に実装するに当たり、その方法と実装中および使用中に起こり得る問題点について調査および考察を行う。

1.4 本研究の目的

本研究では、PVM を Windows 上に実装するに当たり、その方法と実装中および使用中に起こり得る問題点について調査および考察を行う。

2 PVM (Parallel Virtual Machine)

2.1 クラスタ

クラスタとは、複数台のサーバを組み合わせて一つのより大規模なサーバシステムとして利用する技術である。1 台のサーバで障害が発生した場合、他のサーバが処理をフェイルオーバして業務を引き継ぐことができ、サーバの修理・交換中も処理を続けることができるため、システム全体がダウンすることが無い。データはサーバ間で共有するディスク装置に置くか、ネットワークを使用してサーバ間で同期を取る。クラスタはシステムの信頼性を確保するためには欠かせないテクノロジーの一つである。また、複数の計算機に処理を分散させ、システム全体のパフォーマンスを向上させることも可能となる。クラスタリングは従来 UNIX で利用されていたが、現在では本研究にもある Windows 環境でも構築が可能になっている。クラスタ処理を行うためのソフトとして、PVM (Parallel Virtual Machine)[4]、MPI(Message-Passing Interface)[3] 等がある。PVM はクラスタ処理を行うための実装パッケージであり、計算機の集合を単一の大規模並列計算機として用いることができる。また、MPI は、計算機間のインタフェースを規定しており、この規定に従うことにより計算機間の協調が可能となる。

PVM は異機種の計算機間の通信を前提としているため、PVM を実装する計算機の機種や OS を統一する必要は無い。一方、MPI は異機種間での通信は仮定されていないため、場合によっては機種、OS を統一する必要がある。また、PVM が動的プロセス管理により実行中の計算機中を変更できるのに対し、MPI は変更を行うことはできない。しかし、MPI には世界標準規定であるため移植性に富むこと、通信処理速度が速いといった利点があるため、PVM と MPI のどちらが優れているかは一概には言えない。

現在、PVM あるいは MPI により、重要な科学、産業、医学上の問題が解かれており、クラスタ処理における事実上の標準となっている。

2.2 PVM の利用法

PVM は、ネットワーク接続された異機種の並列計算機および逐次計算機を、単一の大きな並列計算資源に統合するパッケージソフトウェアである。

ソフトウェアシステムの構成は大きく 2 つに分けられる。一つはデーモンであり、PVMD3、あるいは PVMD と呼ばれる。デーモンは仮想マシンを構成する全ての計算機上に存在する。ユーザはログイン可能でさえあればどんな計算機にも PVMD3 をインストールすることができる。PVM アプリケーションを実行する場合、最初にユーザはどれか一つの計算機で PVMD3 を起動する。起動された PVMD3 は、ユーザが定義した仮想マシンを構成する計算機のそれぞれにおいて順次 PVMD3 を起動する。最後にどれか一つの計算機に

表示された UNIX または Windows のプロンプトに対してコマンドを入力することにより、PVM アプリケーションを実行する。複数のユーザは、 互いにコンピュータをオーバーラップさせてバーチャルマシンを構成でき、また、各ユーザは一人で複数の PVM アプリケーションを同時に実行することも可能である。ここで PVM 環境について概説する。問題は C(あるいは FOTRAN) で書かれた分割されたプログラムとして書かれ、個々のコンピュータで実行するためにコンパイルされる。ワークステーションのネットワーク上では、各ワークステーションはコンパイル済みのプログラムが置かれるファイルシステムにアクセスできる。プログラムが実行される特定のワークステーションに合わせてプログラムをコンパイルすることだけが要求される。ユーザがプログラム実行に使用する特定のコンピュータを指定しなければ、PVM は自動的に割り当てを行う。プログラムの実行に先立って、初めに問題を解く際に用いるコンピュータをファイル (ホストファイル) に定義し、“ 仮想並列マシン (virtual parallel machine) ”を構成しておかなければならない。

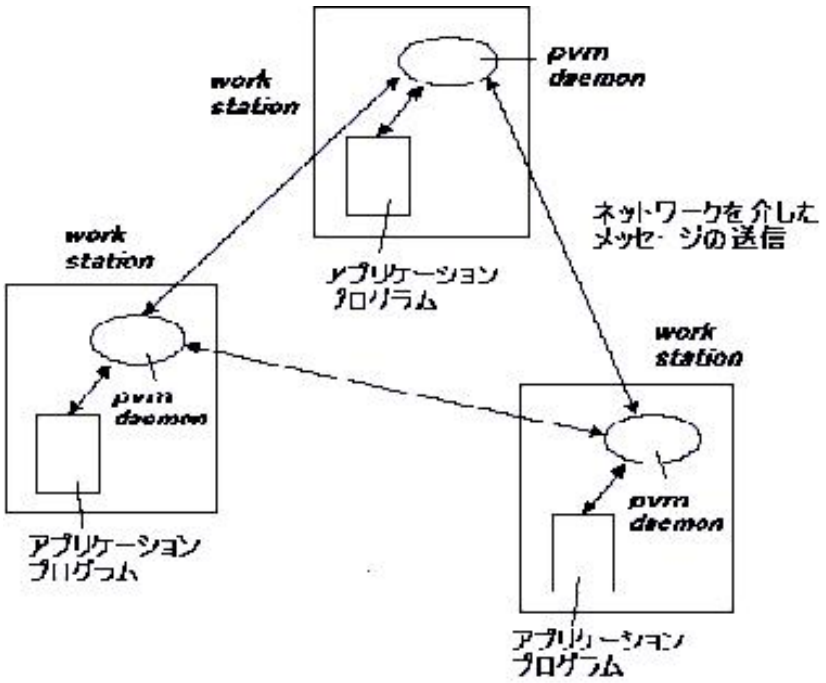

図 1: PVM を用いたワークステーション間のメッセージパッシング

コンピュータ間のメッセージのルーティングは、図 2 に示すように PVM によって、コンピュータ上に実装され仮想マシンを構成する PVM デーモンプロセスによって行われる。各 PVM デーモンはルーティングを行うために分な情報を必要とし、実際、すべてのプロセッサの大域的な情報を必要とする。

2.3 PVM の利点と欠点

ライブラリは汎用なので、一旦 PVM を使って並列化したコードを作れば対応するどのマシンでも並列動作が可能である。PVM に対応したコードは、PVM が動かないマシン上で稼働させることは例え 1 CPU 動作のみでも不可能になる。また、PVM は汎用的故に種々雑多なクラスタ環境での運用も可能です。つまりいろいろな種類のマシンが、ごく普通のネットワーク環境下で混在しているような状況でも運用できるということである。この場合、通信を多用する、あるいは通信量が非常に多い並列計算では十分な並列化による効果を引き出せない可能性がある。つまり個々のマシンの速度が異なれば、通信発生時に一番遅いＣＰＵの処理終了まで待たされ、それが他の足を引っ張る可能性がある。加えて、メーカー製の並列マシンに PVM がインプリメントされていて、実装や、運用、保守がメーカー任せなら良いが、既存のスカラーマシンをクラスタ化して、それに PVM を実装させる場合、その実装のための作業から、実際の運用、保守全てが、研究者当人の仕事となり、負担となります。PVM における最大の問題点は、移植性である。

2.4 PVM の構築

PVM の構築を行うためには、まず PVM のソースファイルが必要となる。PVM はフリーソフトであり、現在、http://www.netlib.org/pvm3/ 等、様々なサイトから UNIX、あるいは Windows 用の PVM のソースファイルをダウンロード可能である。PVM では、それを構築する全ての計算機上でデーモンを作る必要があるため、PVM を構築する全ての計算機にソフトをダウンロードする必要がある。ダウンロードし、コンパイルした後、PVM を実行させるためには、主に環境変数の設定とパスの指定が必要となる。

このとき、計算機の機種、OS が異なれば環境設定も異なるため、個々の計算機ごとにその環境に応じた値に設定する必要がある。

PVM 上でのプログラミングは、現在、C，C++，FORTRUN，JAVA といった言語で可能である。これらの言語中で、計算機間でのメッセージ通信を行う命令を記述することにより並列計算が可能となる。PVM には、ユーザプロセスを PVM タスクにする関数及び PVM タスクを再びユーザプロセスにする関数がある。 また、バーチャルマシンにホストを追加・削除する関数、PVM タスクを起動・終了する関数、他の PVM タスクにシグナルを送信する関数、及びバーチャルマシンの設定とアクティブな PVM タスクに関する情報を得る関数がある。

もしあるホストに障害が発生すると、PVM はそのホストを自動的に検出し、バーチャルマシンから削除するフォールトトレランス機能がある。アプリケーションは、各ホストの状態を PVM に要求することができ、代替のホストを追加することが可能である。 ホストの障害に対するアプリケーションの

復旧は、全てアプリケーション開発者に任されている。 ホストの障害によって強制終了させられたタスクに対して、PVM は自動的な復旧を試みることはない。 PVM 3 では、ダイナミックプロセスグループが実装されていてプロセスは複数のグループに所属することができ、 実行中にいつでもグループを変更できる。 ブロードキャストやバリア同期といった論理的なタスクのグループを扱うための関数は、ユーザが明示的に定義したグループ名を引数としてとることができる。 グループへの所属と離脱のための関数が提供される。タスクは他のグループについての情報を問い合わせることもできる。PVM は、他の PVM タスクにシグナルを送るための 2 つの方法を提供する。一つは、UNIX のシグナルを他のタスクへ送る方法である。 もう一つは、あるイベントに対してユーザが定義したタグを持つメッセージを、タスクの集合に対して通知する方法である。 アプリケーションは、このメッセージをチェックすることができる。この通知イベントには、タスクの終了、ホストの削除 (あるいは障害)、及びホストの追加がある。 タスク間でのメッセージのパックと送信を行う関数を提供する。 PVM モデルでは、任意のタスク間でメッセージを送ることができること、並びにメッセージのサイズ及び数に制限はないことを仮定している。全てのホストの物理メモリは有限であり、使用できるバッファの大きさを制限する。一方、 通信のモデルでは、マシン固有のメモリ制限には束縛されず、十分なメモリを利用可能であると仮定している。PVM の通信モデルは、非同期ブロック送信、 非同期ブロック受信及び非ブロック受信の 3 つの関数を提供する。 ここで我々の定義では、ブロック送信は、送信バッファが再使用のために開放された時点でリターンし、受信側の状態には依らないものとする。非ブロック受信は、到着したデータとともにリターンするか、あるいはデータ未着の場合それを示すフラグとともに直ちにリターンするものとし、 一方ブロック受信は、データが受信バッファに存在する時のみリターンするものとする。これらの 1 対 1 通信関数に加えて、モデルはタスク集合に対するマルチキャスト、 並びにユーザが定義するタスクのグループへのブロードキャストをサポートする。 送信元を指定する、あるいは無視する際には、ワイルドカードを用いることができる。 PVM モデルは、メッセージ順序の保存を保証している。 まず、タスク 1 がタスク 2 へメッセージ A を送信し、その後タスク 1 がタスク 2 へメッセージ B を送信すると、 メッセージ A はメッセージ B よりも早く到着する。 更に、 タスク 2 が受信を行う前に両方のメッセージが到着した場合、 ワイルドカードを指定した受信は常にメッセージ A を返す。 メッセージバッファは動的に割り当てられる。 従って、送信あるいは受信可能なメッセージの大きさは、ホストで利用可能なメモリ量によってのみ制限される。 PVM 3 で記述されたプログラムはたとえば、SUN のネットワークまたは、Intel の Paragon のノードのグループらに、ネットワークで接続された複数の Paragon、 あるいは世界中に分散した異機種マルチコンピュータの組合せのいずれであっても実行可

能である。PVM 3 では、マルチプロセッサ内の通信は、マシン固有の通信関数を使用するように設計されている。 同一マルチプロセッサ内のノード間で交換するメッセージは直接転送され、一方ネットワーク上の他のマシンへのメッセージは、 マルチプロセッサ上でユーザが起動した PVM デーモンに対して送られ、更に他のマシンへと転送される。

3　PVM の構築する上で起こりえる問題点

1. 仕事の分担の自動化は難しく, コスト (手間) がかかる.

2. 仕事には, 段取り, 順序が存在する.

1 に関しては, 実際に並列処理を実現するプログラミングモデルを考えれば, 容易にわかるであろう. 個々の問題には, その問題独自の性質が存在し, 問題間で一般性を見つけ出すことは難しい. 人が, 毎回プログラムの際に, 仕事の分担を考えていては, そのコストは大きなものとなってしまう. そのため, 仕事の分担に対しては, 通信のレイテンシ (遅延) を考えた分割法など, 仕事そのもののタスクの振り分けのみならず, そのアーキテクチャのバックグラウンドまで考える必要が生まれる.「通信時間 くく計算時間」という関係が, 少なくとも成立する仕事でなければ, 仕事を分担した意味がなくなってしまう. 以上から, 仕事の分担は並列処理を考える上で, 第一に問題となるものである. 2 に関しては,1 で述べたことに付随する形で存在する. つまり仕事には順序が存在し, その順序の破壊は, 即ちシステムとしての破壊を生む. 次に述べることは, 逐次プログラムでもいえることであるが, 並列処理では通信を多用するため, タイミングによりデータのハザードや, デッドロックが起きる可能性がある. しかも, これらのミスは頻繁に起きやすい.

4　結果および考察

PVM はそれを構築する各計算機がデーモンを作り TCP/IP 通信を行う。このため、比較的大きな大域幅のネットワークが必要とされる。ネットワークの大域幅小さい場合、通信に大きな時間が掛かり、結果として計算機 1 台よりもかえって多くの時間が掛かる可能性もある。従って、PVM の利用には充分な大域幅持つネットワークを準備せねばならない。並列処理を行うためには並列アルゴリズムが必要である。PVM は分散メモリ型の並列計算機であるため、PRAM の並列アルゴリズムを PVM 上で実行させた場合、効率良く実行できるとは限らない。従って、PVM を使用する場合は、BSP モデルや CGM モデルといった PVM に適応したモデル上で並列アルゴリズムを設計する必要がある。

5　結論

本研究では、PVM を Windows 上に実装するに当たり、その方法と実装中および使用中に起こり得る問題について調査および考察を行った。PVM を Windows 上で起動するためにインストールし、環境変数の設定を行った。環境変数でのエラーは PVM のフォルダに移動するだけで修正できた。

今後の課題としては、起こり得る種々のエラーに対し、それらの原因と対処法を機種、OS 別に分かりやすくことが考えられる。

6　謝辞

本研究をするにあたり、数々のご指導とご鞭撻を頂いた石水隆先生には感謝の気持ちでいっぱいです。また、多大な迷惑をお掛けしましたことを心から深くお詫び申し上げます。また、助言、ご協力をして頂いた研究室の皆さんにも深く感謝いたします。この一年間本当にありがとうございました。

参考文献

[1] F.Dehne, A.Fabri and A.Rau-Chaplin: ”Scalable Parallel Computational Geometry for Corse Grained Multicomputers, ” Proceeding of ACM Symposium on Computational Geometry, pp.298–307 (1993).

[2] J.JáJá: ”An Introduction to Parallel Algorithms,” Addison-Wesley Publishing Company, 1999

[3] P.S.Pacheco, 秋葉博 (訳): ”MPI 並列プログラミング,” 培風館 (2001)

[4] 村田英明:”PVM3.4 & リファレンスマニュアル, ” (1995).

[5] L.G.Valiant: ”A Bridging Model for Parallel Computation, ”, Communications of the ACM, Vol.33, No.8, pp.103–111, (1990).