卒業研究報告書

題目

MPI による並列計算

指 導 教 員

石水 隆 助教

報告者

06-1-037-0043

坂東 憲治

近畿大学理工学部情報学科

平成 22 年 2 月 5 日提出

概要

現在、様々な情報が計算機で取り扱われている。取り扱われる情報の量は日々増大しており、その処理時間を短縮することは計算機を使用する上での重大な課題である。高速な処理を行うためには、複数のプロセッサを持つ並列計算機（Parallel Computer）が用いられる。並列処理の概念自体はもはや新しいものではないが、近年 CPU のマルチコア化が進み、従来は高価なサーバーしか搭載されていなかったマルチコア CPU も、一般的な PC に普及しつつある。そのため、マルチコア CPU と組み合わせて効果を発揮する並列処理がより重要なものとなった。

本研究では、無料提供されている仮想並列計算環境を構築するソフトウェアのひとつである MPI(Message Passing Interface)[1][2]を用いて基本問題の一つである行列積演算を行い、並列計算の有用性を実証するために、ネットワークを利用した仮想並列計算を行う。MPI は無料提供されているソフトウェアであり、ゴードン国立研究所[3]の MPICH2 のページ[4]からダウンロードすることにより容易に使用することが可能である。MPI はライブラリレベルで並列化するため言語を問わず利用でき、並列プログラミングの規格として広く使われている。

目次

1 序論

1.1. 仮想並列計算

現在、様々な情報が計算機で取り扱われている。取り扱われる情報の量は日々増大しており、その処理時間を短縮することは計算機を使用する上での重大な課題である。高速な処理を行うためには、複数のプロセッサを持つ並列計算機（Parallel Computer）が用いられる。しかし、一般的に並列計算機は高価であるために、容易に用いることはできない。そこで、複数の計算機をネットワーク接続することにより仮想的な並列計算機として利用する仮想並列計算(Parallel Virtual Computing)が現在注目されている。

1.2. MPI と PVM

仮想並列計算の構築には、MPI(Message Passing Interface)[1][2]や PVM(Parallel Virtual Machine)[5][6]等のソフトウェアを用いて行うことができる。MPI は Message Passing Interface の略称であり、メッセージ通信のプログラムを記述するために広く使われる「標準」を目指して開発されたもので、メッセージ通信の API 仕様である。MPI はインターフェースの規定であることに対し、PVM は実装パッケージそのものである。PVM は、TCP/IP ネットワークで接続された何台ものコンピュータを仮想的に 1 台のマシンととらえて、並列プログラムを走らせることのできるメッセージ・パッシングの環境とライブラリを提供するものである。

MPI と PVM では、MPI のほうが後に作られているため、MPI は PVM の問題点を学んで作られているところが多い。PVM が動的なプロセス管理が可能であったのに対し、初期の MPI では動的なプロセス管理は不可能であった。しかし、MPI-２では動的なプロセス管理の機能が取り入れられた。動的なプロセス管理とは、アプリケーションの中から動的にプロセスを生成したり、停止したりすることである。この機能は、処理中に必要に応じてプロセス数を調整できるため、効率の良い並列計算には欠かせないものである。

また、PVM は移植性が悪いという欠点があり、その点、PVM の問題点を学んで作られた MPI は、移植性も良く、バッファ処理も優秀であり高速にメッセージを受け渡すことができる。

これらの事情により、現時点において仮想並列計算の世界標準は PVM から MPI に移り変わりつつある。本研究では、無料提供されている仮想並列計算環境を構築するソフトウェアの一つである MPI(Message Passing Interface)[1][2]を用いて基本問題の一つである行列積演算を行い、その性能を実験的に評価する。MPI は無料提供されているソフトウェアであり、ゴードン国立研究所[3]の MPICH2 のページ[4]からダウンロードすることにより容易に使用することが可能である。

1.3. MPICH

MPICH は、ゴードン国立研究所[3]というアメリカの研究所が開発を行い、無償でソースコードを配布したライブラリであり、PVM の問題点を元に、移植しやすさを重視した作りになっている。この MPICH は UNIX や Linux に限らず、Windows 系へのサポートもしており、OS への対応が充実している。本研究では、MPICH の最新のヴァージョンである MPICH２を使用して研究を進めていく。

1.4. 各種インストールと環境設定

MPICH のインストールと環境設定

仮想並列計算環境を作るため、MPICH のホームページよりパッケージをダウンロードし、インストールを行う。

インストール後に、環境変数を用いて MPICH プログラムのあるフォルダへの PATH を指定をする必要がある。この指定は、マイコンピュータのプロパティから行うことができる。プロパティで表示されるシステム環境変数のリストから、変数名 path を選択し、変数値の最後に;C:¥Program Files¥MPICH2¥bin¥を追加すればよい。PATH の指定が正しくできているかの確認は、新規にコマンドプロンプトを立ち上げ、mpiexec と入力したときに、引数の Usage が表示されるかどうかで確認することができる。

Visual C++のインストール

次に、C 言語の環境を作るために、Visual C++のインストールを行う。
VisualC++2005 Express Edition が、マイクロソフトの公式ページ[7]より配布されているので、その仮想 CD をダウンロードしインストールを行うことができる。

Microsoft Platform SDK のインストール

Visual C++の開発環境をあげるために、マイクロソフトの公式ページ[8]より配布されている Platform SDK のインストールを行う。

Platform SDK のインストールその後、Visual C を起動し、Platform SDK の Executable、Include、Library に PATH を通し、更新を行う。これにより、Visual C++の環境が向上される。

1.5. ワークグループの設定

OS が Windows の場合には､使用する計算機に共通のアカウント名とパスワードを持つユーザーを設定しておく必要がある。並列環境の作成のため、それぞれのコンピュータに mpi アカウントを作りパスワードを共通のものとし、ワークグループを「HEIRETSU」として統一する。これにより、コンピュータ同士でのネットワークが構築される。

また、プロセスの実行の際には、すべての使用コンピュータに実行ファイルを置く必要があるため、mpi 共有フォルダをそれぞれのコンピュータに準備する。

2 研究内容

2.1. 目的

本研究では並列計算の有用性を示すために、MPI を用いて以下の計算を行う。行列計算を１つのコンピュータで計算する場合と、MPI を用いて複数のコンピュータに計算する場合と分けて、処理時間にどれだけ差があるかを計測する。

2.2. 計算方法

2.2 計算方法

本研究では、ランダムに生成された正方行列を使用する。また、行列のサイズを 10*10、100*100、500*500、1000*1000 と変化させて、それぞれ測定を行う。

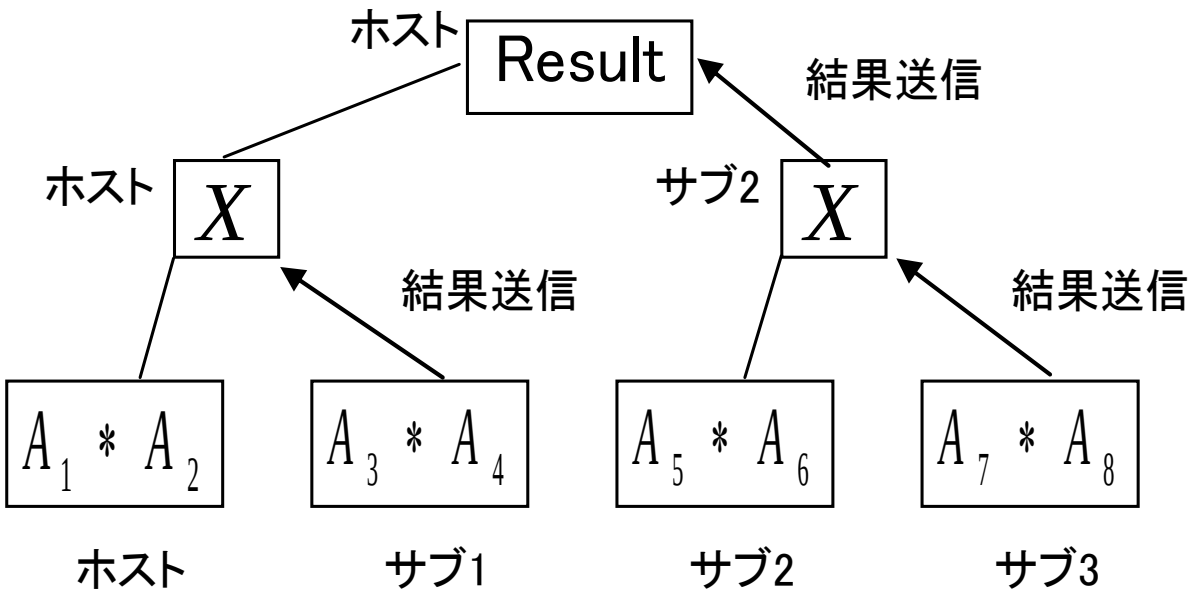

図１：行列計算の概念図

図１に計算の概念図を示す。初期条件では、ホストコンピュータがすべての行列を保持している。ホストコンピュータは、他のサブコンピュータに生成した行列を２つずつ送信する。各コンピュータにて、受け取った行列同士の積を計算し、その結果を上位コンピュータに渡し、また計算を繰り返す。最後に、ホストコンピュータに送り、それらをホストコンピュータで計算する。この結果を表示するまでの時間を計測し、この作業を１台で行ったときと、複数台で行ったときの計測時間を判定する。

3 結果・考察

表 1：行列計算の処理時間(秒)

CPU＼行列数	10	100	500	1000
1 台	0.015	0.066	10.87	167.2
4 台	0.012	0.051	7.29	86.3
向上率	1.3 倍	1.3 倍	1.5 倍	1.9 倍

上記の表 1 が MPI において行列計算を行った時の処理時間である。各場合において 50 回のテストを行い、その平均値を取った。CPU の数に関係なく、行列数が増えるにつれ処理時間が増えている。台数の違いを見ると、CPU が 1 台のときに比べ、4 台のときはいずれも速度が向上しているのがわかる。特に 1000*1000 の行列計算のときの処理時間の向上率は 2 倍近くにもなり、処理数が多くなるほど向上率も大きくなると考えられる。これは処理の少ない場合より、処理の多い場合のほうが、送受信や同期の時間による影響が少ないため、向上率が上昇すると考えられる。また、1 台の計算時間はややばらつきがあることに対し、複数のコンピュータによる処理は安定した処理速度となった。これは、コンピュータの数が少なくなればなるほど、1 台のコンピュータにかかる負担が大きくなるためであろうと考えられる。

4 結論・今後の課題

本研究により、並列計算による処理速度の向上を確認することができ、並列処理は有用性が高いということがわかった。しかし、単に CPU を増やし並列処理を行えば高速になるというわけではなく、並列化によって何らかの問題が生じる可能性もあるので、場合によって計算する CPU の数を変更したり、データの分割、送信、同期方法を考えるなど、今後更なる研究が必要となる。

並列化の研究が進めば、進化するマルチコア CPU とあいまって、より膨大なデータを処理することが可能になる。当然実際マルチコア CPU を用いて並列計算を行うという研究も重要である。

謝辞

本研究を進めるにあたり、石水先生をはじめ、同じ研究室の皆様方には多くの助言をいただき、非常に感謝しております。この場でお礼を申し上げます。

参考文献

[1] P パチェコ 著,秋葉博 訳:MPI 並列プログラミング, 培風館（2001）

[2] 渡邊真也 著：MPI による並列プログラミングの基礎,
http://mikilab.doshisha.ac.jp/dia/smpp/cluster2000/PDF/chapter02.pdf

[3] Argonne National Laboratory,
http://www.mcs.anl.gov/research/projects/mpich2/indexold.html

[4] MPICH2, http://www.mcs.anl.gov/research/projects/mpich2/

[5] PVM, Parallel Virtual Machine, http://www.csm.ornl.gov/pvm/

[6] PVM, http://erpc1.naruto-u.ac.jp/~geant4/pvm/pvm.html

[7] Visual Studio 2008 Express Editions,
http://www.microsoft.com/japan/msdn/vstudio/express/default.aspx

[8] Windows® Server 2003 SP1 Platform SDK Web Install,
http://www.microsoft.com/downloads/details.aspx?FamilyId=A55B6B43-E24F-4EA3-A93E-40C0EC4F68E5&displaylang=en