卒業研究報告書

題目

MPI を用いた画像の並列処理

指 導 教 員

石水 隆 助教

報告者

05-1-037-0044

上田勇人

近畿大学理工学部情報学科

平成 21 年 1 月 31 日提出

概要

近年、計算機は目覚しい発展を遂げつつある。計算機が扱うデータの量が膨大なものになる一方、データの処理に対し高速性も求められるようになった。高速な処理を行うには、主に、1 台の計算機の処理速度を向上させる方法と、複数台の計算機をネットワークで繋げ、並列計算を行う方法の2つがある。1 台の計算機の処理速度の向上には物理的な限界があるため、並列処理による高速化が注目されている。しかしながら、高性能な計算機はとても高価なものであり、一般で利用するのは困難である。その解決策として、ネットワークを用いて複数台の安価な計算機を繋ぎ、計算機全体を1台の並列計算機として扱うことのできるクラスタ処理が注目されてきている。

本研究では、無償で提供されておりクラスタ処理による並列計算を可能にするソフトウェアである MPI(Message Passing Interface)を用いてその性能を実験によって評価する。評価方法は、BMP(Bit Map)画像から JPEG(Joint Photographic Experts Group)画像への変換を並列で行うプログラムを作成し処理速度がどの程度向上したかを測定する。

目次

1 序論

1.1 本研究の背景

1.1.1 並列処理

ある 1 つの処理を、複数のプロセッサを用いて処理を行い、単一のプロセッサでの処理よりも高速に計算処理を行なうことを並列処理(Parallel Processing)という。

並列処理はタンパク質の構造解析や地球の大気の循環など大規模なシミュレーションなど、逐次処理では膨大な時間がかかる場所において逐次計算機よりも短時間で解けることから利用されている。

1.1.2 並列計算機

複数のプロセッサを持ち、並列計算を行うことが可能である計算機を並列計算機(Parallel Computer)という。

並列計算機には大きく分けて 2 つに大別され、全てのプロセッサが同じメモリを通して使用する共有メモリ型並列処理と、各プロセッサが個々に局所メモリを持ち、ネットワークを用いてデータをやり取りしながら処理を行う分散メモリ型並列処理がある。一般に、共有メモリ型並列計算機の方がプロセッサ間の通信にかかるコストが小さく、理想的な並列化が得られやすい。しかし、共有メモリ型並列計算機は設計可能なプロセッサ台数の上限が低く、拡張性にも乏しいことから、現在は分散メモリ型の並列計算機が主流となっている。また、並列計算機は非常に高価であり、ごく一部の大学や研究所でしか利用できない。

しかし、現在では、複数の計算機をネットワーク接続し、計算機郡全体を 1 台の仮想並列計算機とするクラスタ処理が注目されており、クラスタ環境を実現するソフトウェアも無料で提供されている。このため、クラスタ環境による並列処理は身近なものになっている。クラスタ環境を実現するソフトウェアには、現在 PVM(Parallel Virtual Machine)[1]や MPI(Message Passing Interface)[2]といったものがある。

PVM・MPI については次項にて説明する。

1.1.3 PVM

PVM(Parallel Virtual Machine)[1]は、1991 年に米国のオークリッジ国立研究所(Oak Ride National Laboratory)[3]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なうための並列化ライブラリである。

PVM は TCP/IP の通信ライブラリで一般的に使用されている LAN(Local Area Network)があれば並列処理が実行可能であり、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まで多くの種類の計算機で並列処理環境を構築することが出来る。

PVM の構成は 2 つに大きく分けられる。1 つはデーモン(pdmd3)であり、もう 1 つは PVM インターフェイスルーチンライブラリである。PVM を実行中は、PVM によって構成された仮想並列計算機上にある各々の計算機にデーモンが存在し、このデーモンを使用し通信を行なっている。複数のユーザー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。ライブラリは、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。また、各ユーザーは 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 の大きな特徴として、耐故障性(Fault Tolerant)が挙げられる。通常、仮想並列計算機は計算中にある 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 では、任意で計算機の追加や削除が行なえると共に、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なく続行できる。また、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ために、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。

1.1.4 MPI

MPI(Message Passing Interface)[2]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、様々な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と同等の機能を持っており、PVMの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 を使用するユーザーは、通信を考慮せずプログラムを組むことが出来る。

大きく PVM と違う点は、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 PVM と同様に、パーソナルコンピュータからスーパー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[6]や LAM[7]といったものがある。また、MPI のサポートするプログラミング言語は多く、C 言語から C++、Fortran、そして最近では Java などに対応している。

1.2 本研究の目的

本研究の目的は、複数台の計算機をネットワークで繋げ、仮想並列計算機を MPI によって構成し、その性能を評価する。評価方法として BMP 画像から JPEG 画像にエンコードさせる並列 JPEG エンコーダを作成することで、1 台の計算機での逐次処理時におけるエンコードの処理時間が、複数台の計算機を使用しての並列エンコーダにおいてどれだけの処理時間の向上が出来ているかの測定を行なう。

1.3 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、本研究の環境および MPI の導入方法と並列 JPEG エンコーダについて述べる。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

2 実験環境

今回実験を行った環境は以下の通りである。

2.1 使用機器

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、研究室にある計算機にインストールし環境設定を行なう。今回の研究では、Argonne National Laboratory[4]が無料で配布している MPICH2 というソフトウェアを使用することにする。この MPICH2 を使用する理由は、MPI の特徴である移植性の高さ、MPI の新規格である MPI-2 をある程度サポートしている点や、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ためである。

この研究で使用する計算機は、Microsoft 社が提供販売を行なっている OS である Windows 系を使用する。Windows OS を使用する理由としては、現在の一般家庭や教育施設、そして企業などの場所において計算機に取り入れられている Windows は他社の OS よりも圧倒的に幅広く使用されているためである。

今回の研究において、使用する計算機の性能を下記に記す。

1. PC1：OS　Microsoft Windows XP Professional
   CPU　Intel Pentium M　1.5GHz
   Memory　512MB
2. PC2：OS　Microsoft Windows XP Professional
   CPU　Intel Pentium 4　3.2GHz
   Memory　2048MB

3. PC3：OS　　Microsoft Windows XP Professional
CPU　　Intel Pentium M　1.5GHz
Memory　512MB

研究にはこのスペックをもつ計算機計 4 台での実験を行なう。また、仮想並列計算機を構築する際に LAN やハブ(Hub)を使用し計算機をネットワークでつないでいる。この構成図を図 1 に示す。

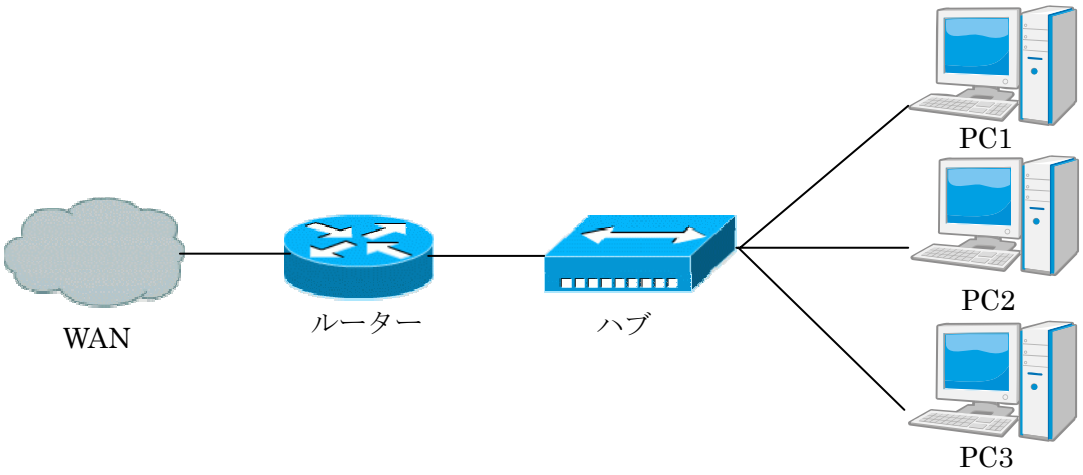


図 1：仮想並列計算機の構成

2.2 実験環境の構築

2.2.1 MPICH2 のインストール及び設定

MPICH2 を使用するには、計算機に MPICH2 をインストールする必要がある。Windows 用 MPICH2 の exe ファイルを公式サイト[5]よりダウンロードした後に exe ファイルをクリックすることにより、インストーラーが起動し自動的にインストールされる。デフォルトでは C:\Program Files\MPICH2 フォルダにインストールを行なわれる設定になっているので、環境に合わせて適当なフォルダを指定する

インストール後の環境設定として、インストール先のフォルダにある bin ファイルに環境変数でパス(PATH)指定をする。

また、Windows OS の場合には全ての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。さらに、プロセスの実行にはすべての計算機に共通したファイル構成で、実行ファイルを置く必要がある。そのため、共有するフォルダを作成する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。

2.2.2 Visual C++ 2005 Express Edition の設定

今回作成する並列 JPEG エンコーダに使用するプログラム言語は C 言語で行い、コンパイルツールは Microsoft 社製の Visual C++2005Express Edition 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設定しなければならない。まず、空のプロジェクトを作成し、ツールオプションからインクルードファイルとライブラリファイルを MPICH2 フォルダにある lib と include フォルダを指定し追加を行なう。次にプロジェクトの設定でリンカ入力の依存ファイルを追加する、追加する依存ファイルは mpi.lib である。これらの設定を行なうことで MPIDH2 による並列プログラミングが可能となる。

2.3 使用する画像

本研究では BMP(Bit Map)画及び JPEG(Joint Photographic Experts Group)画像を使用する。今回の検証では入力画像として、サイズが 2560×1920、24bit のフルカラーの画像 50 枚を使用し、変換枚数を 1･5･10･25･50 と変化させたときの各枚数における処理時間を検証する。

2.3.1 BMP

BMP[8]は Windows が標準でサポートしている形式であり、白黒からフルカラーまで対応している。基本的には無圧縮だが、オプションで圧縮することも可能である。

2.3.2 JPEG

JPEG[9]は静止画像データの圧縮方式の一つであり、”Joint Photographic Experts Group”という ISO(International Organization Standardization)により設立された検討グループの頭文字がそのまま使われている。

JPEG は圧縮の際に、若干の画質劣化を許容する方式と、全く劣化のない方式を選ぶことができ、許容する場合はどの程度劣化させるかを指定することができる。圧縮率は概ね 1/10～1/100 程度であり、写真などの自然画の圧縮には効果的だが、コンピュータグラフィックスには向かない形式である。

現在では、JPEG 画像を繋ぎ合わせて動画にした Motion-JPEG という形式や、JPEG 圧縮に起きる歪みを解消するために、新しい JPEG アルゴリズムである JPEG 2000 という規格がでている。JPEG よりも JPEG 2000 は高圧縮でひずみも無いという理想的な画像圧縮方法であるが、エンコードの処理時間がかかり、また、アルゴリズムも難解なために今でもそれほど広まってはいない。

2.4 並列 JPEG エンコーダ

今回作成した並列 JPEG エンコーダは以下のように処理を行う。また、処理の流れを図 2 に示す。

1. メイン PC は BMP 画像を読み込み、分割する。その際の注意として、BMP 画像は RGB の 3 色を 1 次元配列で保存されている。そのため、縦のサイズを分割する際にピクセル数が PC の台数分で割り切れない場合画像の分割が不可能になる。その際には、割り切れるように計算機の台数を増減する作業を行なわなければならない。
2. メイン PC は分割した BMP 画像をネットワークでつながった各サブ PC に送信する。
3. 各サブ PC はメイン PC より送信された画像データを受信する。
4. 全ての PC は各 PC が持っている分割された BMP 画像を JPEG エンコーダで JPEG 画像に変換する。
5. サブ PC は変換した JPEG 画像をメイン PC に送信する。
6. メイン PC はサブ PC から分割された JPEG 画像を受信する。
7. メイン PC は各 JPEG 画像を結合し 1 枚の JPEG 画像を出力する。

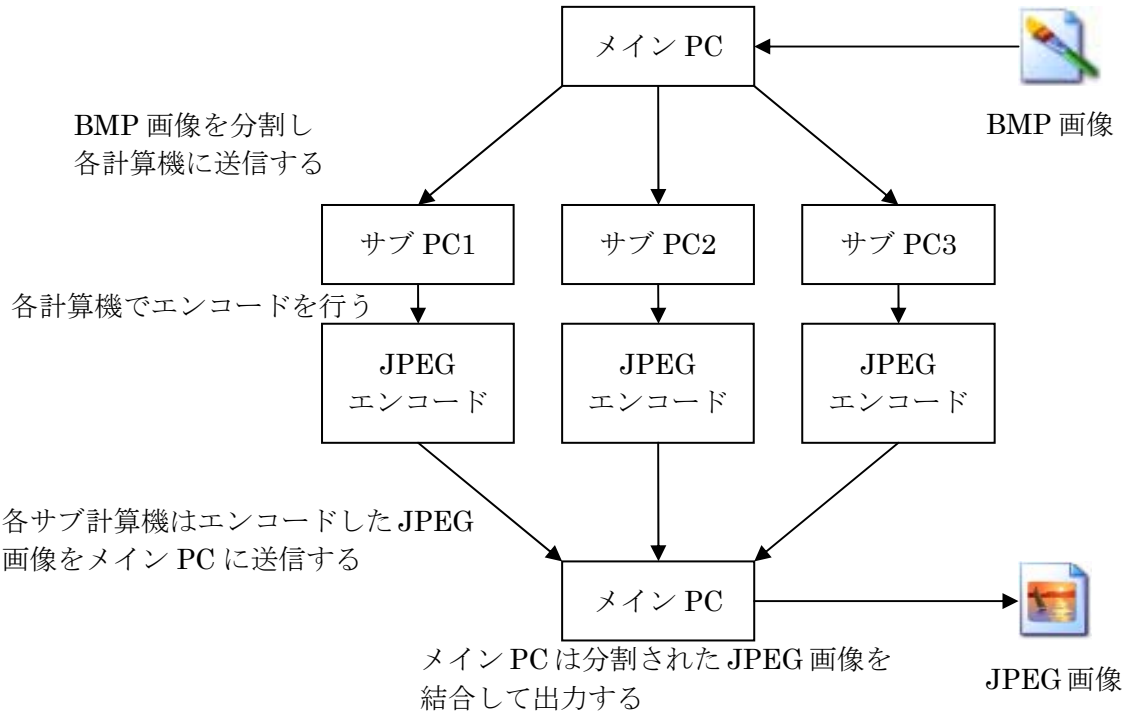


図 2：並列 JPEG エンコーダの処理の流れ

3　結果

今回の検証では、処理を行う BMP 画像の枚数を 1・5・10・25・50 枚と変化させエンコード処理を行う。まず、1 台で動作させた場合の処理時間を計測し、次に JPEG 並列エンコーダを構成する計算機の台数を 2 台、3 台と変化させることでどれだけの処理時間が短縮されたかの計測を行なった。

なお、計測に使用する計算機は前章において示している。次に仮想並列計算機を構成する場合において使用された PC の組み合わせを下記に示しておく。

1. 1 台の場合：メイン PC1 台
2. 2 台の場合：メイン PC1 台とサブ PC1 台
3. 3 台の場合：メイン PC1 台とサブ PC2 台

以下の節では、今回の検証で得られた結果を処理時間、スピードアップ率、演算時間、通信時間それぞれについて示す。

3.1 処理時間

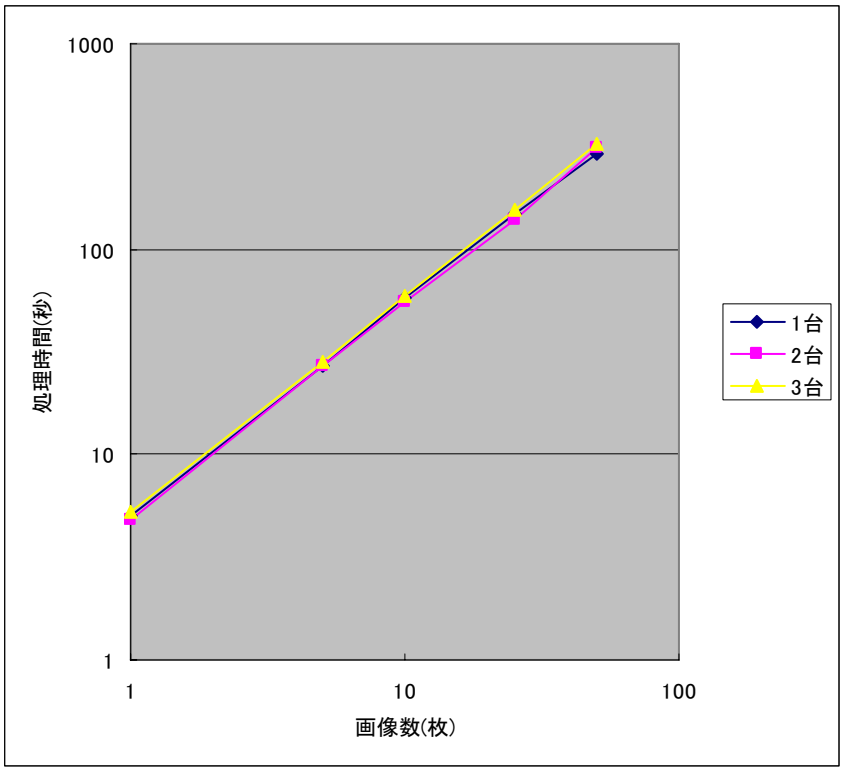
並列 JPEG エンコーダが BMP 画像を読み込み、エンコード処理を行い 1 枚の JPEG 画像として出力されるまでに要した時間を表 1 に、画像の枚数変化に伴う処理時間をグラフ 1 に示す。

この計測結果では、並列 JPEG エンコーダを使用した場合、PC1 台の場合と比べて処理時間の短縮が見られた場合があった。しかしながら、逆に処理時間が悪化してしまった場合も目立った。

表 1：処理時間

		BMP 枚数				
		1 枚	5 枚	10 枚	25 枚	50 枚
PC 数	1 台	5.01	27.11	58.11	150.23	293.73
	2 台	4.85	27.20	55.37	139.89	308.42
	3 台	5.27	28.47	58.85	154.33	323.59

(単位は秒)



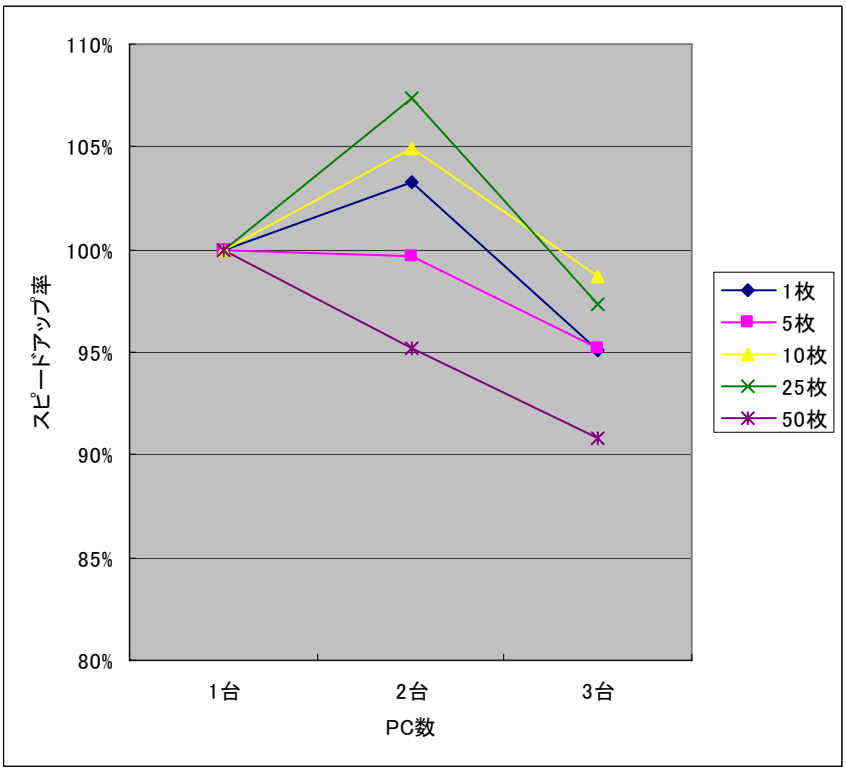
グラフ 1：画像の枚数変化に伴う処理時間

3.2 スピードアップ率

ここでは、PC の台数の増加によるスピードアップ率についての結果を表 2・グラフ 2 に示す。理論上は PC の台数が増加するに従い処理速度は向上するはずである。今回の実験では PC を 1 台から 2 台に増加させた際には、一部例外も見られたが全体的に高速化がみられた。しかしながら、2 台から 3 台に増加させた際は逆に処理時間が悪化してしまった。

表 2：スピードアップ率

		BMP 枚数				
		1 枚	5 枚	10 枚	25 枚	50 枚
PC 数	1 台	100%	100%	100%	100%	100%
	2 台	103.3%	99.7%	104.9%	107.4%	95.2%
	3 台	95.1%	95.2%	98.7%	97.3%	90.8%



グラフ 2：PC の増加に伴うスピードアップ率

3.3 演算時間

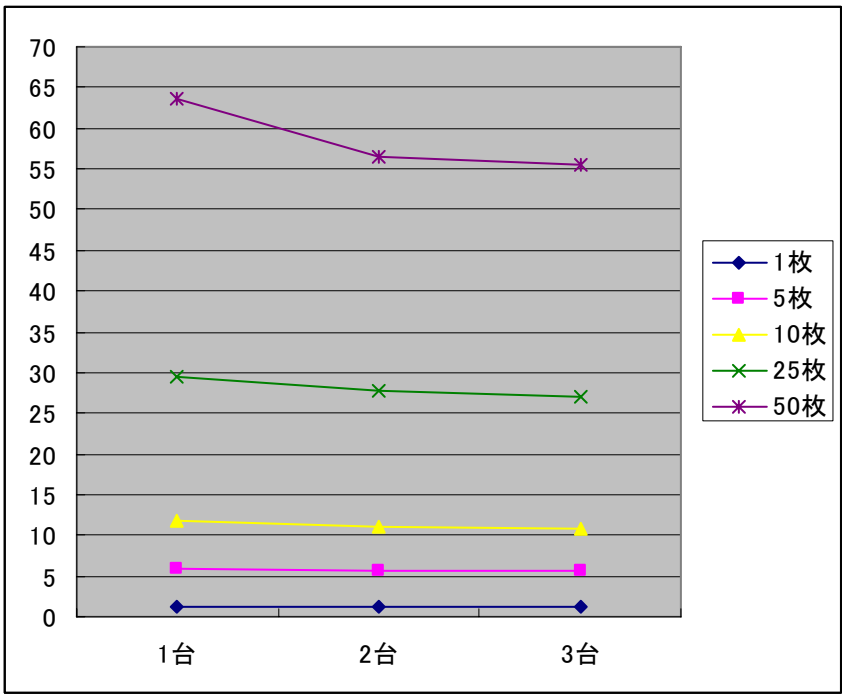
ここでは、全体の処理時間ではなく BMP 画像を JPEG 画像にエンコードする処理の演算時間を表 3・PC の増加に伴う演算時間をグラフ 3 に示す。

これらから読み取れるように、演算時間は PC の台数の増加に伴って着実に減少している。

表 3：演算時間

		BMP 枚数				
		1 枚	5 枚	10 枚	25 枚	50 枚
PC 数	1 台	1.29	5.87	11.68	29.56	63.61
	2 台	1.14	5.73	11.16	27.81	56.48
	3 台	1.16	5.64	10.89	26.90	55.58

(単位は秒)



グラフ 3：PC の増加に伴う演算時間

3.4 通信時間

並列 JPEG エンコーダの動作の中で PC 間の通信を行った時間を表 4 に、処理時間における通信時間の割合を表 5 に示す。

表 5 からは画像数の増加と PC 数の増加に従って処理時間における通信時間の割合が増加していることが読み取れる。

表 4：通信時間

		BMP 枚数				
		1 枚	5 枚	10 枚	25 枚	50 枚
PC 数	2 台	0.16	0.86	1.75	4.56	9.41
	3 台	0.20	1.24	2.60	6.62	13.23

(単位は秒)

表 5：処理時間における通信時間の割合

		BMP 枚数				
		1 枚	5 枚	10 枚	25 枚	50 枚
PC 数	2 台	3.30%	3.16%	3.16%	3.26%	3.05%
	3 台	3.80%	4.36%	4.42%	4.29%	4.09%

4　考察

今回の計測結果では、エンコード処理の演算時間のみを見てみると PC 数の増加に伴い演算時間が短縮された。しかしながら、全体を通しての処理時間を考えてみると処理時間が悪化している場合が目立つ。この原因は、PC 数の増加に伴い通信時間が増加する為であると思われる。処理時間からの割合を見ると 1%程度の増大だが、通信時間が増大すると同期時間も同様に増大するためこの影響が大きいものと考えられる。通信時間を減らす為には、シンプルな構成で高速なネットワーク環境を構築することが有効だと考えられる。また、処理するデータ量をスペックの高い PC と低い PC で差をつけることにより、各 PC のエンコード処理時間の差を吸収し、効率よく通信ができ、全体の処理速度の向上が期待できる。

5　結論・今後の課題

本研究では、PC の台数が 2 台で BMP 画像が 1 枚から 25 枚までと非常に限定的な条件でしか MPI の有効性を実証することが出来なかった。

今回の実験では 100BASE-TX の LAN を使用していたが、ギガビットイーサネットの使用やネットワーク構成の最適化によって通信速度が向上し、通信時間の短縮が期待できる。また、送信するデータ量を高スペックの PC と低スペックの PC で差をつけることでスペックの差を吸収し、効率よく変換でき、処理時間の短縮が出来るものと考えられる。

6　謝辞

本研究におきまして様々な助言を頂いた石水隆先生に感謝の意を表します。ご迷惑も沢山お掛けしましたが一年間本当にありがとうございました。また、情報論理工学研究室の皆様にも感謝いたします。

参考文献

[1] http://www.csm.ornl.gov/pvm/pvm_home.html：PVM 公式ページ
[2] William Gropp, Ewing Lusk, Rajeev Thakur 著,畑崎隆雄 訳
実践 MPI-2　ピアソン・エデュケーション(2002)
[3] http://www.ornl.gov/：オークリッジ国立研究所
[4] http://www.anl.gov/：アルゴンヌ国立研究所
[5] http://www.mcs.anl.gov/research/projects/mpich2/：MPICH2 公式ページ
[6] http://www-unix.mcs.anl.gov/mpi/index.html：MPICH 公式ページ(現在は閲覧不可)
[7] http://www.lam-mpi.org/：LAM 公式ページ
[8] http://e-words.jp/w/BMP.html：BMP 解説
[9] http://e-words.jp/w/JPEG.html：JPEG 解説