卒業研究報告書

題目

MPI による仮想並列処理

指 導 教 員 石水 隆 助教

報告者

04-1-47-278

阪木康寿

近畿大学理工学部情報学科

平成 20 年 2 月 4 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）[4]が用いられる。しかし、一般に並列計算機（超並列マシン・スーパーコンピュータ）は非常に高価であるため容易に利用できない。そこで、複数の計算機をネットワークで接続し、接続された複数の計算機全体を１つの仮想な並列計算機として用いる仮想並列計算(Parallel Virtual Computing)の採用による解決が考えられる。

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ひとつである MPI（Message Passing Interface）[1][2][3]の実装により仮想並列計算機を構築し、その性能の評価実験を行う。評価方法としては、MPI を用いて基本問題の１つである行列積計算を行う。このとき、入力する行列のサイズ・並列計算に用いるプロセッサ数を変化させ、各処理時間から速度向上について検証を行う。

目次

第1章　序論

1.1 並列処理(Parallel Processing)

ある 1 つの処理を、複数のプロセッサを用いて協調して行う事で単一のプロセッサでの処理よりも高速に計算処理を行うことを並列処理（Parallel Processing）という。並列処理を用いることにより、処理時間の大幅な短縮が得られ、また、処理能力も向上すると期待されている。

並列処理は地球規模の環境変動の解明・予測に用いられる地球シミュレータ[0]など、極めて大きな計算処理を必要とする問題の解決に用いられる。

1.2 並列計算機(Parallel Computer)

複数のプロセッサを用いて並列処理を実行可能な計算機を並列計算機（Parallel Computer）[4]という。並列計算機は、全てのプロセッサが同じメモリにアクセスし、計算データなどを一箇所に置く共有メモリ型並列計算機（Shared Memory Parallel Computer）と各プロセッサがそれぞれ個別にアクセス可能なメモリを持ち、データの受け渡しにはネットワークなどの通信を用いる分散メモリ型並列計算機（Distributed Memory Parallel Computer）の 2 種に大別される。共有メモリ型並列計算機は同期問題やデータの通信・遅延といった問題について、メモリを共有しているため対処しやすいが、プロセッサ数の増減などの変化があった場合にはメモリにすべてのプロセッサを接続させることが困難になってしまう。そのため、現在では局所メモリが使用できる分散メモリ型並列計算機が主流となっている。

1.3 仮想並列計算(Parallel Virtual Computing)

一般的に、並列計算機~~自体~~は非常に高価なものであり、維持コストも非常に高くなる。そのため、並列計算機を持つのはごく一部の大学や研究所そして企業しかなく、注目はされていたが利用しにくい状況であった。

現在は、複数の計算機をネットワークで接続し、接続された計算機全体を仮想計算機（Virtual Machine）として用いるクラスタ処理(Cluster Computing)[5]やより大きな規模のネットワークで用いられるグリッド処理（Grid Computing）[5]が幅広く使用されている。現在、様々な企業・グループなどでクラスタ処理を実装できるソフトウェアが無料で提供されていることもあり仮想並列計算機は身近なものになっている。提供されているソフトウェアとしては、共有メモリ型並列処理用として OpenMP[6][7]、分散メモリ型並列処理用として PVM(Parallel Virtual Machine)[9][10]や MPI(Message Passing Interface)[1][2][3]を実装したものがある。昨今では、共有メモリ型並列処理より、分散メモリ型並列処理の方が用いられることが多いため、PVM や MPI の使用が主流となっている。PVM・MPI については次に説明する。

1.4 PVM (Parallel Virtual Machine)

PVM(Parallel Virtual Machine)は、1991 年に米国のオークリッジ国立研究所(Oak Ride National Laboratory)[8]を中心に異機種間の分散処理が目的に開発された、メッセージパッシングによる並列処理を行うための並列化ライブラリである。

PVM はワークステーションクラスタなため、TCP/IP の通信ライブラリで一般的に使用されている LAN 環境があれば並列処理が実行出来るので多くのユーザが利用している。また、異機種間の通信も考慮されているため、対応する計算機は家庭にあるパーソナルコンピュータからスーパーコンピュータなど多くの種類で PVM による並列処理が出来る。

PVM の構成はデーモンとルーチンライブラリの 2 つに大きく分けられる。PVM を導入した計算機上で pvm3 と呼ばれるデーモンを起動すると、仮想並列計算機を構成する全ての計算機上に常駐する pvm3 デーモンを起動し、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ことができる。PVM はこのデーモンを使用し通信を行っている。複数のユーザは互いにオーバーラップさせ仮想並列計算機を構成することが出来~~る~~、また、各ユーザは PVM アプリケーションを 1 人で複数実行することが可能となっている。

PVM のもう 1 つの構成要素であるルーチンライブラリ~~に~~は、メッセージパッシング、プロセスの生成、タスクの協調、仮想計算機の構成ルーチンを提供している。

PVM の大きな特徴として、耐故障性(Fault Tolerant)があげられる、PVM は任意で計算機の追加や削除が

行なえる。通常、仮想並列計算機で計算中に、ある 1 台が停止してしまうと、計算処理が出来なくなってしまうが、PVM は故障した計算機を仮想並列計算機内から迅速に削除され、計算処理自体が停止してしまうことがなく続行できる。

PVM の問題点として、PVM は多くの並列計算機に移植されるようになったとき、各並列計算機ベンダが独自にチューニングを行なった PVM を開発してしまい。PVM で作成をしたプログラムの移植性が乏しくなってしまったことが挙げられる。

1.5 MPI (Message Passing Interface)

MPI(Message Passing Interface)[1][2][3]は、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、メッセージ通信の API 仕様である。そのため、PVM と違いソフトウェアがあるというわけではない。

MPI は 1992 年に結成された MPI Forum(MPIF)により標準使用の定義や検討を作り始めたことで具体化してきた。MPI の開発には、アメリカ、ヨーロッパの 40 の組織から 60 人の人間が関わっており、研究者や主な並列計算機ベンダのほとんどが参加した。MPI は PVM よりも後に開発されたため、PVM と同等の機能を持っており、問題点の改善も含まれている。

MPI は標準を目指して作成されたために様々な通信関数が実装されている、MPI 規約を用いて作成したプログラムは移植性が高いため、MPI を使用するユーザは、通信を考慮せずプログラムを組むことが出来る。

MPI が大きく PVM と異なる点は、異機種間の通信が考慮されていないことである、MPI を用いての仮想計算機の構築には使用する計算機のオペレーティングシステム(Operating System)が同じでなければならないという制約が存在する。

MPI は専用の並列計算機からワークステーション、パーソナルコンピュータに至るまで幅広くサポートしている、無料で提供されている主な実装は MPICH[12]や LAM[13][14]といったものがある。

MPI のサポートするプログラミング言語は多く、C 言語や Fortran そして最近では Java などに対応している。

1.6 PVM と MPI の差異

PVM と MPI の大きな違いとして動的なプロセス管理があったが、最近 MPI の新規格である MPI-2 の仕様において動的なプロセス管理の機能が取り入れられたことから、PVM の優位性の 1 つが失われている。しかし、MPI-2 は前の規格である MPI-1 の関数も使用できるが、新規格である MPI-2 の機能を完全にサポートしている実装が無いという問題がある。

PVM の利用時の大きな欠点は移植性に乏しいということがいえ、MPI は PVM と違い世界的な標準が目的とされ開発されたものであるため移植性が高く評価されている。

MPI の欠点は、MPI は高レベルのバッファ操作を可能とするために同一のワークステーション・クラスタを対象としていることである、MPI はこの性質上仮想計算機を構成する全ての計算機が同じ OS で無ければ動作がしないため、異機種間での並列処理ができないことである。その点 PVM は MPI と違い異機種間における処理を目的で作成されているので、異機種の計算機による並列処理に適していると言える。

MPI は同機種における並列処理が望ましいため近くにある同機種の計算機でのクラスタ処理が、PVM はデーモンを使用しての TCP/IP 通信ができ、異機種の計算機を使用できるという特性を持っているため、様々な異機種の計算機がある場所や通信を利用し離れた計算機も使用できるという点からクラスタ処理とグリッド処理にそれぞれ利用できる。

現在の両方の開発状況は、PVM の開発は昔ほどに進んでいないように思われる。逆に、MPI は現在でも新規格である MPI-2 の研究開発が盛んに行なわれている。

1.7 本研究の目的

本研究では、MPI による仮想並列計算機の性能を実験的に評価する。評価方法として n 行 n 列の正方行列 A を用意し、$A \times A$ の行列積計算を並列化した並列行列積計算プログラムを作成し、1 台の計算機での逐次処理時における行列積計算の処理時間と比較して、複数台の計算機を使用しての並列行列積計算においてどれだけの処理時間の向上が出来ているかの検証を行なう。

1.8 本報告書の構成

本報告書の構成は以下の通りである。2 章では、本研究の使用機器および MPI の導入方法と並列行列積プログラムについて述べる。3 章では、並列計算による処理時間に関する結果を示し、次の 4 章においてその考察を行う。5 章では結論と今後の課題について述べる。

第2章　準備

2.1 仮想並列計算機の構成

まず、研究を行なうにあたって仮想並列計算機を構築するソフトウェアを配布している Web Site から使用するソフトウェアをダウンロードし、各計算機にインストールと環境設定を行う。今回の研究で使用する仮想並列計算機を構築するソフトウェアは、MPI を無料のソフトウェアとして実装された MPICH2[12]というソフトウェアである。MPICH2 は、MPI の特徴である移植性の高さを持つこと、MPI の新規格である MPI-2 をある程度サポートしていること、また、移植性が高く、Windows OS にも導入が可能であり現在でも研究開発が盛んに行なわれていることなどの理由から、本研究では MPICH2 を使用する。

本研究では、多くの情報学科 04 年度入学生が持つノートパソコン Let's Note CF-W2[15]を 8 台用いて実験を行う。なお、MPI による仮想並列計算機を構築する場合、OS の統一を行う必要性がある。今回は Microsoft 社[16]が提供販売を行っている WindowsOS を用いて仮想並列計算機の構築を行った。

仮想並列計算機を構築する際に LAN(Local Area Network)やハブ(Hub)を使用し計算機をネットワークで接続している。この構成図を図 1 に示す。

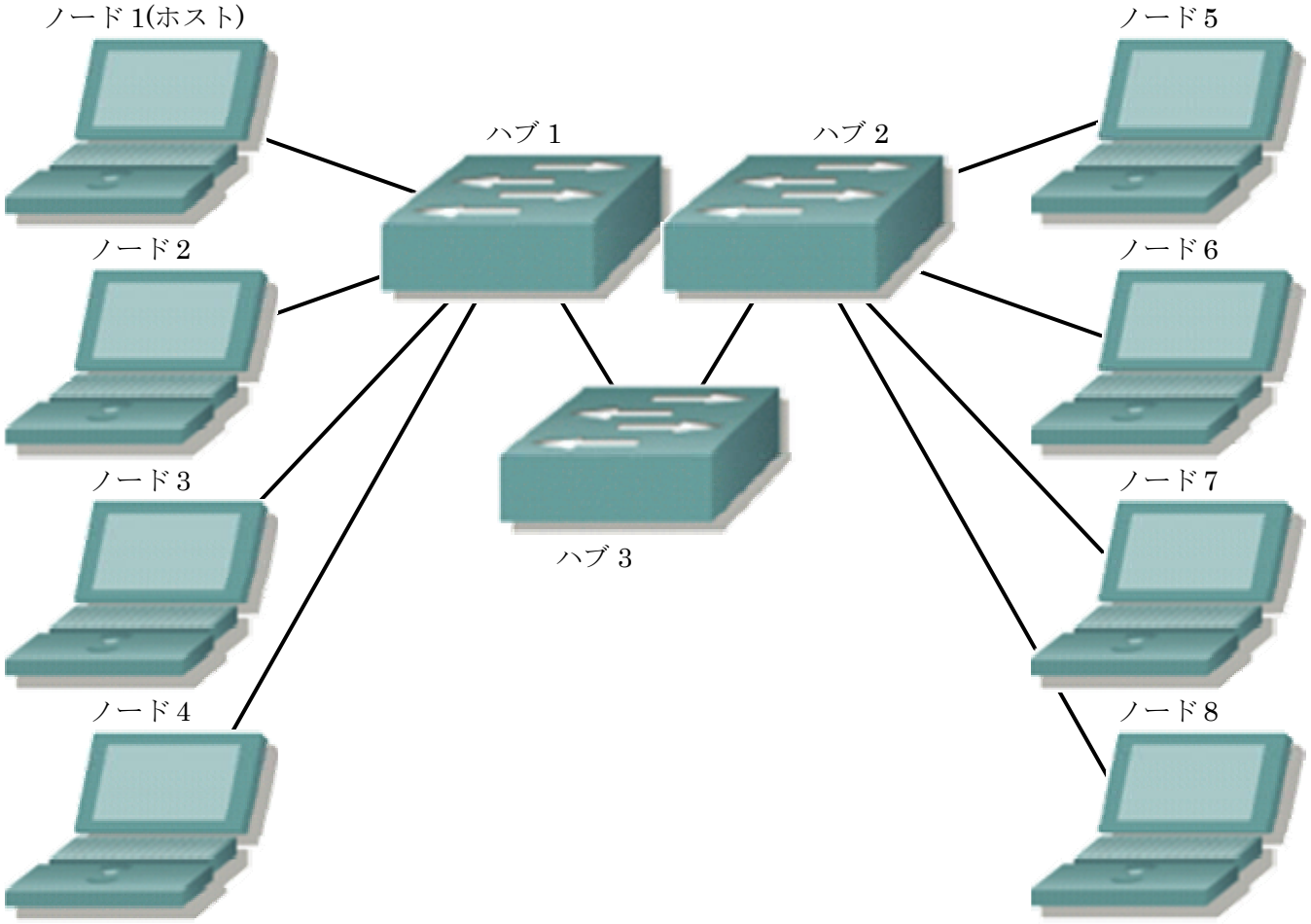

図 1　仮想並列計算機の構成

2.2 MPICH-2 のインストールと環境設定

MPICH2 を使用するには、計算機に MPICH2 をインストール(Install)する必要がある。インストールは以

下の手順で行う。

1. MPICH2 の公式ページ[12]から Windows 用 MPICH2 のファイル mpich2-1.0.6p1-win32-ia32.msi をダウンロードする
2. MPI を使用する全ての計算機でダウンロードしたファイルをインストールする。ダウンロードしたファイルをクリックすることにより、インストーラー(Installer)が起動し自動的にインストールされる。インストール時には MPICH2 をインストールするフォルダを指定しなければならない。本研究では、C:¥Program Files¥MPICH2 フォルダへインストールを行った。
3. MPICH-2 のバイナリのあるフォルダに対して各計算機の環境変数 PATH を指定する。本研究ではフォルダ”C:¥Program Files¥MPICH2¥bin”に対して環境変数 PATH の指定を行った。
4. 各計算機にネットワークを通して共有できるフォルダを設定する。本研究では、各計算機でフォルダ”C:¥mpi”を作成し、このフォルダのプロパティをネットワークを通じて共有できるように設定を行った。この操作を行うことで実行ファイルの受け渡しが迅速に行なえるようになる。
5. 各計算機に MPICH-2 が使用するためのユーザを設定する。本研究では、各計算機に管理者権限を持つユーザ”mpi”を作成し、また、そのパスワードの設定を行った。

今回の研究時には、各計算機共通の共有フォルダ上での実行ファイルによる処理において集団通信関数が使用できない不具合が生じたため、各計算機に個別の共有フォルダを作ることで実行ファイル配布時の作業においてなるべく無駄を省くことにした。

今回作成する並列行列積計算プログラムは C 言語でプログラミングを行い、コンパイルツールは Microsoft 社製の Visual C++.net[17]を使用する。

プログラミングをするにあたって MPICH2 を使用できるように環境を設定する必要がある。設定の手順を以下に示す。

1. 空のプロジェクトを作成する
2. VisualC++のツールオプションから MPICH-2 のインクルードファイルおよびライブラリファイルのあるフォルダ”C:¥Program Files¥MPICH2¥include”および”C:¥Program Files¥MPICH2¥lib”を指定し追加を行う。
3. プロジェクトオプション設定でリンカ入力の依存ファイル”mpi.ch2lib”を追加する。

これらの設定を行なうことで MPICH2 による並列プログラムのコンパイルが可能となる。

2.3 行列積プログラム

本研究では n 行 n 列の正方行列 A を乱数により生成し、$A \times A$ の行列積計算を行うプログラムを作成した。なお、乱数生成は処理を統括するホストのみで行う。

本プログラムでは、行列積計算を並列化するために、問題サイズの分割を行っている。配列のサイズは n^2 なので、プロセッサ数を p とした時に n^2/p のサイズに領域分割し、各プロセッサはそれぞれ同じ大きさの領域の計算を担当する。担当領域は MPI の機能によって各プロセッサに固有に割り振られるランク（p0,p1…と表す）の若い順番に領域を割り当てられる。領域分割の模式図を図 2 に示す。

図 2 行列の各プロセッサへの領域分割の概念図

本研究で作成した並列行列演算プログラムの動きを以下に挙げる。

1. ホスト計算機は、n 行 n 列の正方行列 A を乱数により生成する。
2. 各プロセッサは割り当てられた領域の行列積計算を行うために正方行列 A の全ての要素のデータを持つ必要がある。そのため、ホストは生成した正方行列 A を全てのノードにそのコピーを送信する。データの送受信には MPI によって用意された単方向通信関数 MPI_Send（送信側）と MPI_Recv（受信側）を用いる。
3. 各ノードは割り当てられた領域の行列積演算を行う。
4. 各ノードはホストへ計算結果を返す。この時も、前述の MPI 関数を用いて通信を行う。
5. ホストは、各ノードから受信した計算結果から解となる行列を生成する。ランクの順番で領域分割されているため、ホストはランクの順番に受信した計算結果を結合していくことで最終的な行列積を求めることが出来る。

図 3 に本研究で作成した並列行列積アルゴリズムの実行の概念図を示す。

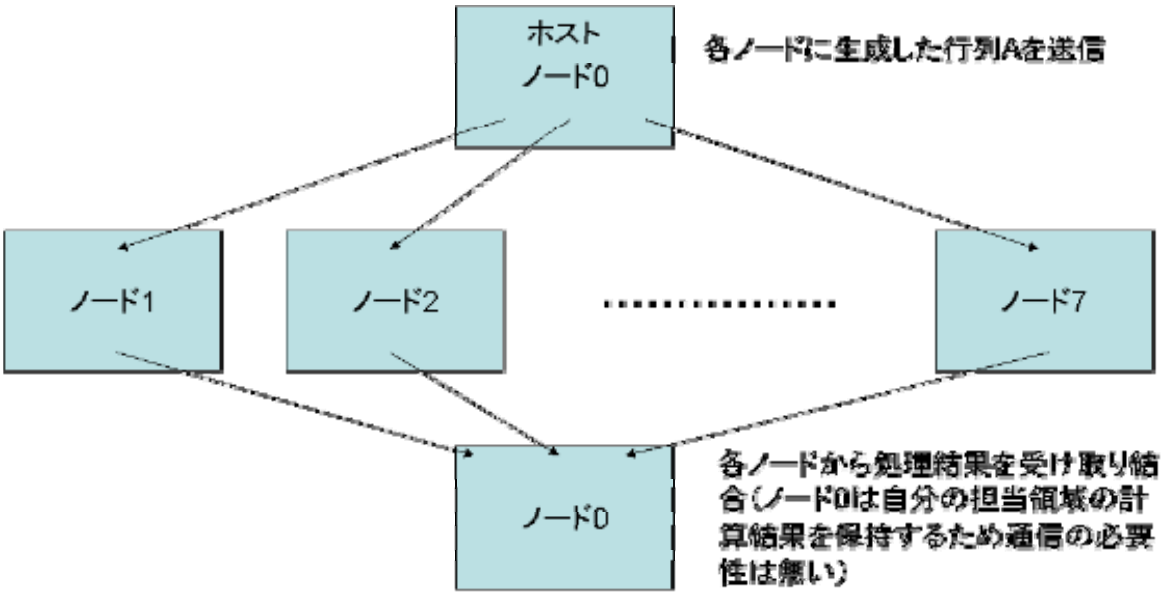

図 3 並列行列積アルゴリズムの実行の概念図

第3章　結果

3.1 検証方法

今回の実験での検証では、行列のサイズを n=1024 と n=2048 のそれぞれの場合において、並列計算に用いるプロセッサ数を 1・2・4・8 と変化させ、それぞれ連続 10 回試行の平均時間・最多大時間・最小時間を計測を行う。

3.2 計測結果

並列行列積計算プログラムを実行した際に、ホストから各ノードへの正方行列 A のコピー作業が全て完了してから行列積計算を行いホストで各ノードの計算結果を全て結合して最終的な $A \times A$ の解を得られるまでの処理時間の結果を以下に示す。

各プロセッサ数において 10 回試行した時の処理時間の平均時間・最大時間・最小時間を表 1 に、CPU 数の変化によっての処理時間の推移を図 4 示す。表 1 より、行列積計算を並列化した場合、全体的に処理時間の向上が行なえていることが分かる。また、図 4 より CPU 数の増加による処理時間の変化は反比例に近いものと言える。

表 1 並列行列積プログラムの実行時間

行列サイズ / CPU 数	1024			2048		
	平均時間	最大時間	最小時間	平均時間	最大時間	最小時間
1	76.227	76.203	76.193	610.610	611.275	610.358
2	40.836	40.848	40.809	323.845	324.314	323.001
4	21.701	28.719	20.893	174.462	175.244	173.778
8	12.572	13.131	12.405	89.279	89.614	89.052

単位：秒

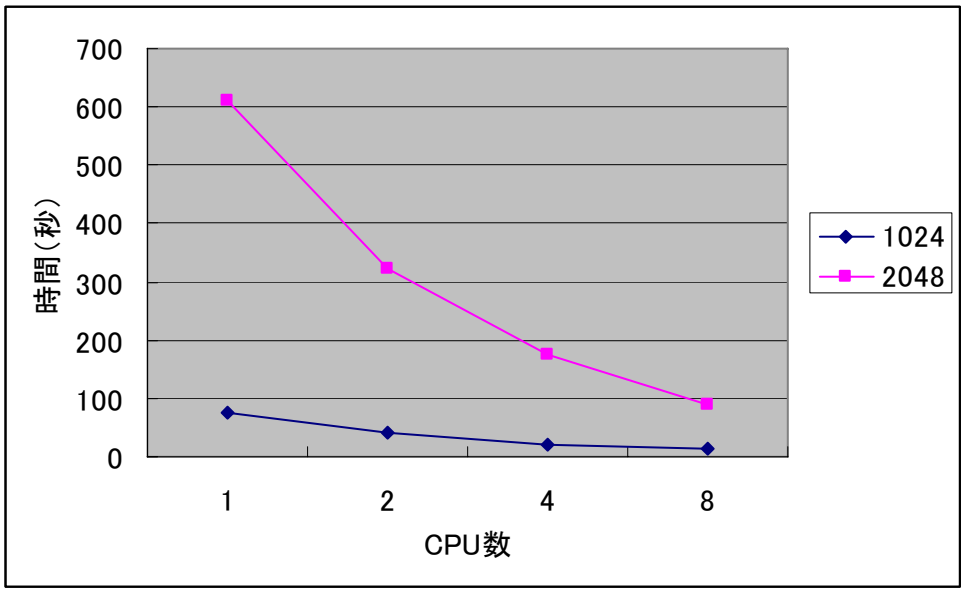

図 4 CPU 数の変化に伴う処理時間の推移

第4章　考察

図 4 より CPU 数と速度上昇が反比例に近い関係であると言える。領域分割を均等に行い均質な性能の計算機を用いているが、通信・遅延時間と結合処理の時間によるオーバヘッドから、完全な反比例にはならなかったと考えられる。また、オーバヘッドの主な部分は通信・遅延時間であることから、CPU 数の増加によるオーバヘッドの増加は正比例に近い関係であると考えられる。

以上から、均質な CPU を用いた場合 CPU 数を増加させ、均等に問題領域を分割することで単純に処理時間を短縮可能であると言える。しかし、極端に CPU 数を増加させた場合、オーバヘッドが膨大な値になることが予測される。そのため、仮想並列計算機の構築において極端に CPU 数を増加させる場合は、計算時間とオーバヘッドとのトレードオフを考える必要性があると言える。

第5章　結論および今後の課題

本研究では、MPI による仮想並列計算機の性能を実験的に評価を行った。評価方法として n 行 n 列の正方行列 A を用意し、$A \times A$ の行列積計算を並列化した並列行列積計算プログラムを作成し、1 台の計算機での逐次処理時における行列積計算の処理時間と比較して、複数台の計算機を使用しての並列行列積計算においてどれだけの処理時間の向上が出来ているかの検証を行った。

本研究の検証により、処理を高速化するために問題サイズの分割による並列化を行うことが有効であることが示せた。また、仮想並列計算機は既存の計算機・ネットワークを用いて並列処理を実装可能であるため、コスト面においても優秀であると言える。

今後の課題としては、ノード数を増加させ続けた場合のオーバヘッドの増加が定量的であるか調査することが挙げられる。

謝辞

本研究をするにあたり、石水隆先生には多忙の中数々のご指導をして頂き、本当にありがとうございます。また、院生の横瀬氏には MPI に関する様々な助言を頂いたことも大変感謝しています。そして、実験に協力して頂いた木村君、鄭君、中川君、新納君、藤山君、前田君、渡辺君にも色々とお世話になりありがとうございました。

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] W.グロップ,E.ラスク,T.タークル著, 畑崎隆雄 訳：実践 MPI-2 メッセージパッシング・インターフェースの上級者向け機能, ピアソン・エデュケーション(2002)

[3] 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

[4] J.JáJá：An Introduction to Parallel Algorithms ,Addison Wesley(1992)

[5] 日本アイ・ビー・エム システムズ・エンジニアリング株式会社 著：グリッド・コンピューティングとは何か, ソフトバンクパブリッシング株式会社(2004)

[6] OpenMP, http://www.openmp.org/blog/

[7] 牛島省 著：OpenMP による並列プログラミングと数値計算法, 丸善株式会社(2006)

[8] OAK RIDGE National Laboratory, http://ww.ornl.gov/

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

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

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

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

[13] LAM / MPI Parallel Computing, http://www.lam-mpi.org/

[14] 船山正樹他 訳, オハイオスーパーコンピュータセンタ著：MPI 入門 / LAM による開発, http://phase.hpcc.jp/phase/mpi-j/mpiprimj.pdf

[15] パナソニック, 製品情報 | レッツノート W2(CF-W2E), http://panasonic.jp/pc/support/products/w2e/index.html

[16] Microsoft, http://www.microsoft.com/ja/jp/default.aspx

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

地球シミュレータセンター, http://www.es.jamstec.go.jp/index.html